UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 1

ARQUITECTURA DEL PROYECTO IDENTIFICACIÓN Y CONTROL DE ACCESO.

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

AUTOR: *Adrian Carmona Luis.*

TUTOR: *Ing. Roberlán Rodríguez Sánchez.*

ASESOR: *Manuel Alejandro Gil Martin.*

Junio, 2008

"Año 50 de la Revolución"

“La actitud frente a la vida es mostrar con el ejemplo el camino a seguir, el llevar a las masas con el propio ejemplo cualesquiera que sean las dificultades a vencer en el camino, quien pueda mostrar el ejemplo de su trabajo repetido durante días sin esperar de la sociedad otra cosa que el reconocimiento a sus méritos, de constructor de esta nueva sociedad, tiene derecho a exigir a la hora del sacrificio.”

Che.

Agradecimientos

Agradezco este trabajo a:

En especial a mi mamá y a mi abuelito por guiarme siempre por el buen camino y confiar siempre en mí.

A mi hermana, por haber podido contar siempre con ella.

A mi familia, por haberme apoyado en todo momento.

A mi novia por estar siempre a mi lado.

A mis amigos, Rodolfo Venero Noriega, Gabriel La O Ramírez, Leonel Columbié, Rolando Barrientos, Yadián García, Yanara Ramil, Misladys Vázquez, Osniel Calvo, Juan José, por todos los momentos, buenos y malos, que hemos compartido juntos, por ayudarme y comprenderme siempre.

A Yoney González Rodríguez, Ivón Leyva García y Rosalida Moreno por su ayuda incondicional en la realización de este trabajo.

A Ariana Ramírez, Lisandra Proenza, María Elena Hernández y Sahily Monferrer por apoyarme en todo momento.

Colectivos:

A Osmany Alonso Guerra y Manuel Alejandro Gil Martin por apoyarme en la realización de este trabajo.
A mi tutor Roberlán Rodríguez Sánchez por guiarme durante el desarrollo de esta investigación.
A todos los profesores que de una forma u otra han contribuido a nuestra formación.

Declaración de Autoría

Declaro ser autor de la presente tesis y autorizo a la Facultad 1 y a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, a hacer uso del mismo en su beneficio.

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

Adrian Carmona Luis
_____
Firma del Autor

Ing. Roberlán Rodríguez Sánchez.
_____
Firma del Autor

Resumen

Las plataformas en software libre se imponen, cada vez son más los proyectos que se desarrollan sobre estas en nuestro país, teniendo gran aceptación dentro de la comunidad informática debido a las grandes ventajas que trae consigo su desarrollo; especialmente en la Universidad de las Ciencias Informáticas (UCI), permitiéndose incorporar sistemas dentro del mercado de software libre.

En el presente trabajo se definirá la arquitectura del proyecto Identificación y Control de Acceso de la UCI, así como integrar y organizar los diferentes sistemas que lo conforman para obtener, como objetivo fundamental, una plena seguridad de los medios materiales y de la comunidad que pertenece al centro.

Durante el desarrollo del trabajo se realizan diferentes actividades que tributan a la construcción de la arquitectura del proyecto Identificación y Control de Acceso de la UCI como: el análisis de las principales tendencias, herramientas, tecnologías y metodologías actuales; la elaboración de un dominio específico que enmarcará las aplicaciones creadas en el proyecto; la creación de una arquitectura que se usará como guía para el desarrollo de las diferentes aplicaciones, así como una breve descripción de cada módulo, logrando la integración de todos los sistemas pertenecientes al proyecto Identificación y Control de Acceso de la UCI y desarrollados sobre plataforma de software libre.

Tabla de contenido

Índice de Figuras

Introducción

En la Universidad de las Ciencias Informáticas (UCI), creada al calor de la Batalla de ideas, conviven más de 15 000 personas de diferentes rincones del país entre profesores, familiares, estudiantes, directivos, técnicos, entre otros y cuenta con una alta suma de medios materiales de gran valor en el mercado mundial. Esta es la razón principal por la cual, la dirección del centro tiene la necesidad de crear un sistema automatizado que tenga como objetivo fundamental lograr una plena seguridad y protección tanto de la comunidad universitaria como de los medios materiales. De ahí surge el Proyecto Identificación y Control de Acceso en la UCI, el cual ha sido por casi cinco años el proyecto que ha dado respuesta ante los cambios y necesidades de los aspectos de seguridad y protección, brindando una serie de servicios de forma activa a la comunidad de la UCI.

Actualmente el Proyecto Identificación y Control de Acceso brinda una serie de importantes servicios a la UCI en general, sus requerimientos han aumentado en números por necesidades del centro y el sistema no puede darle soluciones debido a que no están incorporados dentro este. Además se debe tener en cuenta las nuevas políticas del país a incorporarse al mercado de software libre, la dirección de la Universidad de Ciencias Informáticas decidió que el proyecto se rigiera por las nuevas políticas y se desarrollará en software libre. La actual arquitectura del proyecto Identificación y Control de Acceso de la UCI no se ajustan a los nuevos requerimientos ni a las políticas de software libre incorporadas por el país.

Todos estos aspectos ocasionan que en el proyecto surjan una series de cambios estructurales, los cuales se deben a nuevas concepciones y entornos de desarrollos que van desde la base hasta los niveles más superiores; estos cambios traen como consecuencia redefinir totalmente el proyecto, especialmente en su arquitectura base, de ahí surge la razón fundamental de este trabajo, que es la definición de una nueva arquitectura para el proyecto Identificación y Control de Acceso de la UCI.

¿Qué es la Arquitectura de Software?

La arquitectura de software, denominada también 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, esta establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información y además define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura de software debe ser implementada en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea de computación.

De esta situación surge el siguiente **problema científico:**

¿Qué arquitectura basada en software libre utilizar en el proyecto Identificación y Control de Acceso para lograr un sistema ajustado a sus nuevos requerimientos?

Teniendo en cuenta el problema planteado se define como **objeto de estudio:** El proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI).

Por consiguiente, el **campo de acción** está dirigido a: la Arquitectura del Proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas.

Dada las condiciones se plantea como **objetivo general** de la investigación:

Definir la nueva arquitectura del proyecto Identificación y Control de Acceso de la UCI, teniendo en cuenta su desarrollo en software libre y los nuevos requerimientos.

Después de obtener toda la información relacionada con el tema del trabajo para guiar la investigación se plantea la siguiente **idea a defender:**

Definiendo la arquitectura del proyecto Identificación y Control de Acceso de la UCI es posible obtener un sistema desarrollado en software libre y que cumpla con los nuevos requerimientos funcionales.

Según los objetivos planteados para la investigación se define el siguiente conjunto **de tareas de investigación para lograr su cumplimiento:**

- Realizar un estudio de la actual arquitectura del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI).

- Analizar las arquitecturas que existen dentro del mercado de software libre.

- Definir la arquitectura principal del proyecto Identificación y Control de Acceso de la (UCI).

Métodos de Investigación Científica Empleados:

Métodos Teóricos:

- **Histórico lógico:** Nos permite estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo.

Su objetivo en la investigación es el estudio de cómo ha evolucionado y se ha desarrollado la arquitectura desde su surgimiento hasta la actualidad, en cuanto a sus herramientas, procesos, formas de trabajos, plataformas, etc.

- **Modelación:** Es justamente el método mediante el cual se crean abstracciones con vistas a explicar la realidad. El modelo como sustituto del objeto de investigación. En el modelo se revela la unidad de los objetivos y lo subjetivo.

Se modela la arquitectura con vista a un mejor entendimiento de las tareas y los objetivos a cumplir por sus desarrolladores en los diferentes flujos de trabajos del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas.

Métodos Empíricos:

- **Entrevistas:** Con el fin de precisar el problema a resolver, tener una mejor idea de las herramientas y de la arquitectura que se definirá para el proyecto.

El presente trabajo está estructurado en tres capítulos:

- Capítulo 1. Fundamentación teórica, se realiza un estudio detallado de la arquitectura de proyectos en ámbito nacional e internacional, herramientas, tecnologías y metodologías actuales basadas en software libre, con el fin de definir las más adecuadas para la solución del problema.

- Capítulo 2. Dominio y arquitectura, se define el dominio que enmarcará las aplicaciones, los requerimientos de referencias de las aplicaciones y la arquitectura base del proyecto Identificación y Control de Acceso de la UCI. Se desglosa la arquitectura base en: diseño de las capas lógicas, convenciones de nombres y estándares de códigos de las diferentes tecnologías a usar, estructura de las aplicaciones y los mecanismos de colaboración que pudieran existir entre los diferentes módulos.

- Capítulo 3. Subsistemas y Módulos, se definen los subsistemas pertenecientes al proyecto Identificación y Control de Acceso de la UCI y los módulos que representa cada subsistema y se realiza una breve descripción de cada uno de ellos. Se dará a conocer como se realizará la comunicación entre los módulos y los subsistemas. También se brindará una breve descripción de un algoritmo de encriptación para proteger la información de los subsistemas.

Capítulo 1. Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza un profundo estudio del marco teórico sobre las arquitecturas que se utilizan para desarrollar aplicaciones web sobre tecnología PHP y de las aplicaciones de escritorios sobre tecnología Java, partiendo de un estado del arte de diferentes proyectos a nivel nacional e internacional desarrollados en una arquitectura de software libre, logrando su entendimiento. Se caracterizan las herramientas y tecnologías que se seleccionaron a partir de la propuesta de la Dirección de Informatización para ser utilizadas en la construcción de las diferentes aplicaciones.

1.2 Estado del Arte

El estado del arte, es uno de los primeros temas que se desarrollan en la investigación. Su estudio permite al autor dirigirse al tema de investigación: la propuesta de una arquitectura basada en software libre. En el mismo se brinda una breve descripción del tema en el ámbito nacional e internacional de algunos proyectos.

1.2.1 Ámbito Internacional

Fisterra 2 es una nueva versión de Fisterra, la cual constituye una evolución en las aplicaciones de gestión empresarial en el ámbito del software libre, su objetivo principal es construir una solución más general y flexible que sea adaptable de forma sencilla a nuevos negocios. Esta nueva versión incorpora la tecnología GNOME2, arquitectura multicapa para su implementación y proporciona un framework genérico y un conjunto de ejemplos de aplicaciones para facilitar el desarrollo y adaptación de nuevos módulos sectoriales sin partir de cero.(1)

El diseño de Fisterra 2 se orienta hacia una arquitectura cliente-servidor, establece una separación entre cliente y servidor. El cliente tiene como funciones la interacción con el

usuario final mediante la interfaz gráfica, su implementación es totalmente independiente de la del servidor siempre que cumpla las interfaces de comunicación. El servidor es el encargado de gestionar el proceso de negocio de la aplicación y realizar las operaciones enviadas por los clientes.(1)

La nueva versión de Fisterra utiliza como base el conjunto de tecnologías que conforman el Gnome SDK, GTK (Librería que proporciona los widgets y controles) y Lib-Glade (Librería para la descripción visual de interfaces de usuario, y el uso de estas descripciones en aplicaciones) para las interfaces de los clientes, GObject (Sistema que permite realizar orientación a objetos en C, usado por el proyecto GNOME), libXML y libXSLT(Librerías que implementan operaciones para manipulación de ficheros XML, y transformaciones XSL) para intercambio y manipulación de la documentación y LibGDA (Librería de acceso a bases de datos del proyecto Gnome) y PostgreSql (Gestor de bases de datos relacional) para la persistencia.(1)

1.2.2 Ámbito Nacional

SIGEP (Sistema de Gestión Penitenciaria) es un sistema informático integral cubano creado en software libre donde tiene como propósito gestionar las actividades penitenciarias, abarcar las áreas de control penal, clasificación y tratamiento, salud integral, custodia, control logístico, administración de la Dirección General y la gestión de sus unidades de apoyo así como el acompañamiento post penitenciario.

Dentro de su arquitectura utilizan como ambiente de desarrollo, plataforma Java; Tomcat como contenedor Web; Subversion como controlador de versiones; como base de datos Oracle y como herramientas de modelado al Visual Paradigm y al Erwin. El eclipse como entorno integrado de desarrollo y como framework utiliza el Spring, JUnit, Aspecto, JfreeChart, entre otros.

Presenta una arquitectura multicapa donde se encuentran la capa de Acceso a Datos que contiene un lenguaje de consultas de alto nivel, una arquitectura de caché (Objetos y Consultas), herramientas de desarrollo, ficheros de mapeos flexibles e intuitivos y soporte para diferentes tipos de pruebas. La capa de Servicios de Negocio es muy sencilla en el proceso de desarrollo, mantenimiento, integridad de aplicaciones e interrelación con otros sistemas. Por último está la capa de Presentación la cual contiene Ajax, que esta a su vez consiste en HTML, tecnología Javascript, DHTML y DOM; todas estas tecnologías trabajan juntas para hacer extremadamente eficiente el desarrollo Web obteniendo una transformación de las pasadas interfaces Web en interactivas aplicaciones Ajax.

Se utiliza el patrón Modelo Vista Controlador (MVC); es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. (2)

En los aspectos de seguridad utilizan la autenticación del usuario mediante el usuario, contraseña, dirección IP del equipo y la verificación humana. La seguridad se define y se maneja independiente de la aplicación, se utiliza también canales seguros y seguridad en los servicios.

1.3 Tendencias, tecnologías y metodologías actuales

1.3.1 Metodologías de desarrollo de software

En un proyecto de desarrollo de software la metodología define quién, qué, cuándo y cómo debe hacerlo. No existe una metodología de software universal. Cada equipo de desarrollo escoge la metodología según las características de su proyecto, por lo que es importante determinar el alcance del proyecto antes de escoger la metodología que se va a usar en el desarrollo del mismo. A continuación se muestra una de las metodologías de desarrollo, dando una explicación acerca de la misma.

Rational Unified Process (RUP)

RUP (Rational Unified Process, por sus siglas en inglés) es un proceso de desarrollo de software y junto con UML (Lenguaje de Modelado Unificado, por sus siglas en inglés) constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos; tanto para sistemas tradicionales como para sistemas Web.

Además no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. RUP divide su ciclo de desarrollo en cuatro fases y ha agrupado sus actividades en 9 flujos de trabajo.

Fases.

- **Inicio**: Esta fase tiene como objetivo determinar la visión del proyecto.
- **Elaboración**: En esta etapa el objetivo es definir la arquitectura del sistema.
- **Construcción**: En esta etapa el objetivo es llegar a obtener un producto listo para su utilización que está documentado y tiene un manual de usuario.
- **Transición**: El objetivo es llegar a obtener el reléase ya listo para su instalación. Puede implicar reparación de errores.

Flujos de Trabajo.

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

- **Prueba (Testeo):** Busca los defectos a lo largo del ciclo de vida.

- **Instalación:** Produce reléase del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportara el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Cada flujo de trabajo tiene como resultado un modelo propuesto por RUP:

- Modelo de Casos de Uso del Negocio.

- Modelo de Objetos del Negocio.
- Modelo de Casos de Uso.
- Modelo de Diseño.
- Modelo de Despliegue.
- Modelo de Datos.
- Modelo de Implementación.
- Modelo de Pruebas.

El modelo de casos de uso, el modelo de diseño, el modelo de despliegue y el modelo de implementación son los modelos más importantes para describir la arquitectura de un sistema teniendo en cuenta que son los que proporcionan el desarrollo de las vistas de arquitectura.

El ciclo de vida de RUP tiene tres características fundamentales:

Guiado por casos de uso.

Los casos de uso reflejan las necesidades de los futuros usuarios, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. Los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura.

La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura (casos de uso arquitectónicamente significativo). El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e incremental.

RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos.

El uso de esta metodología asegura que se produzca desde sus primeras fases de desarrollo, un producto de calidad que cumpla con las características de funcionalidad, usabilidad y fiabilidad.

1.3.2 Tecnologías

Java

Java es un lenguaje de programación orientado a objetos de alto nivel, gran rendimiento, sencillo, gran nivel de seguridad, multiplataforma y contiene las herramientas necesarias para desarrollar cualquier tipo de aplicación. En la actualidad se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, ordenadores y redes de cualquier tecnología de programación, por su versatilidad y eficiencia, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes, de manera que hoy en día, más de 2.500 millones de dispositivos utilizan dicha tecnología como son; ordenadores, teléfonos móviles, tarjetas inteligentes, impresoras, Web cams, juegos, sistemas de navegación para automóviles, terminales de lotería, dispositivos médicos, cajeros de pago en aparcamientos, etc.(3)

La tecnología Java a pesar de ser madura y extremadamente eficaz se ha convertido en un recurso valioso ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo prácticamente en cualquier otra plataforma.

- Crear programas para que funcionen en un navegador Web y en servicios Web.

- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.

- Combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizadas.

- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier dispositivo digital.

