Universidad de las Ciencias Informáticas

Facultad 1

Título: "Propuesta de Arquitectura del Sistema Gestión de Fotos del Proyecto Control de Acceso".

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Rolando Ramiro Barrientos Dominguez

Tutor: Lic. Osmanys Alonso Guerra

Ciudad de la Habana
4 de julio del 2008

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste, firmo el presente a los ____ días del mes de _____ del año_____.

_____	_____
Rolando Ramiro Barrientos Dominguez	Lic. Osmanys Alonso Guerra
[Autor]	[Tutor]

Datos de Contacto

Síntesis del Tutor: Osmanys Alonso Guerra.
Licenciado en Ciencias de la Computación.
Recién graduado en Adiestramiento.
Dpto. de Programación Facultad 1

AGRADECIMIENTOS

El presente trabajo de diploma para optar por el título de "Ingeniero en Ciencias Informáticas", no es sólo lo que su nombre indica. En el están contenidas toda una serie de etapas plácidas o turbulentas, hermosas o tristes, por las cuales he transitado a lo largo de estos cinco años.

Hoy, cuando mi vida está a punto de entrar en una etapa diferente y dejar atrás todas estas cosas bellas que se viven sólo en la "UNIVERSIDAD", siento un inmenso orgullo porque con mi resultado he sido capaz de engendrar el fruto de todo el sacrificio, empeño, dedicación, apoyo, confianza, cariño, amistad, educación y amor, que me ha sido depositado a lo largo de estos años.

A todos los que hicieron posibles que hoy albergue tantas emociones lindas dentro mi corazón, que mi vida esté llena de mil razones para seguir adelante y que me sienta tan orgulloso de mí, reciban pues, mi eterno agradecimiento:

A la rosa más fragante y bella del jardín de mi corazón. Al amor de mis amores. A la mitad de mi vida. A mi razón de ser. A mi "cosita" más bella. Gracias por existir, gracias por haberme traído al mundo, gracias por darle a mi vida nuevas manifestaciones del sentimiento: "Amor", gracias por tanto cariño y ser tan complaciente conmigo, gracias por estar siempre ahí cuando más necesité de tu ternura, gracias por sembrar en mí tantos sentimientos lindos. Gracias e infinitamente gracias, a mi "mima" Isabel Cristina.

A mi adorado papá Rolando por tanta educación, respeto, carácter, apoyo y amor. Por haberme enseñado a ser el hombre que hoy soy. Por ser un ejemplo para mí.

A mi querido hermano Victor por ser mi fuente inspiración día a día y constituir un pedazo de mi corazón. Yo nunca….nunca te voy a abandonar, porque te amo brother.

A la otra "cosita" bella que tengo en la vida: mi abuela Dorgeris, por mimarme y malcriarme tanto. Por ser tan especial para mí.

A toda mi familia, en especial a mis tíos Pimpe y Pirry, por sus consejos, apoyo y cariño.

A mi abuelo Ramiro donde quiera que esté, porque a pesar de no haberlo conocido sé que siempre ha estado a mi lado para guiarme los pasos.

A mis queridos primos Luisi y Jose por haberme acogido y apoyado durante todo este tiempo. Gracias de corazón, hermanos.

A mi hermano del alma Alexmay, por siempre estar ahí cuando te necesité. Por estar a mi lado en los momentos más difíciles y celebrarme siempre mis cumpleaños. ¡Jajajaja!

A mis amigos Karina, Yoyi, Danilo, Leonel y Adrián, por su amistad incondicional. Pido gracias a Dios por haberme permitido conocerlos.

A mi tutor Osmanys Alonso Guerra por su apoyo y confianza en mí.

A Gabriel La O, José Ramón Hermosilla y Alieski Sarmiento por su ayuda en el desarrollo del trabajo.

A nuestro Comandante en Jefe Fidel Castro Ruz, la Revolución y la Universidad de las Ciencias Informáticas (UCI), por haberme formado y convertido en lo que hoy ya soy.

DEDICATORIA:

Dedico el presente trabajo de forma muy especial a mis padres Isabel Cristina y Rolando y a mis hermanos Victor Manuel y Rodney. A mis hermanos de corazón Alexmay y Yoyi. A toda mi familia, de forma especial a mis abuelos Dorgeris, Aurora y Victor; a mis tíos Pimpe y Pirry y a mis primos Luisi, Jose y Diusmaida. A mi novia Yanelis por tanto cariño.

A mis amigos, que a pesar de las adversidades, en los buenos y malos momentos, siempre han estado para mí.

Roly.

Resumen

Con el desarrollo del presente trabajo se pretende diseñar la arquitectura de un servicio web y una aplicación web para su administración, que constituyen el sistema “Gestión de fotos” del proyecto “Control de Acceso”. Dicho servicio deberá ser consumido por sistemas pertenecientes a la Universidad de las Ciencias Informáticas (Todos los que trabajan con imágenes de personas). Para ello se estudiarán y seleccionarán las mejores tecnologías, herramientas, metodologías y lenguajes de programación, utilizados para el desarrollo de aplicaciones bajo la política de software libre, con el objetivo de realizar una correcta selección de los más ajustados para dar cumplimiento al presente trabajo. Al mismo tiempo se tomarán las principales ideas de algunos sistemas hechos en software libre que gestionen fotos (en Cuba y el Mundo) con el objetivo de comprender su arquitectura, funcionamiento y protocolo de gestión de la información. Al concluir se habrán obtenido las vistas arquitectónicas y los estándares de desarrollo de la futura aplicación propuesta.

Palabras Claves:
Arquitectura, servicio web, módulo, software libre, gestión, protocolo, vistas arquitectónicas, aplicación.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En los primeros meses del año 2002, la máxima dirección de la Revolución comienza el estudio de la viabilidad de construir una escuela para formar futuros ingenieros en ciencias informáticas en lo que hasta poco tiempo atrás había sido la importante estación rusa de radioescucha: “Lourdes”. Comienzan a tomarse una serie de medidas para en tiempo record tener listas las condiciones para recibir y alojar a sus primeros educandos. Es precisamente así como germina, gracias a la ingeniosa idea de nuestro Comandante en Jefe Fidel Castro Ruz, la primera universidad surgida al calor de la Batalla de Ideas: Universidad de las Ciencias Informáticas (UCI).

Con el comienzo del curso (2002-2003) se hacía imprescindible garantizar un mecanismo (credencial) para la identificación de los estudiantes, profesores y trabajadores de la UCI y que al mismo tiempo eliminara el acceso de personas ajenas al centro, no autorizadas a circular dentro de él. De esta forma surge el proyecto “Acreditación”. Inicialmente las credenciales eran de papel cartón, con los datos impresos de la persona correspondiente, un número y la especificación de su tipo (estudiante, trabajador, profesor o directivo). Con el paso del tiempo se comienzan a imprimir lo primeros “solapines”, que poco a poco fueron siendo perfeccionados.

A inicios del año 2006 surge el proyecto “Control de Acceso” con el objetivo de desarrollar un software para gestionar el control de acceso de personas y vehículos a la universidad, así como la entrada a los comedores. Un año después “Control de Acceso” integra en su trabajo lo referente a “Acreditación”. De esta forma se comienza a realizar una revisión detallada de los procesos a informatizar.

Situación problémica:

1. Las base de datos UCI almacena en sus campos toda la información relacionada con las personas (estudiantes o trabajadores) que pertenecen al centro, excepto sus fotografías que son guardadas en un repositorio; trayendo consigo que si ocurre algún

problema internamente en dicha base de datos, se perderán las informaciones puesto a que no se sabrá a qué imagen corresponden cuáles datos.

2. El programa que existe actualmente para estandarizar las fotos a un tamaño y resolución determinado no posee un diseño arquitectónico, que además permita que se guarden datos dentro de ellas (Metadatos). Dígase datos a los valores de los campos de la BD asociados al individuo (estudiante o trabajador) fotografiado. Para su realización se utilizaron herramientas y tecnologías completamente propietarias.

3. No se cuenta con un repositorio donde se puedan guardar una serie de fotos referentes a una persona junto a sus datos, a modo de historial, de forma tal que cuando se actualice una foto no se pierda la anterior.

Problema científico: ¿Qué Arquitectura para software libre utilizar en el Sistema de Gestión de Fotos del proyecto Control de Acceso?

Objetivo general: Diseñar la arquitectura del sistema Gestión de Fotos del proyecto Control de Acceso.

Objeto de estudio: “Arquitectura de aplicaciones informáticas”.

Campo de acción: “Arquitectura de aplicaciones web”.

Idea a defender:

Se supone que con el diseño de una arquitectura capaz de organizar y relacionar los componentes arquitectónicos del sistema de Gestión de Fotos, del proyecto Control de Acceso, sea posible realizar una aplicación informática que resuelva los problemas actuales de las bases de datos de personas en la UCI y cumpla con los nuevos requerimientos.

Tareas de investigación

1. Analizar aplicaciones hechas en Cuba o el mundo, bajo la política de software libre, que

utilicen arquitectura en capas.

2. Analizar gestores de fotos existentes a nivel nacional o internacional, en vistas a comprender su protocolo de funcionamiento.

3. Realizar el diseño arquitectónico del sistema “Gestión de Fotos” del proyecto Control de Acceso.

Métodos investigativos aplicados:

Métodos teóricos:

Análisis y Síntesis: Para el procesamiento de la información y arribar a las conclusiones de la investigación, así como precisar las características del modelo arquitectónico propuesto.

Histórico – Lógico: Para determinar las tendencias actuales de desarrollo de los modelos y enfoques arquitectónicos.

Métodos empíricos:

Observación. Externa, no Incluida y Abierta: La observación es realizada desde fuera del entorno estudiado, sin que el observador forme parte de él y con previo conocimiento de todos. Este método es aplicado para la percepción selectiva de las propiedades y restricciones del sistema.

Entrevista: Para obtener una idea más detallada y precisa de lo que desean los clientes que el sistema realice.

El contenido del presente documento estará estructurado en tres capítulos:

Capítulo 1: Este capítulo comprenderá el estado del arte del tema tratado y hará referencia a sistemas vinculados al campo de acción en Cuba y el mundo. Se analizarán distintos patrones y estilos arquitectónicos. Además se realizará un estudio y fundamentación de las tecnologías, metodologías, herramientas y lenguajes, más usados actualmente para el desarrollo de software libre.

Capítulo 2: En este capítulo se definirán los patrones (arquitectónicos y de diseño), el estilo arquitectónico, las tecnologías, las metodologías, las herramientas y los lenguajes, que serán utilizados para el modelado y construcción del futuro sistema informático; además se realizará una descripción detallada de lo que deberá hacer el mismo.

Capítulo 3: En este capítulo se plantearán las metas y restricciones arquitectónicas que deberá cumplir el futuro sistema de software. Conjuntamente se obtendrán las vistas arquitectónicas explicadas de forma detallada.

Capítulo1: Fundamentación Teórica del Trabajo.

1.1 Introducción

En el presente capítulo se hace un estudio y análisis de sistemas existentes en Cuba o el mundo que gestiones fotos, con el objetivo de comprender su protocolo de funcionamiento. A su vez se analiza que es la “Arquitectura de Software”, sus principales corrientes, estilos y patrones. Finalmente, se muestran elementos que caracterizan a distintos lenguajes, herramientas, metodologías y tecnologías actuales para el desarrollo de software.

1.2 Estado del Arte

A continuación se analizarán sistemas existentes en Cuba o el mundo que gestionan fotos, con el objetivo de comprender su estructura y funcionamiento.

1.2.1 Sistemas de Gestión de Fotos en el Mundo

1.2.1.1 Sistema Photo-RDF

El Sistema Photo-RDF permite digitalizar y guardar las imágenes en formato JPEG y se describen en RDF, lo que puede hacer que sean más fáciles de encontrar las fotos que están siendo buscadas. Los metadatos dentro de las fotos son escritos con el programa de introducción de datos, que también permite editarlas en caso de alguna corrección necesaria. Las solicitudes desde la Web son servidas por Jigsaw, enviando la foto o los metadatos, dependiendo del tipo de solicitud.

De modo general el sistema permite:

1. Escanear las fotos y guardarlas en formato JPEG. Para obtener mayor calidad, puede usarse cualquier proceso que produzca este formato, incluyendo cámaras digitales.
2. Introducir y editar los metadatos de manera sencilla para cada una de las fotos y almacenar los datos en formato RDF dentro del archivo JPEG.

3. Servir datos a través de “Jigsaw” ya sean de las imágenes JPEG como la descripción RDF que está almacenada en ellas, usando negociación de contenido HTTP para determinar cuál de las dos desea el cliente.

1.2.1.2 El programa de introducción de datos "Rdfpic"

El programa de introducción de datos ha sido diseñado para habilitar la introducción rápida de metadatos para cantidades de fotos, asumiendo que las fotos normalmente serán de una o unas pocas series. La mayoría de los campos muestran por defecto el valor que fue introducido para la foto anterior, y ofrece un acceso rápido a los valores introducidos para las últimas fotos. Sólo deberán cambiarse los valores de unos pocos campos de una foto a la siguiente y la cantidad de escritura se reducirá.

El programa está escrito en Java, pero la interfaz de usuario de hecho se genera mientras el programa está en marcha, directamente desde una versión de los esquemas legible por un ordenador (normalmente no la sintaxis RDF sino una transformación de ella, con información equivalente). Esto significa que el programa no necesita ser cambiado cuando se cambian los esquemas RDF.

La extensión Jigsaw

Utiliza Negociación de Contenido como mejor manera de servir tanto la versión RDF como la imagen completa, usando los navegadores y herramientas existentes; y a su vez no excluye el uso de otras técnicas (como las extensiones HTTP) para recuperar y almacenar metadatos.

Los esquemas RDF (Marco de Trabajo para la Descripción de Recursos)

*Un **esquema** RDF es un conjunto de informaciones relativas a las clases de recursos que sirve para explicitar las relaciones jerárquicas que establecen entre ellos, o bien para matizar el carácter obligatorio u opcional de las propiedades y otras restricciones como el número de ocurrencias, etc. (1)*

1. *Esquema Dublin Core*: Es un esquema general para identificar obras originales, normalmente libros y artículos, pero también películas, pinturas o fotos. Contiene propiedades como el creador, editor, título, fecha de publicación y empresa editora.

2. Esquema Técnico: Este esquema recoge datos técnicos sobre la foto y la cámara, tales como el tipo de cámara, el tipo de película, la fecha de revelado de la película, y el escáner y software usados para digitalizarla.

3. Esquema de Contenido: Este esquema se usa para clasificar el tema de la foto por medio de un vocabulario controlado, que permite localizar fotos basándose en características tales como retrato, retrato de grupo, paisaje, arquitectura, deporte, animales, etc.

1.2.2 Sistemas de Gestión de Fotos en Cuba

Actualmente en Cuba, la mayor concentración de esfuerzos para el desarrollo de aplicaciones informáticas para la gestión de imágenes, bajo la política de software libre, se encuentra en la UCI. Se han obtenido muy buenos resultados en el trabajo conjunto con centros de alto prestigio pertenecientes al polo científico, desarrollándose productos en la gestión de imágenes médicas.

A pesar de esto, es válido aclarar que para el manejo del personal existente en la universidad, se ha hecho muy poco.

1.2.2.1 Casandra

Es un sistema de gestión de fotos creado por el Grupo de Procesamiento de Imágenes de la Universidad de las Ciencias Informáticas, con el objetivo de convertir a Cuba en un país a la vanguardia en la utilización de las más modernas tecnologías de base científico imagenológicas, además de proveer su carácter científico, en el tema de Procesamiento Digital de Imágenes y Señales, al Sistema Nacional de Salud y a otros Centros e Instituciones.

Básicamente, este sistema trabaja con algoritmos matemáticos para procesar las imágenes que se obtienen principalmente de equipos médicos de los hospitales, aunque también pueden ser bajadas de internet o tiradas por el propio equipo con cámaras digitales. Dentro de los algoritmos para procesar las imágenes se encuentran los que le dan formato y resolución. Este sistema no trabaja con metadatos.

Las principales herramientas utilizadas para lograr el software fueron Matlab, C# (como lenguaje predominante), C++ y como IDE Visual Studio 2005.

1.2.2.2 Sistema de Acreditación

Es una aplicación de tipo escritorio para gestión de fotos, realizada por el grupo de informatización de la UCI e implementada en lenguaje de programación C Sharp en la plataforma:”.NET”. Utiliza la herramienta de desarrollo Visual Studio 2005. Esta aplicación permite procesar una imagen de una determinada persona que ha sido fotografiada con una cámara digital, estandarizándola a formato y resolución determinados y finalmente guardándola en repositorio, luego de haber sido descargada hacia la computadora donde se encuentra el sistema. Está desarrollado con herramientas de licencia privativa, no es multiplataforma y no existe una ingeniería de software que haya guiado y documentado su proceso de desarrollo. Su interfaz no es agradable al usuario.

1.3 Arquitectura de software

La arquitectura de software (AS) tiene sus raíces en 1968, cuando el ilustre profesor de la Universidad Tecnológica de Eindhoven y “Premio Turing 1972”: Edsger Dijkstra, propuso que se estableciera una estructuración correcta de los sistemas de software antes de lanzarse a programar, escribiendo código de cualquier manera, es decir, sin una estructura correctamente diseñada.

No es novedad que ninguna definición de la arquitectura de software (AS) es respaldada unánimemente por la totalidad de los arquitectos. El número de definiciones circulantes alcanza un orden de tres dígitos casi llegando a cuatro.

