UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 9

TÍTULO: "PROPUESTA DE LA ARQUITECTURA DE SOFTWARE PARA EL PROGRAMA NACIONAL DE INFORMATIZACIÓN DEL CONOCIMIENTO GEOLÓGICO."

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

AUTORES: DANAIS SANTOS PÉREZ

DARIS SÁNCHEZ BAÑOS

TUTOR: ING. KARLEN TRIMIÑO PÉREZ

CO-TUTOR(ES): MSC. MABEL PÉREZ CAMPOS

DR. HÉCTOR RODRÍGUEZ

CIUDAD DE LA HABANA, 2 DE JULIO 2008.

¨AÑO 50 DE LA REVOLUCIÓN¨

"Hay grandes hombres que hacen a todos los demás sentirse pequeños. Pero la verdadera grandeza consiste en hacer que todos se sientan grandes."

Charles Dickens

DEDICATORIA

A mi mamá Ileana, mi papá Pedro , mi hermanita Dilena y mi novio por darme su amor en todo momento y confiar en mí...los adoro y los adoraré por siempre.

Danais

A mi mamá Ileana, mi papá Odelis y a mi hermana Idelis por quererme tanto y confiar en mí plenamente...los quiero.

Daris

AGRADECIMIENTOS

A nuestros padres por estar siempre junto a nosotras, por brindarnos todo su amor y apoyo durante nuestras vidas y por todos sus esfuerzos para que este trabajo fuera una realidad.

A nuestro tutor Karlen Trimiño Pérez, que nos dedicó parte de su tiempo a orientarnos y guiarnos a culminar satisfactoriamente este trabajo.

A Jorge Infante Osorio por dedicarnos parte de su tiempo, pues nos ayudó a esclarecer algunas dudas para que esta tesis saliera bien.

De Danais:

A mi mamá Ileana que me ha apoyado en todo momento y ha confiado en mí desde que era una niña, por enseñarme a saber por qué luchamos cada momento de nuestra vidas, por darme su amor y creer siempre en que yo sí podía salir adelante. Por ti estoy aquí y te lo agradeceré por siempre. Te adoro.

A mi papá que aunque no habla mucho siempre estuvo ahí apoyando a mi mamá, ayudándome en todo lo que me hacía falta y dándome su cariño. Te quiero mucho papito.

A mi hermanita Dilena que ha sido mi inspiración, para que en un futuro sepa que no hay barrera que la voluntad no rompa, quiero que siempre lleve esas palabras en su corazoncito, eres mi vida mi tata.

A mi tía Maipe que siempre estuvo apoyándome a mí y a mi mamá en lo que nos hiciera falta.

A mi novio Reinier por apoyarme tanto y darme la fuerza y confianza para seguir adelante, en aquellos momentos que sentía que el mundo se me cerraba. Por ser una de las pocas personas que me entiende tal y como soy.

A todos mis verdaderos amigos por su apoyo incondicional.

AGRADECIMIENTOS

A todas aquellas personas que siempre me ayudaron desinteresadamente y me ayudaron a levantarme cuando estaba sin fuerzas; eso nunca lo olvidaré, se lo agradezco desde lo más profundo de mi corazón.

De Daris:

A mi mamá Ileana por estar siempre conmigo, apoyarme, quererme tanto y por confiar siempre en mí. Espero que estés tan feliz como yo.

Te quiero.

A mi papá Odelis por también darme su apoyo, creer en mí y estar siempre cuando te he necesitado.

A mi negra linda (mi hermana Idelis) que espero ser un gran ejemplo para ti y que siempre tengas presente que te quiero muchísimo.

A mimí Bertha (mi abuelita) que siempre esta pensando en mí y que aunque algunas veces peleo mucho con ella, la quiero.

A mi novio Darian por apoyarme en este momento tan importante para mí.

A mis tías, tíos y mis primos que siempre están al pendiente y preocupándose por mí, en fin a toda mi familia que me ha apoyado tanto.

A todas mis amistades que de una forma u otra me han apoyado y a mi antiguo grupo que siempre los recordaré a todos.

A tres persona que aunque no estén aquí presentes se que estarían muy orgullosas de mí (mi abuela Niria, mi abuelo Idelio y mi bisabuela Angelina).

A Eylin Hernández Luque, a Danais Santos Pérez por ser mi compañera de tesis y ayudarme en todo momento.

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos a la Facultad 9 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Danais Santos Pérez	**Daris Sánchez Baños**	**Ing. Karlen Trimiño Pérez**
_____	_____	_____
Firma del autor	Firma del autor	Firma del tutor

OPINIÓN TUTOR

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Propuesta de Arquitectura para el Programa Nacional de Informatización del Conocimiento Geológico.

Autores: Danais Santos Pérez
Daris Sánchez Baños

El tutor del presente Trabajo de Diploma considera que durante el período que se evalúa las estudiantes mostraron una alta independencia en la investigación del tema y en la confección del mismo. Muy constantes y laboriosas, siempre en comunicación con su tutor. Muy responsables, entregando en tiempo toda la información pedida, en los cortes de tesis mostraron el avance de su trabajo y cumplieron con calidad de todas las tareas que se les orientaron.

El trabajo presenta una alta calidad tanto científica, como técnica. Se realizó una investigación exhaustiva de un tema que requiere mucha dedicación. Obteniéndose un estudio detallado de una buena propuesta de arquitectura para el Programa Nacional de Informatización del Conocimiento Geológico, cumpliéndose así los objetivos propuestos. El documento está correctamente estructurado en capítulos bien delimitados, presentando buena ortografía y redacción. Han buscado bibliografía actual y con gran grado científico.

Con la propuesta de esta arquitectura se espera que se obtenga un producto de calidad, contribuyendo esto a la informatización de la sociedad cubana. Por todo lo anteriormente expresado considero que las estudiantes han vencido las tareas previstas para ejercer como Ingenieras Informáticas y propongo que se le otorgue al Trabajo de Diploma, la calificación de Excelente (5 puntos).

_____

Firma del Tutor

RESUMEN

En Cuba se quiere informatizar muchas Empresas e Instituciones, como es el caso de la Oficina Nacional de Recursos Minerales. Esto se puede llevar a cabo con el apoyo que brinda la Universidad de las Ciencias Informáticas. En este trabajo se brinda una propuesta de la arquitectura de software para informatizar la Oficina Nacional de Recursos Minerales, basando la investigación en las arquitecturas empresariales.