Es un lenguaje independiente de la plataforma, lo que quiere decir q sus programas creados en este lenguaje podrán funcionar en cualquier ordenador del mercado. Con Java podemos programar páginas Web dinámicas, con accesos a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema; cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas.(3)

PHP

Es el acrónimo de Hipertext Preprocesor, lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, de código abierto, ejecutado al lado del servidor y soportado por la mayoría de los servidores Web de hoy en día, por ejemplo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd entre otros, con gran volumen de documentación y con una gran librería de funciones con un espíritu generoso ya que estas

funciones son progresivamente construidas por colaboradores desinteresados en nuevas versiones del lenguaje. PHP nos permite embeber sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas programados íntegramente en un lenguaje distinto al HTML. Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones Web con páginas dinámicas gracias a la extensa librería de funciones con la que está dotado; la cual cubre desde cálculos complejos hasta tratamiento de conexiones de red. Algunas de las más importantes capacidades de PHP es la compatibilidad con las bases de datos más comunes, como MySQL, Oracle, Informix, y ODBC. Esta tecnología aunque sea multiplataforma, ha sido concebida inicialmente para entornos UNIX y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones.

CSS

Las hojas de estilo en cascada (*Cascading Style Sheets*, CSS) representan un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.(4)

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.

- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.

- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser leída por un sintetizador de voz.

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

1.3.3 Lenguajes

HTML

El lenguaje HTML, cuyas siglas significan Lenguaje de Formato de Documento de Hipertexto (en inglés Hypertext Markup Language*), s*e utiliza para crear documentos que muestren una estructura de hipertexto. Un documento de hipertexto es aquel que contiene información cruzada con otros documentos, lo cual nos permite pasar de un documento al referenciado desde la misma aplicación con la que lo estamos visualizando.(5)

HTML permite, además, crear documentos de tipo multimedia, es decir, que contengan información más allá de la simplemente textual, como pueden ser imágenes, video o sonido. El lenguaje HTML no es el único lenguaje existente para crear documentos hipertexto. Hay otros lenguajes anteriores o posteriores a HTML (SGML, XML, etc.), aunque para muchos HTML se ha convertido en el lenguaje estándar para la creación de contenido para Internet.(5)

JavaScript

JavaScript es un lenguaje de programación interpretado, ampliamente utilizado en el mundo del desarrollo web por ser muy versátil y potente, tanto para la realización de pequeñas tareas como para la gestión de complejas aplicaciones, es ejecutado por el navegador que utilizamos para ver las páginas; lo que hace que podemos desarrollar aplicaciones de diversos tipos, desde generadores de HTML, comprobadores de formularios, páginas web dinámicas, intercambiar información entre páginas web en distintas ventanas, manipulación de gráficos, texto, programas que gestionan las capas de una página, etc. Pueden desarrollarse incluso aplicaciones que permitan poder tener capas en una página como si fueran ventanas, y dar la sensación de estar trabajando con una aplicación con interfaz de ventanas. JavaScript comparte muchos elementos con otros lenguajes de alto nivel. Hay que tener en cuenta que este lenguaje es muy semejante a otros como C, Java o PHP, tanto en su formato como en su sintaxis, aunque por supuesto tienen sus propias características definitorias.(6)

1.3.4 Estilos Arquitectónicos y Patrones

Desde sus inicios, en la arquitectura de software, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se las llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles o rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo.(7)

Existe una gran variedad de estilos arquitectónicos las cuales tienen diversas clasificaciones entre las que se encuentran:

- Estilos de flujo de datos.
- Estilos centrados en datos.
- Estilos de llamada y retorno.
- Estilos de código móvil.
- Estilos heterogéneos.
- Estilos peer-to-peer.

1.3.4.1 Arquitectura en Capas

Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. El estilo soporta un diseño basado en niveles de abstracción crecientes lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Permite realizar optimizaciones y refinamientos enfocando los cambios en un solo lugar. Proporciona amplia reutilización dada la división bien definida de responsabilidades. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.(7)

Una especialización muy usada de la arquitectura en capas es la arquitectura de tres capas donde se observan muy bien delimitadas las responsabilidades de cada funcionalidad en la aplicación.(7)

En la figura 1 se ejemplifica una arquitectura de tres capas, donde cada capa está muy bien delimitada de las demás. Una capa superior interactúa con una capa inferior mediante interfaces que definen las funcionalidades que la misma debe brindar. Las capas de la aplicación pueden residir tanto en el mismo nodo físico como en nodos separados.(7)

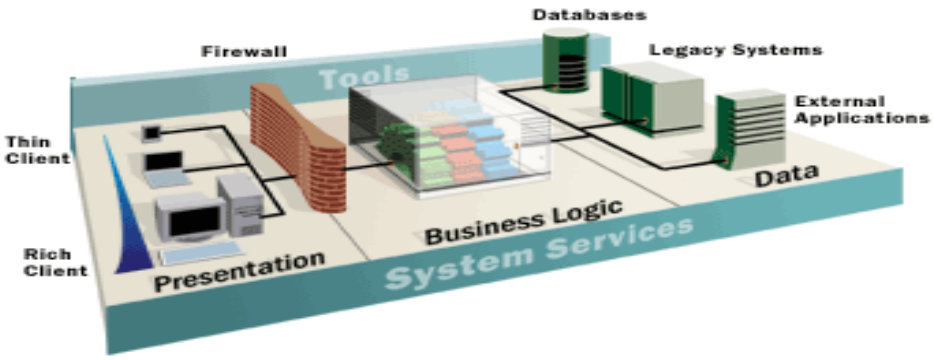

Figura 1.1 Arquitectura de tres Capas

1.3.4.2 MVC (Modelo-Vista-Controlador)

La arquitectura del patrón Modelo-Vista-Controlador es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica. El principal objetivo de este patrón es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador).

Las aplicaciones MVC se dividen en tres grandes áreas funcionales:

El principal objetivo de la arquitectura MVC es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador).

Las aplicaciones MVC se dividen en tres grandes áreas funcionales:

- **Vista:** La presentación de los datos.
- **Controlador:** El que atenderá las peticiones y componentes para la toma de decisiones de la aplicación.

- **Modelo:** La lógica del negocio o servicio y los datos asociados con la aplicación.

Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes.

1.3.4.3 Arquitectura Orientado a Servicios (SOA)

Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP (Protocolo Simple de Acceso a Datos, por sus siglas en inglés) y WSDL (Lenguaje de Descripción de los Servicios Web, por sus siglas en inglés) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

Existen cuatro elementos básicos para la construcción de una Arquitectura Orientada a Servicios:

- **Operación:** Es la unidad de trabajo o procesamiento en una arquitectura SOA.

- **Servicio:** Es un contenedor de lógica. Estará compuesto por un conjunto de operaciones, las cuales las ofrecerá a sus usuarios.

- **Mensaje:** Para poder ejecutar una determinada operación, es necesario un conjunto de datos de entrada. A su vez, una vez ejecutada la operación, esta devolverá un resultado. Los mensajes son los encargados de encapsular esos datos de entrada y de salida.

- **Proceso de negocio:** Son un conjunto de operaciones ejecutadas en una determinada secuencia (intercambiando mensajes entre ellas) con el objetivo de realizar una determinada tarea.

Por lo tanto, una aplicación SOA estará formada por un conjunto de procesos de negocio, que a su vez estarán compuestos por aquellos servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a un buen término. Estas operaciones se ejecutan mediante el envío de los datos necesarios a través de mensajes. El elemento básico de una arquitectura SOA es el servicio.(7)

La Arquitectura Orientada a Servicios propone el uso de los estándares como XML, SOAP, WSDL, etc. Aunque no siempre es necesaria la implementación de todos ellos, es muy recomendable que se utilicen todos.(7)

Estándar XML

El Lenguaje de Etiquetado Extensible (XML, por sus siglas en inglés) es un estándar de comunicación, es un metalenguaje con una importante función en el proceso de intercambio, estructuración y envío de datos en la Web.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y esas partes se componen a su vez de otras partes.

Protocolo SOAP

El protocolo simple de acceso a datos (SOAP), se encuentra en el núcleo de los servicios Web. Este protocolo proporciona un estándar para empaquetar mensajes, y 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.

SOAP fue creado por Microsoft, IBM y otras empresas, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Funciona sobre cualquier protocolo de Internet, generalmente HTTP, que es el único homologado por el W3C.

WSDL (Web Services Description Language)

Es un formato XML que se utiliza para describir servicios Web. WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo 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.

Un programa cliente que se conecta a un servicio Web puede leer el WSDL para determinar las funciones que están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

1.3.5 Ambiente de Desarrollo.

El ambiente de desarrollo (Development Environment) es algo imprescindible en la producción de software. Es donde se definen el conjunto de herramientas y tecnologías (frameworks), versiones a usar y su integración, que intervienen en un proceso de desarrollo de software.

A continuación se presentan un conjunto de herramientas utilizadas en el desarrollo de este trabajo; como son: tres ambientes de desarrollo integrado (IDE) y sus plugins, un contenedor web, un control de versiones, dos gestores de base de datos, herramientas de modelado y los frameworks. Se exponen sus características y ventajas para poder tener un mayor conocimiento de sus prestaciones y valorar su utilidad.

Un ambiente de desarrollo integrado o en inglés Integrated Development Environment (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Visual Basic, Object Pascal, etcétera.

En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk.

A continuación se caracterizan un conjunto de herramientas utilizadas en el ambiente de desarrollo integrado.

Eclipse 3.3

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 plugins esto nos permite que otras aplicaciones pueden ser integradas a eclipse en forma de plugins 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.(8)

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.(8)

Beneficios

- Es una herramienta de código abierto.
- Soporta la construcción de una variedad de herramientas para el desarrollo de aplicaciones.
- Soporta el desarrollo de aplicaciones basadas en GUI y non-GUI.
- Soporta herramientas que manipulan diferentes tipos de archivos como por ejemplo Java, C, PHP, C++, EJB, HTML, GIF, etc.
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.

- Provee a los desarrolladores, herramientas que facilitan la creación de plugins.

- Mediante JDT facilita la creación de aplicaciones programadas en Java.

Desventajas

- Si bien Eclipse es multiplataforma, los plugins no tienen por qué serlo.

- Existen plugins que sólo corren en una plataforma, que aún no han sido desarrollados para más de una.

- Al ser una herramienta de código abierto, se desarrollan plugins que no tienen todas las funcionalidades que tienen en otras herramientas comerciales.

NetBeans 6.0

Es una herramienta con entorno de desarrollo visual para el programador pensado para compilar, depurar, ejecutar y escribir programas fundamentalmente en tecnología java; uno de los lenguajes de programación más poderosos del momento; a pesar que puede servir para cualquier otro tipo de lenguaje de programación. Es un producto libre, de código abierto, multiplataforma y gratuito sin restricciones de uso y usa la plataforma de su propio nombre (**NetBeans**). Este soporta el desarrollo de todos los tipos de aplicaciones de Java; escritorio, web, dispositivos portátiles (móviles, Pocket PC) sin cambiar la forma de programar.

Las versiones mas resientes soportan el desarrollo de aplicaciones empresariales con Java EE 5; incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas de XML, orientación a web servicies y módulos UML. Tiene una arquitectura modular que permite que con posterioridad se le añadan complementos (plug-ins).

Se proporcionará a través de un único programa de instalación que permite configurar más rápidamente el entorno del desarrollo y que ya no requiere la instalación de múltiples paquetes para añadir capacidades al IDE.

1.3.5.1 Plugins

Un plugin es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible.

Subclipse

Es un plugin para Eclipse que adiciona integración para el control de versiones (Subversion, específicamente), permitiendo operaciones de sincronización, actualización, entre otras. Permite bloqueos a recursos para que otros usuarios no puedan modificarlo. Dispone de una vista de comparación entra el recurso local y remoto en caso que exista conflicto entre la versión del recurso local con el remoto. Muestra una vista del historial de versiones de los recursos con un conjunto de atributos de las acciones realizadas sobre el recurso.(7)

Soporta conectarse a varios repositorios de control de versiones al mismo tiempo, permitiendo hacer operaciones sobre el repositorio directamente.

Es un plugin muy útil para el desarrollo colaborativo, en el que intervienen un conjunto de desarrolladores trabajando sobre el mismo proyecto, poniendo a disposición del equipo de desarrollo facilidades para el trabajo en equipo.(7)

PDT (PHP Development Tools)

Es un plugin para Eclipse de código abierto el cual su objetivo es desarrollar y brindar soporte a los proyectos con tecnología PHP, es fácil de usar e intuitivo e integrable con Web Tools de

Eclipse, posee asistente y autocompletado, presenta soporte para el debug incremental del código PHP, presenta una integración con el modulo del proyecto de eclipse lo cual posibilita inspeccionar el uso de las vistas del contorno del fichero y del proyecto, así como la nueva vista PHP Explorer. También contiene un extenso frameworks y APIs que les permite con más facilidad a los desarrolladores extender este plugin para crear nuevas e interesantes herramientas orientadas al desarrollo PHP. Presenta una distribuido mediante un proyecto de código abierto con una extensa comunidad de desarrollador.

PDT nos permite utilizar Eclipse para integrar un gran número de herramientas con el objetivo de obtener un solo entorno de desarrollo, lo que nos posibilitará entregar servicios con gran calidad en menos tiempo.

Entre las diferentes características que ofrece PDT a Eclipse podemos enumerar las siguientes:

- Soporte de las versiones 4 y 5 de PHP indistintamente, ya sea bien de forma genérica a todos los proyectos que se generen o bien de forma individual a cada uno con previa especificación en las propiedades del proyecto en cuestión.

- Soporte completo del sistema de documentación PHPDoc, como característica clave la ayuda contextual a la hora de editar la documentación.

- Gestión y exploración de todas las clases generadas a lo largo de la edición del código o bien que se hayan importado de otra librería de PHP, con eso he de indicar que tales clases las toma Eclipse y pueden ser usadas en todo el proyecto como si fuera parte de la librería estándar de PHP.

- Informe de todos los fallos de sintaxis cometidos mientras se edita el código, aunque podemos modificar este comportamiento para que sea un poco más o menos estricto, pero este aspecto es mejor dejárselo al intérprete PHP.

- Por último comentar que PDT posee un formateado de código fuente, es decir, hay ocasiones en las cuales tenemos el código que no es nada legible, pues posee esta utilidad que nos facilitará un poco la vida formateando por nosotros el código para mayor legibilidad.

Hibernate Tools

Hibernate Tools es un conjunto de herramientas enteramente para trabajar con el framework Hibernate, implementado como una suite integrada de plugins para Eclipse.

Este plugin tiene las siguientes características disponibles

- **Editor de mapeos:** Es un editor para los archivos de mapeos XML de Hibernate, soportando auto completamiento y sintaxis resaltada. Soporta incluso auto completamiento semántico para nombres de clases, propiedades, tablas y columnas.(7)

- **Consola:** La perspectiva de consola de Hibernate permite configurar conexiones a base de datos, provee visualización de clases y sus relaciones. Admite además ejecutar preguntas en formato del lenguaje de preguntas de Hibernate (HQL) interactivamente contra la base de datos y mostrar los resultados.(7)

- **Ingeniería inversa:** Es su característica más importante, permitiendo generar las clases del modelo de dominio y los archivos de mapeos de Hibernate, los EJB3 entity beans anotados, documentación en formato HTML, a partir de una base de datos.(7)

- **Asistentes:** Presenta asistentes como: generador de archivos de configuración rápidamente, configuración de la consola, entre otros.(7)

- **Tareas de Ant:** El Hibernate Tools incluye una tarea de Ant que permite ejecutar la generación de esquemas, mapeos o código Java como parte de su construcción.(7)

Visual Editor para Java

El Visual Editor para Java es un editor de código-céntrica que nos brinda una ayuda para el diseño de aplicaciones que contienen una interfaz gráfica de usuario (GUI). Se basa en el modelo de componentes JavaBeans y apoya la construcción visual utilizando el Resumen Window Toolkit (AWT) o Swing. Permite editar el código fuente y el diseño simultáneamente. Puede ser utilizado no sólo como una herramienta para generar el código, sino como un editor para mostrar el efecto de las modificaciones de código fuente durante el desarrollo.

1.3.5.2 Contenedor Web

Los servidores de aplicaciones 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, Hypertext Transfer Protocol, protocolo de transmisión del hipertexto, el protocolo que sirve para incursionar en los sitios de WWW en Internet). Sin embargo en el mundo de JEE surge un término muy común, contenedor Web, el cual además de interceptar solicitudes enviadas en los protocolos: HTTP, HTTPS, FTP, y otros, es esencialmente un período de ejecución Java que proporciona una implementación del API Servlet7 Java y facilidades para las páginas servidoras de Java o JavaServer Pages (JSP). Además es responsable de inicializar, invocar, y gestionar el ciclo de vida de los servlets Java y las páginas JSP.

Apache Tomcat

Es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y JavaServer Pages. Es desarrollado en un ambiente participativo y abierto.

Apache Tomcat es usado en numerosas aplicaciones Web de gran escala y criticas en diversas industrias y organizaciones que se referencia en su sitio oficial.

La versión 5.x es implementada a partir de las especificaciones Servlet 2.4 y JSP 2.0 y cuenta con un mecanismo de recolección de basura perfeccionado, basado en su reducción. Posee una capa envolvente nativa para Windows y Unix para la integración de las plataformas.