Una AS, también denominada arquitectura lógica, consiste en un conjunto de patrones y

abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La arquitectura de software establece los fundamentos para que analistas, diseñadores, programadores, etc., trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Define de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea. En fin, es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

Con el paso de los años esta importante disciplina continúa teniendo auge, pero no es hasta el año 2000 que la IEEE conforma la definición oficial de “Arquitectura del Software”, reflejada en su documento ANSI/IEEE Std 1471-2000 y que plantea:

“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”. (2)

Existen otras definiciones importantes de AS, entre las que podemos encontrar la planteada por Thomas G. Lane:

“Arquitectura de software es el estudio de la estructura a gran escala y el rendimiento de los sistemas de software. Aspectos importantes de la arquitectura de un sistema incluye la división de funciones entre los módulos del sistema, los medios de comunicación entre módulos, y la representación de la información compartida.”(3)

Principales corrientes arquitectónicas

Con el paso de los años los principales exponentes del estudio de la arquitectura de software a nivel internacional fueron encontrando seguidores de sus ideas y a la vez se fueron surgiendo numerosas y diversas corrientes arquitectónicas. A continuación se explican, de

manera general, algunas de las más importantes:

- *Arquitectura estructural, basada en un modelo estático de estilos, ADLs y vistas*

  Constituye la corriente fundacional y clásica de la disciplina. Los representantes de esta corriente son todos académicos, mayormente de la Universidad de Carnegie Mellon en Pittsburg: Mary Shaw, Paul Clements, David Garlan, Robet Allen, Gregory Abowd y John Ockerbloom. En toda la corriente, el diseño arquitectónico no sólo es el de más alto nivel de abstracción, sino que además no tiene por que coincidir con la configuración explícita de las aplicaciones; rara vez se encontrarán referencias a lenguajes de programación o piezas de código.

- *Arquitectura como una Etapa de ingeniería y Diseño Orientada a Objetos*

  Esta corriente es una alternativa de la descrita anteriormente. Es el modelo de James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman y otros. Es una corriente estrechamente ligada mundo de UML y Rational. En esa postura, la arquitectura se restringe a las fases iniciales y preliminares del proceso y concierne a los niveles más altos de abstracción. Importa más la abundancia, el detalle de diagramas y técnicas disponibles, que la simplicidad de de la visión de conjunto. Las definiciones revelan que la AS concierne a decisiones sobre organización, selección de elementos estructurales, comportamiento, composición y estilo arquitectónico susceptible de ser descriptas a través de las cinco vistas clásicas del modelo 4+1 de Kruchten.

- *Arquitectura Procesual*

  Esta corriente surge desde comienzos de este siglo, con sede en el SEI (**S**oftware **E**ngineering **I**nstitute) y la participación de algunos arquitectos de Carnegie Mellon de la primera generación, unido a nombres nuevos de la segunda como son: Rick Kazman, Len Bass, Paul Clements y Felix Bachmann, entre otros. Intenta establecer modelos de ciclo de vida y técnicas de diseño, análisis, selección de alternativas, validación,

comparación, estimación de calidad y justificación económica específica para la arquitectura de software.

- *Arquitectura Basada en Patrones*

Se basa en la redefinición de los estilos como patrones POSA (Patrón Orientado a la Arquitectura de Software), el diseño consiste en identificar y articular patrones preexistentes, que se definen en forma parecida a los estilos de arquitectura.

1.3.1 Necesidad del uso de una arquitectura

La arquitectura de software es de suma necesidad en un proceso de desarrollo de software porque:

- Ayuda a comprender el Sistema.
- Permite organizar el desarrollo del software.
- Fomenta la reutilización de código.
- Hace evolucionar el sistema.

1.3.2 El rol de arquitecto del software.

El arquitecto del software posee uno de los roles más importantes dentro del equipo de desarrollo, debido a la gran cantidad de responsabilidades y actividades que recaen sobre él en el “Proceso de Desarrollo del Software”. Es el encargado de seleccionar la arquitectura más adecuada para que el sistema responda a las necesidades del usuario de forma óptima, a los requisitos funcionales y no funcionales; además de ser capaz de lograr resultados esperados, atendiendo a restricciones dadas y en el tiempo establecido. Está presente en todo el ciclo de vida del software y debe mantener el mando en los flujos de trabajo de requerimiento, análisis - diseño e implementación; de cada iteración que se realice en el proyecto. Realiza un modelo entendible y manejable de la arquitectura del software, resumiendo la complejidad del producto y resaltando los detalles más importantes del mismo.

Debe tomar decisiones críticas y cruciales, que muchas veces deciden la dirección que tomará el proyecto respecto al funcionamiento y mantenimiento de la aplicación, apoyándose para ello en el equipo de trabajo y las restricciones más importantes. Finalmente, debe detallar correctamente los cambios realizados con el objetivo de facilitar el entendimiento y comprensión de las demás personas que interactúen con su trabajo. Crea vistas arquitectónicas del sistema para detallar a los clientes u otros interesados y la arquitectura del software en diferentes etapas del desarrollo del sistema.

A continuación, se exponen las principales responsabilidades de un arquitecto de software:

- El arquitecto posee la responsabilidad técnica del sistema y debe ser capaz de desarrollar e implementar todas las funcionalidades de la aplicación económicamente.

- El arquitecto debe trabajar en conjunto con todos los implicados en el proyecto para obtener experiencia de cada uno.

- Debe ser flexible en caso de existir algún cambio en la aplicación que influya en la arquitectura.

- El arquitecto obtiene una arquitectura sólida después de haber pasado por varias iteraciones del producto, ya en la fase de elaboración se debe tener la arquitectura base y estable, porque este es precisamente el hito de esta fase de RUP.

- En caso de tener un sistema complejo esta metodología aconseja que exista un equipo de arquitectos para dividir las tareas porque la arquitectura en este caso puede llegar a ser demasiado compleja.

- El arquitecto debe tener conocimiento y experiencia en el desarrollo de sistemas, porque él es quien va a explicarle la arquitectura a cada uno de los miembros del equipo de trabajo.

- El arquitecto debe tener presente además que la arquitectura está dirigida por los casos de uso.
- Para el arquitecto la línea base de la arquitectura es el documento más importante en su trabajo.
- El arquitecto es responsable de seleccionar los estándares de codificación para el desarrollo del sistema.

1.3.3 Estilos arquitectónicos

Dentro de la arquitectura de software, el estudio de los estilos arquitectónicos es de suma importancia, puesto a que muestran las posibles formas en que nuestro sistema puede estar estructurado y a su vez son previos a la elección de las herramientas y plataforma a utilizar para el desarrollo del mismo. Los estilos arquitectónicos son arquitecturas de software comunes, marcos de referencias arquitectónicas, formas comunes o clases de sistemas, que están compuestos por una serie de elementos y restricciones arquitectónicas. Desde hace algún tiempo se conocen como las 4C de la arquitectura de software:

- Componentes.
- Conectores.
- Configuraciones.
- Restricciones. (Constrains).

La clave del trabajo arquitectónico está en la correcta elección del estilo arquitectónico, por lo que cuando se va a establecer una arquitectura para el sistema, es necesario tener en cuenta los estilos que existen (actualmente aproximadamente 20 estilos abstractos), para así poder buscar el que más se ajusta a nuestro problema, teniendo en cuenta que el sistema también puede tener varios de estos combinados que igualmente dan solución al mismo. Dentro de cada estilo se pueden aplicar gran cantidad de patrones y las relaciones entre ellos.

Algunos de los principales estilos arquitectónicos que se usan en la actualidad están divididos por: “Clases de Estilos”, las que engloban una serie de estilos arquitectónicos específicos:

Estilos de Flujo de datos: Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos.

- Tuberías y Filtros: Una tubería es una popular arquitectura que conecta componentes computacionales a través de conectores, de modo que el procesamiento de datos se ejecuta como un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Este estilo se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada.

Estilos de Llamada y Retorno: Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala.

- Arquitectura en Capas: En este estilo arquitectónico cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto del sistema y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimizan las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema.

- Arquitectura Orientada a Objetos: Los componentes de este estilo son los objetos, o más bien instancias de los tipos de datos abstracto, 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.

- Arquitectura basada en Componentes: Los sistemas de software basados en componentes se sustentan en principios definidos por una ingeniería de software específica. Los componentes son las unidades de modelado, diseño e implementación, las interfaces están separadas de las implementaciones, y conjuntamente con sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución. En este estilo las interfaces son el elemento básico de conexión entre los componentes, por lo que se debe conocer cuáles ofrece cada uno de ellos y las que necesita para su operatividad.

Estilos Peer To Peer: Esta familia se conoce también como componentes independientes, enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes.

- Arquitecturas Basadas en Eventos: Estas se han llamado también arquitectura de invocación implícita, estas se vinculan con sistemas basados publicación-suscripción. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa un componente puede anunciar mediante difusión uno o más eventos.

- Arquitecturas Orientadas a Servicios (SOA): Esta construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces, es una relación entre servicios y consumidores de servicios, ambos lo suficientemente amplios como para representar una función de negocio completa.

- Arquitecturas Basadas en Recursos: Define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es nada menos

que la WWW, donde los URLs identifican los recursos y HTTP es el protocolo de acceso. El argumento central es que HTTP mismo, con su conjunto mínimo de métodos y su semántica simplísima, es suficientemente general para modelar cualquier dominio de aplicación.

De modo general existen muchos estilos arquitectónicos de forma tal que cada uno resuelve un problema específico y estén orientados más bien a las distintas formas en que se pueda estructurar el sistema que se está desarrollando. Durante el diseño y la programación estos estilos serán refinados mediante técnicas de patrones, refactorización, etc. Los estilos arquitectónicos que más se usan actualmente en el desarrollo de software a nivel mundial son: arquitectura orientada a servicios, la arquitectura basada en componentes, la arquitectura orientada a objetos y la arquitectura en capas; además de otros tipos de arquitecturas derivadas de los estilos existentes, es decir, de la combinación de varios de ellos, como son: la C2, GenVoca y REST.

1.3.4 Patrones arquitectónicos

Cuando se va a hablar de “patrones arquitectónicos” no se puede dejar de mencionar a Christopher Alexander, quien fuera su precursor y en 1977 escribiera una serie de artículos referentes a este tema, lo que más bien basado en la arquitectura real (edificios, apartamentos, etc.), y no es hasta 1990 que se comienza a aplicar lo planteado por Alexander a la Arquitectura del Software.

Un patrón arquitectónico es una solución probada que se puede aplicar con éxito a un determinado tipo de problema que aparece repetidamente en algún campo. El establecimiento de estos patrones comunes es lo que posibilita el aprovechamiento de la experiencia acumulada en el diseño de aplicaciones. Un patrón codifica conocimientos específicos acumulados por la experiencia en un dominio, por tanto podemos decir que, un sistema bien estructurado está lleno de patrones. Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente y luego describe el núcleo de la solución a ese problema, de tal manera que puede usarse esa misma solución tantas veces como ocurra ese problema.

“Son los que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema”.(4)

Los elementos de un patrón son:

1. Nombre
   a. Define un vocabulario de diseño.
   b. Facilita la abstracción.

2. Problema
   a. Describe cuando aplicar el patrón.
   b. Conjunto de fuerzas: objetivas y restricciones.
   c. Pre-requisitos.

3. Solución
   a. Elementos que constituyen el diseño (plantilla).
   b. Forma canónica para resolver fuerzas.

4. Consecuencias
   a. Resultados.
   b. Extensiones.
   c. Consensos.

Al contrario de los estilos arquitectónicos, los patrones son muchos y muy variados, por lo que se hace muy difícil revisarlos todos a la hora de hacer una determinada aplicación, por dicha razón, se recomienda el uso de los patrones que estén predeterminados en cada uno de los estilos que se seleccionen para la arquitectura.

1.4 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado fue originalmente creado por Rational Software a pesar de que actualmente sea atendido por el Grupo de Administración de Objetos (OMG). Es un lenguaje que permite visualizar, especificar, construir y documentar, modelos de sistemas de software, incluyendo su estructura y diseño, que capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Es un lenguaje de propósito general para el modelado visual y orientado a objetos, que permite una abstracción del sistema y sus componentes, al mismo tiempo posibilita establecer una serie de requerimientos y estructuras necesarias para plasmar en un sistema de software previo al proceso intensivo de escribir código. Permite construir modelos por ingeniería inversa a partir de programas existentes.

UML es un conjunto de herramientas, que permite modelar (analizar y diseñar) sistemas orientados a objetos.(5)

Este lenguaje de modelado permite:

- Modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc). Puede conectarse con lenguajes de programación (ingeniería directa e inversa).
- Cubrir todas las vistas necesarias para desarrollar y luego desplegar los sistemas, dado a que es un lenguaje muy expresivo.
- Cubrir las cuestiones relacionadas con el tamaño, propias de los sistemas complejos y críticos.
- Equilibrar expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.5 Metodologías de Desarrollo de Software

Para la correcta fabricación de un producto software es necesario modelar el proceso a través de una metodología, la cual será la encargada de guiar el “Proceso Unificado de Desarrollo de Software” sobre el cual se apoyará el equipo de desarrollo. En la actualidad existen muchas de ellas entre las que se encuentran: XP y RUP, las cuales fueron seleccionadas para ser estudiadas con el fin de escoger una para nuestra propuesta.

1.5.2 Metodología XP

La metodología XP (Programación Extrema), es una de las variantes de las metodologías ágiles con más aceptación en la comunidad internacional de desarrollo. Su creador, Kent Beck la comenzó a gestar junto con Ward Cunningham en 1990 y tomó su forma final en 1996. Los fundamentos de la XP según Beck son: mejorar la comunicación, buscar la simplicidad y retroalimentación. Una de las herramientas más importantes de la XP es el desarrollo orientado a pruebas, que utiliza las pruebas unitarias como eje de todo desarrollo. Siendo una directriz de esta herramienta no escribir código para el que no tengamos previamente una prueba unitaria, no sólo mejoramos la seguridad del desarrollo, sino que también documentamos el código de producción con el código de pruebas.

Entre sus principales **Valores** se encuentran:

- Comunicación: Crear software requiere de sistemas comunicados.
- Simplicidad: Empezar con lo necesario y requerido y trabajar desde ahí.
- Retroalimentación: Del sistema, del cliente, y del equipo.
- Valentía: Programa para hoy y no para mañana.
- Respeto: El equipo debe trabajar como uno, sin hacer decisiones repentinas.

Sus **Actividades** fundamentales son:

- Codificación: La parte más importante de XP.
- Pruebas: Nunca se puede estar seguro de algo hasta haberlo probado.
- Escuchar: Escuchar los requisitos del cliente acerca del sistema a crear.
- Diseño: Crear una estructura del diseño para evitar problemas.

Esta metodología se basa en ir construyendo el software y hacerle pruebas al mismo tiempo, de modo que el cliente (que se convierte en parte del equipo) dé su opinión y de esa manera haya una retroalimentación que permita construir el software como él desee, XP está diseñada para entregar al cliente el software que necesita, cuando lo necesita. De todas las metodologías ágiles, ésta es la que ha recibido más atención. Programación Extrema se usa actualmente para la creación y desarrollo rápido y práctico de software.

1.5.3 RUP (Proceso Unificado de Rational)

"Es un proceso de ingeniería de software orientado a objetos. Consiste en un conjunto de actividades necesarias para transformar los requerimientos del usuario en el sistema de software. Está especializado para diversos tipos de software de sistemas, diversas áreas de aplicación, diferentes tipos de organizaciones y diferentes tamaños de proyectos".(6)

La metodología RUP divide el trabajo del equipo de desarrollo de software en 4 fases fundamentales:

- Inicio: En esta etapa se determina la visión del proyecto.
- Elaboración: En esta etapa se determina la arquitectura base.
- Construcción: En esta etapa se obtiene la capacidad operacional inicial.
- Transición: En esta etapa se obtiene el *release* del proyecto.

Cada una de estas etapas se desarrolla en iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

El Proceso Unificado de Rational (RUP), está compuesto por tres elementos fundamentales:

- Actividades: Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: Son las personas o entes involucrados en cada proceso
- Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Dentro de cada iteración ocurrida, en cada fase se llevan a cabo nueve flujos de trabajo, seis (6) son llamados “Flujos de Trabajo Ingenieriles” y los restantes tres (3) son los conocidos “Flujos de Apoyo”.

Flujos de Trabajo Ingenieriles:

Modelado del negocio: Este flujo identifica los procesos de negocio, los que estarán sujetos a automatización y quiénes intervienen en los mismos.

Requerimientos: Se identifican las restricciones que se imponen y lo que el sistema debe hacer.

Análisis y Diseño: Describe como el programa será realizado y define como será programado.

Implementación: Define como estarán los nodos ubicados y la ubicación de los objetos y clases en paquetes.

Prueba: Se localizan los defectos del software.

Instalación: Se entrega una versión operacional.

Flujos de trabajo de apoyo:

Administración de proyecto: Encargado de organizar el trabajo y de que se termine el proyecto en el tiempo previsto.

Administración de configuración y cambio: Describe el uso y actualización concurrente de los elementos, control de versiones entre otras actividades.

Ambiente: Describe los procesos y herramientas que soportarán al equipo de trabajo del proyecto.