Este trabajo centra su atención hacia el desarrollo de una propuesta arquitectónica escalable que cumpla con todos los requisitos necesarios para el funcionamiento del Programa Nacional de Informatización del Conocimiento Geológico (PNICG). La arquitectura de software propuesta debe permitir a los desarrolladores del proyecto implementar todos los procesos de una forma relativamente fácil con tiempos de desarrollo cortos, aumentando la productividad y eficiencia del mismo.

PALABRAS CLAVES

Arquitectura de software.

ÍNDICE

Introducción

Antecedentes

El primer estudio en que aparece la expresión “arquitectura de software” en el sentido en que hoy se conoce es sin duda el de Perry y Wolf en 1992, aunque el trabajo se fue gestando desde 1989. En él, los autores proponen concebir la arquitectura de software por analogía con la arquitectura de edificios, una analogía de la que luego algunos abusaron, otros encontraron útil y para unos pocos ha devenido inaceptable.

Todavía no se ha escrito una historia aceptable que reseñe escuetamente la prehistoria de la arquitectura de software a principios de los 90. En este estudio se ha optado, más bien, por inspeccionar las fuentes más de cerca, con el objeto de señalar y comprender mejor por qué algunas ideas que surgieron hace cuatro décadas demoraron un cuarto de siglo en materializarse.

Si bien la arquitectura de software remonta sus antecedentes al menos hasta la década de 1960, su historia no ha sido tan continua como la del campo más amplio en el que se inscribe, la ingeniería de software. Se trata entonces de una práctica joven, de apenas unos doce años de trabajo constante, que en estos momentos experimenta una nueva ola creativa en el desarrollo cabal de sus técnicas en la obra de Rick Kazman, Mark Klein, Len Bass y otros metodólogos.

En la última década cambió la visión que los desarrolladores tenían de los sistemas de software. Esta nueva visión se llamó "arquitectura". Desde los pequeños sistemas hasta los más complejos poseen una estructura y un comportamiento que los hace clasificables según su "arquitectura". Este nuevo aspecto hace posible el estudio de sistemas ya implementados así como el desarrollo de nuevos, según la categoría del problema que resuelven y el tipo de "arquitectura".

Los distintos niveles de abstracción de la funcionalidad de los sistemas, están asociados con diferentes aspectos y componentes de su "arquitectura". Esta asociación se manifiesta en forma clara en las distintas etapas del proceso de desarrollo de software.

Desde que surgió el tema de arquitectura, diversas fueron las personas que vieron el tema con otra perspectiva, pues se necesitaba desarrollar un modelo de arquitectura completamente diferente a lo que existía, un modelo denominado arquitectura empresarial que proporciona una organización de todos los procesos de negocio y a la vez garantiza mejoras en la calidad, eficacia y responsabilidad en la empresa.

INTRODUCCIÓN

La arquitectura empresarial describe a la empresa como una estructura coherente. La arquitectura documenta el estado actual de la organización, el estado deseado y la brecha entre ambos. Sin embargo los sistemas empresariales emplean cuatro perspectivas para disminuir las necesidades de la empresa y la tecnología.

Estas perspectivas describen los procesos necesarios para alcanzar las metas corporativas.

- Negocio.
- Aplicación.
- Información.
- Tecnología.

Situación Problémica

En la Universidad existen muchos proyectos los cuales presentan una variada arquitectura de software, el Polo Productivo particularmente quiere realizar un modelo para integrar las aplicaciones en una arquitectura de software única que se adecué a las necesidades del Programa Nacional de Informatización del Conocimiento Geológico (PNICG), para lograr una uniformidad en el mismo y garantizar que el producto obtenido tenga la calidad requerida.

Problema Científico

¿Cómo lograr una arquitectura factible para el Programa Nacional de Informatización del Conocimiento Geológico, la cuál garantice la total funcionalidad de las aplicaciones avalando la soberanía tecnológica y se obtenga un producto de calidad?

Objeto de estudio

Investigación de la arquitectura de software para desarrollar una propuesta factible para el proyecto PNICG.

Objetivo General

Plantear el desarrollo de una propuesta arquitectónica para el Programa Nacional de Informatización del Conocimiento Geológico garantizando que esta sea flexible y reusable, además de utilizar herramientas libres y que el producto tenga la calidad requerida según la complejidad del proyecto.

La gran mayoría de las herramientas utilizadas deben ser libres para obtener la soberanía tecnológica por lo que tanto se está luchando, pues muchas herramientas que están catalogadas dentro de las

mejores pertenecen a software propietarios. Lo que garantiza que el país no se vea obligado a trabajar con software privativos, pues son muy costosos, en caso de que en algún momento desapareciera el bloqueo, se tendría que pagar millones de dólares por su uso y la economía no lo soportaría.

Para dar cumplimiento al objetivo general se propone los siguientes *objetivos específicos*:

1. Estudiar los diferentes elementos de las arquitecturas de software.
2. Plantear una arquitectura de software para el PNICG.

Tareas que se desarrollarán:

1- Proposición del hardware que se necesita para realizar las aplicaciones.
2- Definición de una plataforma de desarrollo libre.
3- Investigación sobre el mejor lenguaje e interfaz para la programación de las aplicaciones.
4- Investigación del software más eficaz para garantizar el Modelado.
5- Investigación del software para garantizar la Gestión de Proyecto
6- Investigación del software para garantizar el tratamiento de los errores.
7- Investigación del software para garantizar el control de versiones.
8- Investigación del software para garantizar la Gestión Documental.
9- Investigación del software para garantizar la gestión de Salvas Automáticas.

Campo de Acción

Desarrollo de una propuesta de arquitectura de software para el Programa Nacional de Informatización del Conocimiento Geológico.

Idea a Defender

Realizando un análisis profundo de la arquitectura de software del Programa Nacional de Informatización del Conocimiento Geológico se obtendrá un producto de calidad que cumpla con todas las funcionalidades requeridas.

Diseño Metodológico

Los métodos científicos utilizados en esta investigación son los teóricos y los empíricos. Dentro de los métodos teóricos se utilizan los de análisis-síntesis, el histórico-lógico, y la modelación. Además cómo métodos empíricos están la observación y las entrevistas. A continuación se especifica el por qué de la selección de los mismos.