1.3.5.2 Control de Versiones

Subversion

Es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software; es uno de los controladores mas utilizados en proyectos de software libre. (9)

Características:

- Fuerte integración con Apples; lo cual permite definir controles de acceso avanzados y navegación vía web para consultar el deposito de archivos.

- Contempla y corrige con éxito la transparencia al eliminar y cambiar nombres de archivos.

- En las copias ligeras sobre ramificaciones este controlador de versiones independientemente del número de ramificaciones creadas mantiene un árbol diferencial de cambios, minimizando así el espacio consumido en el depósito.(9)

- En las copias diferenciales de archivos binarios este controlador de versiones es capaz de mantener un control diferencial sobre cualquier archivo binario del depósito así reduciendo el consumo de espacio.

1.3.5.3 Gestores de Base de Datos

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Postgre SQL 8.2

Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. Incluye características de la orientación a objetos, como la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.(10)

Ventajas:

- DBMS Objeto-Relacional. Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son las consultas SQL declarativas, el control de concurrencia multi-versión, el soporte multi-usuario, las transacciones, optimización de consultas, la herencia, los arreglos entre otros.

- Cliente/Servidor. Usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

- Altamente Extensible. Soporta los tipos de datos base, así como: fechas, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. Además operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

- Soporte SQL Comprensivo. Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

- Integridad Referencial. Es utilizada para garantizar la validez de los datos de la base de datos.

- Lenguajes Procedurales. Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Además tiene habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

- MVCC (Multi-Version Concurrency Control) Control de Concurrencia Multi-Versión. Es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios, es decir permite la lectura sin que sea bloqueada por los que escriben que están actualizando registros.

- Write Ahead Logging (WAL) Esta característica incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en caso de que no existan las condiciones para la conexión con la base de datos, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos desde el punto en que se quedó.

- Es un gestor magnífico bajo licencia Berkeley Software Distribution (BSD), que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios Web que posean alrededor de 500.000 peticiones por día. Además por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.

Desventajas:

- Consume bastantes recursos y carga con mucha facilidad el sistema.

- Velocidad de respuesta un poco deficiente al gestionar bases de datos relativamente pequeñas, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes.

Por lo expuesto anteriormente se decide utilizar este último gestor de base de datos (PostgreSQL), porque es Open Source, bajo la licencia BSD, es multiplataforma, con un uso y distribución gratis. Los usuarios mismos pueden mejorar las fallas que estos gestores puedan tener, debido a que pueden arreglar el código fuente a su conveniencia, es unos de los pocos gestores de bases de datos que permite la creación de nuevos tipos de datos. Este gestor posee una integridad referencial muy buena permitiendo una mayor seguridad en los datos. Para esta elección se tuvo en cuenta fundamentalmente que es el único gestor de base de datos aprobado por la Dirección de Informatización de la Universidad de las Ciencias Informáticas. (10)

DB4O

Se trata de una base de datos que está específicamente diseñado para proporcionar persistencia a los programas desarrollados orientados a objetos. Persistencia de objetos es la capacidad de guardar los objetos en un sistema a fin de que existan incluso después de que la aplicación que los creó deje de usarlos. Este motor de base de datos presenta las siguientes características:

- **Consumo mínimo de recursos:** DB4O está diseñado para ser embebido en clientes u otros componentes de software, de manera totalmente invisible para el usuario final. Es por eso que viene como una librería fácil de incorporar, con un tamaño que ronda los 400 Kb. Como el motor corre en el mismo proceso de la aplicación, el usuario cuenta

con control completo sobre la administración de memoria, y puede realizar procesos de profiling y debugging del desempeño sobre todo el sistema. Si la aplicación está corriendo, la base también lo está, sin excepción. Y aún más importante, DB4O es extremadamente flexible a la hora de actualizar una base existente con un modelo de objetos que ha cambiado. Siempre asume que no hay un administrador de base de datos y, por lo tanto, permite a la aplicación cambiar del modelo viejo al modelo nuevo de modo transparente.(12)

- **Alto rendimiento:** El rendimiento de DB4O es equiparable al de los mejores sistemas de base de datos tradicionales.(12)

- **Fácil implementación:** Sólo hay que agregar la librería única de DB4O (.jar o .dll) al entorno de desarrollo, abrir el archivo de base de datos y almacenar cualquier objeto (sin importar su complejidad) con una sola línea de código.(12)

- **Portabilidad:** Corre de manera embebida y nativa en plataformas orientadas a objetos. Se pueden desarrollar aplicaciones para desplegar en varias plataformas (por ejemplo, en PDAs) o en combinaciones heterogéneas de clientes Windows y servidores Java.(12)

- **Confiabilidad:** Finalmente, DB4O soporta todas las propiedades ACID. Múltiples usuarios simultáneos de una base DB4O son efectivamente aislados, y sus operaciones son serializadas de forma transparente por la librería. Las transacciones se terminan con los métodos commit() y rollback() de la clase ObjectContainer en caso de que el sistema se caiga durante una actualización de la base de datos, cuando el ObjectContainer de DB4O es reabierto, se completan de forma correcta todas las transacciones interrumpidas.(12)

1.3.5.4 Herramientas de Modelados

Los medios con los que siempre se ha realizado el intercambio de información de diseño e ideas usando la notación UML han sido populares: pizarras, cuadernos y trozos de papel, etcétera. Pero UML se utiliza mejor a través de una herramienta de modelado, la cual puede ser usada para capturar, guardar, rechazar, integrar automáticamente información, y diseño de documentación. (Modelado de Sistemas con UML 2002).

Una característica que UML brinda para beneficiar a los modeladores es escoger una herramienta de modelado. Tiempos atrás, el modelador primero tenía que seleccionar una notación de metodología, y después estaba limitado a seleccionar una herramienta que la soportara. Ahora con UML como estándar, la elección de notación ya se ha hecho para el modelador. Y con todas las herramientas de modelado soportando UML, el modelador puede seleccionar la herramienta basada en las áreas claves de funcionalidad soportadas que permiten resolver los problemas y documentar las soluciones. (Modelado de Sistemas con UML 2002).

Como una buena caja de herramientas, una buena herramienta de modelado ofrece todas las herramientas necesarias para conseguir hacer eficientemente varios trabajos, sin dejarte nunca sin la herramienta correcta.

Las herramientas de modelado deberían soportar las siguientes funcionalidades:

- Soporte para toda la notación y semántica de UML.

- Soporte para una cantidad considerable de técnicas de modelado y diagramas para complementar UML, incluyendo tarjetas CRC, modelado de datos, diagramas de flujo, y diseño de pantallas de usuario. Posibilidad de reutilizar información obtenida por otras técnicas todavía usadas, como modelado tradicional de procesos.

- Facilitar la captura de información en un repositorio subyacente permitiendo la reutilización entre diagramas.

- Posibilidad de personalizar las propiedades de definición de elementos subyacentes de modelos UML.

- Permitir a varios equipos de analistas trabajar en los mismos datos a la vez.

- Posibilidad de capturar los requisitos, asociarlos con elementos de modelado que los satisfagan y localizar cómo han sido satisfechos los requisitos en cada uno de los pasos del desarrollo.

- Posibilitar la creación de informes y documentación personalizados en tus diseños, y la salida de estos informes en varios formatos, incluyendo HTML para la distribución en la Internet o Intranet local.

- Posibilidad para generar y realizar ingeniería inversa (por ejemplo C++, Java, etcétera.) para facilitar el análisis y diseño interactivo, para volver a usar código o librerías de clase existentes, y para documentar el código.

Visual Paradigm Suite

Visual Paradigm Suite es un conjunto de herramientas de modelado que permiten realizar el modelado dentro del proceso de desarrollo de software.

A continuación se describen las principales herramientas presentes dentro de esta suite:

Visual Paradigm for UML Enterprise Edition:

Es una herramienta de modelado diseñada para un gran número de usuarios, incluyendo ingenieros de sistemas, analistas de sistemas, analistas de negocio, arquitectos de sistemas. Además se integra con IDEs (Eclipse, JBuilder, NetBeans, IntelliJ IDEA, JDeveloper and WebLogic Workshop) para soportar la fase de implementación de desarrollo de software. La transición desde el análisis al diseño y después a la implementación es cuidadosamente integrada dentro de la herramienta CASE, de esta manera se reduce significativamente el esfuerzo en todas las etapas del ciclo de vida del desarrollo del software. Incluye además un conjunto de herramientas que soportan Object-Relational Mapping (ORM), como Hibernate, lo cual incluye generación completamente orientada a objetos, listo para usar librerías para obtener y modificar registros de base de datos para una gran variedad de gestores de base de datos, y la sincronización entre los diagramas de clases y los diagramas de entidad-relación (ERD).

Presenta además generación de código e ingeniería inversa del código. Permite generar los EJB y así mismo los descriptores de despliegue para varios servidores de aplicación. Distinto a muchas herramientas de modelado pueden extenderse sus diagramas hechos desde el Visio y de Rational Rose. Soporta la última versión de UML 2.0.(7)

Visual Paradigm Smart Development Environment (SDE) Enterprise Edition:

Visual Paradigm SDE ofrece integración de la herramienta de modelado UML con los IDEs (Visual Studio®, Eclipse/WebSphere®, Borland JBuilder®, NetBeans/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper, BEA WebLogic Workshop™). Visual Paradigm SDE se incrusta él mismo dentro de tu IDE favorito para proveer ambiente de desarrollo y modelado unificado. Dando todas las prestaciones y servicios que brinda el Visual Paradigm for UML Enterprise Edition.

Visual Paradigm DB Visual Architect:

DB Visual Architect es un simple y fácil de usar ambiente de Object-Relational Mapping (ORM), como Hibernate, el cual actúa como un puente entre el modelo de objetos, el modelo de datos y el modelo relacional. Este ayuda a los desarrolladores a construir aplicaciones de base de datos de alta calidad, mucho más rápido, mejores y barato. Disminuye el costo de desarrollo a un 75 % por automatizar el proceso de mapeo entre los objetos de java y las tablas de la base de datos visualmente y a través de diagramas. Los desarrolladores ahorran enormes cantidades de tiempo usando DB Visual Architect para manipular el código de SQL y JDBC transparentemente con (POJO), como DB Visual Architect está habilitado con ingeniería inversa de una base de datos relacional a un diagrama de clases y un diagrama entidad-relación (ERD) y viceversa (crear una base de datos relacional de modelos de relación de entidad (ERD) o de modelos de clases.(7)

1.3.7 Framework

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework 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. 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.

Symfony 1.0.16

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.(13)

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.(13)

Características principales:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa.
- Código fácil de leer que incluye comentarios y que permite un mantenimiento muy sencillo.

- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Hibernate 3.2

Hibernate es un potente framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySQL, etcétera.(14)
Sus principales características son:

- Permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

- Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones de escritorio.

- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.

- Soporte para modelos de objetos con una granularidad muy fina Permite una gran variedad de mapeos para colecciones y objetos dependientes.

- Provee un sistema de caché de dos niveles y puede ser usado en un clúster. Permite inicialización perezosa (lazy) de objetos y colecciones.

- Proporciona el lenguaje HQL en cual provee una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

- Presenta un potente mecanismo de transacciones de aplicación llegando incluso a permitir la interacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución).

Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.(14)

1.4 Herramientas y Tecnologías para Servicios Web

Servicio Web

Un servicio Web (en inglés Web service) es una colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, soportando así la interoperabilidad máquina – máquina (interoperabilidad sintáctica).(15)

La interoperabilidad se consigue mediante la adopción de estándares abiertos.(15)

XML

Un lenguaje de marcas extensible (en inglés Extensible Markup Language), es el formato estándar para los datos que se vayan a intercambiar.

SOAP

Siglas de Simple Object Access Protocol, es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en

diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

WSDL

Son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web, describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo 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.

UDDI

Son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration, protocolo para publicar la información de los servicios Web. Permite a las aplicaciones comprobar qué servicios web están disponibles.

JAX-RPC

JAX-RPC, siglas Java API for XML-based RPC, es una de las APIs de java para registrar y utilizar servicios Web, la cual define bibliotecas API de Java que los desarrolladores pueden utilizar en sus aplicaciones para desarrollar y consumir servicios web haciendo que los principales detalles de implementación sean invisibles tanto para el cliente como para el desarrollador de los mismos, al consumir estos servicios web no importar que el servidor este escrito en otro idioma o que se ejecute en una plataforma diferente; estos servicios pueden ser consumidos tanto por clientes de java como clientes que no sean de tipo java.

JAX-RPC utiliza un protocolo de mensajería XML, tal como SOAP (Simple Object Access Protocol), para transmitir una llamada a procedimientos remotos a través de una red.

El entorno de ejecución de JAX-RPC genera stubs y ties; que no son mas que clases de bajo nivel que permiten la comunicación entre el cliente y el servicio y ambos realizan funciones parecidas, con la diferencia que los stubs operan en el lado del cliente y los ties en el lado del servidor es decir; un stubs que reside en el cliente es un objeto local que representa un servicio remoto y actúa como un proxy para el servicio, un objeto tie que reside en el servidor actúa como un proxy en el servidor.

En JAX-RPC existen unas herramientas llamadas wscompile que se utiliza para general los stubs utilizando documentos WSDL (web service definition language) y wsdeploy que se utiliza para crear los ties y también puede ser utilizada para crear los documentos WSDL (web service definition language).

Como opera un sistema de ejecución JAX-RPC:

- Convierte la llamada del método remoto del cliente en un mensaje SOAP y posteriormente lo envía al servicio como una petición http.

- En el lado del servidor, el sistema de ejecución JAX-RPC recibe la petición, traduce el mensaje SOAP a una llamada al método y finalmente lo invoca.

Después de que el servicio web ha procesado la ejecución, el sistema de ejecución JAX-RPC realiza una serie de pasos similares a los anteriores para regresar un resultado al cliente.

1.5 Conclusiones

Con el análisis desarrollado en este capítulo se puede determinar cuales herramientas, tecnologías y metodologías actuales, son las más factibles para definir la arquitectura del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas; lo anterior permite definir los elementos más adecuados para lograr el más alto nivel de la reutilización de componentes regidos por el documento de Arquitectura de la Dirección de

Informatización de la Universidad de las Ciencias Informáticas. A su vez se ha llegado a la definición más práctica de qué elementos se deben tener en cuenta a la hora de realizar una arquitectura de software, lo cual se pondrá en práctica en el desarrollo de los siguientes capítulos.

Capítulo 2. Dominio y Arquitectura.

2.1 Introducción

En este capítulo se establece el domino que representa la Arquitectura del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas, donde serán detallados los requerimientos de referencias que no son más que los requerimientos comunes en las diferentes aplicaciones del dominio. También se desarrolla la arquitectura base para el proyecto Identificación y Control de Acceso de la UCI, la cual se descompone en cuatro aspectos: el diseño de las capas lógicas, convenciones de nombres y estándares de códigos, estructura de las aplicaciones y mecanismos de colaboración entre los subsistemas y módulos.

2.2 Definición del Dominio

A continuación se define el dominio en que se enmarca la Arquitectura del proyecto Identificación y Control de Acceso de la UCI, rigiéndose por el documento oficial de arquitectura de la Dirección de Informatización (Arquitectura para los Sistemas que Conforman la Intranet Universitaria); de la Universidad de las Ciencias Informáticas, donde se especifican las herramientas, tecnologías, framework y metodologías a usar en las diferentes aplicaciones.

La Arquitectura del proyecto Identificación y Control de Acceso de la UCI se enmarca en las aplicaciones web desarrolladas en tecnología PHP y en las de escritorio desarrolladas en tecnología Java, todas creadas en dicho proyecto.

A continuación se indica cada tecnología, herramienta, framework, plugin o componente utilizado en las diferentes aplicaciones.

Aplicaciones Web

La tecnología que ha sido usada para estas aplicaciones es PHP, debido a que es una tecnología libre y de código abierto, lo que permite con gran libertad modificar o agregar en su código fuente nuevas funcionalidades de acuerdo a las necesidades del proyecto, logrando con esto obtener aplicaciones más seguras y constantemente mejoradas. Una vez desarrollada la aplicación en esta tecnología puede funcionar en la mayoría de los sistemas operativos existentes, esta presenta una gran comunidad de desarrolladores los cuales comparten el conocimiento adquirido en el proceso de desarrollo, facilitando con esto utilizar funciones creadas para economizar el tiempo de desarrollo.

PHP fue diseñado para trabajar sobre la web y una de sus ventajas es que trae un conjunto muy amplio de funciones para ser utilizadas en diferentes tareas relacionadas con la web y un gran número de framework que les posibilita simplificar el desarrollo de las aplicaciones. En estas aplicaciones se utilizará Symfony, el cual es un framework completo ya que toma prestada las mejores ideas del mundo que presentan otros framework y añadirlas a las suyas, desarrollado en PHP5, se utilizará la versión 1.0.16; una de las más actualizadas, surgida de mejoras de las versiones inferiores a esta por parte de la comunidad, diseñado para optimizar el desarrollo de las aplicaciones, proporcionando varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de las aplicaciones complejas, este es compatible con la mayoría de los gestores de base de datos entre ellos; PostgreSql que es el que se utilizará en estas aplicaciones para almacenar la información. Es multiplataforma por lo que se puede ejecutar en diversos sistemas operativos. Se utilizará Eclipse incorporándole al mismo el plugin PDT, obteniendo como resultado un gratuito, cómodo y potente ambiente de desarrollo para crear aplicaciones en tecnología PHP que pueda utilizarse en cualquier sistema operativo, integrándole nuevas herramientas que brinda a los desarrolladores un trabajo más cómodo en las aplicaciones.