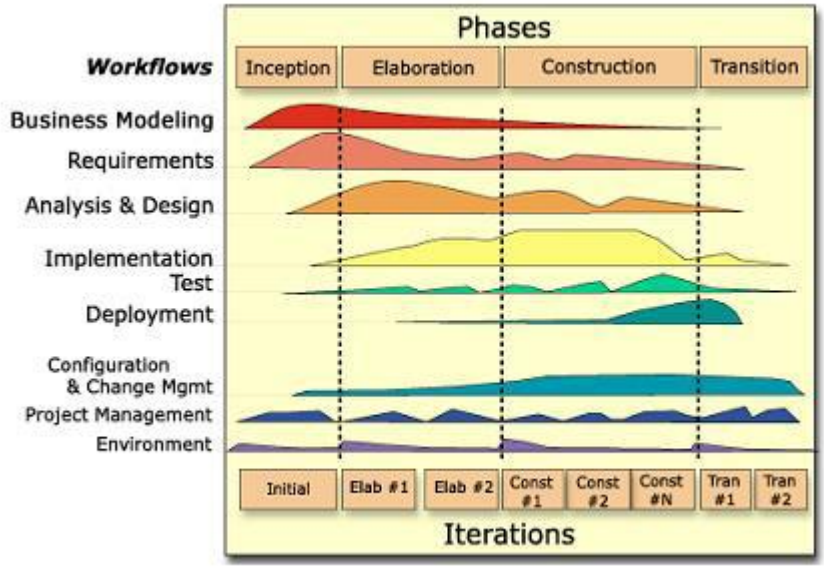

Figura 1 Fases e iteraciones de la Metodología RUP

1.6 Herramientas de Modelado

1.6.1 Visual Paradigm

El Visual Paradigm es una suite completa de herramientas CASE que da soporte al modelado visual con UML 6.0 ofreciendo distintas perspectivas del sistema. Independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final. Posee

entre sus principales características la posibilidad de crear un conjunto bastante amplio de artefactos, utilizados con mucha frecuencia durante la confección de un Software. Sus componentes se encuentran relacionados, por lo que se hace muy fácil la creación de cualquier tipo de diagrama, ya que cada componente utilizado en el diagrama que se esté creando, sugiere nuevos posibles componentes a utilizar, por lo que ya no es necesario localizarlos en la barra donde pueden aparecer un número grande de componentes. Brinda un número considerable de estereotipos a utilizar, lo que permite un mayor entendimiento de los diagramas. Posibilita generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así cómo obtener diagramas a partir del código. Intercambia información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose. Presenta disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java. Permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable, y facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información.

1.6.2 Rational Rose

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML, que permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común. Los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. Por otra parte Rational Rose permite la generación de código a partir de un diseño en UML en lenguajes como C++, VisualBasic, Java, Ada, genera IDL's para aplicaciones CORBA. Soporta realizar ingeniería inversa por lo que se puede obtener un diseño a partir del código de un programa. Está disponible en la plataforma Windows: en Microsoft Windows NT 4.0, Windows 95, Windows 98.

1.7 Lenguajes de Programación Web

En la actualidad Internet es la vía de comunicación más usada mundialmente, por el sin número de ventajas y funcionalidades que ofrece a los usuarios a partir de las aplicaciones (principalmente web) que soporta.

Existen un conjunto de lenguajes de programación que proporcionan a las aplicaciones web gran interactividad, ya sea del lado del cliente o del servidor. Para software libre los más usados del lado del cliente (encargados de visualizar y comprobar la información en el navegador y los formularios) se encuentran HTML, Java Script y CSS; mientras que del lado del servidor (procesan la lógica del negocio) podemos encontrar PHP, JSP, y Perl.

1.7.2 Lenguajes de programación web del lado del cliente

1.7.2.1 Html

EL lenguaje de Marcas Hipertextuales (Html) fue creado por Tim Berners-Lee en el año 1990 principalmente para mostrar información, animaciones en forma de hipertexto. Permite utilizar estilos en formato CSS (hojas de estilos en cascada) en las páginas y se pueden actualizar los contenidos con mayor facilidad. Actualmente es usado por todos los navegadores para mostrar la información final. Es muy sencillo y permite describir texto presentado de forma estructurada y agradable, con enlaces que permitan navegar en otros documentos o fuentes de información relacionadas y con inserciones multimedia. Se basa en especificar la estructura lógica del contenido en el texto y dejar que luego la presentación final de dicho hipertexto se realice por un programa especializado.

1.7.2.2 Java Script

Es un lenguaje interpretado lo que significa que no necesita ser compilado para obtener el resultado. Es basado en prototipo donde las nuevas clases se generan clonando la clase base y extendiendo su funcionalidad. El tiempo de respuesta es sumamente pequeño y es interpretado por el navegador. Presenta como problema que el código es visible y puede ser leído por cualquiera aún cuando este protegido con las leyes del derecho de autor.

Es un lenguaje débilmente tipiado, basado en objetos y guiado por eventos lo que permite el dinamismo de las páginas que incluyan este tipo de código, útil para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet. El código script tiene capacidades limitadas, por razones de seguridad, por lo que es necesario usarlo conjuntamente con HTML.

1.7.2.3 CSS

Las Hojas de Estilo en Cascada, son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (por extensión en XHTML). El W3C es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la *estructura* de un documento de su *presentación*.

CSS proporciona tres caminos diferentes para aplicar las reglas de estilo a una página Web:

1. **Una hoja de estilo externa**, que es una hoja de estilo que está almacenada en un archivo diferente al archivo donde se almacena el código HTML de la página Web. Esta es la manera de programar más potente, porque separa completamente las reglas de formateo para la página HTML de la estructura básica de la página.

2. **Una hoja de estilo interna**, que es una hoja de estilo que está incrustada dentro de un documento HTML. (Va a la derecha dentro del elemento <head>). De esta manera se obtiene el beneficio de separar la información del estilo, del código HTML propiamente dicho. Se puede optar por copiar la hoja de estilo incrustada de una página a otra, (esta posibilidad es difícil de ejecutar si se desea para guardar las copias sincronizadas). En general, la única vez que se usa una hoja de estilo interna, es cuando se quiere proporcionar alguna característica a una página Web en un simple fichero, por ejemplo, si se está enviando algo a la página web.

3. **Un estilo en línea**, que es un método para insertar el lenguaje de estilo de página, directamente, dentro de una etiqueta HTML. Esta manera de proceder no es excesivamente adecuada. El incrustar la descripción del formateo dentro del documento de la página Web, a nivel de código se convierte en una tarea larga, tediosa y poco elegante de resolver el problema de la programación de la página. Este modo de trabajo se podría usar de manera ocasional si se pretende aplicar un formateo con prisa, al vuelo. No es todo lo claro, o estructurado, que debería ser, pero funciona.

1.7.2 Lenguajes de programación web del lado del servidor

1.7.2.1 PHP

El Pre-Procesador de Hipertextos (PHP), es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Puede ser utilizado en cualquiera de los principales sistemas operativos: Linux, en variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, etc; por lo que se puede decir que es un lenguaje multiplataforma. Es soportado por la mayoría de los servidores Web de hoy en día, ejemplo: Apache, (IIS), Personal Web Server y muchos otros. Como es un lenguaje que se ejecuta en el servidor, no necesita que un navegador en particular lo soporte.

Es un lenguaje de programación del lado del servidor, orientado a objetos, muy sencillo y fácil de aprender, con sintaxis similar a C y a PERL. Cumple con las políticas de software libre por lo que se puede obtener en la web y su código esta disponible bajo la licencia GPL. Brinda excelente soporte de acceso a base de datos. Su código es mucho más legible que el de PERL. Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envío de correo, XML, creación de PDF, etc). Actualmente está siendo utilizado con éxito en varios millones de sitios web.

1.7.2.2 Perl

Es un lenguaje interpretado optimizado para la lectura y extracción de información de archivos de texto, generando reportes basados en la información proporcionada por ellos. Es un lenguaje bastante utilizado para muchos sistemas manipuladores de tareas como lenguaje de contenido dinámico. Este lenguaje tiene una orientación más práctica (facilidad de uso y eficiencia) que estética (elegancia). Es muy útil para muchas tareas de administración de sistemas y muy usado para manejo y gestión de procesos (estado de procesos, conteo y extracción de parámetros característicos, etc).

Inicialmente solo corría sobre la plataforma Unix, pero en la actualidad es un lenguaje multiplataforma. Soporta diferentes paradigmas de programación, tanto estructurada como

orientada a objetos y es fácil de utilizar. Es un lenguaje extensible, pues permite hacer llamadas a múltiples programas desarrollados en otros lenguajes de programación.

1.7.2.3 JSP

JSP es un acrónimo de Servidor de Páginas de Java. Fue creado por la compañía Sun Microsystems y utiliza como lenguaje de programación JAVA con paradigma orientado a objeto. Es multiplataforma lo que le da gran flexibilidad y seguridad. Es un lenguaje desarrollado para aplicaciones grandes. Presenta una estructura que permite separar la lógica de presentación en páginas JSP y el código o lógica del negocio en clases JAVA por lo que se considera un lenguaje avanzado para las páginas dinámicas en el servidor lo que permite mayor seguridad de los datos. Posee un grupo de marcos de trabajo que facilitan el trabajo. Las aplicaciones se pueden ejecutar en cualquier ambiente, independientemente del sistema operativo que se utilice, puesto a que es soportado por una máquina virtual que es la encargada de compilar todo el código.

1.8 Entornos Integrados de Desarrollo (IDE)

Un entorno de desarrollo integrado es un programa compuesto por un conjunto de herramientas para ser utilizadas por un programador. Puede dedicarse exclusivamente a un sólo lenguaje de programación o bien a varios de ellos.

“*Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes”.(7)*

1.8.1 Eclipse

Metafóricamente, Eclipse es como una tienda de software libre para herreros, donde no solamente se hacen productos, sino que además se hacen las herramientas para hacer los productos. Es un proyecto de desarrollo de software open-source (código abierto), que está dividido en tres partes: el proyecto Eclipse Proyect, Eclipse Tools, y Eclipse Technology Proyect.

El Eclipse Proyect está subdividido a su vez en tres sub-proyectos que son la propia plataforma, JDT (Java Development Tool) y PDE (Plugin Development Enviroment). Mediante este IDE se pueden crear diversas aplicaciones como pueden ser, sitios web, programas Java, PHP, C++ y Enterprise Java Beans. Su principal aplicación es JDT que es la herramienta para crear aplicaciones en Java, además como su plataforma esta construida en base a módulos esto nos permite que otras aplicaciones pueden ser integradas a eclipse en forma de módulos y los mismos son reconocidos automáticamente por el IDE al iniciarse. Su interfaz de usuario esta compuesta de un conjunto de vistas, editores y perspectivas. Los editores permiten crear, modificar y salvar objetos, las vistas proveen información acerca de los objetos con los que se están trabajando y las perspectivas proveen distintas formas de organización del proyecto. Eclipse es administrado y dirigido por un consorcio de compañías de desarrollo de software con un interés comercial en promover Eclipse como plataforma compartida para herramientas de desarrollo de software.

1.8.2 ZendStudio

ZendStudio divide sus funcionalidades en dos partes: la del cliente y la del servidor las cuales se instalan por separado. La parte del cliente contiene la interfaz de edición y la ayuda y además permite hacer depuraciones simples de scripts. Para disfrutar de toda la potencia de la herramienta de depuración se hace necesaria la parte del servidor que instala Apache y el modulo PHP o los configura para trabajar juntos en depuración en caso de que ya estén instalados.

El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. El editor, la parte del programa que permite escribir los scripts, es muy útil para la programación PHP. A la hora de escribir brinda gran ayuda puesto a que permite:

- Editar varios archivos y moverse fácilmente entre ellos.

- Marcar a qué elementos corresponden los inicios y cierres de las etiquetas, paréntesis o llaves.

- Moverse al principio o al final de una función.
- Identificar automáticamente el código.

En la interfaz encontramos un explorador de archivos, una ventana de depuración, otra para mostrar el código de las páginas y los menús. Además coloca puntos de parada en los scripts y realiza las acciones típicas de depuración.

Dispone de una herramienta muy interesante de depuración que nos permite ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión.

1.9 Frameworks de Desarrollo

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, este puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Fuera de las aplicaciones en la informática, un framework puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo.

1.9.1 Symfony

Es un completo framework desarrollado completamente con PHP5 diseñado para optimizar el desarrollo de las aplicaciones Web, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja, automatiza las tareas más comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Compatible con la mayoría de los gestores de base de datos (MySQL,

PostgreSQL, Oracle y SQL Server de Microsoft) y con la mayoría de las plataformas, fácil de instalar, sigue la mayoría de las mejores practicas y patrones de diseño para la Web, sencillo de usar y suficientemente flexible para adaptarse a los casos mas complejos. Permite su integración con librerías desarrolladas por terceros, con código fácil de entender y leer.

Utilizado fundamentalmente para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

1.9.2 Cake PHP

La fuerza de CakePHP no sólo reside en su versatilidad y potencia, sino también en que la comunidad de programadores lo ha acogido con muchas ganas, y se ha creado una comunidad muy activa, lo que propicia que se desarrollen más módulos y se tenga código muy competitivo y eficaz. ¿Por qué está acogida? Entre otras razones, está la de que su licencia tipo BSD 100% libre, mucho menos restrictiva que la GPL. Por otra parte, la compatibilidad con las versiones 4 y 5 de PHP, la disponibilidad de plantillas flexibles, la instalación cómoda y casi sin requisitos en servidor (sólo necesita tener el mod_rewrite activado), junto con otras características como son el llevar un módulo de validación incorporada para el uso de sesiones, listas de control de acceso, unido a otros componentes de seguridad y sesión, helpers en Vistas para AJAX, Javascript, Formularios HTML. URLs amigables, puedes tener la web entera cacheada, para que su acceso sea más rápido.

En resumidas cuentas, Cake PHP es un framework tipo MVC con scaffolding y CRUD.

1.10 Servidores web

Los servidores web se crearon con el fin de gestionar las peticiones del usuario y devolverle a los mismos recursos a través de un protocolo de comunicación (HTTP: “Protocolo de Transferencia del Hipertexto, el protocolo que sirve para incursionar en los sitios de WWW en Internet).

El servidor Web es un programa que corre sobre el servidor que escucha las peticiones

HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. El servidor Web va a ser fundamental en el desarrollo de las aplicaciones del lado del servidor, que vayamos a construir, ya que se ejecutarán en él.

“Un servidor web, este término podría referirse a la máquina que almacena y maneja los sitios web, y en este sentido es utilizada por las compañías que ofrecen hospedaje. Alternativamente, el servidor web podría referirse al software, como el servidor de http de Apache, que funciona en la máquina y maneja la entrega de los componentes de las páginas web como respuesta a peticiones de los navegadores de los clientes”.(8)

1.10.1 Apache

Es el servidor web más usado hoy en día en el mundo, encontrándose por encima de sus competidores, tanto gratuito, como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma (multiplataforma). Tiene capacidad para servir páginas tanto de contenido estático (para lo que serviría sencillamente un ordenador 486), como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información. Apache es uno de los servidores web más potentes del mercado, ofreciendo una perfecta combinación entre estabilidad y sencillez.

Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos producto de su diseño modular. Este diseño permite a los administradores de sitios web elegir que características van a ser incluidas en el servidor, seleccionando los módulos que serán cargados en los procesos de compilación o ejecución.

1.10.2 Servidor de Información en Internet (IIS)

Es el servidor Web de Microsoft que corre sobre plataformas Windows. De hecho, el IIS viene integrado con Windows NT 4.0. Dado que el IIS está tan íntimamente integrado con el

sistema operativo, es relativamente fácil de administrar. Sin embargo, actualmente el IIS está disponible sólo para el uso en la plataforma Windows NT, mientras que los servidores Web de Netscape corren en la mayoría de las plataformas, incluyendo Windows NT, OS/2 y UNIX.

1.11 Servicios Web

Un servicio web es una colección de protocolos y estándares que sirven para intercambiar datos entre distintas aplicaciones desarrolladas en lenguajes de programación diferentes y ejecutada sobre cualquier plataforma. Estas aplicaciones o tecnologías intercambian datos entre sí, utilizando estándares de comunicación, con el objetivo de ofrecer servicios. Describe una forma estandarizada de aplicaciones usando XML, SOAP, WSDL y UDDI, sobre el protocolo Internet. XML es usado para etiquetar los datos, SOAP es el encargado de la transferencia de la información, WSDL describe los servicios disponibles y UDDI es el responsable de que los servicios WEB disponibles sean listados.

Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios los solicitan llamando a dichos procedimientos a través de la Web. Los servicios web permiten a las organizaciones el intercambio de datos, incluso sin tener presente los sistemas que se encuentran detrás del cortafuegos.

“Es un sistema software diseñado para soportar la interoperabilidad máquina - máquina a través de una red. Este tiene una interfaz descripta en un formato que puede ser procesado por una máquina (específicamente WSDL). Otros sistemas interactúan con el servicio web utilizando mensajes SOAP los cuales se encuentran establecidos previamente”.(9)

De modo general presentan las siguientes ventajas:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los Servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Al apoyarse en HTTP, los Servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta y se garantiza la plena interoperabilidad entre aplicaciones.

1.12 Protocolo de Acceso Simple a Objetos (SOAP)

SOAP (Protocolo de Acceso Simple a Objetos), es un protocolo que define la estrategia para que dos objetos de diferentes procesos puedan comunicarse por medio de intercambio de datos XML. Es uno de los protocolos utilizados en los servicios web puesto a que usa el código fuente en XML, siendo esto una gran ventaja, debido a que facilita su lectura por parte de cualquier persona y a la vez genera el inconveniente de que los mensajes resultantes son más largos. El intercambio de mensajes se realiza mediante tecnología de componentes. El término Objeto en el nombre significa que se adhiere al paradigma de la programación orientada a objetos. Se encuentra en el núcleo de los Servicios Web. Es el primero de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo, incluso por aquellas que raramente cooperan entre si, tales como: Microsoft, IBM, SUN Microsystems, SAP y Ariba.