1. El método "Analítico - Sintético" se utiliza para descomponer toda la información en partes más simples, analizando todas las relaciones y componentes y la otra parte del método es la

operación inversa, unir esas partes previamente analizadas haciéndolas más pequeñas con ideas más sintetizadas.

2. Se aplicará el "Método Histórico – Lógico" estudia la trayectoria real de determinados patrones que guiarán la construcción de la arquitectura de software en el decursar de la historia.
3. Método "Modelación" es un método en el que se crean modelos con vistas a investigar la realidad, en este caso se dará a conocer una propuesta de arquitectura de software que es un modelo.
4. Método "Observación" estimula la curiosidad, impulsa el desarrollo de nuevos hechos de interés. Provoca el planteamiento de problemas científicos y se utiliza en conjunto con las entrevistas, pues todo esto está muy vinculado con procesos externos (opiniones).
5. Mediante "Entrevistas" se planifica la conversación para obtener toda la información que hace falta para la investigación. Este encuentro está dirigido a líderes de cada proyecto del Polo Productivo PNICG, así como a directivos del mismo para desarrollar el proceso de desarrollo de software en la Universidad. **Ver Planilla en Anexo 1.**

Con el objetivo de lograr resultados que le dieran credibilidad a esta investigación fue necesario seleccionar una población, de la cual se extrajo una muestra que fue analizada. De los resultados obtenidos a partir de este análisis se pudo inferir como se iba a comportar dicha población. A continuación se presentan:

Población de investigación: Conjunto de proyectos de software que forman parte del Polo Productivo PNICG de la Universidad de las Ciencias Informáticas.

Muestra: Líderes y parte del personal de los seis proyectos de software que conforman PNICG.

Unidad de estudio: Polo Productivo PNICG.

Técnica de muestreo

Muestreo Estratificado: Su fundamento consiste en subdividir una población heterogénea en una serie de sub-poblaciones homogéneas para garantizar que todas las características de la población heterogénea estén representadas en la muestra, por ejemplo, se divide la población en roles (en este caso líder de proyecto) y tres estudiantes según su desempeño en el mismo y tiempo de duración, es

un estrato de la población, donde esto asegura que la muestra presente las características específicas de la población estudiada.

El documento está constituido por dos capítulos.

El primer capítulo abordará los elementos relacionados con el estudio del estado del arte y la fundamentación teórica. En él se hace un análisis de los principales conceptos de la arquitectura de software, arquitectura empresarial, aplicación empresarial, y algunos detalles acerca de los estilos y vistas arquitectónicas. Se realiza un estudio de las arquitecturas más comunes para determinar cual es la adecuada para desarrollar el proyecto antes de iniciar el trabajo, apoyando nuestra investigación en aplicaciones empresariales. Además se dispone de un análisis sobre las principales tecnologías, metodologías de desarrollo, lenguajes de programación, definiendo finalmente cuáles son los apropiados para el desarrollo de la solución propuesta. Posteriormente se presentan aspectos importantes del segundo capítulo mostrando la solución propuesta para resolver el problema, según la plataforma de desarrollo, lenguajes de programación, la tecnología y las herramientas seleccionadas. De este modo se plantea los Requisitos No Funcionales y Atributos de Calidad.

1

CAPÍTULO 1
Estado del Arte y Fundamentación Teórica.

Introducción

En este capítulo se analizan los aspectos fundamentales a tener en cuenta para el desarrollo de la propuesta de arquitectura de software para el Programa Nacional de Informatización del Conocimiento Geológico, basadas en el objeto de estudio y el campo de acción. Además de realizar un análisis de algunas definiciones de arquitectura de software, arquitectura empresarial, aplicaciones empresariales, vistas, estilos arquitectónicos y las arquitecturas más comunes para seleccionar la más adecuada. Conjuntamente se hace un análisis de algunos lenguajes, tecnologías y herramientas posibles a usar.

1.1 Alcance

En la investigación se describe la propuesta arquitectónica para el PNICG, así como un análisis de las mejoras que se introducen en los componentes del Software, Hardware y el entorno de desarrollo a utilizar.

1.2 Arquitectura de Software

Una Arquitectura de Software, también denominada Arquitectura Lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, probadores, programadores, y el resto de los integrantes del equipo o grupo de desarrollo trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. **[1]**

Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas.**[2]**

La arquitectura dirige el desarrollo de los sistemas software y contribuye a que estos se lleven a cabo en los límites establecidos de costos y tiempo, y fundamentalmente debe garantizar que se cumpla con los requisitos funcionales y no funcionales de los usuarios. La arquitectura de software es el resultado del trabajo durante todo el ciclo de vida del proyecto, y principalmente en las primeras iteraciones, de un grupo de trabajo encabezado por el arquitecto (o grupo de arquitectura, en dependencia de las dimensiones del proyecto).**[1]**

Definiciones de Arquitectura de Software

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

La definición oficial de Arquitectura del Software es la IEEE Std 1471-2000 que expresa que: “La Arquitectura del Software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, además de los principios que orientan su diseño y evolución”.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad. (Kruchten, Philippe)

1.3 Definiciones de Arquitectura Empresarial

Es la aplicación rigurosa de un método que permite la descripción de las actuales y/o futuras estructuras y hábitos de una empresa en relación a sus procesos, sistemas de información, personal y unidades organizativas.**[3]**

1.4 Definiciones de Aplicaciones Empresariales

Es un paquete de software que nos ayuda en el desempeño del negocio. Con una aplicación empresarial se puede simplificar y automatizar tareas para aumentar el rendimiento de los procesos del negocio, para expandir el negocio que ya se tiene o incluso transformarlo. **[4]**

1.5 Vistas Arquitectónicas

Quizá uno de los modelos más conocidos es el "4+1" de Philippe Kruchten, vinculado al Rational Unified Process (RUP), que define cinco vistas diferentes: **Ver Anexo 2**

· Vista de casos de uso o de escenarios: A través de está, los arquitectos pueden desarrollar las siguientes cuatro vistas.

· Vista lógica: Describe el modelo de objetos.

· Vista de proceso: Muestra la concurrencia y sincronía de los procesos.

· Vista física: Muestra la ubicación del software en el hardware.

· Vista de Implementación: Describe la organización del entorno de desarrollo. **[5]**

1.6 Estilos Arquitectónicos

Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles/rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo. (Roy Thomas Fielding).