Se utilizará Apache Tomcat como uno de los más completos contenedores Web.

Para obtener un buen diseño de las aplicaciones se utilizará código HTML directamente en las páginas, soportando también otras tecnologías como las CSS o Hojas de estilos, la cual será muy útil para la creación más exacta de las diferentes páginas web perteneciente a un mismo sitio, facilitando a los diseñadores reducir sustancialmente su carga de trabajo al diseñar los sitios. Otra tecnología utilizada es el lenguaje Java Script para crear efectos e interactividades, validaciones, entre otras funcionalidades.

Aplicaciones de Escritorio

Estas aplicaciones han sido desarrolladas sobre la tecnología Java, que es una potente tecnología multiplataforma, por lo que permite ejecutarse en cualquier sistema operativo sin tener que efectuarse cambios en las aplicaciones. Se puede acceder al código fuente de las librerías que brinda dicha tecnología permitiendo modificar las funciones y adaptarlas a las necesidades que se presenten en el desarrollo de las aplicaciones. Presenta gran documentación en la red, que posibilita a los desarrolladores optimizar el esfuerzo del trabajo ganando tiempo al reutilizar diversas funciones ya implementadas que están expuestas en la red para su buen uso.

Se utilizará como ambiente de desarrollo integrado el NetBeans 6.0 o el Eclipse 3.3; el primero contiene una paleta de componentes gráficos muy apetitosa con diversos objetos lo cual permite obtener interfaces gráficas de gran calidad, organización de código fuente, completamiento de código fuente; brindándole a los desarrolladores agilizar su trabajo. También contiene un gran volumen de información en la red, es multiplataforma por lo que permite ser utilizado en cualquier sistema operativo y se le pueden incorporar una serie de plugins que facilitará un trabajo mucho más cómodo y confortable a la hora de desarrollar las aplicaciones sobre él; el segundo también presenta la mayoría de las características que el anterior pero para el desarrollo de sus interfaces visuales será acompañado del plugin Visual Editor para Java, el que es gratuito, presentando un gran número de objetos visuales que brindan gran calidad a las interfaces gráficas de las aplicaciones. También se usará el plugin Subeclipse, el cual será muy útil y fácil para los desarrolladores a la hora de guardar cualquier

cambio frecuente que se realice en el código fuente de la aplicación permitiendo obtener una mejor organización en el código debido a que en la mayoría de las aplicaciones participarán diversos desarrolladores en su creación simultáneamente. Se utilizará Hibernate como framework de acceso a datos o para la persistencia a los objetos de las base de datos.

Independientemente del tipo de aplicación, ya sea web o de escritorio, como gestor de base de datos se utilizará PostgreSql ya que las aplicaciones almacenan gran volumen de información y DB4O para las aplicaciones que necesitarán base de datos locales para un correcto funcionamiento de la misma. Se utilizará el Visual Paradigm para brindar soporte al modelado visual de los principales flujos de trabajo en el desarrollo de software; utilizando el UML como lenguaje de modelaje y la metodología RUP, y Subversión para el control de versiones así como para actualizar el trabajo realizado por los desarrolladores.

Con el uso correcto de todas las herramientas y tecnologías escogidas para el desarrollo de las aplicaciones; tanto web como de escritorio enmarcadas dentro de nuestro dominio, las mismas se desarrollarán con la calidad esperada, dándole las funcionalidades requeridas por el cliente y ambas serán compatibles con la mayoría de los sistemas operativos, lo cual es uno de los objetivos fundamentales a alcanzar por el proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas.

2.3 Requerimientos de referencia del dominio

Las aplicaciones desarrolladas en el proyecto Identificación y Control de Acceso de la UCI se desarrollarán en el dominio definido anteriormente para cada aplicación. En las diferentes áreas de estas aplicaciones se destacarán el uso de patrones; semejantes en necesidades, para situaciones o problemas similares. Estas frecuentes situaciones o problemas convergen en requerimientos comunes para todas las aplicaciones que pertenecen al dominio, sin importar el lenguaje o tecnología a usar para su construcción, esos requerimientos serán expuestos a continuación:

- **Seguridad:** En la mayoría de las aplicaciones, será necesario manejar los requerimientos de Autenticidad, Integridad y Confidencialidad de la información.

- **Software**: Las aplicaciones se desarrollarán sobre software libre.

- **Almacenamiento de Datos:** Es necesario guardar los datos de cada aplicación en base de datos y de la forma más óptima posible.

- **Auditoría:** Es preciso tener un control de todos los cambios que sucedan en cada aplicación y saber quien es el responsable de los mismos.

- **Capa de presentación amigable y rica en estilo***:* La capa de presentación debe cumplir esta característica ya que es la que va a interactuar de forma directa con diversos tipos de usuarios, para que cada interfaz gráfica pueda soportar las exigencias de los clientes y que sean fáciles de manejar los cambios en ellas.

- **Alto rendimiento de procesamiento de las peticiones del usuario**: Es necesario que las peticiones del usuario se respondan en el menor tiempo posible por lo que todas las capas de las aplicaciones deben interactuar de la manera más eficiente.

- **Interacción con otros módulos:** Es muy común la interacción con sistemas externos o entre módulos del mismo proyecto por lo que se hace necesario que existan estándares en el uso de las tecnologías necesarias para cumplir este requerimiento.

- **Administración de los aspectos configurables de la aplicación:** Toda aplicación necesita ser configurable tanto en la etapa de despliegue como en tiempo de desarrollo de la manera más simple posible.

2.4 Arquitectura Base

Se presenta una arquitectura base para orientar la construcción de los principales elementos que tendrán las capas lógicas y las bases de datos de las diferentes tipos aplicaciones del proyecto Identificación y Control de Acceso de la UCI.

Las capas lógicas se orientarán a través de un diseño base; organizando la forma de codificar según las propuestas de convenciones o estándares de código, brindando una estructura física para soportar el código, creando así un esqueleto base y definiendo mecanismos de colaboración entre los componentes integrados en ella. Está basada fundamentalmente en el estilo arquitectónico; “Arquitectura en N capas”.

Los elementos de nuestras bases de datos estarán constituidos por; nombre de las bases de datos, tablas, campos de las tablas, las vistas y procedimientos almacenados, los cuales están orientados a través de reglas de estricto cumplimiento para sus construcciones.

2.4.1 Diseño de las Capas Lógicas

La estructura del diseño de las diferentes capas lógicas que presentará las aplicaciones del proyecto Identificación y Control de Acceso se muestra en la Figura 2.1.

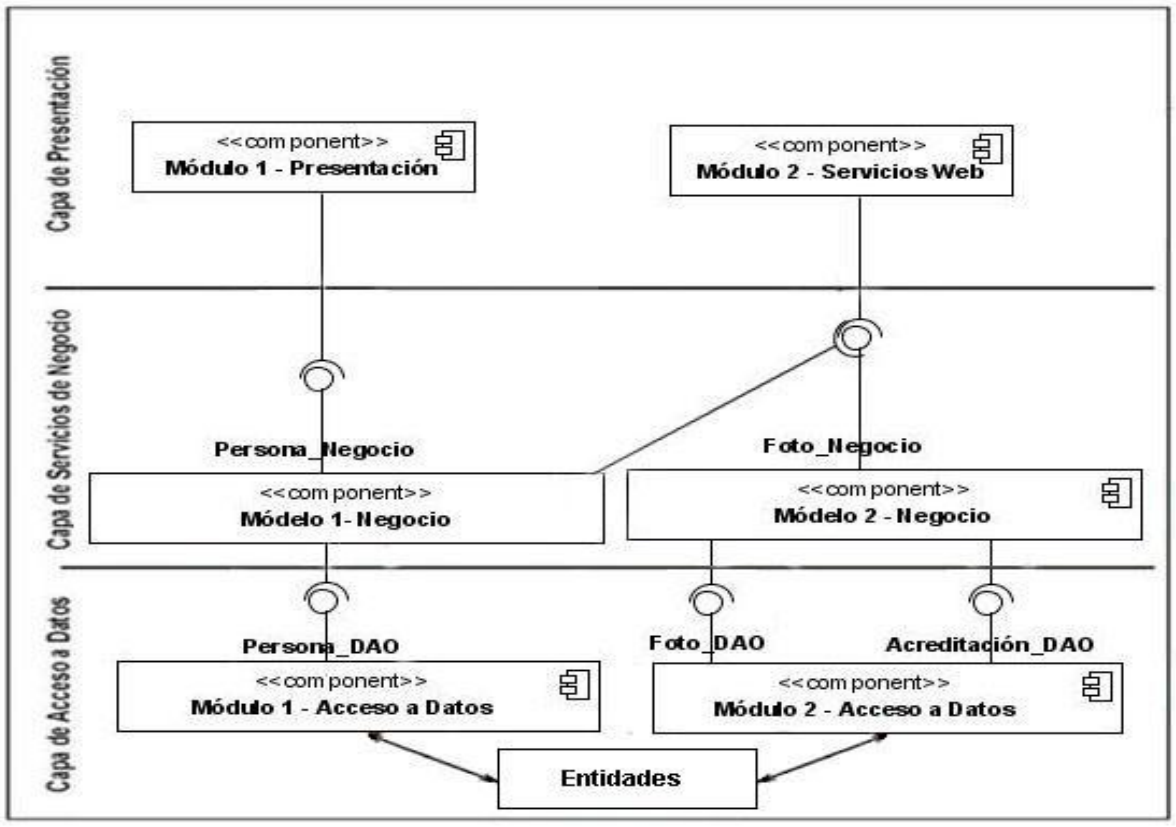

Figura 2.1 Arquitectura de las capas lógicas

Las entidades no son más que los objetos persistentes del dominio; las cuales representan el modelo de objetos o conceptos reales tales como, una persona o una credencial. Estos objetos en conjuntos pueden llegar a pensarse como otra capa lógica que puede presentar las aplicaciones o incluirlas dentro de la capa de acceso a datos. Estos objetos son pasados hasta la capa de presentación, en la cual mostrará al usuario los datos o información que contienen, pero no podrán ser modificados, ya que esto solo ocurre dentro de los contextos transaccionales definidos en la capa de negocio.

En las aplicaciones web al ser uso del Framework Symfony el mismo está basado en un patrón clásico de diseño para estas aplicaciones, conocido como la arquitectura MVC (Modelo-Vista-Controlador). El principio más importante de la arquitectura MVC es la

separación del código del programador en capas, al ser uso de otros patrones de diseño la programación se puede simplificar y las capas de modelo, la vista y el controlador se pueden subdividir en más capas(13).

En las aplicaciones del proyecto Identificación y Control de Acceso de la UCI, todas sus capas lógicas correrán en un servidor web si es una aplicación web y si es una aplicación de escritorio correrán en un servidor de aplicación es decir, la arquitectura que este presenta no será una arquitectura distribuida y por tanto sus diferentes subsistemas no serán distribuidos. Esto no representa un problema ya que pudiera adaptarse a un subsistema distribuido que lo requiera, pero no es muy recomendable.

Las capas arquitectónicas se encuentran separadas por partes lógicas; de esta forma cada capa puede ser modificada tantas veces sea necesario sin afectar las demás capas, una capa no se encuentra regida a lo que le ocurre a la capa superior, su dependencia es únicamente con la capa inmediata inferior. Estas dependencias entre capas se realiza a través de interfaces, asegurando que el acople sea el más bajo posible.

Posteriormente se indicará con más referencias cada una de las capas lógicas propuestas anteriormente comenzando desde la capa de Acceso a datos hasta llegar a la capa de Presentación, respectivamente según la Figura 2.

2.4.1.1 Capa Acceso a Datos

Manteniendo la filosofía de los desarrolladores de que es mejor programar orientado a interfaces que a clases, en esta capa lógica se mantendrá esa filosofía, tanto para las aplicaciones de escritorio como para las web enmarcadas dentro del dominio definido anteriormente.

En esta capa se hará uso del patrón DAO que no es más que una solución al problema del diferencial de impedancia entre un programa de aplicación orientado a objeto y una base de

datos relacional, que ayudará a abstraer el modelo relacional como un modelo de objeto a través de una interfaz común que suministrará a la aplicación del medio de almacenamiento, ya sea una base de datos o un archivo.

El término de Objeto de Acceso a Datos (DAO), es ampliamente usado en el desarrollo de software, estos encapsulan la persistencia de los objetos de dominios o entidades, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos. Los DAOs estarán disponibles para los objetos (especialmente para los objetos del negocio) haciendo uso de la inyección de dependencias con los objetos de negocio y las instancias de los DAOs.

La capa de acceso a datos no debería de contener ningún tipo de lógica de negocio.

Las interfaces de los DAOs tendrán básicamente los siguientes métodos:

- **Métodos para descubrir:** Estos localizan los objetos almacenados para ser usados por la capa de servicios de negocio.

- **Métodos para persistir o salvar:** Estos hacen persistentes a los objetos transitorios.

- **Métodos para eliminar:** Estos eliminan a los objetos guardados en el medio de almacenamiento (base de datos).

- **Métodos para conteos y otras funciones agregadas:** Estos devuelven los resultados de operaciones que son más eficientes implementarlas usando funcionalidades de la base de datos (procedimientos almacenas, etcétera.) que iterar sobre los mismos objetos.

En la Figura 2.2, representa un ejemplo de los elementos que debe presentar un diagrama de clase de esta capa lógica.

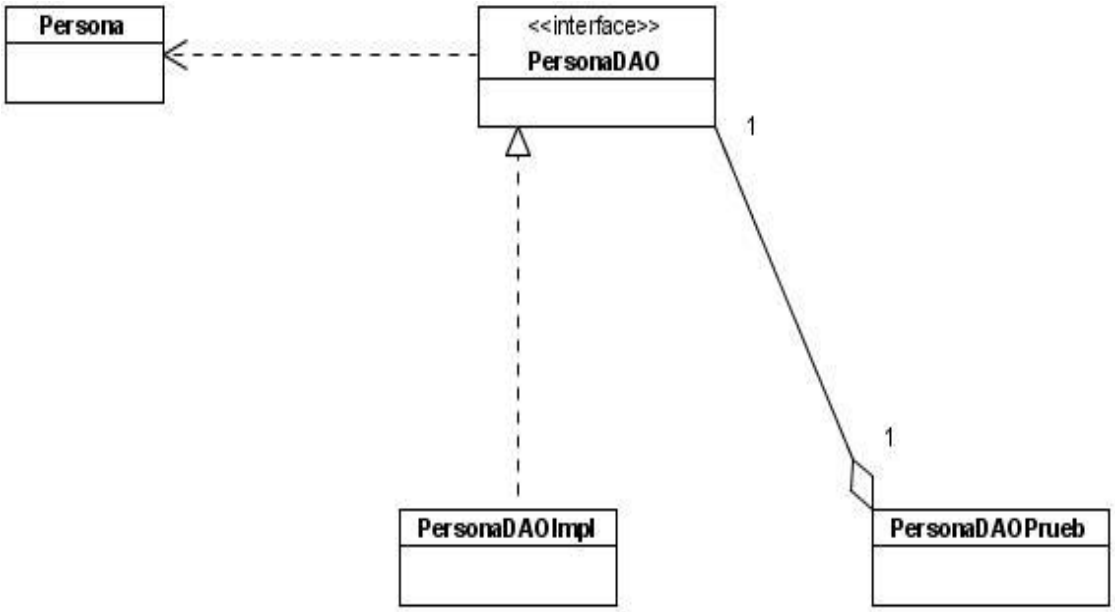

Figura 3.2 Ejemplo de elementos que conforman la Capa Acceso a Datos

Persona: Es la entidad del dominio a ser persistida o guardada por los métodos expuesto en la interfaz PersonaDAO.

PersonaDAO: Es la interfaz que expone a la capa de servicios de negocio los métodos del DAO para persistir la entidad de dominio Persona.

PersonaDAOImpl: Esta clase es la implementación de la interfaz que realmente se conecta a la base de datos lo que permite dar funcionalidad real a los métodos de PersonaDAO.

PersonaDAOPrueb*:* Es la clase que prueba todos los métodos de la clase PersonaDAOImpl, con la utilización de valores de pruebas en casos extremos, es la responsable de asegurar que los métodos del objeto de acceso a datos están funcionando correctamente, para utilizar el DAO desde la capa de servicios de negocio.

2.4.1.2 Capa de Servicios de Negocio

Esta capa lógica presenta un rol muy importante tanto en las aplicaciones web como en las de escritorios enmarcados dentro del dominio definido anteriormente ya que ella contiene los objetos del negocio, estos objetos separan los datos de la lógica del negocio usando un modelo de objetos.