Algunas de las Ventajas de SOAP son:

- *No está asociado con ningún lenguaje:* los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.

- *No se encuentra fuertemente asociado a ningún protocolo de transporte:* La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

- *No está atado a ninguna infraestructura de objeto distribuido*: La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.

- *Aprovecha los estándares existentes en la industria:* Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.

- *Permite la interoperabilidad entre múltiples entornos:* SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dichos estándares, pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.

1.13 Controladores de Versiones

1.13.1 Subversion (SVN)

El SVN es un sistema de software libre y de código fuente abierto que maneja ficheros y directorios a través del tiempo. Existe un árbol de ficheros en un *repositorio* central que es como un servidor de ficheros ordinario que recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus 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. Brinda la

capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas puestos de trabajo. Es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros.

Es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular Sistema de Versiones Concurrentes (CVS) que posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. Se sigue la historia de los archivos y directorios a través de copias y renombrados, las modificaciones (incluyendo cambios a varios archivos) son atómicas.

Permite selectivamente el bloqueo de archivos, se usa en archivos binarios que al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez y es unos de los controladores más utilizados en proyectos de software libre.

1.13.2 Sistema de Versiones Concurrentes (CVS)

El CVS, es una aplicación informática que implementa un sistema de control de versiones, mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren. CVS se ha hecho popular en el mundo del software libre.

CVS utiliza una arquitectura cliente-servidor: un servidor guarda las versiones actuales del proyecto y su historial. Los clientes se conectan al servidor para sacar una copia completa del proyecto. Esto se hace para que eventualmente puedan trabajar con esa copia y más tarde ingresar sus cambios.

Varios clientes pueden sacar copias del proyecto al mismo tiempo, los clientes pueden sacar y comparar versiones sin necesidad de teclear una contraseña; solamente el ingreso de cambios requiere una contraseña en estos casos.

CVS también puede mantener distintas ramas de un proyecto. Por ejemplo, una versión difundida de un proyecto de programa puede formar una rama y ser utilizada para corregir errores. Todo esto se puede llevar a cabo mientras la versión que se encuentra actualmente en desarrollo y posee cambios mayores con nuevas características se encuentre en otra línea formando otra rama separada.

1.14 Lenguaje de Descripción de Servicio Web (WSDL)

Lenguaje basado en XML para describir servicios web. Permite describir la interfaz pública de los mismos; eso significa que detalla los protocolos y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Principales ventajas del WSDL

- Facilita escribir y mantener servicios mediante una aproximación estructurada para definir interfaces web.
- Facilita el acceso a esos servicios web reduciendo el código que hay que escribir para hacer un cliente.
- Facilita hacer cambios para ampliar los servicios, reduciendo la posibilidad de que los clientes dejen de funcionar al llamar a esos servicios.

1.15 Patrones de Diseño

Un patrón de diseño es una solución estándar para un problema común de programación, en forma de técnica, que permite flexibilizar el código haciéndolo satisfacer ciertos criterios. Además puede ser interpretado como un proyecto o estructura de implementación que logra

una finalidad determinada o un lenguaje de programación de alto nivel. Es una manera más práctica de describir ciertos aspectos de la organización de un programa.

Con la aplicación de los patrones de diseño se pretende:

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

Los patrones de diseño se utilizan cuando se percibe un problema en el proyecto, algo que debería resultar sencillo no lo es. No es obligatorio utilizarlos siempre, solo en el caso de tener el mismo problema o similar al que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones de diseño puede ser un error garrafal.

1.16 Conclusiones

De modo general en el presente capítulo se realizó un estudio de sistemas informáticos, en cuba y el mundo, que trabajen con imágenes, para comprender su estructura y forma de gestionar sus procesos. Unido a todo lo anteriormente dicho, se puede agregar que se estudiaron un grupo notable de herramientas, lenguajes, metodologías, estilos arquitectónicos y patrones, con el objetivo de obtener una visión de los posibles a utilizar en el diseño y construcción del sistema propuesto.

Capítulo 2 Definición de la Arquitectura

2.1 Introducción

En el presente capítulo se hace una descripción profunda y general del sistema propuesto. Se justifican y detallan los lenguajes, herramientas, metodologías y tecnologías serán utilizados para la elaboración del presente trabajo. Se especifican los patrones arquitectónicos y de diseño que permitirán modelar el futuro sistema informático.

2.2 Línea Base de la Arquitectura

La Línea Base de la Arquitectura es un esqueleto del sistema con pocos músculos de software, pues no es más que la versión interna del sistema al final de la fase de elaboración, en fin tiene las versiones de todos los modelos que un sistema terminado contiene al final de la fase de construcción. Incluye el mismo esqueleto de subsistemas, componentes y nodos que un sistema definitivo, pero no existe toda la musculatura.

En la misma se exponen los estilos arquitectónicos de la aplicación, así como los principales elementos de la arquitectura. Se describen los principales patrones de arquitectura y patrones de diseño, unido a las tecnologías y herramientas de software que se utilizarán en el sistema a desarrollar.

Los usuarios de la Línea Base son:

- Los arquitectos del proyecto, que le sirve de guía para la toma de decisiones arquitectónicas y son los encargados del mantenimiento y refinamiento de esta línea base.

- Los integrantes del equipo de desarrollo, quienes la usan como guía para la implementación del sistema.

- Los clientes tienen en ella una garantía de la calidad y el conocimiento sobre en qué tecnología está desarrollada su solución.

Alcance

La Línea Base de la Arquitectura describe la estructura del sistema en un alto nivel de abstracción. Describe detalladamente el organigrama de la arquitectura según los estilos arquitectónicos que se utilizarán.

Se propone la utilización de un conjunto de patrones que resuelven problemas que se podrían presentar a lo largo del desarrollo del sistema y las principales tecnologías y herramientas que soportan los estilos y patrones especificados y que además deben cumplir con las restricciones del sistema.

2.2.1 Herramientas de desarrollo a utilizar

2.2.1.1 Eclipse 3.3.0 con PDT 1.0

Eclipse es un entorno de desarrollo integrado, de código abierto. Fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas. Actualmente es desarrollado por la “Fundación Eclipse”, una organización independiente sin ánimo de lucro, que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

La versión **Eclipse 3.3.0** dispone de un editor de texto y resaltado de sintaxis. Presenta compilación del código en tiempo real. Es compatible con el controlador de versiones Subversion (SVN). Permite la Refactorización.

PDT (Herramientas de desarrollo con PHP).

El PDT es un módulo basado en la popular plataforma Eclipse, por lo que no es solo uno de los más potentes para el desarrollo con PHP, sino también uno de los más flexibles.

La versión actual (1.0) posee un editor sensible al contexto, que provee resaltamiento de código, asistente de código y autocompletado de código. Permite la integración con el modelo del proyecto Eclipse, que posibilita inspeccionar el uso de las vistas del contorno del fichero y

del proyecto, así como la nueva vista PHP Explorer. Brinda soporte para la compilación incremental del código de PHP. Alberga extensos marcos de trabajo y APIs que permiten a los desarrolladores e ISVs (vendedores de software independientes) extenderlo fácilmente, para crear nuevas e interesantes herramientas orientadas al desarrollo de PHP.

2.2.1.2 Visual Paradigm UML 6.0

Es un entorno de creación de diagramas UML que permite identificar actores, casos de uso, entidades y trabajadores a partir de descripciones textuales. Posee un diseño centrado en casos de uso y enfocado al negocio. Permite enriquecer la documentación del modelo con imágenes, además de apoyar la documentación de texto enriquecido para los modelos y esquemas, puede visualizar su concepto mediante la incorporación de imágenes en la documentación.

Permite Guardar / cargar plantillas para la documentación del modelo, permitiendo su definición y estructura para ahorrar tiempo y esfuerzo. Apoya el modelado visual con los procedimientos almacenados, de modo que los puede crear y editar fácilmente en un entorno intuitivo. Presenta capacidades de ingeniería directa (versión profesional) e inversa, disponiendo de intercambio de información con Windows Visio. Permite la integración con los principales IDEs, así como la generación de código fuente en el lenguaje PHP. Es una herramienta multiplataforma. Usa un lenguaje estándar común para todo el equipo de desarrollo, facilitando su comunicación. La UCI paga la licencia para la ejecución de su uso.

2.2.1.3 Symfony 1.0.16

Symfony es un marco de trabajo desarrollado completamente con PHP5, por lo que cumple con las políticas de la Programación Orientada a Objetos (POO). Es fácil de instalar y configurar en la mayoría de las plataformas (con la garantía de que funciona correctamente en los sistemas Windows). Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos. Es muy fácil de entender, lo que permite su integración con librerías desarrolladas por terceros, además de poseer un código

fácil de leer incluyendo comentarios de phpDocumentor y que permite un mantenimiento muy sencillo. Es completamente independiente del sistema gestor de bases de datos.

2.2.1.4 Apache 2.2.6

Apache es el servidor web por excelencia, con algo mas de un 60% de los servidores de internet confiando en él. Es multiplataforma, elabora índices de directorios y además es de sencilla administración puesto a que la misma es basada en la configuración de un único archivo. Entre sus características mas sobresalientes están:

Fiabilidad: Alrededor del 90% de los servidores con más alta disponibilidad en el mundo, funcionan con Apache.

Gratuidad: Apache es totalmente gratuito y se distribuye bajo la licencia Apache Software License, que permite la modificación del código.

Extensibilidad: Se pueden añadir módulos para aumentar, las ya gigantescas, capacidades de Apache. Hay una extensa variedad de módulos que permiten desde generar contenido dinámico (con PHP, Java, Perl, Pyton), monitorizar el rendimiento del servidor, atender peticiones encriptados por SSL, hasta crear servidores virtuales por IP o por nombre (varias direcciones web son manejadas en un mismo servidor) y limitar el ancho de banda para cada uno de ellos. Dichos módulos incluso pueden ser creados por cualquier persona con conocimientos de programación.

2.2.1.5 Subversion 1.4.6

Esta versión presenta una herramienta llamada SVNsync, que proporciona la capacidad de replicar la historia de un repositorio a otro. El origen y el destino de los depósitos puede ser local, remoto o cualquier combinación de estos. Presenta alto rendimiento en la copia de trabajos enormes, el formato de copia que presenta esta versión permite que los clientes

busquen rápidamente una copia de trabajo. Detecta las modificaciones de archivos y trata con archivos de gran tamaño, además de presentar mejoras el manejo de archivos binarios.

Posee más de 40 nuevas soluciones a fallos, de modo que si ocurre algún problema se active un sistema de recuperación automático. Contiene nuevos interruptores y opciones de líneas de comandos para el cliente que lo hacen más fácil y cómodo que las otras versiones.

Es un superconjunto con lo mejor de todas las versiones anteriores y se considera la actual mejor puesta en libertad. Incluye numerosas ventajas entre las que se pueden destacar: la posibilidad de leer los depósitos creados por versiones anteriores, uso de cualquier lenguaje de programación aplicado a diversos tipos de proyectos y contiene las soluciones a fallos que no están presentes en cualquier despacho anterior.

2.2.2 Metodologías de Desarrollo

2.2.2.1 RUP

El Proceso Unificado Racional (RUP) establece tres elementos fundamentales para el desarrollo de la arquitectura de software como parte de la solución:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e Incremental.

Esto implica que se pueda desarrollar el software en pequeñas porciones estrechamente relacionadas, exista un mayor nivel organizacional del trabajo y el sistema sea cada vez más eficiente en la búsqueda de errores. Una particularidad de esta metodología es que en cada ciclo de iteración se hace imprescindible el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. Es muy utilizada por ser flexible e iterativa, lo que permite que en la medida que se vaya construyendo el software por ciclos se puedan detectar errores. Desde sus inicios ha tenido una gran aceptación.

2.2.3 Lenguajes de Programación Web

2.2.3.1 PHP5

PHP5 ("Pre-Procesador de Hipertexto 5") es un lenguaje de programación de lado de servidor, sencillo, de sintaxis cómoda y similar a la de otros lenguajes como C o C++, es rápido a pesar de ser interpretado, es multiplataforma y dispone de una gran cantidad de librerías y se le pueden agregar extensiones fácilmente. Es un lenguaje basado en herramientas con licencia de software libre, es decir, no hay que pagar licencias, ni estamos limitados en su distribución y se puede ampliar con nuevas funcionalidades si se desea. Es ideal tanto para el que comienza a desarrollar aplicaciones Web como para el desarrollador experimentado. Se puede utilizar como módulo de Apache, lo que lo hace extremadamente veloz. Por estar completamente escrito en C, se ejecuta rápidamente utilizando poca memoria.

Puede ser compilado y ejecutado en diversas plataformas, incluyendo diferentes versiones. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo. Se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD. Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos.

PHP5 hace un cambio en el manejo de los objetos a diferencia de PHP4, o sea, PHP4 trata los objetos igual que otros tipos de datos básicos como los enteros o los arreglos, de modo que cuando se realizan operaciones sobre un objeto el mismo es copiado íntegramente, mientras que en PHP5 las variables que nombran objetos son en realidad referencias. Otro de los elementos significativos propio de PHP5 es que incluye modificadores de control de acceso para implementar el encapsulamiento.

2.2.3.2 Java Scripts

Es un lenguaje del lado del cliente interpretado, facilitando que e no necesita ser compilado para obtener respuestas. Es basado en un prototipo donde las nuevas clases se generan clonando la clase base y extendiendo su funcionalidad. El tiempo de respuesta es sumamente pequeño y es interpretado por el navegador. Tiene como desventaja que puede ser leído por cualquiera dado a que su código es visible. Es un lenguaje débilmente tipiado, basado en

objetos y guiado por eventos, lo que permite el dinamismo de las páginas que incluyan este tipo de código.

2.2.3.3 Html

Es un lenguaje de programación web del lado del cliente. El Lenguaje de Marcado de Hipertexto es el lenguaje de marcado predominante para la construcción de páginas web. Usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes o multimedia. Se basa en especificar la estructura lógica del contenido en el texto y dejar que luego la presentación final de dicho hipertexto sea realizado por un programa especializado.

2.2.3.4 CSS

Es un lenguaje de programación web del lado del cliente que permite el control centralizado de la presentación de un sitio web completo, agilizando de forma considerable, la actualización del mismo. A través de él, una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o la elección del usuario, ejemplo: para ser impresa, mostrada en un dispositivo móvil, o "leída" por un sintetizador de voz.

2.2.4 Seguridad del Sistema

La información que manejará el sistema propuesto será extremadamente delicada, por que se necesita algún mecanismo de seguridad que garantice el flujo íntegro y almacenamiento seguro de la información. Para ello se usará la encriptación de los datos con el propósito de transformar la información donde el contenido de la misma lucirá como un conjunto de caracteres extraños, sin ningún sentido o lógica de lectura.

Existen dos tipos de cifrados de mensajes: el cifrado simétrico y el cifrado asimétrico, el cifrado simétrico necesita siempre utilizar nuevas claves para cada información que se quiera enviar, con este no se puede firmar mensajes por lo que el emisor no conoce el origen de los datos, tiene graves problemas con la distribución de las claves, por lo que se corre el riesgo de que la información trasmitida pueda ser interceptada por otra persona y como la clave para

descifrar el mensaje es la misma con el que se cifra y es conocida por más de una persona, la información puede ser desencriptada.

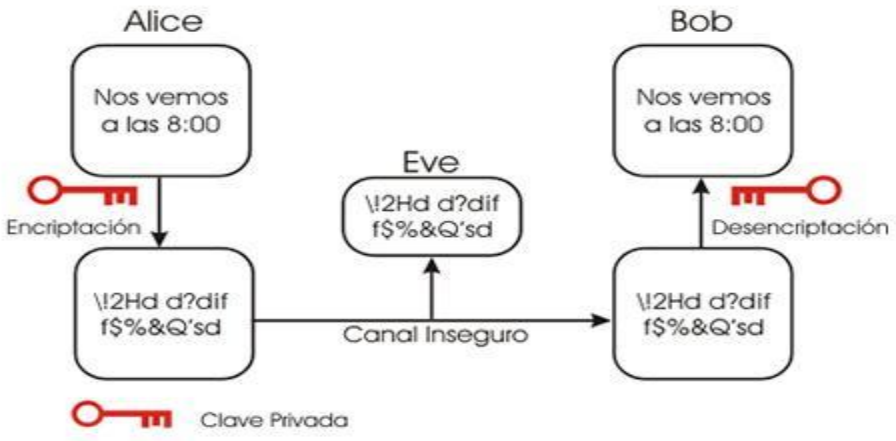

Figura 2 Cifrado simétrico

Se usará el cifrado asimétrico, que resuelve el problema del intercambio de claves ya que este usa dos claves, una pública y otra privada. Las dos claves pertenecen a quien se le envía el mensaje, una clave es pública y la puede conocer cualquiera, la otra clave es privada y el propietario debe guardarla para que nadie tenga acceso a ella porque en esta radica la seguridad del sistema. El remitente usa la clave pública del destinatario para cifrar el mensaje, y una vez cifrado, sólo la clave privada del destinatario podrá descifrarlo.