Un estilo arquitectónico define a una familia de sistemas en términos de un patrón de organización estructural. Específicamente, un estilo arquitectónico determina el vocabulario de componentes y conectores que podrán ser usados y combinados. (Mary Shaw y David Garlan)
Un estilo arquitectónico provee una colección de elementos edificadores del diseño en bloque, reglas y restricciones para componer los bloques constructivos, y las herramientas para analizar y manipular los diseños creados en el estilo. (Robert T.Monroe)

Los estilos están en un plano de abstracción más alto, definiendo como configurar la arquitectura, mientras los patrones están más cercanos al diseño, incluso podría decirse que más cercano a

algo físico, pues estos pueden representarse mediante código en un lenguaje de programación determinado. **[2]**

1.6.1 Estilo Llamada - Retorno

Este estilo es interesante en los sistemas que se centran en la interacción de un usuario con el propio sistema.

Se podrá dividir la arquitectura en dos partes, la primera representa la interfaz del usuario con el que se realiza la llamada al sistema, la segunda contiene la lógica de negocio que se realiza tras la correspondiente llamada del usuario. Esta familia de estilos enfatiza la modificabilidad, la escalabilidad, la reusabilidad y la usabilidad del sistema. Al diseñar una arquitectura de este tipo es importante saber que datos son los que servirán para interactuar con el usuario y cuales servirán solo como lógica de aplicación. En esta familia de arquitecturas se destacan los patrones modelo-vista-controlador y arquitectura en capas.**[6]**

1.6.2 Estilo Tubería y Filtros

Este estilo se enmarca dentro de las llamadas arquitecturas de flujo de datos. Es sin duda alguna el estilo que se definió más temprano y el que puede identificarse topológica, procesual y taxonómicamente con menor ambigüedad **[7]**.

Ha prevalecido el nombre de tubería-filtros por más que se sabe muy bien que los llamados filtros no realizan forzosamente tareas de filtrado, como ser eliminación de campos o registros, sino que ejecutan formas variables de transformación, una de las cuales puede ser el filtrado. **[8]**

Una tubería es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. [...] Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura mascota cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos *stack* lo fue en las especificaciones algebraicas o en los tipos de datos abstractos.**[8]**

El estilo de arquitectura tuberías y filtros, se basa precisamente, en dividir todo el proceso asociado a la entrega de mensajes en pequeñas tareas que serán implementadas en unos

componentes llamados filtros que estarán interconectados entre sí a través de tuberías o canales. Al tener divididas las tareas en componentes independientes se podrá construir infinidad de combinaciones de cadena de procesamiento simplemente añadiendo, quitando u ordenando en una nueva secuencia todos estos elementos reutilizando cada una de las funcionalidades en cada uno de los pasos.

Cada filtro recibe los mensajes a través de su canal o tubería de entrada, los procesa y posteriormente los deposita en su tubería de salida. Este canal de salida coincidirá con el canal de entrada del siguiente filtro que continuará a su vez con la cadena de tareas.

En el caso específico del servidor de integración el proceso está dividido en tres tareas fundamentales: **Ver Anexo 3**

- Recibir el mensaje, validarlo y enviarlo.

El sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes **[9].**

1.7 Patrones Arquitectónicos

Los patrones arquitectónicos capturan existencia, experiencia comprobada en el desarrollo del software y ayudan a promover buenas prácticas de diseño. En la actualidad los patrones se encuentran muy difundidos y se les puede encontrar por cientos.

Algunos autores consideran que un patrón es un par problema – solución, resultado de la experiencia en el diseño de arquitecturas de sistemas y propone los patrones arquitectónicos como descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. **[6]**

Los patrones arquitectónicos están relacionados con la interacción de los objetos dentro o entre niveles arquitectónicos, están dirigidos a la solución de problemas arquitectónicos como adaptabilidad a requerimientos cambiantes, performance, modularidad y acoplamiento. Mientras las soluciones propuestas tienen que ver con patrones de llamadas entre objetos (similar a los patrones de diseño), decisiones y criterios arquitectónicos, empaquetado de funcionalidad, entre otros aspectos. (Frank Buschman)

1.7.1 Patrón en Capas

Los sistemas o arquitecturas en capas constituyen uno de los patrones que aparecen con mayor frecuencia.

El patrón en capas es definido 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.**[1]**

En algunos ejemplares, las capas internas están ocultas a todas las demás, menos para las capas externas adyacentes, y excepto para funciones puntuales de exportación; en estos sistemas, los componentes implementan máquinas virtuales en alguna de las capas de la jerarquía. En otros sistemas, las capas pueden ser sólo parcialmente opacas.

En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. El uso de arquitecturas en capas, explícitas o implícitas, es muy frecuente. **[10]**

En un patrón en capas, los conectores se definen mediante los protocolos que determinan las formas de la interacción. Los diagramas de sistemas clásicos dibujan las capas en adyacencia, sin conectores, flechas ni interfaces; en algunos casos se suele representar la naturaleza jerárquica del sistema en forma de círculos concéntricos.
Las restricciones topológicas del patrón pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma; se supone que si esta exigencia se relaja, el patrón deja de ser puro y pierde algo de su capacidad heurística **[11].**

A veces se argumenta que el cruce recargado de muchos niveles involucra eventuales degradaciones de performance; pero muchas más veces se sacrifica la pureza de la arquitectura en capas precisamente para mejorarla: colocando, por ejemplo, reglas de negocios en los procedimientos almacenados de las bases de datos, o articulando instrucciones de consulta en la capa de la interface del usuario.

El número mínimo de capas es obviamente dos, y en ese umbral la literatura arquitectónica sitúa a veces al sub-patrón cliente-servidor como el modelo arquetípico del patrón de capas y el que se encuentra con mayor frecuencia en las aplicaciones en red. Un componente servidor, que ofrece

ciertos servicios, escucha que algún otro componente requiera uno; un componente cliente solicita ese servicio al servidor a través de un conector. El servidor ejecuta el requerimiento (o lo rechaza) y le devuelve una respuesta.**[11]**

Las ventajas del patrón en capas.

- 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.
- Admite muy naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas. **[11]**

Las desventajas del patrón en capas.