Es la responsable de delimitar las transacciones y proveer un punto de entrada para las operaciones sobre el sistema. Esta capa no debería tener conocimiento sobre lo concerniente a la presentación y debería ser reutilizable.

Las principales funcionalidades que presentará la Capa de Servicio Negocio son:

- **Lógica de negocio específica de procesos de negocios:** A veces es más oportuno para las Entidades contener con lógica de negocio aplicable a muchos casos de uso específicos. Sin embargo, existen casos de usos específicos que son realizados en la capa de servicios de negocios. Pero se propone que en esta definición de arquitectura las Entidades no presenten ningún tipo de lógica de negocio, sino que esta responsabilidad recaiga sobre los objetos de negocio, permitiendo usar a las Entidades como Transfer Objects que se mueven entre las capas arquitectónicas de la aplicación.

- **Puntos de entrada muy bien definidos para las operaciones de negocio implementadas:** Los objetos de negocio ofrecerán las interfaces que se usarán en la capa de presentación.

- **Control de transacciones:** Aunque a veces la lógica de negocio pueda ser movida a Entidades, el control de transacciones no debería. Sino que las políticas transaccionales de la aplicación deberían ser planteadas al nivel de los objetos de negocio.

- **Ejecución de restricciones de seguridad:** Las restricciones de seguridad en esta capa es en los puntos de entradas a la capa media, es decir en los objetos de negocio.

En la Figura 2.3, se mostrará un ejemplo de los elementos que debe presentar un diagrama de clases en esta capa lógica.

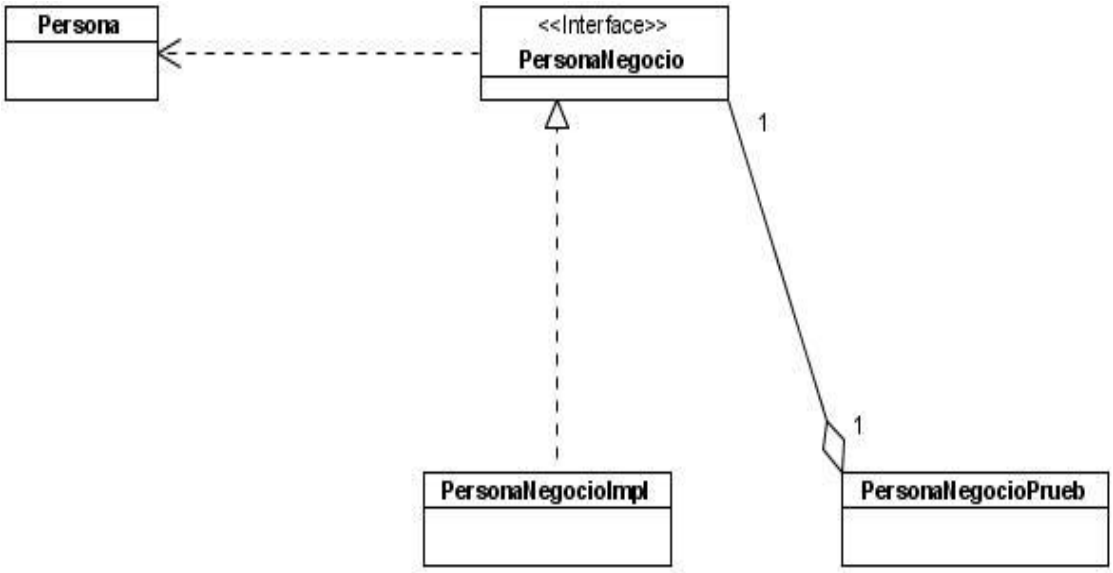

Figura 4.3 Ejemplo de elementos que conforman la Capa de Negocio

Persona: Es la entidad del dominio que utilizarán los métodos expuestos en la interfaz de negocio PersonaNegocio.

PersonaNegocio: Es la interfaz que expone a la capa de presentación los métodos para las operaciones de negocio sobre la entidad de dominio Persona. Puede darse el caso de que un objeto de negocio exponga métodos de negocio de varias entidades del dominio o de procesos especificados de negocio.

PersonaNegocioImpl: Esta clase es la implementación de la interfaz de negocio desarrollando los métodos que contendrán la lógica de negocio.

PersonaNegocioPrueb: Es la clase que prueba todos los métodos de la clase PersonaNegocioImpl, con la utilización de valores de pruebas en casos extremos, es la responsable de asegurar que la lógica del negocio implementada en el objeto PersonaNegocioImpl responda a los requerimientos q debe cumplir. En estas pruebas el objeto de negocio (PersonaNegocioImpl), no tiene que usar necesariamente los DAOs reales, ya que el objetivo de estas pruebas no es probar la parte de persistencias de las operaciones de negocio implementadas.

2.4.1.3 Capa de Presentación

La capa de presentación descansará sobre una capa de servicios de negocio, esto nos permite que esta capa lógica sea fina y que no contenga lógica del negocio, sino lo referente a aspectos de presentación, por ejemplo, el código para manipular las interacciones de escritorios, web o servicios web según el tipo de aplicación a desarrollar por cada módulo perteneciente al proyecto Identificación y Control de Acceso. Además se puede desarrollar más de una capa de presentación en una misma aplicación sin impactar en las capas arquitectónicas inferiores, las cuales estas capas arquitectónicas inferiores no deberán tener conocimiento sobre la capa de presentación.

Capa de Presentación Web:

Las aplicaciones web enmarcadas dentro de nuestro dominio tradicionalmente usarán; HTML para estructurar el contenido de las páginas con independencia del diseño visual y los dispositivos de presentación, CSS las cuales serán las encargadas del diseño estático de los contenidos estructurados en HTML y JavaScript para añadir dinamismo al despliegue del contenido en pantalla. Esta capa de presentación web es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado en las aplicaciones.

Capa de Presentación de Escritorio:

Las aplicaciones de escritorios enmarcadas dentro de nuestro dominio, si son desarrolladas en el ambiente de desarrollo integral (IDE) eclipse, el mismo estará acompañado por el plugin Visual Editor para Java, el cual es el editor de interfaces gráficas de código abierto que presenta una serie de objetos visuales muy apetitosos en funcionalidades y si se utiliza NetBeans como IDE se utilizará su propio diseñador de interfaces con una serie de objetos con las mismas características similares o quizás con funcionalidades más avanzadas que los antes mencionados.

Esta capa de presentación de escritorio es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado en las aplicaciones.

Interfaces de Servicios Web:

En el proyecto Identificación y Control de Acceso reflejará la filosofía de servicios web, por ejemplo, el soporte para clientes a través de invocación a métodos a través de clientes remotos mediante servicios web u otros protocolos de serialización de XML como SAOP, http, etc. Esto debería ser visto como una capa de presentación alternativa sobre las interfaces bien definidas de la capa de servicios de negocio.

2.4.2 Convenciones de nombres y estándares código

Con el fin de lograr un lenguaje común y uniforme para las distintas aplicaciones; ya sea aplicaciones Web como las aplicaciones de escritorios que se desarrollarán en el proyecto Identificación y Control de Acceso, se especifican conversiones de nombres y estándares de código para los distintos recursos de las aplicaciones.

2.4.2.1 Clases de la capa de acceso a Datos

- Las interfaces que enmarcan las operaciones sobre los objetos de acceso a datos, correspondientes al patrón de diseño Data Access Object terminan con la palabra *DAO.*

Ejemplo: PersonaDAO

- Las implementaciones reales de las interfaces DAO comienza con el nombre de la interfaz correspondiente y termina con la palabra *Impl.*

Ejemplo: PersonaDAOImpl

- Las clases utilizadas para efectuar pruebas a las interfaces DAO comienzan con el nombre de la interfaz correspondiente y termina con la palabra *Prueb*.

Ejemplo: PersonaDAOPrueb

2.4.2.2 Clases de la capa de Servicios de Negocio

- Las interfaces que representan las operaciones del negocio terminarán con la palabra Negocio.

Ejemplo: PersonaNegocio.

- Las implementaciones de las interfaces de negocio comenzarán con el nombre de la interfaz correspondiente y terminaran con la palabra Impl.

Ejemplo: PersonaNegocioImpl.

- Las clases utilizadas para probar las funcionalidades del negocio terminan con la palabra Prueb.

Ejemplo: PersonaNegocioPrueb.

2.4.2.3 Recursos de la Capa de Presentación

Aplicación Web:

- La página de inicio o principal tendrán el nombre “index.phtml”.
- Las páginas HTML que constituyen recursos estáticos o dinámicos tendrán la extensión “.phtml”.

- Las extensiones de las imágenes a utilizar vienen dadas por el formato de las mismas.

- Las páginas de las aplicaciones se nombrarán al inicio con la palabra “sf” y seguidamente el nombre de la acción o acciones que realizará la misma.

- Los archivos de hojas de estilos creados con CSS se nombrarán dependiendo de la acción que realice y terminarán con la extensión “.css”.

- Los archivos de JavaScript se nombrarán dependiendo de la acción que realice y terminarán con la extensión “.js”.

Aplicación de Escritorio:

- La Form principal se llamará “Jprincipal.java”.

- Las clases asociadas a formularios se nombrarán correspondiéndose con la acción respectiva que realiza, anteponiéndole la letra “J” y terminarán con la extensión “.java”.

- Las extensiones de las imágenes a utilizar vienen dadas por el formato de las mismas.

Interfaces de Servicios Web

- Las interfaces que representarán a las operaciones de negocio que se van a brindar o consumir como servicio terminarán en la palabra “Servicio”.

Ejemplo: FotoServicio.

- Las implementaciones de las interfaces de servicio que expondrán las funciones de negocio y las implementaciones que se van a consumir de un servicio, comenzarán con el nombre de la interfaz correspondiente y terminarán con la palabra “Impl”.

Ejemplo: FotoServicioImpl.

2.4.3 Estándares de código en PHP

Para el desarrollo de las aplicaciones web enmarcadas dentro del dominio definido para el proyecto Identificación y Control de Acceso, especificaremos una quía de estándares de desarrollo para los programadores en lenguaje PHP, definiendo el modo en que básicamente debe ser realizada dichas aplicaciones.

- Siempre utilizar las etiquetas **<?php …..?>** para abrir un bloque de código. No utilice el método de etiquetas cortas, por que esto depende de las directivas de configuración en el archivo PHP.ini y hace que el script no sea tan portable.

- Los nombres de las clases deben de iniciarse con letra mayúscula.

- Los nombres de las variables y de las funciones se iniciarán con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciarse con letra mayúscula (el nombre puede escribirse separado por signos de guión mayor).

- Si una función, en una clase, es privada; deberá comenzar con el signo de guión mayor para una fácil identificación.

- Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.

- Las estructuras de control deben tener un espacio entre el keyword de la estructura y el signo de apertura de paréntesis para distinguir entre las llamadas de las funciones y el signo de llaves deben estar sobre la línea de la estructura.

- Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro; espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (**;**).

- El estilo de los comentarios debe ser como el estilo de comentarios para C (***/* */*** ó ***//***).

2.4.4 Estándares de código en Java

Para el desarrollo de las aplicaciones de escritorios enmarcada dentro de nuestro dominio, especificaremos una quía de estándares de desarrollo para los programadores en lenguaje Java, definiendo el modo en que básicamente debe ser realizada dichas aplicaciones.

- Las líneas de código no deben ser muy extensas, por lo general inferiores a 80 caracteres. En caso de necesitar más, se bajan a la siguiente línea. Podemos aprovechar un cierre de paréntesis o una coma etc.

- Se deben emplear espacios o tabuladores para que se distingan con claridad los distintos bloques de sentencias. Es recomendable emplear espacios en blanco debido a las distintas interpretaciones que hacen los editores sobre las tabulaciones.

- Los nombres de las clases comienzan con mayúscula. Si el nombre es compuesto de varias palabras, por lo general se escriben juntas y con el primer carácter en mayúscula.

Ejemplo: class MyClase{…}

- Las instancias de las clases, los objetos, comienzan con minúscula y en caso de estar compuesto de varias palabras, el resto comenzaran con mayúscula:

Ejemplo: public String myCadena;

- El nombre de las variables y clases debe ser significativo, permitiendo tener una idea de su función con solo leerlo. Esto no siempre es posible, pero siempre es mejor dar un nombre cercano al objetivo que un nombre al azar.

1. Las variables siempre se debe inicializar si no queremos llevarnos sorpresas y excepciones.

2. No se puede emplear clases declaradas dentro de otra clase.

3. El estilo de los comentarios debe ser como el estilo de comentarios para C (***/* */*** ó ***//***).

4. Es muy recomendable documentar cada uno de los métodos y variables relevantes de la clase.

5. Es obligatorio detallar el uso de cada constante.

6. El comentario de la clase en general debería tener los siguientes campos:

    - Nombre del autor.
    - Fecha de creación.
    - Descripción breve de sus funciones.
    - Licencia bajo la que se ampara.
    - Historial de cambios, en los que se incluiría el nombre del autor de los cambios, la fecha y un comentario sobre los mismos.

7. Los comentarios de bloques y líneas sueltas, se debe ser preciso, y no sobre comentar el código.

2.4.5 Interfaces Gráficas de Usuario

- Las interfaces gráficas de las aplicaciones enmarcadas dentro de nuestro dominio son las ventanas al mundo del usuario. Es importante cuidar los detalles estéticos y ser precisos, porque lo que queremos es que el usuario se sienta cómodo con nuestro software.

- El espacio del texto y los gráficos no debería exceder el 40 % del formulario.

- Se debe pedir solo la información necesaria.

- Los textos de los formularios deben ser claros y concisos.

- Los datos importantes o de uso más frecuente deben de estar en primer lugar.

- Las interfaces debe ser intuitivas, designando de la misma forma a los componentes con la misma función y comportamiento a lo largo de los formularios. Esto minimiza el tiempo de aprendizaje.

- Proporcionarle al usuario ayuda para realizar las tareas, en este aspecto son muy útiles los tooltips incorporados por la mayoría de los lenguajes de alto nivel actuales.

    - El acceso a las distintas funciones debe ser previsible. Esto crea sensación de progreso y permite al usuario experimentar con la aplicación aumentando el grado de soltura con la misma.

2.4.6 Bases de Datos

Las bases de datos seguirán las siguientes reglas de nombre.

- Es reglamentado utilizar el identificador “UCI” al inicio de los nombres de cada una de las base de datos.

- Los nombres de las base de datos serán cortos, descriptivos y en singular.

- Los nombres de las base de datos comenzarán con letra inicial mayúscula y seguidamente con letras minúsculas.

- Si el nombre de las base de datos contiene más de una palabra, estas se separaran por el símbolo “_”.

Ejemplo: UCI_Foto.

2.4.6.1 Tablas

- Seguirán las siguientes reglas de nombre.

- Es reglamentado utilizar el identificador *"Tb_"* al inicio del nombre de cada uno de las tablas.

- El nombre de la tabla serán cortos, descriptivos y en singular.

- El nombre da las tablas comenzarán con letra inicial mayúscula y seguidamente con letras minúsculas.

- El nombre de las tablas serán en singular.

    1. Se antepondrá una *"n"* para todas aquellas tablas de nomencladores.

        Ejemplo: Tb_nEstado.

    2. Se antepondrá una *"d"* para todas aquellas tablas de datos.

        Ejemplo: Tb_dPersona.

    3. Se antepondrá una *"r"* para todas aquellas tablas que representen relaciones y además el nombre consistirá en el nombre de las tablas que se relacionarán, unidas por el símbolo *"_"*.

        Ejemplo: Tb_rPersona_Estado.

2.4.6.2 Campos de las Tablas

- Para el carnet de identidad se usará el identificador: CI.

- Para campos que sean booleanos se antepondrán el identificador "Es_" delante del nombre.

  Ejemplo: Es_Becado.

- Los campos identificadores que sean de tipo "Id" se nombrarán con el prefijo "Id_".

  Ejemplo: Id_Estado.

- Las palabras en un campo comenzarán con mayúscula, seguidamente con minúscula y cuando son palabras compuestas estarán separadas por "_".

2.4.6.3 Vistas

- Es reglamentado utilizar el identificador "Vs_" al inicio del nombre de cada una de las vistas.

- Los nombres de las vistas serán lo más corto y descriptivo posible.

  Ejemplo: Vs_DatosPersona.

2.4.6.4 Procedimiento Almacenado

- Es reglamentario utilizar el identificador "Pa_" al inicio del nombre de cada uno de los procedimientos almacenados.

- El nombre de los procedimientos almacenados estarán compuesto por el nombre de la Entidad principal sobre la cual actúa el procedimiento almacenado y la acción que efectúa el procedimiento almacenado (operación), estos separados por el símbolo "_".

  Ejemplo: Pa_Persona_Ins.

- El nombre de las principales operaciones de los procedimientos almacenados serán.

  1- Procedimiento Almacenado Inserción: Ins.

  Ejemplo: Pa_Persona_Ins.

  2- Procedimiento Almacenado eliminación: Elm.

  Ejemplo: Pa_Persona_Elm.

  3- Procedimiento Almacenado selección: Sel.