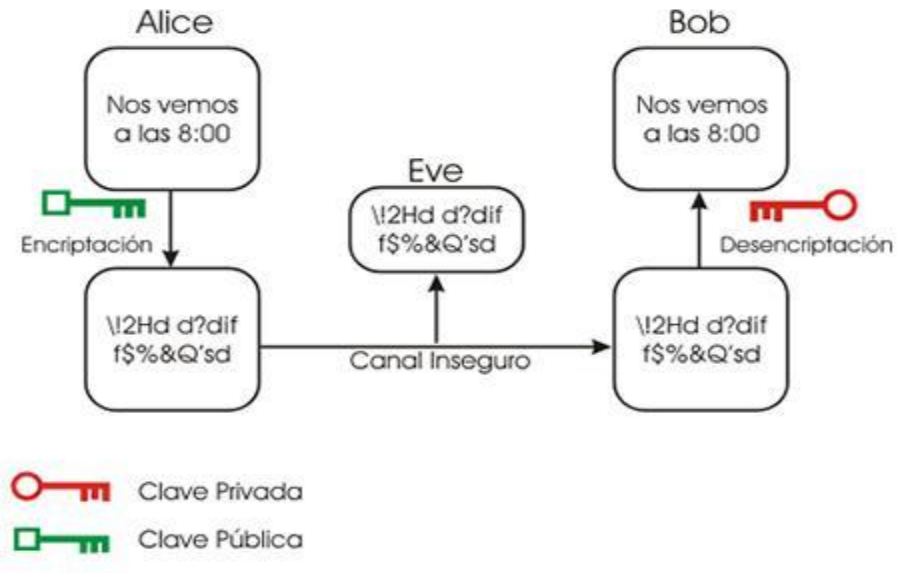

Figura 3 Cifrado asimétrico

Dentro de los cifrados asimétricos, los algoritmos más utilizados son:

ECC (Elliptical Curve Cryptography)
RSA (Rivest, Shamir, Adleman)

El ECC es un algoritmo que se utiliza muy poco, aunque es equivalente en fortaleza al RSA pero requiere menos potencia de cálculo en sus operaciones, es lento y no posee una gran seguridad por lo que se corre riesgo de poder ser descifrado el mensaje durante su recorrido. Por tales motivos se usará el RSA que es hoy en día el algoritmo de mayor uso en la encriptación asimétrica, es el de mayor seguridad de los algoritmos asimétricos y el más rápido. Se puede usar para el manejo de firmas, que permite comprobar quien es el emisor de los datos.

Los mensajes enviados usando el algoritmo RSA se representan mediante números y su funcionamiento se basa en el producto de dos números primos grandes (mayores que10^{100}) elegidos al azar para conformar la clave de descifrado. Emplea expresiones exponenciales en aritmética modular. Basado en la exponenciación modular y módulo fijos, el sistema RSA crea sus claves de la siguiente forma:

1. Se buscan dos números primos lo suficientemente grandes: **p** y **q** (de entre 100 y 300 dígitos).
2. Se obtienen los números **n = p * q y Ø = (p-1) * (q-1).**
3. Se busca un número de tal forma que no tenga múltiplos comunes con Ø.
4. Se calcula **d = e-1mod Ø**, con mod = resto de la división de números enteros.
5. Con estos números obtenidos, n es la clave pública y d es la clave privada. Los números **p**, **q** y **Ø** se destruyen. También se hace público el número **e**, necesario para alimentar el algoritmo.

El cálculo de estas claves se realiza en secreto en la máquina en la que se va a guardar la clave privada.

En cuanto a la longitud de las claves, el sistema RSA permite longitudes variables, pero se utilizarán claves de no menos de 1024 bits ya que se han roto claves de hasta 512 bits, aunque se necesitaron más de 5 meses y casi 300 ordenadores trabajando juntos para hacerlo.

La seguridad de este algoritmo radica en que no hay maneras rápidas conocidas de factorizar un número grande en sus factores primos utilizando computadoras tradicionales. Se basa en el uso de dos claves diferentes, que poseen una propiedad fundamental: una clave puede desencriptar lo que la otra ha encriptado. Las claves públicas y privadas tienen características matemáticas especiales, de tal forma que se generan siempre a la vez por parejas, estando cada una de ellas ligada estrechamente a la otra. Realizar la exponenciación modular es fácil, sin embargo, su operación inversa (la extracción de raíces) no es factible a menos que se conozca la factorización de la clave privada del sistema. De modo general con este cifrado se garantiza:

- Integridad: Es decir que el mensaje no sea alterado.
- Confidencialidad: Garantiza que solo quienes posean la llave puedan descifrar el mensaje.
- Autenticidad: Permite comprobar que el emisor es quien dice ser y no otro.

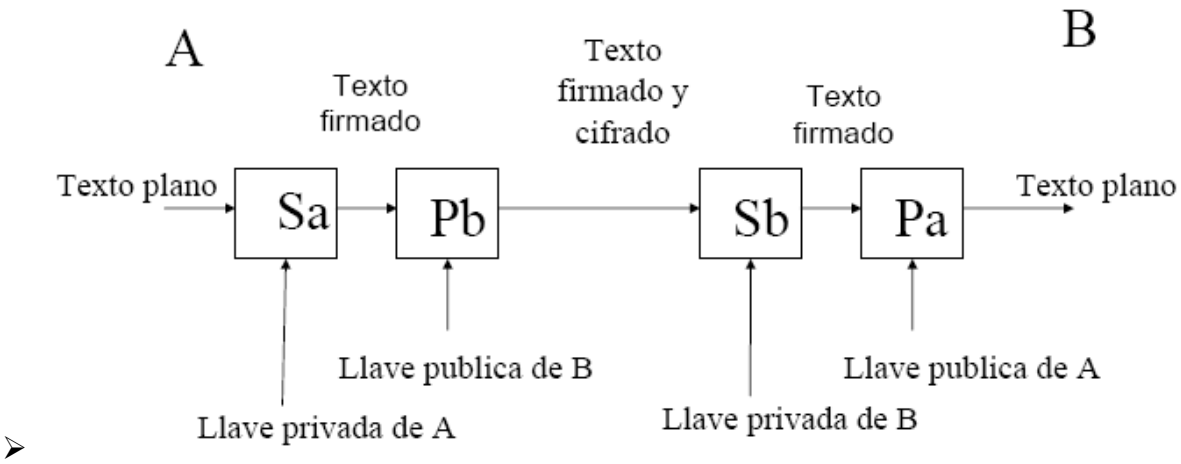

Figura 4 Firmado y cifrado de un mensaje

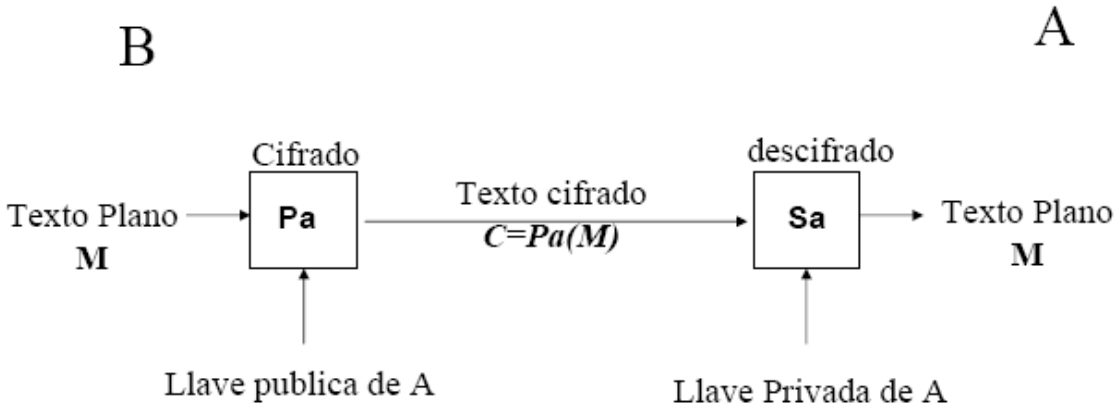

Figura 5 Cifrado de un mensaje

2.2.5 Arquitectura en Capas

El estilo Arquitectura en Capas facilita la modularidad del sistema, la localización de errores y mejora considerablemente el soporte del mismo. La esencia de su estructura está dada en que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las interacciones entre ellas ocurren generalmente por invocación de métodos.

Las principales ventajas de este estilo arquitectónico son mostradas a continuación:

- Permite el desarrollo paralelo del sistema por cada capa, de forma tal que al mismo tiempo se pueda trabajar en todas y cada una de ellas.

- Existe una gran independencia inter-capas, de forma tal que de efectuarse un cambio en alguna de ellas, no repercute en el resto.

- Cada capa brinda única y exclusivamente información a su inmediata superior y a la vez se sirve de su inmediata inferior.

- Facilita el mantenimiento y soporte del proyecto.

- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nuevas funcionalidades).

- Soporta un diseño basado en niveles de abstracción crecientes, lo cual permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.

- Proporciona amplia reutilización. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

2.2.6 Patrones Arquitectónicos

2.2.6.1 Modelo Vista Controlador (MVC)

El MVC ha sido propio de las aplicaciones en Smalltalk desde 1992, antes que se generalizaran las arquitecturas en capas múltiples. Un propósito común en numerosos sistemas es el de tomar datos de un repositorio y mostrarlos al usuario, permitiéndole al mismo introducirle modificaciones.

La lógica de una interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si realizamos un diseño ofuscado, es decir, que mezcle los componentes de interfaz y de negocio, traerá como consecuencia que cuando necesitemos cambiar el interfaz tengamos que modificar trabajosamente los componentes del negocio, implicando mayor trabajo y riesgo de error. Por lo que el MVC separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. Con esta idea, separa su estructura en tres clases diferentes:

- **Modelo:** datos y reglas de negocio.
- **Vista:** muestra la información del modelo al usuario.
- **Controlador:** gestiona las entradas del usuario.

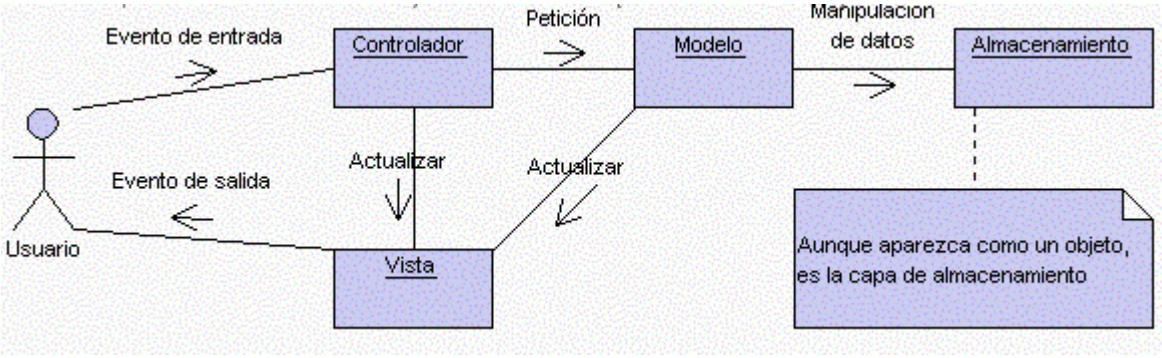

Figura 6 Modelo Vista Controlador (MVC)

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. Veamos cada componente:

1. El **modelo** es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

- Definir las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si se solicita Gestionar Foto de Seguridad, chequear que el solicitante sea un Directivo de Seguridad y Protección".
- Llevar un registro de las vistas y controladores del sistema.

2. El **controlador** es responsable de:

- Recibir los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contener reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

3. Las **vistas** son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tener un registro de su controlador asociado (normalmente porque además lo instancia).
- Enviar hacia el modelo las solicitudes o cambios hechos por el usuario.

2.2.6.2 Arquitectura en Dos (2) Capas

La arquitectura en 2 capas que va a ser utilizada para la realización del sistema propuesto será una arquitectura especial, diferente a la clásica, que está compuesta por la capa Lógica de la Aplicación y la capa de Acceso a Datos. La Lógica de la Aplicación es la encargada del intercambio de sistemas externos con la aplicación determinando las acciones a ejecutarse de acuerdo con las peticiones realizadas, maneja la seguridad del sistema y efectúa solicitudes (a la capa de Acceso a Datos) de realización de operaciones. La capa de Acceso a Datos es la encargada de manipular toda la información que respecta con el acceso a los repositorios de datos.

2.2.7 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

2.2.7.1 Patrón Decorator

Permite que cuando se tengan que agregar nuevas funcionalidades al sistema, no sea necesario añadir más clases y subclases, evitando así que se acabe con una jungla de estas. Con este patrón se trata de evitar el efecto de herencia-jungla, puesto a que permite la asignación dinámicamente de nuevas responsabilidades a un objeto. Es una alternativa más flexible a crear subclases y extender las funcionalidades de una clase. De esta forma se añaden atributos o comportamientos adicionales a un objeto concreto, no a una clase.

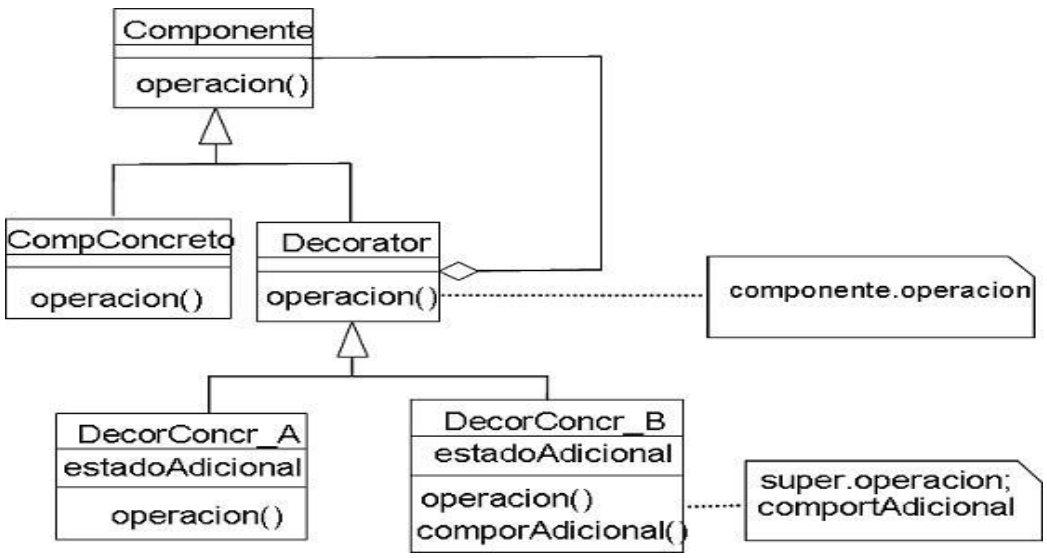

Figura 7 Estructura del Patrón Decorador (Decorator)

2.3 Descripción del Sistema

Con el presente trabajo se desarrolla una propuesta de arquitectura para el sistema Gestión de Fotos del proyecto Control de Acceso, que guiará al equipo de desarrollo para la realización de la aplicación propuesta. El sistema informático que solucionará los problemas que existen actualmente con la gestión de fotos en la Universidad de las Ciencias Informáticas (UCI), estará constituido por dos módulos: el Servicio Web “Gestor de Fotos” y la aplicación web “Sistema Administrador Servicio Web”. Para su mejor comprensión serán analizados ambos por separados.

Servicio Web “Gestor de Fotos”: Este servicio web tendrá como objetivo principal brindar una serie de servicios otros sistemas internos que lo necesiten, acerca de las fotos correspondientes a las personas (estudiantes o trabajadores) pertenecientes a la UCI. El mayor peso de los procesos a informatizar cae precisamente sobre lo referente al Sistema de Acreditación (SA) y al Sistema de Seguridad y Protección (SSP). El SA necesita del servicio web (SW) para guardar la foto principal de alguna persona con sus respectivos datos y el SSP podrá ejecutar procesos parecidos pero para fotos de seguridad (ejemplo: foto de huella digital).

El Sistema de Administración de la Base de Datos UCI (SABDUCI) realizará peticiones de actualización de datos, de forma tal que los datos almacenados en el repositorio coincidan con los de la dicha base de datos. De modo general cualquier sistema previamente autorizado podrá pedirle al SW que busque una determinada “foto principal”. Internamente el servicio web tendrá implementada cada una de estas funciones para poder ejecutar las solicitudes de los diferentes sistemas clientes de acuerdo a sus permisos, además buscará y procesará las informaciones necesarias en un repositorio de fotos, donde las mismas se encontrarán clasificadas y organizadas según su tipo.

Sistema Administrador Servicio Web: Esta parte del sistema propuesto será la encargada de permitirle al administrador del sistema la configuración del SW. El administrador del sistema a través de esta aplicación podrá añadir nuevos tipos de fotos a ser procesadas por el WS, unido con los datos que deberá poseer cada una como metadatos y efectuar un cambio en

alguno de los datos de un tipo de foto determinado. Además deberá permitirle realizar su otra actividad importante que consiste en adicionar un nuevo sistema cliente con sus privilegios para que consuma al SW o realizar cambios en los permisos de de alguno de los sistemas antes adicionado.

2.4 Conclusiones

En el anterior capítulo se definieron, luego del profundo estudio realizado en el capítulo 1, las herramientas, lenguajes, metodologías, arquitectura y patrones que serán utilizados para el desarrollo de la futura aplicación informática. Finalmente, se realizó una descripción a grandes rasgos del sistema informático propuesto para dar solución a las problemáticas existentes.

Capítulo 3 Descripción de la Arquitectura

3.1 Introducción

El conjunto de modelos que describen la Línea Base de la Arquitectura se denomina ***Descripción de la Arquitectura***. El papel de la descripción de la arquitectura es guiar al equipo de desarrollo a través del ciclo de vida del sistema. Esta descripción puede adoptar diferentes formas, puede ser un extracto (un conjunto de vistas), o puede ser una reescritura de los extractos de forma que sea más fácil leerlos. Estas vistas incluyen elementos arquitectónicamente significativos (casos de uso, subsistemas, interfaces, algunas clases y componentes, nodos y colaboraciones). La descripción de la arquitectura tiene cinco secciones, una para cada modelo: una vista del modelo de casos de uso, una vista del modelo de análisis (opcional), una vista del modelo de diseño, una vista del modelo de despliegue, y una vista del modelo de implementación.