- No admiten un buen mapeo en una estructura jerárquica. Incluso cuando un sistema se puede establecer lógicamente en capas, consideraciones de performance pueden requerir acoplamientos específicos entre capas de alto y bajo nivel.
- Los cambios en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel, en especial si se utiliza una modalidad relajada; también se admite que la arquitectura en capas ayuda a controlar y encapsular aplicaciones complejas, pero complica no siempre razonablemente las aplicaciones simples. **[11]**

La separación y especialización de interfaces de usuario, reglas de negocios y estructuras de datos conservan todavía plena relevancia. Este patrón posee virtudes de distribución, preservación de identidad, seguridad, performance, escalabilidad, sincronicidad, balanceo de carga, robustez y acidez transaccional que siguen siendo competitivas y que no se valoran.

1.7.2 Patrón Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de la capa de presentación de una aplicación en Interfaz, la lógica y el modelo, en tres componentes distintos (Modelo, Vista Controlador).

El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo son los datos provenientes de la lógica de negocio recogidos en el controlador y el controlador invoca a la lógica de negocio, utilizando los objetos de negocio.

Descripción del patrón:

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos. Objeto que contiene los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Objeto responsable de mostrar el modelo como respuesta a una petición. Muestran los dados del modelo suministrados por un controlador.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Objeto responsable de procesar las entradas de datos en forma de peticiones HTTP, invocando las funcionalidades necesarias, expuestas por la capa de servicios de negocio, devolviendo el modelo requerido para ser mostrado.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al modelo.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente

Ventajas:

- Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Permite el escalamiento de la aplicación en caso de ser requerido.

Desventajas:

- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- La curva de aprendizaje del patrón de diseño es más alta que usando modelos más sencillos.

Beneficios que brinda:

- Menor acoplamiento.
    1. Desacopla las vistas de los modelos.
    2. Desacopla los modelos de la forma en que se muestran e ingresan los datos.
- Mayor cohesión.

1. Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).

- Las vistas proveen mayor flexibilidad y agilidad.
    1. Se puede crear múltiples vistas de un modelo.
    2. Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente.
    3. Las vistas pueden anidarse.
    4. Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.
    5. Se puede sincronizar las vistas.
    6. Las vistas pueden concentrarse en diferentes aspectos del modelo.
- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales.
    1. Una vista para cada dispositivo que puede varias según sus capacidades.
    2. Una vista para la Web y otra para aplicaciones de escritorio.
- Más claridad de diseño.
- Facilita el mantenimiento.
- Mayor escalabilidad.

En la Organización Nacional de Recursos Minerales (ONRM) lo más adecuado será utilizar el patrón MVC en el nivel superior pues las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario. Este patrón se implementará a la hora de desarrollar el Portal que tendrá implícito todas las aplicaciones existentes, estas últimas serán desarrolladas con el patrón en capas por las facilidades que brinda.

1.8 Posibles Arquitecturas a Usar

Generalmente, no es necesario innovar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más utilizadas son:

- **Cliente- servidor:** Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- **Arquitectura en capas:** Especialización de la arquitectura cliente-servidor donde la carga se divide en capas con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.
- **Orientada a Servicios**

1.8.1 Cliente –Servidor

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

Ventajas de la arquitectura cliente-servido:

- Centralización del control: Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un cliente web o no autorizado no pueda dañar el sistema.
- Escalabilidad: Se puede aumentar la capacidad de clientes y servidores por separado.

La arquitectura cliente - servidor es la arquitectura de red que separa al cliente del servidor. Cada caso del software del cliente puede enviar peticiones a un servidor.
Esta arquitectura se encuentra dentro de la clasificación de estilo de llamada y retorno. El cliente y el servidor generalmente están localizados en diferentes sistemas, sin embargo pueden encontrarse en el mismo. El cliente es la entidad que hace la petición por un servicio. El servidor es la entidad que provee el servicio correspondiente a la petición. El servicio debe procurar el resultado, el cual es retornado al cliente

1.8.2 Arquitectura en Capas

La arquitectura en capa define el patrón en capas como una organización jerárquica. Lo que posibilita un diseño basado en niveles de abstracción creciente, posibilitando a los implementadores, particionar un problema en una secuencia de pasos incrementales. Este estilo de desarrollo en varios niveles, facilita que en caso que ocurra algún cambio, sólo se tendrían que

realizar las correcciones necesarias en el nivel requerido sin tener que revisar código de otros niveles.

La programación por capas es un estilo de programación en la que el objetivo primordial es dividir, fraccionar y llegar a separar la Presentación (donde muestras y obtienes datos), de la Lógica de Negocio (donde realizas operaciones) y con todo esto se obtendría independencia, por lo que si hay cambios de arquitectura se puede acceder a los datos o cambiar el negocio o modificar la presentación y solo sería en esa fracción de la capa.

En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

La arquitectura en capas es algo referente a la arquitectura de software, pero realmente si una aplicación no tiene arquitectura de capas adecuada para el tipo de necesidad se podría decir que no cumple con la programación orientada a objetos. Una arquitectura como está puede tener tantas capas como se necesite, dependiendo siempre de la complejidad de la aplicación.

1- Capa de presentación: En esta capa se diseña todo lo que constituye la interfaz gráfica y la interacción del usuario con el software. Se comunica únicamente con la capa de negocio. Representa el conjunto de componentes que genera la información que se representará en la interfaz de usuario del cliente. La norma es que la mayor parte de las interacciones con el usuario se realicen en las JSP debido a su flexibilidad, ya que integran de forma natural etiquetas XHTML, HTML, XML y JSF.

La capa de presentación se implementa mediante tecnologías JSP, Servlets, Java Standard Tags Libraries, Spring Tags y Scriptlets con las que se construyen y conectan las interfaces gráficas con la capa de lógica del negocio.

La función de la capa de presentación es la de proveer una interfaces para realizar la transferencia de datos. Es la encargada de interactuar con el usuario, obtener, validar y enviar los datos al servidor.

Los componentes de la interfaz de usuario deben mostrar datos al usuario, obtener y validar los datos procedentes del mismo e interpretar las acciones de éste que indican que desea realizar una operación con los datos. Asimismo, la interfaz debe filtrar las acciones

disponibles con el fin de permitir al usuario realizar sólo aquellas operaciones que le sean necesarias en un momento determinado.

Los componentes de interfaz de usuario:

- No inicializan, participan ni votan en transacciones.
- Presentan una referencia al componente de proceso de usuario actual si necesitan mostrar sus datos o actuar en su estado.
- Pueden encapsular tanto la funcionalidad de visualización como un controlador.