  Ejemplo: Pa_Persona_Sel.

  4- Procedimiento Almacenado Actualizar: Act.

  Ejemplo: Pa_Persona_Act.

2.4.7 Estructura de las aplicaciones

La estructura definida para las aplicaciones pertenecientes al proyecto Identificación y Control de Acceso nos permitirá organizar cada tipo de clase o componente que se necesite en el desarrollo de cada aplicación, donde esta estructura le facilita a los desarrolladores un marco de trabajo más cómodo e interactivo.

2.4.7.1 Organización de la aplicación en unidades organizacionales

Con el objetivo de dividir y organizar las diferentes aplicaciones que se enmarcan dentro del dominio definido, se crearán paquetes por cada unidad organizacional que se defina; las cuales no son más que subsistemas y módulos. Un subsistema se refiere a un conjunto de funcionalidades del sistema que tienen características muy propias y a su vez están subdivididas en módulos. Un módulo encapsula un conjunto de funciones que debe realizar el subsistema; estos son agrupados por presentar características muy similares y se definen en la etapa de diseño. A continuación se expone esta estructura (Ver Figura 2.4).

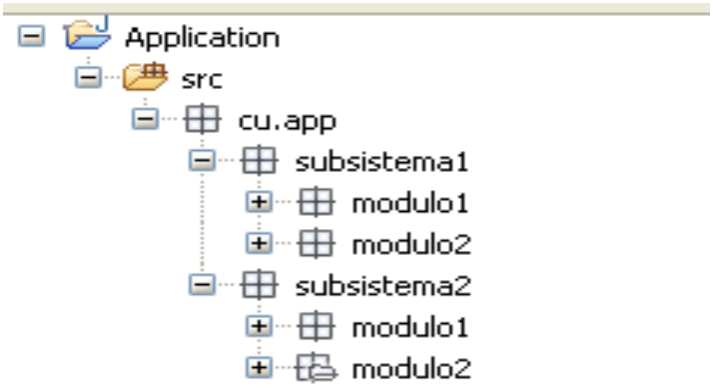

Figura 5.4 Representación de subsistemas y módulos

2.4.7.2 Estructura de organización de paquetes

Dentro de la estructura de paquetes que se define en las aplicaciones del proyecto Identificación y Control de Acceso la unidad más elemental es el módulo. Un módulo se puede comparar con una pequeña máquina que puede actuar por si sola, siempre y cuando estén activas las demás máquinas de las que esta depende para su funcionamiento. En un módulo existirán todas las capas lógicas definida en la arquitectura base para las aplicaciones del proyecto Identificación y Control de Acceso. Cada componente que se desarrolle en un módulo tiene un lugar muy bien definido en esta estructura organizacional de paquetes (Ver Figura 2.5).

- **configuración:** Se encuentran todos los archivos que tienen que ver con la configuración del módulo, los archivos utilizados para la internacionalización, ficheros de propiedades y cualquier otro destinado a estos fines.

- **dao:** Se hallarán las interfaces DAOs.

- **dao.imp:** Se hallarán las implementaciones de la interfaz DAOS.

- **dao.prueba:** Se hallarán las clases de prueba utilizadas para efectuar las pruebas de unidad a las implementaciones de los DAOs.

- **negocio:** Se encontrarán las interfaces de los servicios de negocio.

- **negocio.imp:** Se encontrarán las implementaciones de las interfaces de negocio.

- **negocio.prueba:** Se hallarán las clases de prueba utilizadas para efectuar las pruebas de unidad a los métodos implementados en las clases de negocio.

- **servicio:** Se encontrarán las interfaces que representan las operaciones de negocio que se van a exponer o construir como servicios.

- **presentación:** Se encontrarán las interfaces gráficas que representarán cada aplicación.

- **presentación.validar:** Se encuentran las clases para validar datos (Validadores).

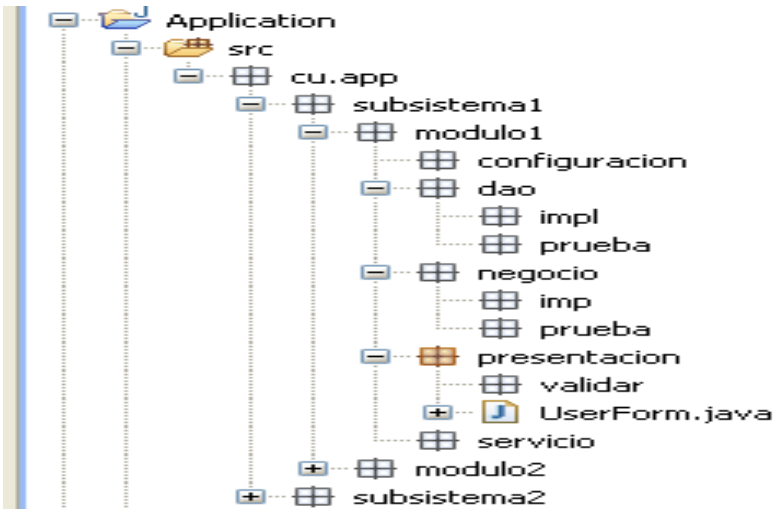

Figura 6.5 Representación de la estructura de paquete

2.5 Mecanismo de colaboración

Los mecanismos de colaboración son estrategias propuestas para establecer la forma de comunicación entre los componentes del software. Según la organización planteada en la sección anterior, que dividen al sistema en estructuras físicas denominadas subsistemas y módulos, se proponen las estrategias de comunicación o colaboración entre ellos. Como ya se ha explicado, un subsistema agrupa una colección de módulos. Por consiguiente, si se detecta que entre los módulos de un mismo subsistema existe dependencia entre ellos, es decir, que algunos módulos necesitan funcionalidades implementadas en otros, entonces crearemos dos posibles propuestas de estrategia de colaboración a usar. Un módulo se puede ver como la encapsulación a nivel de componente de software de un conjunto de casos de uso.(7)

2.5.1 Dependencia directa

La dependencia directa es cuando en un módulo específico de cualquier tipo de aplicación que se enmarca dentro del dominio definido presente funcionalidades expuestas en una o más interfaces que otros módulos necesiten y estos las utilizan directamente. En la Figura 2.6 se muestra un ejemplo de las dependencias directas que existen entre el Modulo1 y Modulo2 con

el Modulo3 mediante la interfaz Expediente_Negocio. Esta figura es un ejemplo de una de las estrategias de colaboración entre módulos pertenecientes a un mismo subsistema.(7)

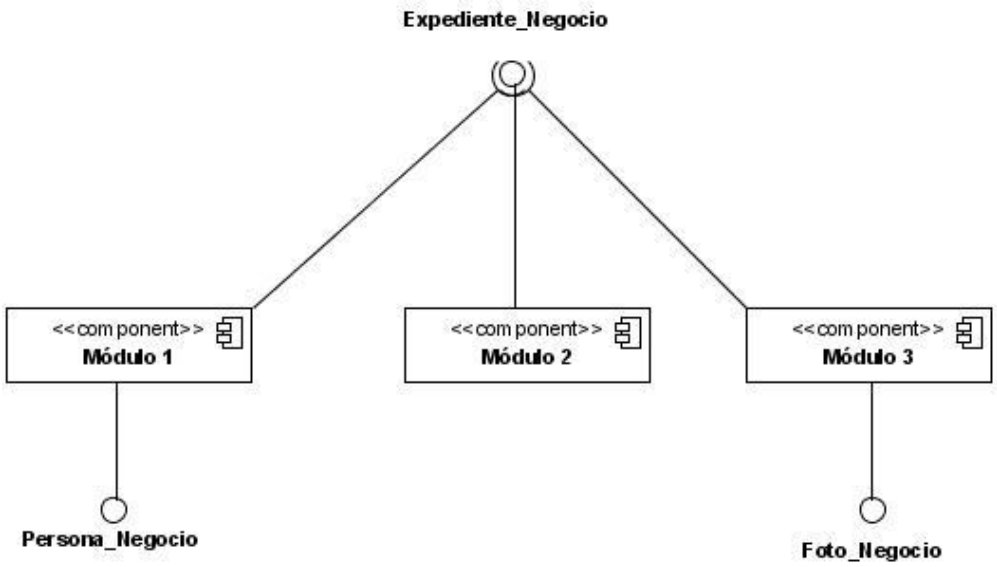

Figura 7.6 Representación de dependencia directa

2.5.2 Dependencia indirecta

La dependencia indirecta es cuando existen dependencias directas entre varios módulos que pertenecen a cualquier tipo de aplicación que se enmarca dentro del dominio definido y no se desea que exista este tipo de dependencia debido a que su impacto arquitectónico no ponga en riesgo algún requerimiento funcional o no funcional de las aplicaciones.

Tomando como ejemplo la Figura 2.6, se desea eliminar las dependencias directas que existen con el Módulo 2 a través de las funcionalidades expuestas en la interfaz Expediente_Negocio, por que como se puede ver en la figura, el Módulo 2 es imprescindible para los otros dos módulos ya que estos necesitan de la interfaz Expediente_Negocio y la aplicación tiene como uno de sus requerimientos no funcionales que en algunos casos la aplicación no contenga el Módulo 2.

Para proporcional una solución adecuada a este problema, como muestra la Figura 2.7, se creará un nuevo módulo que llamaremos “Común”, donde en él se moverá la interfaz Expediente_Negocio y todas las clases y recursos implicados en soportar las funcionalidades expuestas por la interfaz. De esta forma los módulos que anteriormente dependían directamente del Módulo 2 a causa de usar las funcionalidades de la interfaz Expediente_Negocio (Figura 2.6), ahora cambiará ya que tendrán dependencia directa del módulo Común (figura 8), por lo que se obtendrá como resultado lograr cumplir con el requerimiento no funcional que presente la aplicación de poder eliminar en algún momento el Módulo 2 y para que los demás módulos cumplan con sus funcionamientos sin ningún problema.(7)

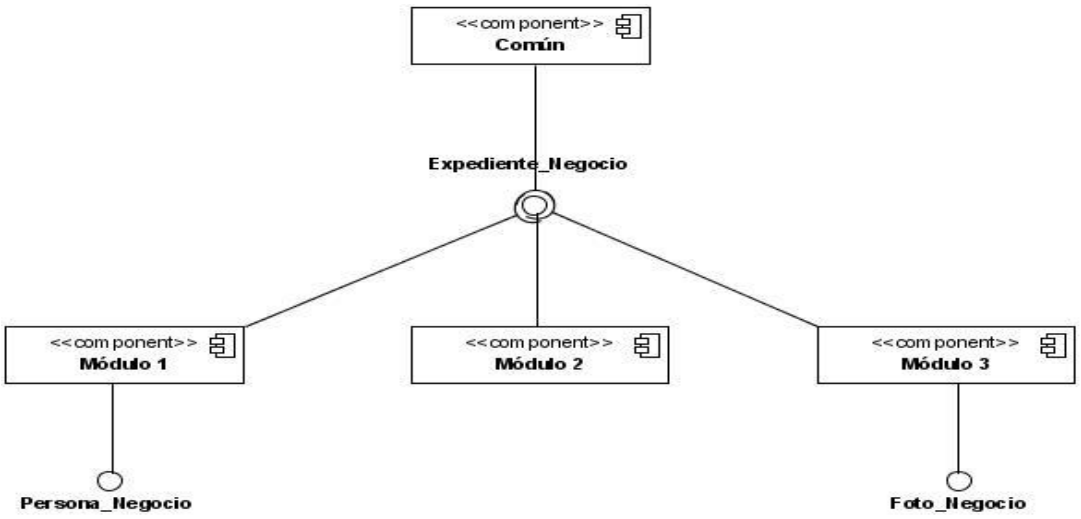

Figura 8.7 Representación de dependencia indirecta

Este módulo Común presenta la misma estructura básica de los módulos planteadas anteriormente, el cual no se identificó inicialmente, pero es de suma importancia su creación para eliminar dependencias directas entre los módulos específicos en las diferentes aplicaciones y moverlas hacia un módulo Común para cumplir con los requerimientos especificados en las diferentes aplicaciones que se encuentran enmarcadas dentro del dominio definido.

Todo este proceso es entre los módulos específicos de un subsistema por lo que el módulo Común creado pertenece también al subsistema. Además, este proceso también es aplicable para establecer este mecanismo de colaboración a nivel de subsistemas.

La comunicación entre módulos y subsistemas es a través de las interfaces de la capa de servicios de negocio, que son puntos de entradas a la lógica de negocio implementada en ellos. De esta forma se logra un mayor desacople entre ellos, al ocultar de tras de las interfaces las implementaciones reales de cada funcionalidad.

2.6 Conclusiones

Con el desarrollo de este capítulo se ha determinado la definición del dominio adecuada para las aplicaciones del proyecto Identificación y Control de Acceso de la UCI y los requerimientos de referencias identificados para este dominio. Además, se expuso la arquitectura que tendrá el proyecto, permitiendo de esta forma realizar el diseño de las capas lógicas de cada una de las aplicaciones guiadas por la arquitectura planteada anteriormente. Se ha explicado la propuesta de convenciones de nombres o estándares de codificaciones tanto para código fuente, como para recursos y los mecanismos de colaboración entre las distintas divisiones organizacionales, establecidas en la arquitectura del proyecto Identificación y Control de Acceso de la UCI.

Capítulo 3. Subsistemas y Módulos

3.1 Introducción

En este capítulo se determinarán los módulos que representan a cada subsistema del proyecto Identificación y Control de Acceso, así como una breve descripción de cada uno de ellos en cuanto a sus objetivos fundamentales a cumplir para obtener un subsistema adecuando a sus requerimientos funcionales. Se describirá cada subsistema en cuanto el tipo de aplicación que representará y los nodos físicos que utilizará para obtener un correcto funcionamiento en su trabajo. Se aplicarán los mecanismos de colaboración explicados en el capítulo anterior para establecer la comunicación entre los módulos pertenecientes a un mismo subsistema. Conjuntamente, se establecerá la comunicación entre los distintos subsistemas. Al mismo tiempo, se propone para los aspectos de la seguridad de la información en los subsistemas uno de los algoritmos de encriptación más usados; la encriptación asimétrica.

3.2 Subsistemas y Módulos

El proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI) está conformado por 5 subsistemas y 9 módulos en general; cada uno de los módulos se centra en tareas específicas del subsistema a que pertenecen, con el propósito de dar cumplimiento a los requerimientos funcionales que representa cada subsistema, estos son:

Subsistema de Control de Acceso.

El objetivo principal de este sistema consiste en controlar las entradas y salidas de las personas, vehículos y visitas que acceden diariamente a la Universidad de las Ciencias

Informáticas (UCI), ofreciendo un control estricto y una robusta seguridad evitando que ocurran violaciones o incidencias en la UCI. Este sistema está compuesto por tres módulos los cuales son:

- Módulo de Control de Acceso de Personas (MCAPersona), su objetivo fundamental es darle funcionamiento a los requerimientos funcionales del sistema en cuanto a gestionar los accesos puntuales de las personas pertenecientes a la Universidad de las Ciencias Informáticas.

- Módulo de Control de Acceso de Vehículos (MCAVehículos), su objetivo fundamental es darle funcionamiento a los requerimientos funcionales del sistema en cuanto a gestionar los accesos de los vehículos pertenecientes a la Universidad de las Ciencias Informáticas y vehículos externos a la UCI, autorizados por la jefatura principal de la institución.

- Módulo de Control de Acceso de Visitantes (MCAVisitantes), su objetivo fundamental es darle funcionamiento a los requerimientos funcionales del sistema en cuanto a gestionar los accesos de los visitantes a la Universidad de las Ciencias Informáticas.

Portal de Seguridad y Protección.

El Portal de Seguridad y Protección tiene como objetivo principal mantener al tanto a toda la comunidad de la Universidad de las Ciencias Informáticas (UCI) de temas de interés informativos referente a la Seguridad y Protección en la UCI. Este portal está compuesto por dos módulos los cuales son.

- Módulo de Seguridad y Protección (MSP), su objetivo fundamental es brindar una serie de información respecto a temas de la seguridad y la protección en la UCI.

- Módulo Administración (MAdmón), su objetivo fundamental es para que la Dirección de Seguridad y Protección administre lo referente a los requerimientos funcionales del sistemas en cuanto a la administración de la información, a la administración de los enlaces en otros sitios, administrar los reportes y la documentación que van hacer mostrados o recibidos y administrar el plan de la guardia llevando un estricto control de la misma en cuanto a su planificación diaria y mensual.

Acreditación.

El objetivo principal de este subsistema es realizar un conjunto de actividades que se deben cumplir a la hora de acreditar a una persona específica, entre estas actividades se encuentra la confección de su credencial. Estas actividades están encaminadas a ejecutar un grupo de procesos para la correcta acreditación de todo el personal que esta presenta en la Universidad de las Ciencias Informáticas (UCI). Este sistema esta compuesta por solo un módulo que es.