Propósito

Este documento brinda información sobre la arquitectura del sistema mediante las diferentes vistas arquitectónicas: vista de CU, vista lógica, vista de implementación y vista de despliegue; y al mismo tiempo permite tomar decisiones arquitectónicamente significativas.

Con la elaboración del presente documento se pretende:

- Mejorar la comprensión del sistema.
- Organizar el desarrollo del software.
- Fomentar la reutilización del código.
- Hacer evolucionar el sistema.

Alcance

Por el presente documento se regirá el equipo de desarrollo del proyecto Control de Acceso para la realización del Servicio Web Gestor de Fotos y la aplicación web para administrarlo, a lo largo de todo el ciclo de vida de ambos.

3.2 Metas y restricciones arquitectónicas

A continuación se plantean las metas y restricciones con las que se deberá cumplir para la correcta puesta en funcionamiento del sistema de software propuesto y que además son significativas para la arquitectura puesto a que constituyen los requerimientos no funcionales del sistema antes mencionado.

3.2.1 Requerimientos de Hardware

PC Administrador del Sistema

1. Periféricos: Mouse y Teclado
2. Tarjeta de Red.
3. 128 MB de RAM.
4. 30 GB o más de espacio en disco duro.

Servidor web y Gestor de Fotos

1. Periféricos: Mouse y Teclado.
2. Tarjeta de Red.
3. 2 GB de RAM.
4. 30 GB de espacio en disco duro.

PC Repositorio de Fotos

1. Periféricos: Mouse y Teclado.
2. Tarjeta de Red.
3. 1GB de RAM.
4. 30 GB de espacio en disco duro.

3.2.2 Requerimientos de Software

PC Administrador del Sistema

1. Sistema Operativo: Cualquiera (La Aplicación Administrador Web Service es una aplicación web).
2. Navegador Web: Internet Explorer o Mozilla Firefox (Ambas en cualquier versión).

Servidor web y Gestor de Fotos

1. Sistema Operativo: Linux.
2. Servidor web: Apache.

PC Repositorio de Fotos

1. Sistema Operativo: Linux o Windows (superior al 95).

3.2.3 Redes

1. Tipo de cable a utilizar UTP categoría 5, con trasmisiones de datos de hasta 100 Mbps.
2. Debe estar protegida contra fallos de corriente y conectividad.

3.2.4 Seguridad

1. Se realizará un fuerte tratamiento de excepciones con el objetivo de garantizar que la información manejada esté protegida del acceso no autorizado.
2. Se implementará un mecanismo de acceso al Repositorio de Fotos, que estará dado por la diferenciación de las acciones que el sistema realizará para cada Sistema Cliente que se sirva de él.
3. La encriptación de contraseñas.
4. La asignación de los Sistemas Clientes y sus permisos sobre el sistema se definirán desde el módulo: Administración del web service.
5. La información que se maneje en el sistema será objeto de cuidadosa protección contra estados inconsistentes.
6. La conexión entre la PC Administrador del Sistema y el Servidor Web deberá ser segura, por lo que utilizará el protocolo HTTPS.
7. La conexión entre los sistemas clientes (incluido el Sistema de Administración del Web Service) y el web service Gestor de Fotos deberá ser a través del Protocolo de Acceso Simple a Objetos (SOAP).

8. La conexión entre el web service Gestor de Fotos y el Repositorio de Fotos deberá ser mediante el protocolo TCP/IP.

3.2.5 Portabilidad

1. El sistema deberá poder ser accedido desde cualquier Sistema Operativo.

3.2.6 Usabilidad

1. El sistema podrá ser usado por cualquier Sistema Cliente previamente autorizado y en cualquier instante del día.

3.2.7 Apariencia de la “Aplicación Administrador Servicio Web”

1. Diseño sencillo que permita el mínimo de entrenamiento para utilizar el sistema.
2. Deberá permitir el uso de los colores corporativos.
3. Presentará un mapa del sitio que facilite la navegabilidad dentro del mismo.
4. Texto claro y de fácil comprensión.

3.2.8 Rendimiento

1. Se necesitan tiempos de respuestas muy pequeños.
2. La velocidad de procesamiento de la información deberá ser alta.

3.3 Vista de Casos de Uso

Los casos de uso van guiando la arquitectura del software durante todo el ciclo de vida del proyecto, puesto a que responden a funcionalidades del sistema. A partir de ellos podemos saber como va a ser la interacción de los usuarios con el futuro sistema informático. Los casos de uso arquitectónicamente significativos se realizan en las primeras iteraciones del proyecto y después se van desarrollando los demás, por lo que es muy importante efectuar su selección adecuadamente para que a partir de ellos se puedan centrar los cimientos de la arquitectura de software. Se debe definir dentro de los casos de uso más importantes, cuales son los que necesitan mayor rendimiento y capacidad de hardware para su funcionamiento

dentro del sistema.

Existen varias vistas de la arquitectura planteadas por la metodología RUP, una de ellas es la Vista de Casos de Uso, que está compuesta por los actores y casos de uso más significativos del sistema (con su realización puedan provocar algún cambio para el sistema o repercuta directamente en algún otro caso de uso).

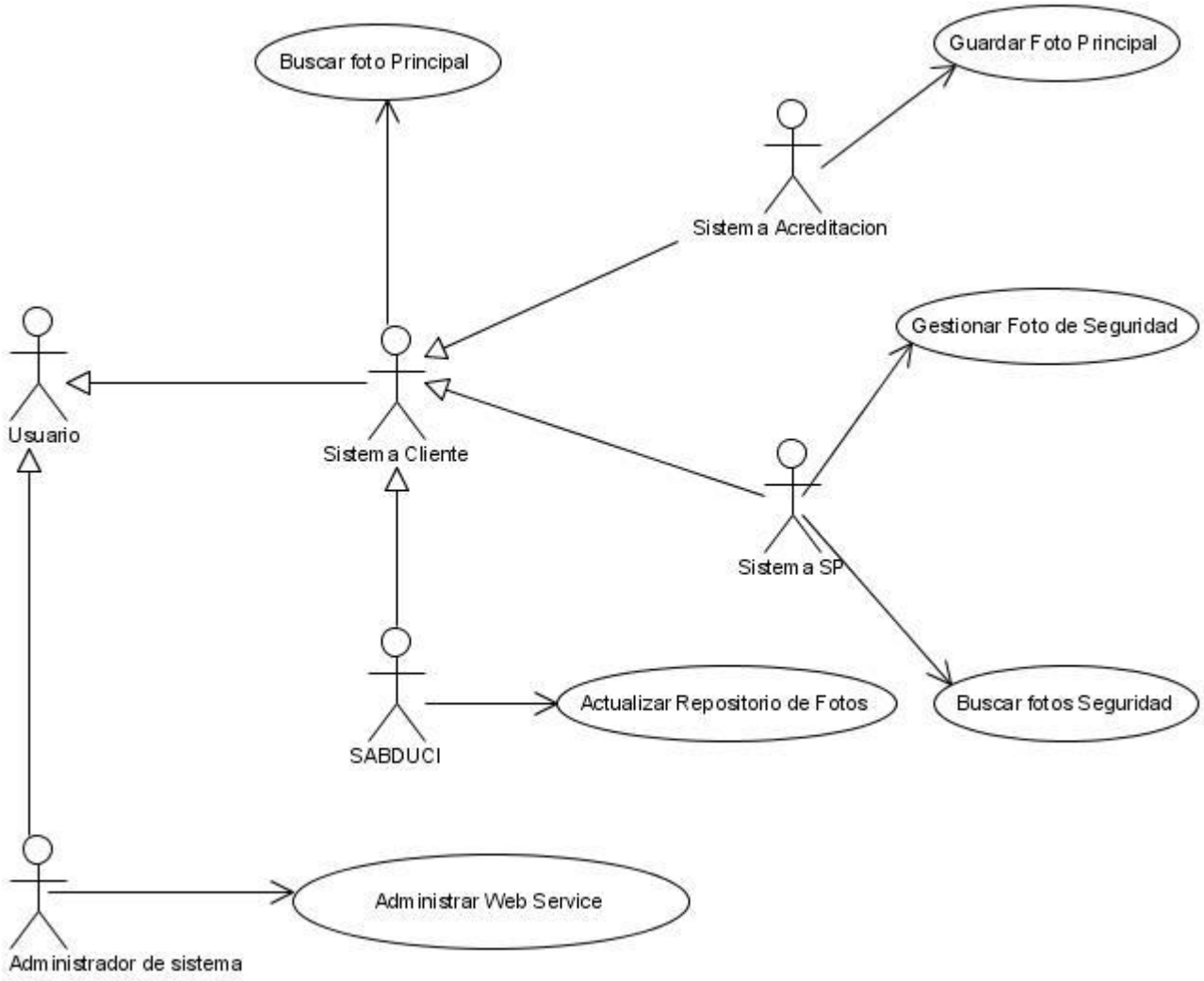

Figura 8 Vista de Casos de Uso Arquitectónicamente Significativos

Casos de Uso Arquitectónicamente Significativos:

3.3.1 Descripción de CU Guardar Foto Principal

El caso de uso se inicia cuando el Sistema de Acreditación le solicita al servicio web (WS) guardar una imagen en el repositorio. Inicialmente se selecciona que es lo que va a ser procesado: un estudiante o trabajador y acto seguido se envían los datos necesarios para la ejecución del servicio a través de la aplicación. Si es estudiante se deben enviar los siguientes datos: facultad, grupo, nombre (s), número de solapín, primer apellido, segundo apellido, provincia, municipio, usuario y CI. En el caso del trabajador los datos a enviar son los siguientes: área de trabajo, nombre (s), número de solapín, primer apellido, segundo apellido, provincia, municipio, usuario y CI. Luego de esto, el SW recepciona los datos y los revisa para ver si son correctos. Si no existe problema alguno, procede a estandarizar la imagen en la resolución, ancho, largo y formato (predeterminados por el administrador del sistema) y seguidamente le introduce como metadatos, los datos recepcionados anteriormente. Una vez ocurrido todo este proceso previamente explicado, el SW procede a guardar la imagen en el repositorio:

- Para los estudiantes: Explora el repositorio para ver si existe una subcarpeta con el nombre de su facultad y luego revisa si hay una para su grupo, de no existir alguna de ellas, la crea, unido con una para dicho estudiante que tendrá como nombre el o los nombre(s) del mismo, concatenado con su número de solapín (ejemplo: rolandoramiro50263). Finalmente guarda la imagen de modo que su dirección podría quedar así: d:\repositoriofotos\facultad1\1505\ rolandoramiro50263\principal50263.jpg y envía confirmación de la operación exitosa.

- Para los trabajadores: Explora el repositorio para ver si existe una subcarpeta con el nombre de su área de trabajo y de no encontrarla la crea, unido con otra que tendrá como nombre el o los nombre(s) del mismo, concatenado con su número de solapín (ejemplo: osmanys70654). Finalmente guarda la imagen de modo que su dirección podría quedar así: d:\repositoriofotos\servicios\osmanys70654\principal70654.jpg y envía confirmación de la operación exitosa.

En caso de que el Sistema de Acreditación envíe datos incorrectos del estudiante o trabajador en cuestión o el mismo ya esté registrado, el SW devuelve un mensaje de notificación de error.

3.3.2 Descripción de CU Buscar Foto Principal

El caso de uso se inicia cuando algún sistema cliente, previamente autorizado, solicita buscar una imagen de una determinada persona. El SW realiza los procesos de búsqueda de manera diferente atendiendo a si es un estudiante o un trabajador:

- Para los estudiantes: El Sistema Cliente, luego de haberse conectado con la Base de Datos UCI (BDUCI) y obtenido los datos correspondientes, deberá enviarlos en el siguiente orden: 1. Facultad, 2. Grupo, 3. Nombre(s) y 4. Número de solapín. Estos datos, unidos con otros que están predefinidos, son utilizados por el SW para obtener la dirección en se encuentra la foto del estudiante dentro del repositorio (ejemplo: d:\repositoriofotos\facultad1\1505\rolandoramiro50263\principal50263.jpg).

- Para los trabajadores: El Sistema Cliente, luego de haberse conectado con la BDUCI y obtenido los datos correspondientes, deberá enviarlos en el siguiente orden: 1. Área de trabajo, 2. Nombre(s) y 3. Número de solapín. Estos datos unidos con otros que están predefinidos son utilizados por el SW para obtener la dirección en se encuentra la foto del trabajador dentro del repositorio (ejemplo: d:\repositoriofotos\servicios\ osmanys70654\principal70654.jpg).

Acto seguido se obtiene la imagen y es enviada al sistema solicitante para que realice sus operaciones. En caso de no encontrarla envía un mensaje de notificación.

3.3.3 Descripción de CU Gestionar Foto de Seguridad

Este caso de uso es invocado única y exclusivamente por el Sistema de Seguridad y Protección (SSP). Inicia cuando el sistema solicita al servicio web la realización de alguna

operación sobre una foto de seguridad (foto específica, ejemplo: Huella Digital). Existen dentro de él dos procesos importantes: Guardar Foto de Seguridad y Actualizar Foto de Seguridad.

- Guardar Foto de Seguridad: Este proceso inicia cuando el SSP solicita su ejecución y envía hacia el SW la foto de seguridad con los datos correspondientes (dependiendo del tipo que sea) del estudiante o trabajador fotografiado. El servicio web comprueba la correctitud de los datos analizando el tipo de foto de seguridad que es y si no encuentra problema alguno procede con la operación de estandarizar la imagen e introducirle sus datos como metadatos. Seguidamente guarda la foto en el repositorio en la subcarpeta correspondiente a la persona fotografiada, de forma tal que la dirección en el Repositorio de Fotos pueda quedar estructurada de la siguiente manera: d:\repositoriofotos\facultad1\1505\rolandoramiro50263\huelladigital50263.jpg (para los estudiantes) o d:\repositoriofotos\servicios\ osmanys70654\huelladigital70654.jpg (para los trabajadores). En caso de existir algún problema en alguno de los datos, se interrumpe el proceso y se envía mensaje de notificación.

- Actualizar Foto de Seguridad: Este proceso inicia cuando el SSP le solicita al servicio web su ejecución. Cuando una foto de seguridad está a punto de caducar se hace necesario su actualización, para ello el solicitante envía al SW los datos (luego de haberlos obtenido de la BDUCI) de la persona que será procesada, para de esta forma proceder con la localización de la dirección de la imagen específica en el Repositorio de Fotos. Encontrada la fotografía y con la confirmación de actualización recibida del SSP, el WS extrae sus metadatos y se los introduce a la nueva imagen luego de haberla estandarizado. Finalmente toma la antigua fotografía, la coloca en el Historial de Fotos correspondiente al estudiante o trabajador en cuestión (ejemplo de dirección: d:\repositoriofotos\servicios\osmanys70654\historial70654\050608huelladigital70654.jpg) y en su lugar pone a la nueva. En caso de no encontrarla o que la operación de actualización sea cancelada, se interrumpirá el proceso y se procederá al envío de una notificación.

3.3.4 Descripción de CU Buscar Foto de Seguridad

Este caso de uso es invocado única y exclusivamente por el Sistema de Seguridad y Protección. Inicia cuando este sistema solicita al WS buscar una foto de seguridad de una determinada persona, en el repositorio de fotos. El SW realiza los procesos de búsqueda de manera diferente atendiendo a si es un estudiante o un trabajador:

- Para los estudiantes: El Sistema Cliente, luego de haberse conectado con la BDUCI y obtenido los datos correspondientes, deberá enviarlos en el siguiente orden: 1.Facultad, 2. Grupo, 3. Nombre(s) y 4. Número de solapín. Estos datos, unidos con otros que están predefinidos, son utilizados por el SW para obtener la dirección en se encuentra la foto del estudiante dentro del repositorio (ejemplo: d:\repositoriofotos\facultad1\1505\rolandoramiro50263\huelladigital50263.jpg).

- Para los trabajadores: El Sistema Cliente, luego de haberse conectado con la (BDUCI) y obtenido los datos correspondientes, deberá enviarlos en el siguiente orden: 1. Área de trabajo, 2. Nombre(s) y 3. Número de solapín. Estos datos unidos con otros que están predefinidos son utilizados por el SW para obtener la dirección en se encuentra la foto del trabajador dentro del repositorio (ejemplo: d:\repositoriofotos\servicios\ osmanys70654\ huelladigital70654.jpg).

Acto seguido se obtiene la imagen y es enviada al sistema solicitante para que realice sus operaciones. En caso de no encontrarla envía un mensaje de notificación.

3.3.5 Descripción de CU Administrar Servicio Web

Este caso de uso es iniciado por el Administrador del Sistema (AS) a través de la Aplicación Administrador Servicio Web (AASW) que hace función de intérprete entre él y el SW. El AS es el responsable de realizar dos actividades imprescindibles para el correcto funcionamiento del servicio web:

1. Realizar la asignación de permisos a Sistemas Clientes para usar el servicio web: Esta tarea permite al AS adicionar nuevos Sistemas Clientes para que consuman el SW,

unido a sus permisos. Además le brinda la posibilidad de incorporarle nuevos permisos a sistemas antes adicionados.

2. Estandarización de los distintos tipos de fotos: Con el cumplimiento de esta tarea el AS introduce al SW los datos relacionados con las características que deben tener los distintos tipos de fotos que puedan ser guardadas, ya sea por el Sistema de Acreditación o por el Sistema de SP. Estas características son la resolución, el ancho, el largo, el formato y los datos específicos (datos que serán introducidos dentro de cada tipo de foto como metadatos).