Las interfaces web deben ser consistentes con la información que se requiere, no se deben utilizar más campos de los necesarios, así como la información requerida tiene que ser especificada de manera clara y concisa, no debe haber más que lo necesario en cada formulario. El objetivo principal de este componente es facilitar al usuario la interacción con la misma.

Dentro de la parte técnica, la capa de presentación contiene los objetos encargados de comunicar al usuario con el sistema mediante el intercambio de información, capturando y desplegando los datos necesarios para realizar alguna tarea.

2- Capa de lógica de negocio: En esta capa residen los objetos de negocio que interactúan con los objetos de dominio. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse para un correcto funcionamiento lógico de la aplicación.

Los objetos de negocio, separan los datos y la lógica de negocio haciendo uso del modelo de objetos definido, dichos objetos de negocio son los encargados de procesar la información proveniente tanto de la Presentación, como de la Capa de Acceso a Datos.

Los objetos de negocio son los objetos del Modelo que implementan la lógica de negocio. Sus funciones fundamentales son:

- Realizar la validación de los datos introducidos por el usuario.
- Ejecutar la petición realizada por el usuario. Para ello podrá valerse de transacciones contra Sistemas Host, consultas a bases de datos, consultas a proveedores de contenidos, etc.

- Generar los objetos que utilizarán las Vistas para mostrar los resultados obtenidos.

Su función es garantizar que toda la lógica de negocio esté bien localizada y no mezclada con objetos de otras capas. Esto conlleva grandes ventajas, pues define los objetos de negocio, y crea las interfaces de servicio con sus implementaciones, las cuales hacen uso de los objetos de dominio.

Además de encontrarse toda la lógica de la aplicación y el modelo de objetos encargados de la manipulación de los datos existentes, así como el procesamiento de la información ingresada o solicitada por el usuario en la capa de presentación.

En la misma se reciben solicitudes de la capa de presentación, la cual procesa y obtiene los resultados invocando a la capa de Acceso a Datos. La comunicación con otros sistemas que actúan en conjunto, se hace mediante esta capa, es en ella están expuestas el conjunto de funcionalidades o métodos a exportar.

La responsabilidad de esta capa es implementar lógica de negocio que será consumida por la interfaz del sistema y por otros sistemas a través de los servicios compartidos. Recibe los datos que ingresó el usuario del sistema mediante la capa de presentación, luego los procesa y crea objetos según lo que se necesite hacer con los datos.

Al encapsular los datos, la aplicación asegura mantener la consistencia de los mismos, así como obtener información precisa de las bases de datos e ingresar en las mismas solamente la información necesaria, a través de la capa de acceso a datos, asegurando así no tener datos duplicados ni en las bases de datos, ni en los reportes solicitados por el usuario.

Existen tres tipos de componentes de negocio fundamentales:

- Lógica de Negocio: Implementan la funcionalidad de negocio del sistema.
- Entidades de negocio: Representan las entidades del sistema.
- Workflow: Implementan procesos de negocio en los cuales participan entidades y lógica de negocio.

3- Capa de Acceso a Datos: La capa de Acceso a Datos es el puente entre la capa de Lógica de Negocio y el Sistema de Base de Datos. Encapsula la lógica de acceso a datos.

Aquí se encuentran componentes que hacen transparente el acceso a la base de datos. Este es el lugar idóneo para implementar los objetos de acceso a datos, permitiendo ingresar, obtener, actualizar y eliminar información del Sistema de Bases de Datos.

Los Objetos de Acceso a Datos (DAOs) encapsulan la persistencia de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos. Las implementaciones de los DAOs estarán disponibles para los objetos de negocio.

En esta capa se utiliza el framework Hibernate pues se está trabajando con programación orientada a objetos y bases de datos relacionales, observando que estos son dos paradigmas diferentes.
El modelo relacional trata con relaciones, tuplas y conjuntos y es muy matemático por naturaleza. Sin embargo, el paradigma orientado a objetos trata con objetos, sus atributos y relaciones entre objetos. Cuando se quiere que los objetos sean persistentes utilizando para ello una base de datos relacional, se muestra una desavenencia entre estos dos paradigmas, la también llamada diferencia objeto-relacional (object – relational gap"). Un mapeador objeto-relacional **(ORM)** ayudará a evitar esta diferencia.

La misma es la responsable de unir todos los índices de funciones y estructuras que devuelven los servicios y combinarlos en una macro estructura. Su función es mantener sincronizada en todo momento la información existente en el sistema web y la que se encuentra en la base de datos. También permite el intercambio de información con clientes y proveedores.

Esta capa de acceso a datos se relaciona directamente con el patrón DataMapper pues permite manejar toda la carga y almacenamiento entre el modelo de dominio y la base de datos, logrando que ambos varíen independientemente.

DataMapper

Permite disponer de dos objetos. El primero representa la lógica de negocio. El segundo el acceso a datos. Se trata de realizar una correlación entre objetos de lógica de negocio y tablas de la base de datos.

Data Mapper ofrece la posibilidad de incluir fácilmente lógica de negocio junto con el acceso a datos. En el caso de Active Record esa lógica de negocio se mezclará con el acceso a datos en la misma clase. Si el mapeo entre las clases y las tablas de la base de datos es sencillo, será

suficiente. En caso contrario, habrá que separar ambas lógicas (lógica de negocio de lógica de acceso a datos) utilizando Data Mapper.

- Data Mapper sería propio de soluciones DAO, muy vistas en Java. En Java son populares frameworks de mapeo de clases como Hibernate.

Ventajas de este patrón:

- Diferencia estructural (i.e. no vale solo con IDataRow).
- Diferencia de tecnología (i.e. Access, SQL, Oracle).
- Diferencia de flujo (i.e. Local 100Mb LAN vs. Trans-continental).
- Aísla el acceso a datos.
- Hace más fácil el testing.
- La única forma de conseguir aislamiento real de la fuente de datos.
- Permite persistir un “Domain Model”.
- El resto de patrones se orientan a una única tabla, DataMapper permite muchas más opciones a la hora de decidir la estrategia de carga, enlazar las referencias entre objetos.
- Requiere solucionar otros problemas que ya están solucionados con otros patrones (Unit of Work, Identity Map).
- Permite separar un objeto del dominio de sus correspondientes datos en la base.
- Mapea un registro en la base, a un objeto del dominio.