- Módulo de Acreditación (MAcreditación) su objetivo fundamental es darle funcionamiento a los requerimientos funcionales del subsistema donde se encuentran; gestionar grupo, gestionar Credenciales, mostrar un historial de credenciales que ha tenido una persona perteneciente a la Universidad de las Ciencias Informáticas, mostrar un listado de credenciales por cada grupo y imprimir credenciales.

Registro de Incidencias.

El sistema de Registro de Incidencias para la Universidad de las Ciencias Informáticas (UCI), su objetivo principal es administrar toda la información referente a las incidencias ocurridas en la UCI. Mediante el mismo se podrá crear un expediente para cada una de las personas que comentan incidencias, se imprimirán reportes en determinados intervalos de tiempo, realizando búsquedas teniendo en cuenta algunos parámetros de entrada. Este sistema está compuesto por un solo módulo que es.

- Módulo General Reportes (MGReportes), este módulo es el encargado de brindarle funcionamiento a todos los requerimientos funcionales del subsistema, donde se encuentran; registrar las incidencias cometidas por las personas, crear los expedientes de cada persona que han cometidos incidencias, organizar las incidencias por tipos y crear diferentes reportes teniendo en cuenta algunos parámetros de búsquedas o sin parámetros de búsquedas e imprimir los diferentes reportes en caso necesario.

Gestión de Fotos.

El subsistema Gestión de Fotos tiene como objetivo fundamental que las fotos de las personas que se manejan en la Universidad de las Ciencias Informáticas (UCI) tengan el formato, la organización y la seguridad requerida, además de permitir un historial de fotos por persona y por recursos. También brindará la posibilidad de insertarle meta datos a las fotos y leerlos de forma tal de que en caso de alguna contingencia que altere el orden de los datos se pueda saber a quien pertenece cada una de las fotos. Este sistema está compuesto por dos módulos que son.

- Módulo Fotos (MFotos), su objetivo principal es crear un servicio web donde permitirá brindar a otros sistemas las operaciones que se realizarán en la gestión de fotos, entre ellas están insertarle meta datos, proporcionarle dimensiones y resolución a las imágenes, crear un historial de fotos para cada persona, etc.

- Módulo de administración (MFAdmón), su objetivo principal es administrar y configurar los datos con que se gestionarán las fotos.

3.3 Descripción de los Subsistemas

En la descripción de los diferentes subsistemas del proyecto Identificación y Control de Acceso se dará a conocer el tipo de aplicación que representa cada subsistema; ya sea una

aplicación web o una aplicación de escritorio y la distribución física que cada uno de ellos presentará.

3.3.1 Subsistema de Control de Acceso

El Subsistema de Control de Acceso es una aplicación de escritorio, la cual contiene el gestor de base de datos PostgreSql que almacena toda la información necesaria de la aplicación y una base de datos local DB4O que estará situada en el mismo nodo de la aplicación, esta base de datos local su propósito fundamental es, en caso de fallar la conexión entre la aplicación y el servidor de base de datos principal, tener donde almacenar y buscar información. La aplicación utiliza un servicio web que se encuentra publicado en la UDDI de la Universidad de las Ciencias Informáticas donde el mismo brinda una serie de funcionalidades muy útil para nuestro subsistema y para agilizar el trabajo con la aplicación, se utilizará el dispositivo Optimus-S lo cual es un lector portátil de código de barra con el objetivo de controlar los accesos. (Ver Figura 3.1)

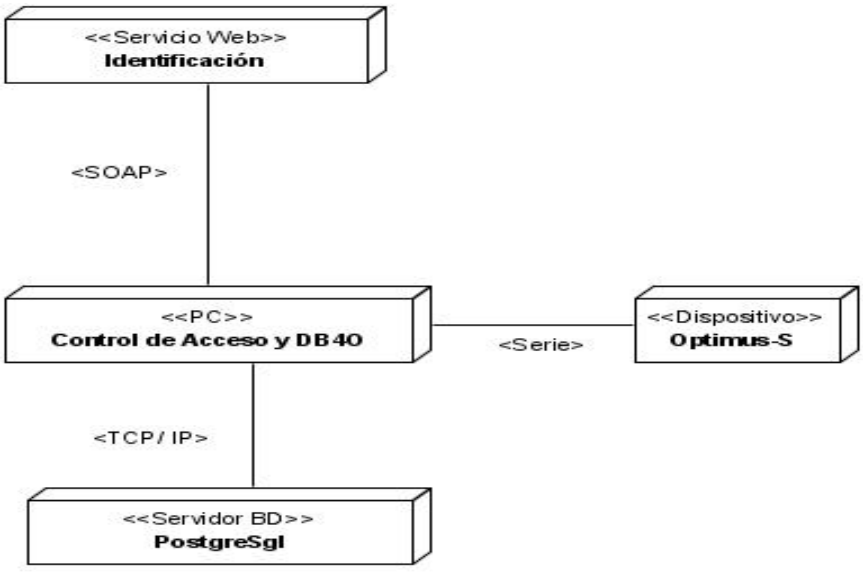

Figura 9.1 Diagrama de Despliegue del Sub. Control de Acceso

- **Servicio Web Identificación:** Este nodo representa al Servicio Web de Identificación el cual nos posibilita una serie de funcionalidades coherente a las credenciales de las personas de la Universidad de las Ciencias Informáticas.

- **PC:** En este nodo se ubicará la aplicación y la base de dato local DB4O.

- **Servidor DB:** Este nodo contiene la base de dato principal de la aplicación.

- **Dispositivo:** Este nodo representa al dispositivo Optimus-S el cual será de gran utilidad para agilizar el trabajo en la aplicación.

Descripción de elementos e interfaces de comunicación

<<SOAP>>: Este es el protocolo usado para el intercambio de información cuando se consume el servicio web. Este es empleado entre la comunicación del servicio web (Identificación) y la aplicación.

<<SERIE>>: Este es una interfaz de comunicaciones de datos digitales utilizada frecuentemente por computadoras o periféricos, donde la información es transmitida bit a bit enviando uno solo a la vez. Este es empleado entre la comunicación de nuestra aplicación y el dispositivo Optimus-S.

<<TCP/ IP>>: Es la base de Internet para que sirve para enlazar computadoras con diferentes sistemas operativos y proporciona transmisiones fiables de datos sobre la red. Este es empleado entre la comunicación del nodo que tiene la base de dato principal de la aplicación y el nodo que tiene la base de dato local.

3.3.2 Portal de Seguridad y Protección

El Portal de Seguridad y Protección (Portal SP) es una aplicación web, el cual contiene el gestor de base de datos PostgreSql que almacena toda la información necesaria del Portal. El Portal se alimentará de los servicios web que nos brinde los subsistemas de Acreditación y Control de Acceso con el objetivo de utilizar algunas de sus funcionalidades, las mismas contienen una serie de información o datos muy importantes para lograr un correcto funcionamiento en el Portal. El Portal posibilita que diversos usuarios visiten las informaciones expuestas en él de manera segura para que no exista ningún tipo de ataque por usuarios maliciosos y será administrada la aplicación por el especialista donde este podrá acceder a funcionalidades que los usuarios no tendrán permiso para acceder a ellas. Se usará también un dispositivo para imprimir información de gran interés por parte de la jefatura de Seguridad y Protección de la Universidad de las Ciencias Informáticas. (Ver Figura 3.2).

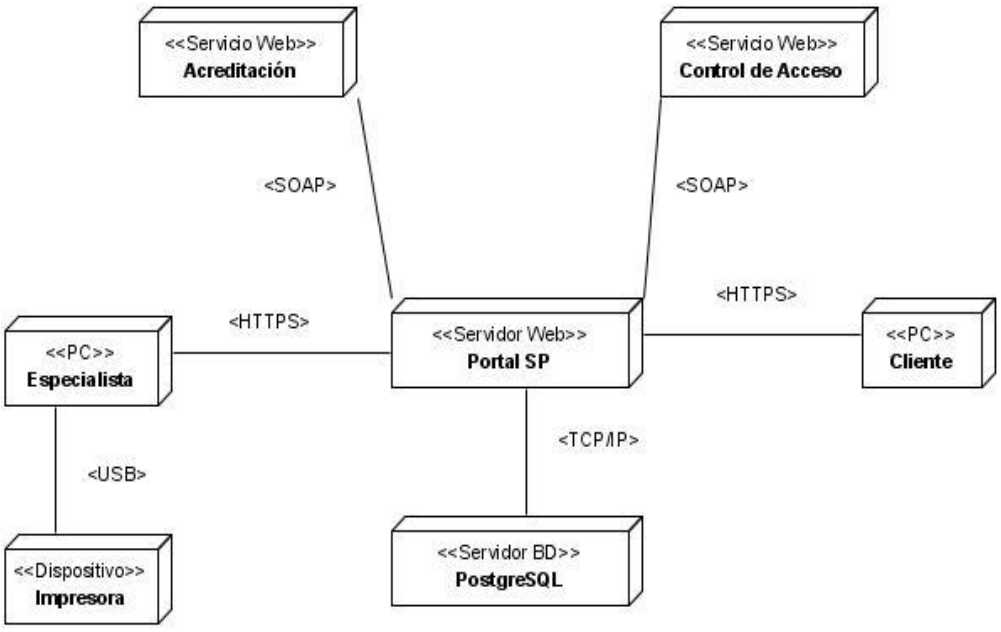

Figura 10.2 Diagrama de Despliegue del Portal de Seguridad

- **Servicio Web Acreditación:** Este nodo representa al Servicio Web del subsistema de Acreditación el cual brindará una serie de funcionalidades con los datos de las personas acreditadas en la Universidad de las Ciencias Informáticas.

- **Servicio Web Control de Acceso:** Este nodo representa al Servicio Web del subsistema de Control de Acceso el cual brindará una serie de funcionalidades que nos permitirá llevar un control estricto y una robusta seguridad, evitando que ocurran violaciones o incidencias.

- **Servidor Web:** Este nodo contiene la aplicación web (Portal SP).

- **Servidor de Base de Datos:** Este nodo contiene la base de datos del Portal, la cual almacenará toda la información referente al mismo.

- **PC Cliente:** Este nodo representa la computadora desde la cual se conecta todos los clientes que quieran visitar el portal.

- **PC Especialista:** Este nodo representa la PC donde el especialista administrara el Portal.

- **Dispositivo Impresora:** Este nodo representa a una impresora que es quien imprimirá los diferentes reportes e informaciones que se necesite.

Descripción de elementos e interfaces de comunicación

<<SOAP>>: Este es el protocolo usado para el intercambio de información cuando se consume el servicio web. Este es empleado entre la comunicación de los servicios web (subsistema de Acreditación, subsistema de Control de Acceso) y el Portal.

<<HTTPS>>: Este es el protocolo de transferencia en la arquitectura cliente servidor, adicionando encriptación de los datos, para asegurar el tráfico de los mismos. Este protocolo es empleado en la comunicación entre el nodo que tiene el Servidor Web y de Base de Datos y los nodos que contienen a la PC del especialista y la PC de los Clientes que se conectarían al Portal.

<<USB>>: Es un puerto que sirve para conectar periféricos a una computadora e incluye la transmisión de energía eléctrica al dispositivo conectado, él hace sencillo el poder agregar más de un dispositivo a una computadora. Este es empleado entre la comunicación de la PC del Especialista y el dispositivo de impresión (Impresora).

<<TCP/ IP>>: Es la base de Internet para que sirve para enlazar computadoras con diferentes sistemas operativos y proporciona transmisiones fiables de datos sobre la red. Este es empleado entre la comunicación del nodo que tiene la base de dato y el nodo que tiene a la aplicación.

3.3.3 Subsistema de Acreditación

El subsistema de Acreditación es una aplicación de Escritorio, el cual contiene el gestor de base de datos PostgreSql que almacena toda la información coherente al subsistema. La aplicación se alimentará de una serie de servicios web que están publicados en la UDDI de la Universidad de las Ciencias Informáticas con el objetivo de utilizar una serie de funcionalidades e información que estos servicios web nos brindan. Para darle un correcto y adecuando funcionamiento a nuestro subsistema se utilizará un dispositivo para imprimir las credenciales de las personas de la Universidad de las Ciencias Informáticas. (Ver figura 3.3)

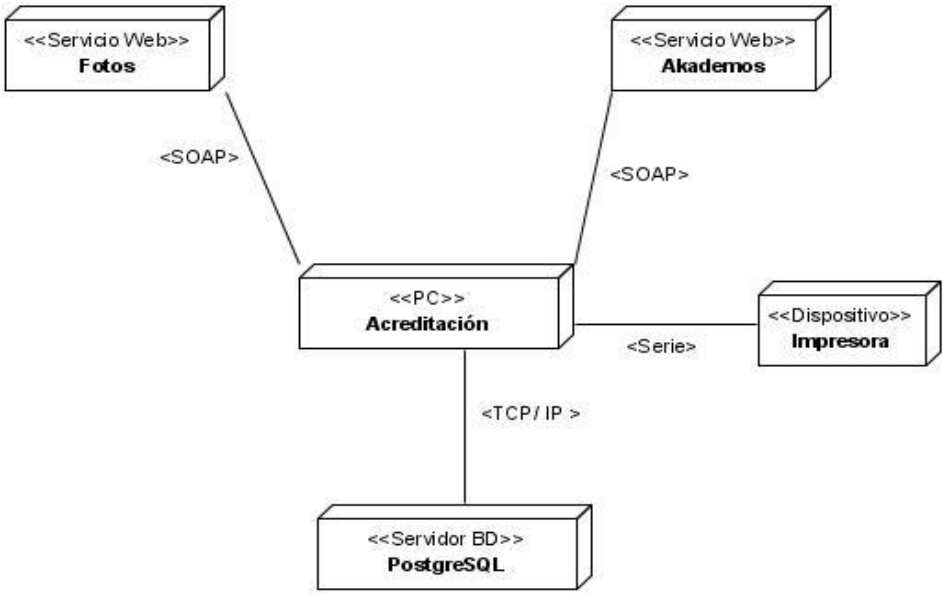

Figura 11.3 Diagrama de Despliegue de Acreditación

- **Servicio Web Fotos:** Este nodo representa al Servicio Web del subsistema de Gestión de Foto el cual es utilizado en el sistema para obtener las fotos de una persona determinada para crearle su credencial.

- **Servicio Web Akademos:** Este nodo representa al Servicio Web de Akademos el cual contiene todos los datos de las personas que ingresan a la Universidad de las Ciencias Informáticas.

- **PC Especialista:** En este nodo es donde se ubicará la aplicación y es donde la especialista hará uso de la misma.

- **Servidor de Base de Datos:** Este nodo contiene la base de datos del subsistema.

- **Dispositivo Impresora:** Este nodo representa a una impresora que es quien imprimirá las credenciales de las personas de la Universidad de las Ciencias Informáticas.

Descripción de elementos e interfaces de comunicación

<<SOAP>>: Este es el protocolo usado para el intercambio de información cuando se consume el servicio web. Este es empleado entre la comunicación de los servicios web (subsistema de Acreditación, subsistema de Control de Acceso) y el Portal.

<<TCP/IP>>: Es la base de Internet que sirve para enlazar computadoras con diferentes sistemas operativos y proporciona transmisiones fiables de datos sobre la red. Este es empleado entre la comunicación del nodo que tiene la base de dato principal de la aplicación y la PC del Especialista que es donde estará situada la aplicación.

<<Puerto de Serie>>: Este es una interfaz de comunicaciones de datos digitales utilizada frecuentemente por computadoras o periféricos, donde la información es transmitida bit a bit enviando uno solo a la vez. Este es empleado entre la comunicación de la PC del Especialista que es donde estará situada la aplicación y el dispositivo de impresión (Impresora).

3.3.4 Registro Incidencias

El subsistema Registro de Incidencias es una aplicación web, el cual contiene el gestor de base de datos PostgreSql que almacena toda la información coherente al subsistema. La aplicación solamente podrán acceder a ella dos tipos de usuarios por su gran contenida de información confidencial, ellos son; el Especialista de SP que es quien administrará el sitio y el Oficial Operativo que es quien registra las incidencias cometidas por las personas de la Universidad de las Ciencias Informáticas, se utilizará un dispositivo para imprimir diferentes tipos de reportes, esta operación solamente puede realizarla el Especialista. (Ver Figura 3.4).

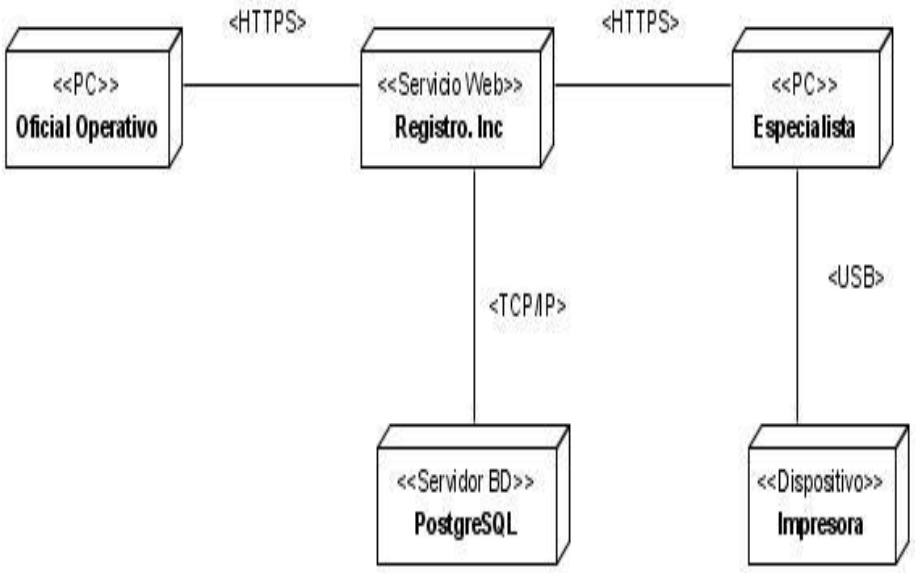

Figura 12.4 Diagrama de Despliegue del Registro.Inc

- **Servidor Web** Este nodo contiene la aplicación web.