Inicialmente el AS solicita la realización de cualquiera de estas dos actividades a través de la AASW. En caso de selección de la actividad uno, el AS solicita la incorporación de un nuevo Sistema Cliente o la asignación de permisos a uno ya presente, acto seguido el servicio web procesa la información y envía un mensaje de notificación. En caso de selección de la actividad dos, el AS solicita la adición de un nuevo tipo de foto o efectuar un cambio en algún valor de los estándares de algún tipo antes registrado.

3.3.6 Descripción de CU Actualizar Repositorio de Fotos

Es un caso de uso que puede ser iniciado única y exclusivamente por el Sistema de Administración de la Base de Datos UCI (SABDUCI). Con su ejecución es posible la actualización de los datos de las fotos que aparecen en el Repositorio de Fotos, para ello se analizarán las posibles variantes, cada una por separado:

- Baja del Centro: Cuando un estudiante o trabajador causa baja del centro, el SABDUCI se encarga de eliminar todos sus datos de dicha base de datos y por lo que deberá enviar una solicitud al SW para eliminar las fotos que aparecen en el repositorio. Para ello envían los datos de la persona en cuestión y con esto se procede a la localización de la carpeta correspondiente. Una vez localizada se copia en el Repositorio de Expedidos (ejemplo: d:\repositorioexpedidos\michel50657) y se elimina del Repositorio de Fotos.

- Cambio de Grupo Docente o Facultad: Cuando estudiante es cambiado de grupo o

facultad se hace necesario actualizar el Repositorio de Fotos, para ello el SABDUCI envía los datos del estudiante en cuestión y lo que será actualizado. El SW procede con localización en el repositorio de la carpeta relacionada, finalmente cortándola de donde está actualmente y pegándola en su correspondiente lugar actual. Ejemplo: (d:\repositoriofotos\facultad1\1505\rolandoramiro50263)
(d:\repositoriofotos\facultad1\1501\rolandoramiro50263).

- Cambio de Puesto de Trabajo: Cuando un trabajador es cambiado de puesto de trabajo se hace necesario actualizar el Repositorio de Fotos, para ello el SABDUCI envían los datos del trabajador en cuestión y lo que será actualizado. El SW procede con la localización de la carpeta relacionada en el Repositorio de Fotos, finalmente cortándola de donde está actualmente y pegándola en su correspondiente lugar actual. Ejemplo:

  d:\repositoriofotos\servicios\ osmanys70654
  d:\repositoriofotos\direccionlaboratorios\ osmanys70654

- Cambio de Datos: Cuando el SABDUCI haya efectuado algún cambio en alguno de los datos de una determinada persona (ejemplo: alguna corrección en el nombre, apellidos o CI), enviará una solicitud de actualización al SW, unido a los datos de la persona en cuestión y los que deberán ser actualizados. Una vez recibida la solicitud, el SW localizará la carpeta correspondiente y procederá a extraer los metadatos de las todas las fotografías pertenecientes a dicha persona que contienen el dato a modificar. Finalmente se actualizarán los datos, se introducirán nuevamente como metadatos a sus respectivas imágenes, se guardarán en su carpeta en el repositorio y se enviará un mensaje de notificación. En caso de no localizar la carpeta correspondiente a la persona en cuestión, se emitirá un mensaje de aviso.

3.4 Vista Lógica

Esta vista representa un subconjunto del artefacto Modelo de Diseño, representando los elementos de diseño más importantes para la arquitectura del sistema. Se describen las clases más importantes y su organización en paquetes en cada uno de los módulos.

Módulo “Gestor de Fotos”

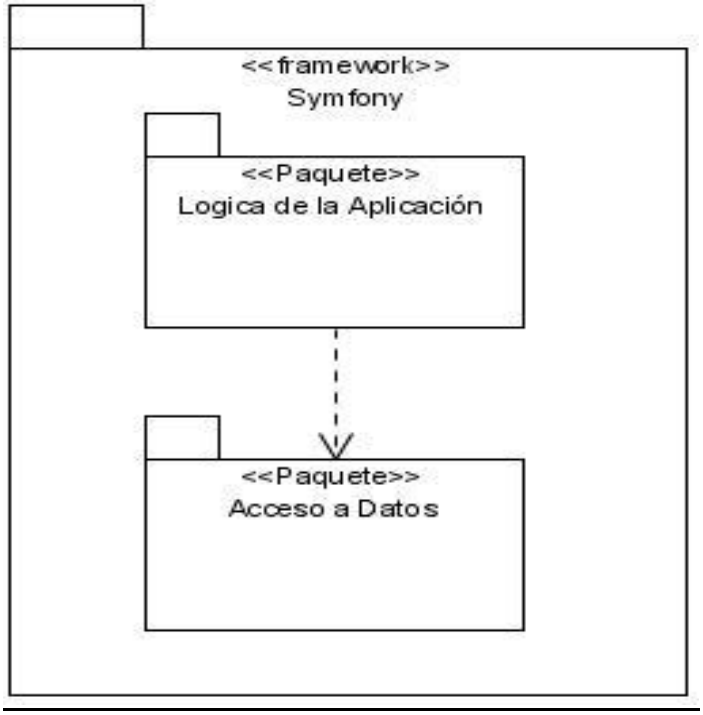

Figura 9 Vista Lógica del Módulo “Gestor de Fotos”

Paquete de Lógica de la Aplicación

ckwebserviceCF: Esta clase, es el controlador frontal del servicio web, lo que indica que constituye el único punto de entrada al mismo. Determina la acción a ejecutarse de acuerdo con las peticiones de los Sistemas Clientes. Maneja la seguridad del sistema.

actions: Esta clase contiene la lógica de la aplicación. En ella se verifica la integridad de las peticiones realizadas por los Sistemas Clientes, se efectúan solicitudes (al modelo) de realización de operaciones y se preparan los datos que van a ser enviados a los sistemas anteriormente mencionados.

Paquete de Acceso a datos

ManipuladorRepositorio: Es la clase que contendrá la lógica del negocio del servicio web. Posee la lógica de intercambio de información con los repositorios de fotos (Repositorio de Fotos y Repositorio de Expedidos).

Módulo "Aplicación Administrador del Servicio Web"

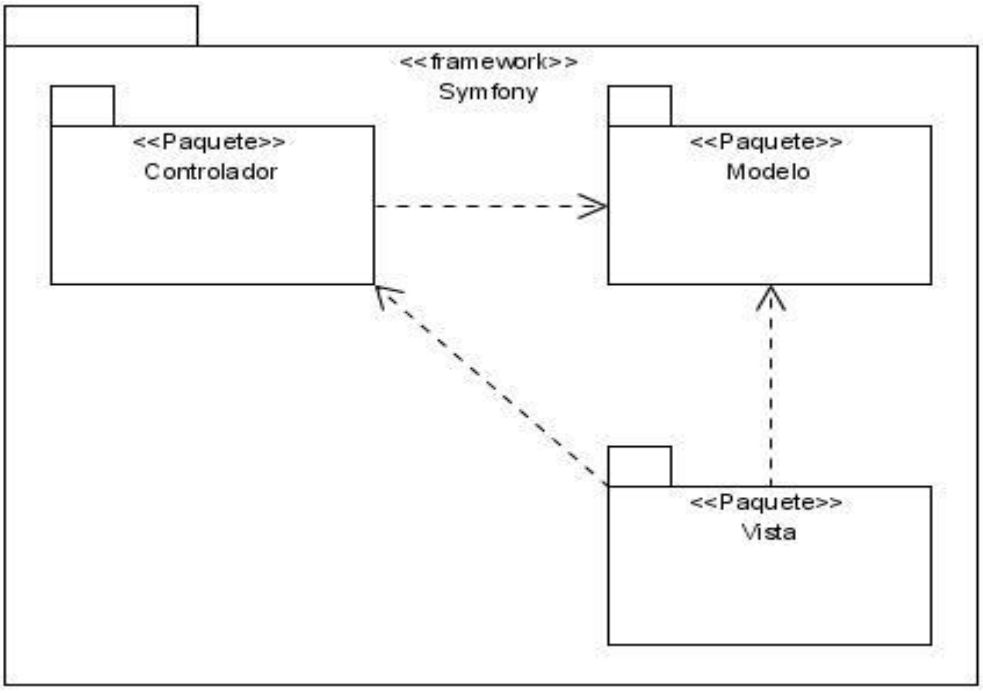

Figura 10 Vista lógica del Módulo "Administrador del Web Service"

Paquete Vista

HTML_Form_AgregarSistCliente: Esta clase permite que el Administrador del Sistema (AS) pueda agregar los datos de un nuevo Sistema Cliente, para que el mismo sea registrado y

pueda consumir el SW.

HTML_Form_ReasignarPrivilegios: Esta clase permite al Administrador del sistema efectuar un cambio en alguno de los permisos que tenga asignado un determinado Sistema Cliente.

HTML_Form_AgregarNuevoTipoFoto: Esta clase permite que el AS pueda agregar un nuevo tipo de foto (con sus respectivos datos) a ser procesada por el web service.

HTML_Form_GestionarCambiosDatos: Esta clase permite al AS efectuar un cambio en alguno de los datos correspondientes a un determinado tipo de foto, que haya sido agregado anteriormente al sistema.

Layout: Es una plantilla global que almacena el código HTML común para todas las páginas de la aplicación, de modo que no haya que repetirlo en cada una de las páginas.

Paquete Controlador

PS_AdministradorWebServiceCF: Esta clase constituye el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse en dependencia a las peticiones del usuario. Además maneja la seguridad del sistema.

Actions: Contiene la lógica de la aplicación, verifica la integridad de las peticiones del usuario, se efectúan solicitudes (al modelo) de realización de operaciones y prepara los datos requeridos por la vista.

Paquete Modelo

ManipuladorFichero: Es la clase que contendrá la lógica del negocio de la aplicación web. Posee la lógica de intercambio de información con el fichero de configuración de la aplicación.

3.5 Vista de Implementación

Vista de Implementación del Módulo "Gestor de Fotos"

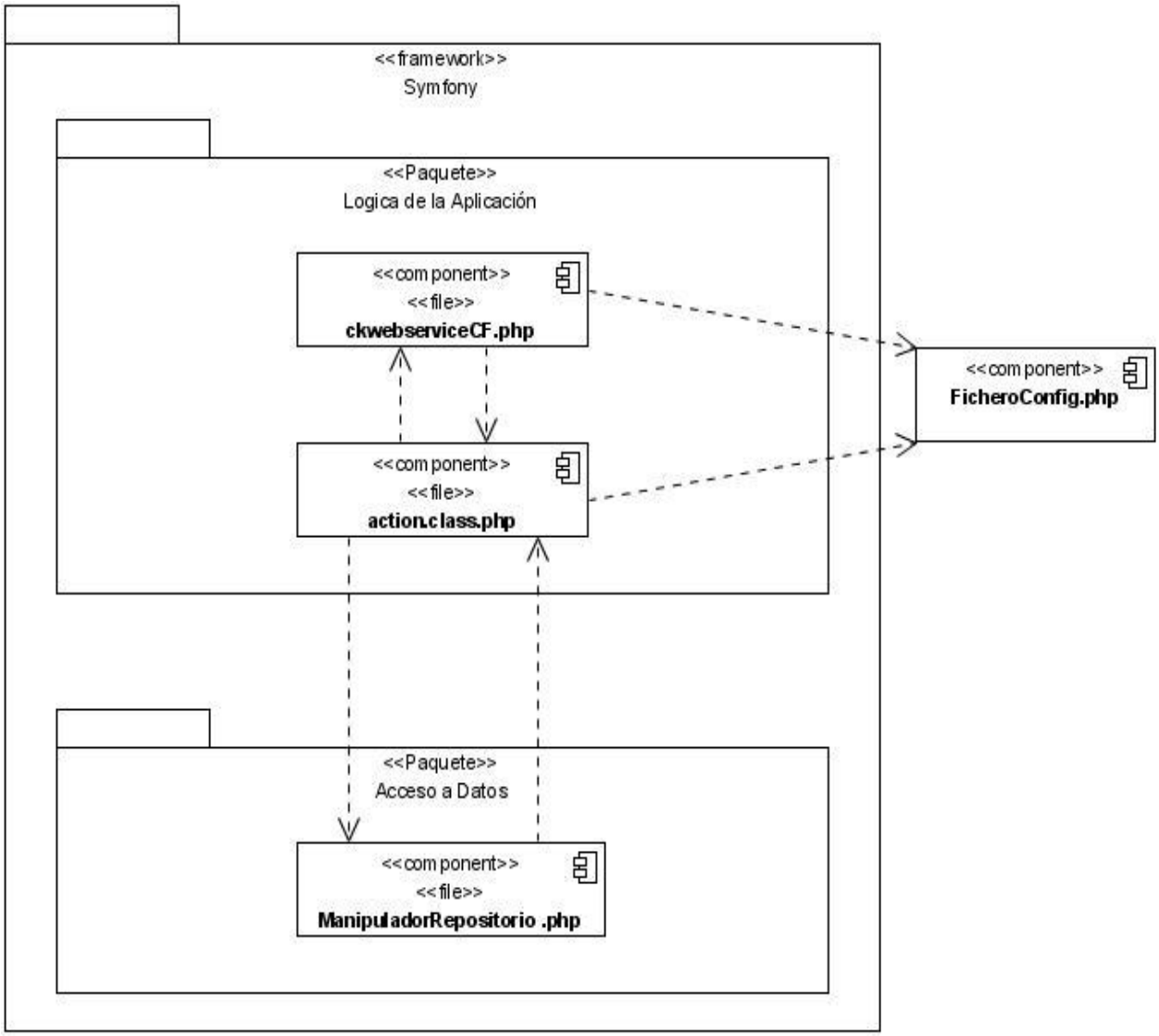

Figura 11 Vista de Implementación de módulo "Gestor de Fotos"

Ficheroconfig.php: Es un componente externo al diseño arquitectónico de este módulo, del cual dependerán en gran medida las acciones que realizará SW, puesto a que contiene las configuraciones de lo que podrá hacer el mismo. Aparecerán cuales Sistemas Clientes tienen acceso a consumirlo y sus permisos. Además se encontrarán definidos los diferentes tipos de fotos y los datos que deberán poseer cada uno de ellos.

Lógica de la Aplicación

ckwebserviceCF.php: Este componente constituye el controlador frontal del servicio web, por lo que es el único punto de entrada al mismo. Determina la acción a ejecutarse atendiendo a las peticiones de los Sistemas Clientes que consumen el SW y maneja la seguridad del sistema.

action.class.php: Este componente contiene la lógica de la aplicación, verifica la integridad de las peticiones, se efectúan solicitudes (al modelo) de realización de operaciones y prepara los datos requeridos por los Sistemas Clientes.

Acceso a Datos

ManipuladorRepositorio.php: Es el componente que contendrá la lógica del negocio del servicio web. Posee la lógica de intercambio de información con los repositorios de fotos (Repositorio de Fotos y Repositorio de Expedidos).

Vista de Implementación del Módulo "Aplicación Administrador del Servicio Web"

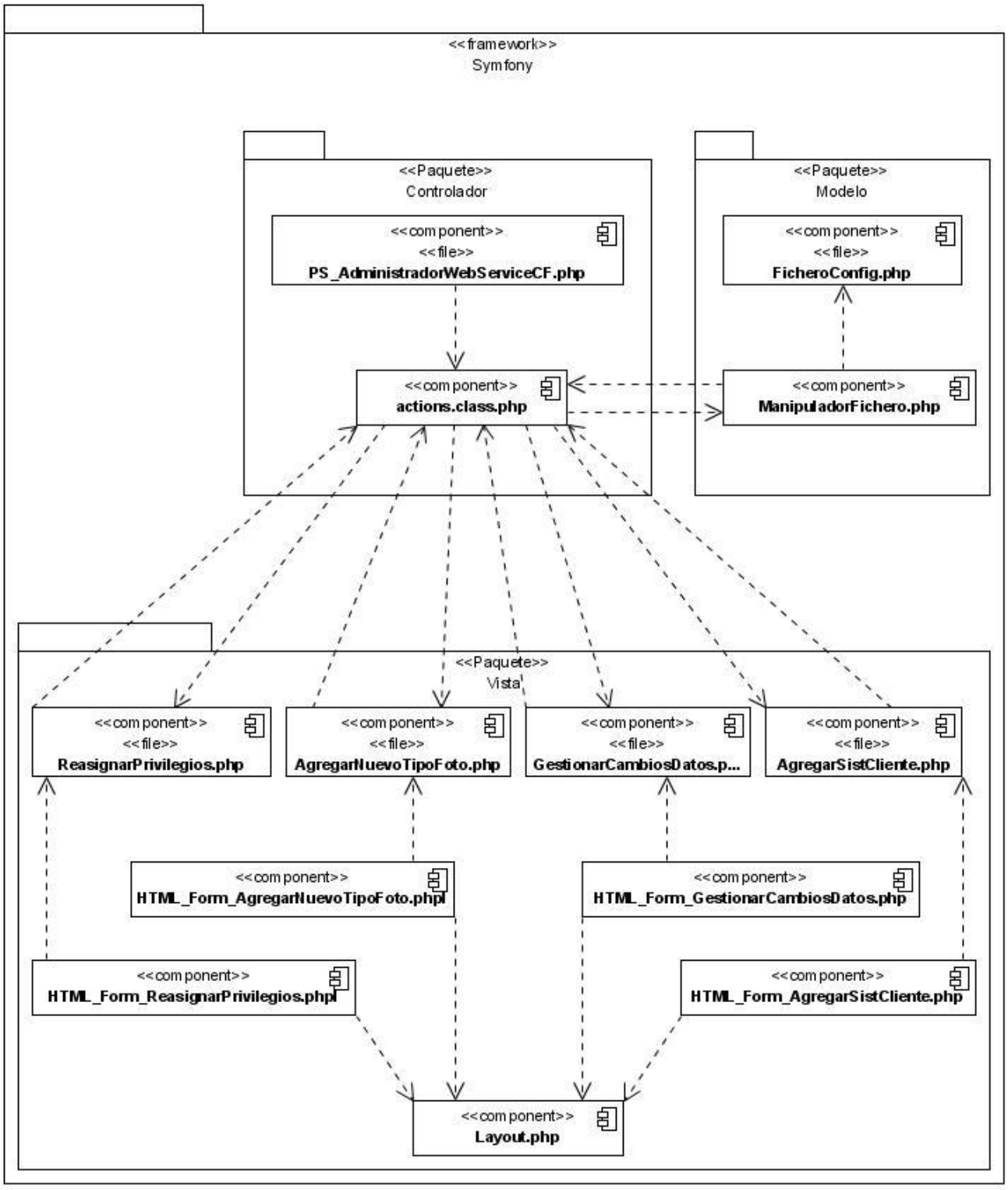

Figura 12 Vista de Implementación del Módulo "Aplicación Administrador del Servicio Web"

Modelo

FicheroConfig.php: En este componente estarán las configuraciones de lo que podrá hacer el SW. Aparecerán cuales Sistemas Clientes tienen acceso a consumir el SW y sus permisos. Además se encontrarán definidos los diferentes tipos de fotos y los datos que deberán poseer cada uno de ellos.