La aplicación utiliza una arquitectura multi-capa. Para una arquitectura colocada, las capas de presentación, de lógica de negocio y de acceso a datos están físicamente localizadas aislando las responsabilidades de cada capa. La arquitectura colocada hace que la aplicación sea más simple y escalable. **[12] Ver Anexo 4**

La Capa de Acceso a Datos e Hibernate

Hibernate ofrece facilidades para recuperación y actualización de datos, control de transacciones, repositorios de conexiones a bases de datos, consultas programáticas y declarativas, y un control de relaciones de entidades declarativas. **[12]**

Se utilizan Reflection y la generación de *bytecodes* en tiempo de ejecución, y la generación del SQL ocurre en el momento de la arrancada. Esto permite desarrollar objetos persistentes siguiendo el lenguaje común de Java: incluyendo asociación, herencia, polimorfismo, composición y el marco de trabajo Collections de Java. **[12]**

Todas estas capas pueden residir en un único ordenador (no es lo típico). Si bien lo más usual es que existan más ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor).

Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores.

Además si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores, los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.**[9]**

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de acceso a datos.

1.8.3 Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios (en inglés *Service Oriented Architecture* o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.**[66]**

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación. En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado.

La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.**[66]**

Al contrario de las arquitecturas orientadas a objetos, la SOA está formada por servicios de aplicación débilmente acoplados y altamente interoperable. Para comunicarse entre sí, estos

servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (WSDL). La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Plataforma Java o Microsoft .NET). **[66]**

Con esta arquitectura, se pretende que los componentes de software desarrollados sean reusables, ya que la interfaz se define siguiendo un estándar. SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y dar soporte a las actividades de integración y consolidación.

Esta arquitectura está definida para dar soporte a servicios, aunque no es lo mismo que los WebServices, si es cierto que están íntimamente relacionados con los mismos y que le han dado auge a la arquitectura SOA. **[13]**

Esta arquitectura proporciona independencia total de la plataforma y del lenguaje, además de que permite generar servicios reutilizables y compartidos mediante el uso de lenguajes y protocolos totalmente estandarizados como HTTP, SOAP, XML, UDDI o WSDL.

Por otro lado y totalmente relacionado con SOA se tiene el concepto de EAI (Enterprise Architecture Integration) como arquitectura de integración de sistemas, para lo cual se define un componente tecnológico llamado Bus de Integración que dispone de una serie de conectores para servir de enlace entre diferentes sistemas con tecnologías diferentes.

Este bus de integración proporciona muchas y diferentes ventajas dentro del modelo de negocio:

- Permite la implementación de servicios para los clientes finales mediante la integración/orquestación de los procesos de negocio, de manera rápida y fiable.
- Control, visibilidad y seguimiento de los procesos de negocio de extremo a extremo, siendo un soporte importante para los usuarios.
- Soporte a la toma de decisiones a diferentes niveles organizacionales, porque permite conocer indicadores, tanto técnicos como funcionales, del estado de los procesos en tiempo real.
- Evita la falta de coherencia de información. Actualización de la información en tiempo real.
- Reducen el tiempo y coste total de desarrollo y mantenimiento porque está basado en estándares y herramientas gráficas (generación automática de código).

- Visión del proceso de negocio de extremo a extremo, siendo más fácil el seguimiento y la validación por el usuario de negocio. Más fácil la reingeniería y optimización.

Al contrario de las arquitecturas orientadas a objetos, la SOA está formada por servicios de aplicación débilmente acoplados y altamente interoperable. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación.

Una de las principales características de SOA es que resuelve las problemáticas generales y particulares de conectividad. Asimismo, permite una real reusabilidad masiva y una gran independencia de las plataformas, rentabilizando las ya existentes. También requiere de algún entrenamiento a nivel de diseño y de administración. Además de no implementarse con Open-Source puede implicar inversiones significativas.

SOA es una arquitectura que esta inmersa en un proceso de maduración, demora muchos años para su verdadera implementación en una empresa. Se sustenta en la unión de muchos procesos los cuales deben estar bien integrados. Esta arquitectura en un futuro pudiera implementarse pues satisface completamente las necesidades de toda empresa. Una de las problemáticas en la no admisión de la misma para desarrollar el proyecto es que requiere de personal totalmente calificado y condiciones esenciales para el desarrollo de la misma, con las cuales no se cuenta en la Universidad de las Ciencias Informáticas.

1.9 Arquitectura Empresarial

La ONRM presenta características de un sistema distribuido y con grandes implicaciones de escalabilidad, por lo que exige una propuesta de Arquitectura de Software para poder iniciar su desarrollo. Dicha empresa necesita de una estrategia de consolidación para aprovechar el recurso humano, el conocimiento, la administración y una disponibilidad de servicio con el objetivo fundamental de hacer mejor las cosas, más rápido y con menores costos.
Para ello se necesita disponer de una arquitectura a nivel organizacional. Razón por la que el PNICG decide adoptar una política de arquitectura empresarial y una política para los procesos de integración. La arquitectura trata de integrar toda esta diversidad de aplicativos y fuentes de información.

Más allá de la tecnología se puede observar que la estrategia corporativa y la tecnología de información son más importantes que las herramientas y productos que sirven como mecanismos para desarrollar las soluciones.

Una Arquitectura Empresarial es un registro de modelos y documentos totalmente integrados en cuatro objetos de arquitectura principales (Negocio, Aplicación, Información y Tecnología). Desarrollar una arquitectura empresarial proporciona un proyecto organizativo que puede ayudar a mejorar la calidad, eficacia y responsabilidad del negocio.

Una solución de arquitectura empresarial bien desarrollada permite a las organizaciones responder rápido, eficaz y positivamente a las oportunidades y desafíos presentados por los actuales cambios de mercado, consolidaciones del sector y avances tecnológicos. **[4]**
Una arquitectura empresarial, almacenada y mantenida en un completo repositorio totalmente integrado, es un valor fundamental para una organización, que proporciona:

- Capacidad para responder a los cambios.
- Coste reducido de la gestión de la infraestructura de IT.
- Comunicación y compresión mejorada entre los integrantes de la empresa.
- Una base excelente para la mejora de procesos de negocio y el análisis.**[14]**

1.10 Componentes Estructurales de la Arquitectura Empresarial

1.10.1 Negocio

1.10.1.1 Proceso de negocio

En este nivel se describen las actividades más importantes de la empresa (el núcleo del negocio). Se sugiere modelar la cadena de valor del negocio y de ahí obtener los procesos del negocio.