- **Servidor de Base de Datos:** Este nodo contiene la Base de datos.

- **PC Especialista:** En este nodo es donde el Especialista administrará la aplicación web e imprimirá los diferentes tipos de reportes.

- **PC Oficial Operativo:** En este nodo es donde el Oficial de Operaciones registrará las incidencias cometidas.

- **Dispositivo Impresora:** Este nodo representa a una impresora que es quien imprimirá los diferentes reportes que necesite el Especialista de SP.

Descripción de elementos e interfaces de comunicación

<<HTTPS>>: Este es el protocolo de transferencia en la arquitectura cliente servidor, adicionando encriptación de los datos, para asegurar el tráfico de los mismos. Este protocolo es empleado en la comunicación entre el nodo que tiene el Servidor Web y de Base de Datos a los nodo que representan a la PC del Especialista y a la PC del Oficial Operativo.

<<USB>>: Es un puerto que sirve para conectar periféricos a una computadora e incluye la transmisión de energía eléctrica al dispositivo conectado, hace sencillo el poder agregar más de un dispositivo a una computadora. Este es empleado entre la comunicación de la PC del Especialista y el dispositivo de impresión (Impresora).

<<TCP/IP>>: Es la base de Internet que sirve para enlazar computadoras con diferentes sistemas operativos y proporciona transmisiones fiables de datos sobre la red. Este es empleado entre la comunicación del nodo que tiene la base de dato y el nodo que contiene a la aplicación web.

3.3.5 Subsistema de Gestión de Fotos

El subsistema Gestión de Fotos es un servicio web que brindará información a los sistemas clientes que se conecten a él, cuenta con el gestor de base de datos PostgreSql que almacena toda la información referente a este subsistema. El subsistema será administrado por Administrador del Sistema. (Ver figura 3.5).

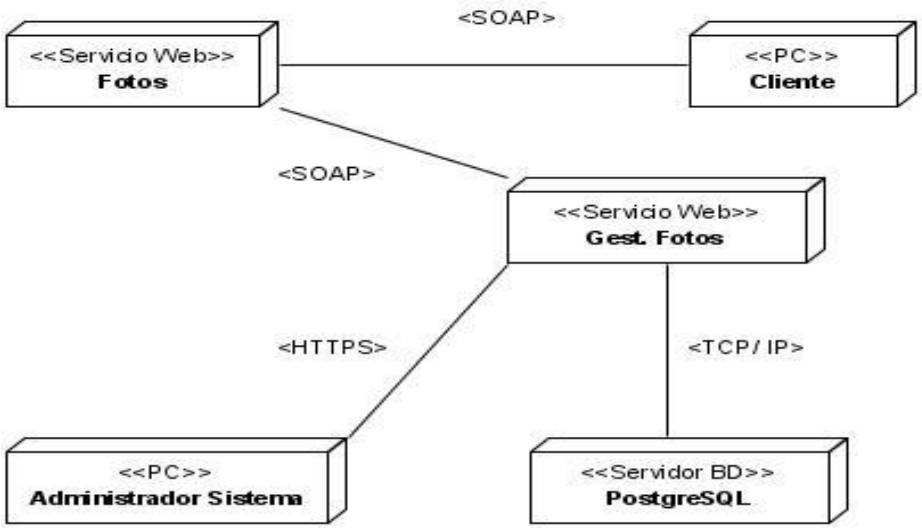

Figura 13.5 Diagrama de Despliegue de Gest. Fotos

- **Servicio Web Fotos:** Este nodo alojará el servicio web perteneciente al subsistema.

- **PC Clientes:** Son las computadoras con aplicaciones que se conectan al servicio web para consumirlo.

- **Servidor Web:** Este nodo contiene la aplicación web (SGF).

- **Servidor BD:** Este nodo contiene la base de datos (BD-PostgreSql) de la aplicación.

- **PC Administrador Sistema*:*** Este nodo representa la computadora desde la cual el Administrador del Sistema realizará su función, que no es más que administrar la aplicación.

Descripción de elementos e interfaces de comunicación

<<SOAP>>: Este es el protocolo usado para el intercambio de información cuando se consume el servicio web. Este es empleado entre el Servidor Web y de Base de Datos del Módulo de Administración y cualquier otro cliente que solicite el servicio web.

<<HTTPS>: Este es el protocolo de transferencia en la arquitectura cliente servidor, adicionando encriptación de los datos, para asegurar el tráfico de los mismos. Este protocolo es empleado en la comunicación entre el Servidor Web y la PC del Administrador del Sistema.

<<TCP/IP>>: Es la base de Internet que sirve para enlazar computadoras con diferentes sistemas operativos y proporciona transmisiones fiables de datos sobre la red. Este es empleado entre la comunicación del nodo que tiene la base de dato y el nodo que contiene a la aplicación web.

3.4 Comunicación entre módulos

En el proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI) para lograr un correcto funcionamiento, muchos de los módulos necesitan funcionalidades de otros módulos, por lo que surge una gran necesidad de comunicación a través de dependencias directas; la misma se establecerá entre módulos pertenecientes a un mismo subsistema con el objetivo de lograr una mayor organización en el trabajo, estas comunicaciones se realizarán a través de las interfaces en la capa de negocio de cada uno de los módulos, con dependencias directas o indirectas según han sido planteadas en el capítulo anterior.

3.5 Comunicación entre Subsistemas

La comunicación entre los subsistemas del proyecto Identificación y Control de Acceso se realizará a través de servicios web, cada uno de ellos presentan su propia base de datos y tendrán expuestas una serie de funciones que facilitarán el intercambio de información entre sí. A pesar de las diferencias en la tecnología y los lenguajes de desarrollo que existen entre los subsistemas; no se ocasionaría ninguna barrera para lograr una correcta comunicación entre los subsistemas a través de los servicios web. (Ver Figura 3.6)

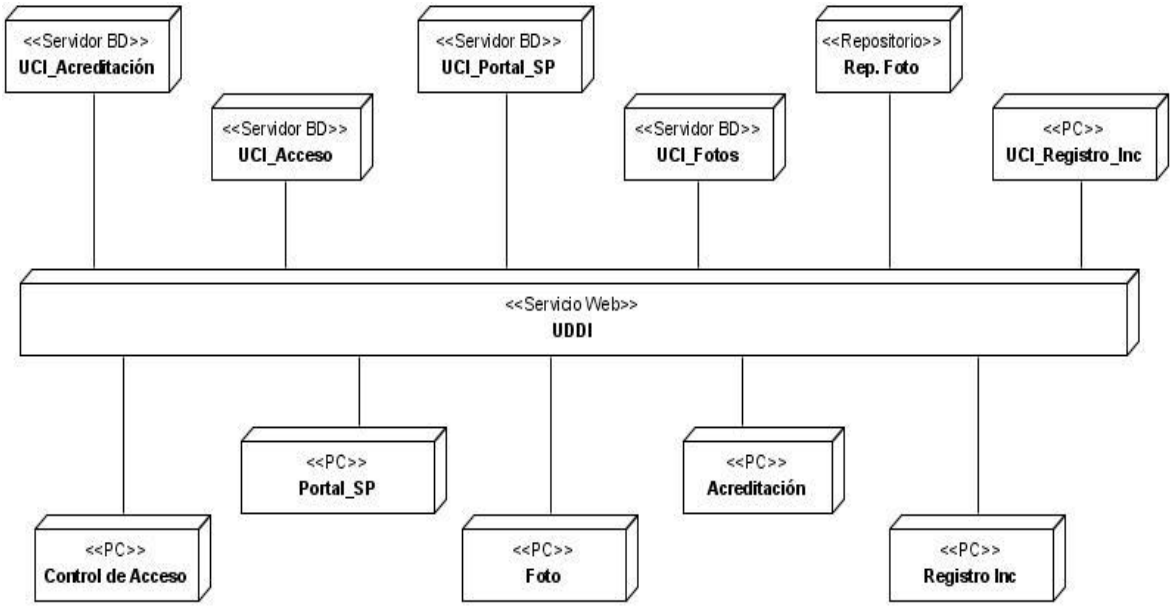

Figura 14.6 Comunicación entre subsistemas

3.6 Seguridad en la información de los Subsistemas

La Encriptación es una medida de seguridad muy utilizada para que al momento de almacenar o transmitir informaciones sensibles como contraseñas, números de identificación legal, número de tarjetas de créditos, códigos de barras, reportes administrativos, etcétera; para que estos no puedan ser obtenidos fácilmente por terceros. La encriptación hace uso de diversas fórmulas matemáticas con el propósito de transformar la información.

En los subsistemas del proyecto Identificación y Control de Acceso de la UCI se manejará un gran volumen de información muy valiosa y datos confidenciales, los mismos no están aislados a los ataques o intersecciones por atacantes en la red al ser compartidas estas informaciones entre los subsistemas o entre aplicaciones externas. Para asegurar la transmisión segura de información en los subsistemas se utilizará la encriptación de los datos, lo cual se usará la encriptación asimétrica que es uno de los algoritmos de encriptación más usados, este requiere de dos claves, una clave privada; que es conocida por una única persona, manteniéndose en secreto y la otra llamada clave pública que se le da a conocer a

todos, la cual soluciona la distribución de la clave privada de los sistemas mediante un canal seguro, también requiere de una fórmula matemática o algoritmo matemático lo más compleja posible de reproducir, donde esta fórmula desmenuza la información; el contenido de los mismos lucirían como un conjunto de caracteres extraños, sin ningún sentido o lógica de lectura. Al introducir los datos se genera la clave pública y privada necesaria, la clave pública podrá ser distribuida sin ningún inconveniente entre todos los interlocutores, la privada deberá ser muy bien guardada. Cuando se envía una información o un dato, el emisor busca la clave pública de cifrado del receptor y una vez que dicho mensaje llega al receptor, éste se ocupa de descifrarlo usando su clave privada.

La utilización de algoritmos matemáticos que presentan mayor complejidad al ser programados y manipulados dentro de las aplicaciones permitirá que la información no sea descifrada con elementos sencillos.

3.7 Conclusiones

En el desarrollo de este capítulo se expusieron los diferentes subsistemas que presenta el proyecto Identificación y Control de Acceso, así como cada uno de sus módulos. Además, se realizó una descripción de cada subsistema definiendo el tipo de aplicación que representará y la distribución de cada uno de los nodos físicos que contiene. Igualmente, se especifica como se va a realizar la comunicación entre los módulos pertenecientes a un mismo subsistema y la integridad entre los mismos. Finalmente, se abordó brevemente la encriptación de la información con el objetivo de manejar los aspectos de la seguridad de la información en los subsistemas.

Conclusiones Generales

A raíz de los nuevos requerimientos funcionales incorporados al proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas y la emigración a software libre por políticas de la UCI, se llevó a cabo un estudio de la problemática planteada anteriormente. Producto de la investigación realizada, se pudo proponer la arquitectura del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas; definiendo en la propuesta las herramientas, tecnologías, framework, patrones de arquitectura, convenciones de código, entre otros, a utilizar; que a su vez dan cumplimiento al objetivo de la investigación.

Con el fin de dar cumplimiento al objetivo general del presente trabajo se definieron las tareas de la investigación, con las que se arribó a las siguientes conclusiones:

- Se realizó un estudio de las metodologías, técnicas y tendencias actuales, lo que permitió la selección de las herramientas que más se ajustan a las necesidades en el desarrollo de las diferentes aplicaciones que pertenecen al proyecto Identificación y Control de Acceso de la UCI.

- Se analizaron bibliografías referentes a temas de arquitecturas desarrolladas en software libre, lo que permitió conocer el estado del arte asociado al tema.

- Se evaluaron proyectos reales desarrollados en software libre para obtener información referente a los aspectos de organización, estándares de código, comunicación entre sus subsistemas y módulos, con el objetivo de definir la arquitectura adecuada para el proyecto Identificación y Control de Acceso de la UCI.

- Se propuso el diseño de la arquitectura para el proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI) presentando los estilos arquitectónicos, las tecnologías, los lenguajes, las herramientas de desarrollo, entre

otros, que más se ajustan al documento oficial de arquitectura de la Dirección de Informatización (Arquitectura para los Sistemas que Conforman la Intranet Universitaria) de la Universidad de las Ciencias Informáticas.

Recomendaciones

A partir del trabajo realizado se sugieren las siguientes recomendaciones con el objetivo de mejorar la arquitectura del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI).

- Probar los resultados de esta investigación, o sea que sea aplicada la arquitectura propuesta a los diferentes subsistemas y módulos del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI).

- Para dar cumplimiento al punto anterior se recomienda que los arquitectos del proyecto Identificación y Control de Acceso de la Universidad de las Ciencias Informáticas (UCI) apliquen la arquitectura propuesta. De esta forma se podrá determinar que los resultados obtenidos sean los deseados.

- Se recomienda continuar con el refinamiento constante de la arquitectura propuesta durante el ciclo de desarrollo, pues con el continuo desarrollo de las tecnologías, la arquitectura propuesta deberá adecuarse al avance tecnológico venidero.

- El sistema con la arquitectura que se propone exige para un correcto funcionamiento, recursos del hardware, a pesar de que la tecnología de desarrollo que se utilizarán en la arquitectura propuesta es libre, por lo que se recomienda obtener los equipos adecuados para lograr el mejor resultado de la propuesta que se brinda en la investigación.

Referencias Bibliográficas

1. CRESPO, J. M. C. *Fisterra 2. Software Libre para gestión empresarial.* 2004, Disponible en: http://community.igalia.com/twiki/pub/Fisterra/ProjectArticles/osic2004_slides.pdf.

2. *Propuesta Arquitectura SIGEP (Sistema de Gestión Penitenciaria)*. En 2008.

3. ZUKOWSKI, J. *Java 2.* 2003.

4. PÉREZ, J. E. *Introducción de CSS.* 2008, Disponible en: http://www.librosweb.es/css.

5. GARCÍA, J. F. V. G. R. *Curso completo de HTML* Disponible en: http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html.pdf.

6. PÉREZ, J. E. *Introducción a JavaScript.* 2008, Disponible en: http://www.librosweb.es/javascript.

7. RIVERO, L. A. P. I. P. *ArBaWeb: ARQUITECTURA BASE SOBRE LA WEB* UCI, 2007.

8. GÓMEZ, C. Á. *Introducción al ambiente de desarrollo Eclipse* Disponible en: http//www.geocities.com/neos_software/articles/ambientejee5.pdf.

9. RAMÍREZ, A. *Subversión* Disponible en: http://polaris.dit.upm.es/~rubentb/docs/subversion/TutorialSubversion/index.html.

10. FERNÁNDEZ, T. J. R. P. R. T. *Introducción a PostgreSQL* Disponible en: http://users.servicios.retecal.es/tjavier/presbbdd/out-htmls/index.html.

11. SÁNCHEZ, J. *MySql Quía Rápida* Disponible en: http://64.233.169.104/search?q=cache:DqQ56OHIZIIJ:www.jorgesanchez.net/bd/mysql.pdf+PDF+de+MySQL&hl=es&ct=clnk&cd=8&gl=cu&lr=lang_es.

12. GERBER, C. L. C. L. G. F. M. F. *Bases de datos orientadas a objetos.* 2007,

13. POTENCIER, F. Z. F. *Symfony, la guía definitiva.* 2008, Disponible en: http://www.librosweb.es/symfony.

14. ALBIOL, F. R. *INTRODUCCIÓN A HIBERNATE1.* 2003.

15. BRUN, R. E. XML y Servicios Web. En *Sevilla. 2002*

Bibliografía

Arquitectura para los Sistemas que Conforman la Intranet Universitaria. 2007.

Bass L., Clements P. y Kazman R. (2003). Software Architectura in Practice. Second Edition. Camegie Mellon University: Addison Wesley.

Camacho E, Caradeso F. y Nuñez G. (2004). Guía de estudio de la Arquitectura de Software.

DAVID GALLARDO, E.B., ROBERT MCGOVERN. Eclipse In Action: A Guide for Web Developers. 2003.

FLOWER, M. Patter of Enterprise Application Architecture. 2002. p.

Modelado de Sistemas con UML 2002. http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/index.html.

Pressman R. (2005). Ingeniería del Software. Un enfoque practico. Parte 1. La Habana Cuba: Félix Verala.