ManipuladorFichero.php: Es el componente que contendrá la lógica del negocio de la aplicación web. Posee la lógica de intercambio de información con el fichero de configuración de la aplicación.

Controlador

PS_AdministradorWebServiceCF.php: Es el componente del controlador frontal y constituye el único punto de entrada a la aplicación. Carga la configuración, atiende las peticiones del usuario, determina la acción a ejecutarse y maneja la seguridad del sistema.

Actions.class.php: Este componente contiene la lógica de la aplicación, verifica la integridad de las peticiones y prepara los datos requeridos por la vista.

Vista

AgregarSistCliente.php: Es el componente que contiene la parte construida del formulario para agregar un nuevo Sistema Cliente para que consuma el SW.

ReasignarPrivilegios.php: Es el componente que contiene la parte construida del formulario para asignar un nuevo permiso a un Sistema Cliente previamente autorizado a consumir el SW.

AgregarNuevoTipoFoto.php: Es el componente que contiene la parte construida del formulario para agregar un nuevo tipo de foto a ser procesada por el SW.

GestionarCambioDatos.php: Es el componente que contiene la parte construida del formulario para gestionar algún cambio en los datos de un tipo de foto anteriormente adicionado.

Layout.php: Es el componente de la plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada una de ellas.

HTML_Form_AgregarSistCliente.php: Es el componente que contiene el resultado de la unión entre los componentes *AgregarSistCliente.php* y *Layout.php*.

HTML_Form_ReasignarPrivilegios.php: Es el componente que contiene el resultado de la unión entre los componentes *ReasignarPrivilegios.php* y *Layout.*php.

HTML_Form_AgregarNuevoTipoFoto.php: Es el componente que contiene el resultado de la unión entre los componentes AgregarNuevoTipoFoto.php y *Layout*.php.

HTML_Form_GestionarCambioDatos.php: Es el componente que contiene el resultado de la unión entre los componentes GestionarCambioDatos.php y *Layout*.php.

3.6 Vista de Despliegue

En esta vista se realiza una representación gráfica de los nodos físicos en los que estará desplegado el sistema propuesto y la comunicación entre ellos; además se brinda una explicación de en que consiste cada uno de ellos.

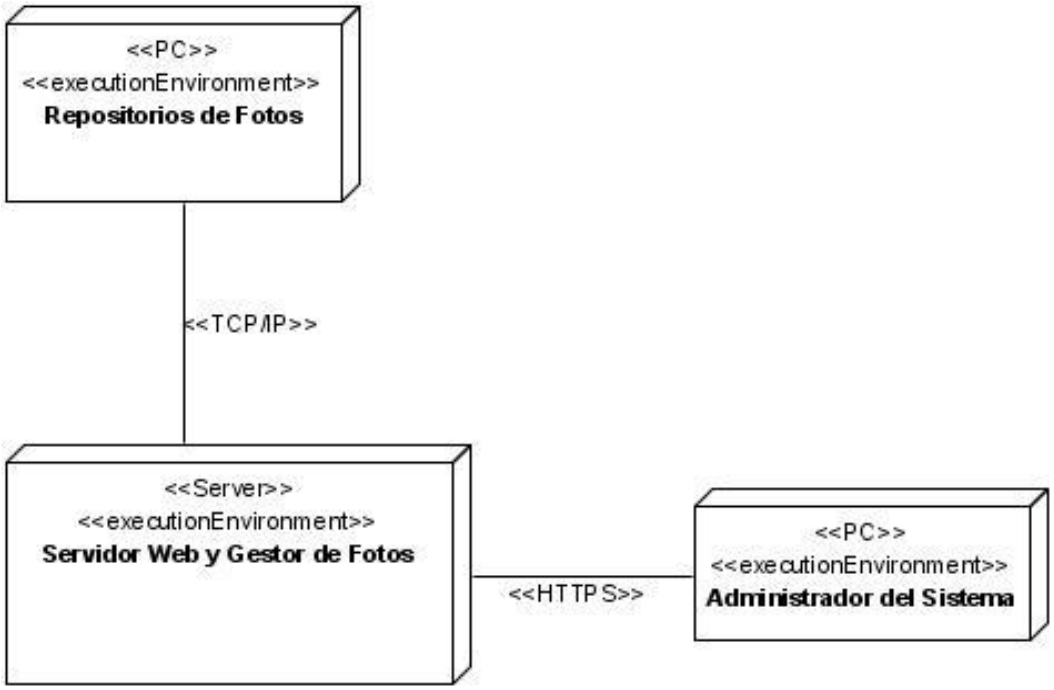

Figura 13 Vista de Despliegue

PC “Repositorio de Fotos”: En este ordenador se encontrará alojado el repositorio de fotos, en el cual se guardarán de forma ordenada en subcarpetas, las fotos (principales o de seguridad) con los metadatos dentro. Además estará también el Repositorio de Expedidos, donde serán guardadas las fotos que hayan sido eliminadas por causa de baja del centro, de alguna persona.

Server “Servidor Web y Gestor de Fotos”: En este servidor estará alojada la aplicación web que brindará la posibilidad de gestionar cambios en el sistema (Adicionar un nuevo tipo de foto, cambiar datos de los estándares de las fotos, adicionar un nuevo Sistema Cliente y

efectuar cambio de permisos para un sistema existente) desde la PC del Administrador del Sistema. Además contendrá el servicio web “Gestor de Fotos” que será consumido por los sistemas clientes existentes en la UCI, previamente autorizados.

PC “Administrador del sistema”: Es la PC mediante la cual el Administrador del Sistema puede conectarse a la aplicación web para configurar el servicio web: Gestor de Fotos.

3.7 Conclusiones

En el presente capítulo se plantearon las metas y restricciones arquitectónicas, de forma tal que se establece lo que debe tener el futuro sistema para que pueda funcionar de forma óptima. Además se obtienen las vistas de la arquitectura (Vista de Casos de Uso Arquitectónicamente significativos, Vista Lógica, Vista de Implementación y Vista de Despliegue) de forma detallada.

CONCLUSIONES GENERALES

El desarrollo de la arquitectura de software es una de las etapas más importantes en el proceso de desarrollo de software, pues es aquí donde los profesionales aportan todos sus conocimientos, creatividad y experiencia para formular la mejor propuesta de solución que se dará al cliente, de forma tal que se pueda organizar el desarrollo del trabajo, fomentar la reutilización y hacer evolucionar el sistema.

Luego de culminada la realización del presente trabajo y analizado todo el contenido expuesto anteriormente el él, se arriban a las siguientes conclusiones:

- Se realizó un estudio profundo acerca de: patrones (arquitectónicos y de diseño), estilos arquitectónicos, herramientas, lenguajes, tecnologías y metodologías, que existen en la actualidad bajo la política de software libre, con el objetivo efectuar una selección de los más ajustados al presente trabajo y hacer uso de las mejores técnicas de diseño arquitectónico. Con tales prácticas se elaboró una propuesta que ha permitido diseñar y documentar, de forma correcta, la arquitectura del sistema que se desea construir.

- Se propuso una arquitectura de dos capas para el diseño del servicio web: “Gestor de Fotos.

- Se propuso una arquitectura basada en el patrón arquitectónico: “Modelo Vista Controlador” para la Aplicación Administrador Servicio Web.

- Se dio cumplimiento al objetivo general trazado, de modo que se obtuvieron todos los artefactos necesarios.

- Existe gran satisfacción por parte del cliente.

RECOMENDACIONES

De modo general, el diseño arquitectónico propuesto, describe toda una serie de aspectos significativos referentes al futuro sistema informático para la gestión de fotos en la UCI. No obstante, como parte de un proceso de mejoras continuas, se proponen las siguientes recomendaciones:

- Aplicar los métodos de evaluación de la arquitectura para identificar las posibles debilidades, tanto en la descripción arquitectónica como en el propio diseño.

- Mantener un constante refinamiento de la arquitectura a lo largo del ciclo de desarrollo del futuro sistema de software.

- Realizar una evaluación de los costos de la tecnología que será utilizada (principalmente del hardware que la soportan) a pesar de que en su totalidad sea libre.

REFERENCIAS BIBLIOGRÁFICAS

1. RODRÍGUEZ, E. M. M. *RDF: Un modelo de metadatos flexible para las bibliotecas digitales del próximo milenio. .* Disponible en: http://www.bib.uc3m.es/~mendez/publicaciones/7jc99/rdf.htm.

2. *Published Software Architecture Definitions.* Disponible en: http://www.sei.cmu.edu/architecture/published_definitions.html.

3. REGALADO, I. Y. V. y CABANA, I. E. F. *La arquitectura de software como disciplina científica.* 2008, Disponible en: http://www.gestiopolis.com/administracion-estrategia/arquitectura-de-sofware-como-disciplina-cientifica.htm.

4. *Patrones arquitectónicos.* Disponible en: http://isg3.pbwiki.com/Patrones%20Arquitect%C3%B3nicos.

5. GRACIA, J. *Desarrollo de Software Orientado a Objetos.* 2005, Disponible en: http://www.ingenierosoftware.com/analisisydiseno/uml.php

6. *La Ingeniería de Software y RUP* 2007, Disponible en: http://www.slideshare.net/dersteppenwolf/la-ingeniera-de-software-y-rup

7. *Entorno de desarrollo integrado.* 2007, Disponible en: http://www.babylon.com/definition/entorno_de_desarrollo/Spanish

8. *Servidor WEB.* Disponible en: http://www.pergaminovirtual.com.ar/definicion/Servidor.html.

9. *Introducción a los Web Services en PHP* 2005, Disponible en: http://www.desarrolloweb.com/articulos/1852.php.

BIBLIOGRAFÍA

1. Patrón "Modelo-Vista-Controlador". Disponible en: http://www.proactiva-calidad.com/java/patrones/mvc.html.

2. PERL. Disponible en: http://www.wikilearning.com/buscador.php?txtPalClave=Perl&tipo.

3. El Servidor Web Disponible en: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.

4. WEBSERVICES.ORG. publicado el: mayo, 2008 de última actualización: mayo, 2008. Disponible en: http://webservices.org/.

5. Textos Científicos. Disponible en: http://www.textoscientificos.com/redes/redes-virtuales/tuneles/encriptacion.

6. Herramientas WEB para la enseñanza de PROTOCOLOS DE COMUNICACIÓN. Disponible en: http://neo.lcc.uma.es/evirtual/cdd/tutorial/presentacion/rsa.html

7. Desarrollo web. Disponible en: http://www.desarrolloWeb.com/articulos/1557.php.

8. Lenguaje de Programación. 2003, Disponible en: http://www.lenguajes-de-programacion.com

9. Symfony plugins 2007, Disponible en: http://trac.symfony-project.com/wiki/SymfonyPlugins.

10. wikipedia. 2008, Disponible en: http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado.

11. Visual Paradigm. 2008, Disponible en: http://www.visual-paradigm.com/.

12. Slideshare. 2008, Disponible en: http://www.slideshare.net/vivi_jocadi/rational-rose/.

13. epistemoWIKIA. 2008, Disponible en: http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=Eclipse_SDK.

14. ACHOUR, M.; BETZ, F., et al. Manual de PHP. 2006.

15. CASTILLO, Á. D. Webs dinámicos con PHP. 2003, Disponible en: http://www.programacion.net/php/tutorial/php4/2/.

16. DIJKSTRA, E. “The Structure of the The Multiprogramming System”. Communications of the ACM. 1983.

17. ESPAÑOLA, O. WORD WIDE WEB consortium. 2008, Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

18. ---. Ingeniero Software. 2005, Disponible en: http://www.ingenierosoftware.com/analisisydiseno/uml.php.

19. KRUCHTEN, P. Wikipedia. 2008, Disponible en: http://es.wikipedia.org/wiki/Arquitectura_software.

20. MASON, M. Osmosis Latina. 2005, Disponible en: http://www.osmosislatina.com/subversion/basico.htm.

21. PESSMAN. El proceso Unificado del Rational. 2005.

22. PRESSMAN. El Proceso Unificado del Rational II 2005.

23. REYNOSO, B. Introducción a la Arquitectura de Software Buenos Aires:

24. SARWAT, R. La nueva oleada de algoritmos criptográficos. Disponible en: http://www.revista-ays.com/DocsNum10/PersEmpresarial/sarwat.pdf

25. TORRES, J. F. Sistema criptográfico de llave publica RSA. Análisis y diseño de algoritmos. Disponible en: http://www.ing.ula.ve/~ibc/ayda/c26rsa.pdf.

26. WHITE, S. y LEMUS-OLALDE, C. “The software architecture process”.

GLOSARIO DE TÉRMINOS

Sistema Operativo: Es un conjunto de programas destinados a permitir la comunicación entre un usuario y un computador para gestionar sus recursos de una forma eficaz.

IBM (**I**nternational **B**usiness **M**achines): Es una empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática.

Hardware (soporte físico): Se utiliza generalmente para describir los artefactos físicos de una tecnología. Es el conjunto de elementos físicos que componen una computadora Disco Duro, CD-ROM, etc.

ADL: Lenguaje de Descripción Arquitectónico.

Software (soporte lógico): Los componentes intangibles de una computadora, es decir, el conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica.

Informática: Es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales. Es la unión sinérgica del cómputo y las comunicaciones.

UTP (del inglés **U**nshielded **T**wisted **P**air, par trenzado no apantallado): es un tipo de conductor utilizado, principalmente para comunicaciones.

RAM (**R**andom **A**ccess **M**emory): Memoria de acceso aleatorio ó memoria de acceso directo.

TCP/IP: Es un conjunto de protocolos de red que permiten la transmisión de datos entre redes de computadoras.

Estereotipo: Define un nuevo significado de la semántica para el elemento a modelar.

Cliente: Es un ordenador que accede a recursos y servicios brindados por otro llamado Servidor, generalmente en forma remota.

Servidor: Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

SEI (**S**oftware **E**ngineering **I**nstitute) es un instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso Estadounidense en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército estadounidense la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Financiado por el Departamento de Defensa estadounidense y administrado por la Universidad Carnegie Mellon.

IEEE: El Instituto de Ingenieros Eléctricos y Electrónicos es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol): protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

HTTPS: Versión segura del protocolo HTTP, utiliza un cifrado basado en SSL.

Protocolo: Conjunto de normas y procedimientos útiles para la transmisión de datos, conocido por el emisor y el receptor.

WWW (**W**orld **W**ide **W**eb): es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes u otros contenidos multimedia.

Smalltalk: Es un sistema informático que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales. Metafóricamente, se puede considerar que un Smalltalk es un mundo virtual donde viven objetos que se comunican mediante el envío de mensajes.

Programación Orientada a Objetos (POO): Es un paradigma de programación que define los programas en términos de clases de objetos, que son entidades que combinan *estado* (datos), *comportamiento* (procedimientos o métodos) e *identidad* (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto difiere de los lenguajes procedurales tradicionales, en los que los datos y los procedimientos están separados y sin relación. Estos métodos están pensados para hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

Navegador web: También conocido como Hojeador o web browser, es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet. Esta red de documentos es denominada World Wide Web (WWW) o Telaraña Mundial. Los navegadores actuales permiten mostrar y/o ejecutar: gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces.

Plugin: C*omponente enchufable* (o **plug-in** en inglés "enchufar", también conocido como addin, add-in, addon o add-on) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

CASE (Computer-Aided Systems Engineering): Es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de sistemas.

SOAP: Protocolo de Acceso Simple a Objetos, es un protocolo estándar define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Bajo acoplamiento: Dependencia mínima entre componentes del sistema.

API's: Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

Credencial: Documento que sirve para que a un empleado se le reconozca como perteneciente a su centro de trabajo y se reconozca su plaza.

Metodología: En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

Framework: Estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado.

Repositorio de Fotos: Es la carpeta donde se guardarán todas las fotos de las personas pertenecientes a la Universidad de las Ciencias Informáticas (UCI).

Repositorio de Expedidos: Es la carpeta donde se guardarán todas las fotos de las personas que han causado baja de la UCI.

URI (**U**niform **R**esource **L**ocator*):* Es un localizador uniforme de recursos. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.