Los procesos son un conjunto de actividades relacionadas y agrupadas que en conjunto reciben un insumo y producen una salida. Dichos procesos pueden automatizarse mediante sistemas de cómputo desarrollados a la medida o mediante la compra de sistemas existentes en el mercado como ERP (Planeación de Recursos Empresariales) o BPM (Business Process Management).

Los procesos de negocio pueden ser vistos como un recetario para hacer funcionar un negocio y alcanzar las metas definidas en la estrategia de negocio de la empresa.**[4]**

Hay tres tipos de procesos de negocio:

1. Procesos estratégicos: Estos procesos dan orientación al negocio. Por ejemplo, "Planificar estrategia", "Establecer objetivos y metas".
2. Procesos centrales: Estos procesos dan el valor al cliente, son la parte principal del negocio. Por ejemplo, “Repartir mercancías”
3. Procesos de soporte: Estos procesos dan soporte a los procesos centrales. Por ejemplo, “Contabilidad”, “Servicio técnico”. [**4]**

Los procesos de negocio consisten en subprocesos, decisiones y actividades.

Un subproceso es parte de un proceso de mayor nivel que tiene su propia meta, propietario, entradas y salidas.

Las actividades son partes de los procesos de negocio que no incluyen ninguna toma de decisión ni vale la pena descomponer (aunque ello sea posible). Por ejemplo, “Responde al teléfono”, “Haz una factura”.

Un proceso de negocio es usualmente el resultado de una Reingeniería de Procesos. El modelado de procesos es usado para capturar, documentar y rediseñar procesos de negocio.

Para aplicar los procesos se debe tener claras las tareas, una estructura jerárquica y una tendencia a la interacción y comunicación vertical. **[4]**

1.10.2 Aplicación

1.10.2.1 Protocolos

1)- TCP/IP

TCP/IP es el protocolo común utilizado por todos los ordenadores conectados a Internet, de manera que puedan comunicarse entre sí. Hay que tener en cuenta que en Internet se encuentran conectados ordenadores de clases muy diferentes y con hardware y software incompatibles en muchos casos, además de todos los medios y formas posibles de conexión.

Aquí se encuentra una de las grandes ventajas del TCP/IP, pues este protocolo se encargará de que la comunicación sea posible. TCP/IP es compatible con cualquier sistema operativo y con cualquier tipo de hardware.

Cualquier máquina de la red puede comunicarse con otra distinta y esta conectividad permite enlazar redes físicamente independientes en una red virtual llamada Internet. Las máquinas en Internet son denominadas "hosts" o nodos.**[16]**

TCP/IP proporciona la base para muchos servicios útiles, incluyendo correo electrónico, transferencia de ficheros y login remoto. El correo electrónico está diseñado para transmitir ficheros de texto pequeños. También pueden proporcionar chequeos de seguridad controlando las transferencias. El login remoto permite a los usuarios de un ordenador acceder a una máquina remota y llevar a cabo una sesión interactiva. **[16]**

Las utilidades de transferencia sirven para transferir ficheros muy grandes que contengan programas o datos.

Ventajas e inconvenientes.

- El conjunto TCP/IP está diseñado para enrutar y tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales.
- Se utiliza a nivel mundial para conectarse a Internet y a los servidores web. Es compatible con las herramientas estándar para analizar el funcionamiento de la red.
- Un inconveniente de TCP/IP es que es más difícil de configurar y de mantener que NetBEUI o IPX/SPX; además es algo más lento en redes con un volumen de tráfico medio bajo. Sin embargo, puede ser más rápido en redes con un volumen de tráfico grande donde haya que enrutar un gran número de tramas.**[16]**

El conjunto TCP/IP se utiliza tanto en redes empresariales como en campus universitarios o en complejos empresariales, donde se utilizan muchos enrutadores y conexiones a mainframe o a ordenadores UNIX, como así también en redes pequeñas o domésticas, y hasta en teléfonos móviles.

2)- File Transfer Protocol

FTP (File Transfer Protocol) es un protocolo de transferencia de archivos entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar archivos desde él o para enviarle nuestros propios archivos independientemente del sistema operativo utilizado en cada equipo.**[17]**

El Servicio FTP es ofrecido por la capa de Aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado.**[18]**

Para solucionar este problema son de gran utilidad aplicaciones como scp y sftp, incluidas en el paquete SSH, que permiten transferir archivos pero cifrando todo el tráfico.

Servidor FTP

Un servidor FTP es un programa especial que se ejecuta en un equipo servidor normalmente conectado a Internet (aunque puede estar conectado a otros tipos de redes, LAN, MAN). Su función es permitir el intercambio de datos entre diferentes servidores/ordenadores.

Por lo general, los programas servidores FTP no suelen encontrarse en los ordenadores personales, por lo que un usuario normalmente utilizará el FTP para conectarse remotamente a uno y así intercambiar información con él.**[19]**

Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que los clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes; o como servidor de backup (copia de seguridad) de los archivos importantes que pueda tener una empresa. Para ello, existen protocolos de comunicación FTP para que los datos se transmitan cifrados, como el SFTP (Secure File Transfer Protocol).**[19]**

Cliente FTP

Cuando el navegador no está equipado con la función FTP, o si se quiere cargar archivos en un ordenador remoto, se necesitará utilizar un programa cliente FTP.

Un cliente FTP es un programa que se instala en el ordenador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos.**[19]**

Para utilizar un cliente FTP, se necesita conocer el nombre del archivo, el ordenador en que reside (servidor, en el caso de descarga de archivos), el ordenador al que se quiere transferir el archivo (en caso de querer subirlo al servidor), y la carpeta en la que se encuentra.**[19]**

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Windows, DOS, Linux y Unix. Sin embargo, hay disponibles clientes con opciones añadidas e interfaz gráfica. Aunque muchos navegadores tienen ya integrado FTP, es más confiable a la hora de conectarse con servidores FTP no anónimos utilizar un programa cliente.**[19]**

3)- HyperText Transfer Protocol

El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW). HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

Es un protocolo orientado a transacciones y sigue el esquema petición - respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador o un spider) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante una URL. **[20]**

Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos y la traducción automática de un documento. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores.

El desarrollo de aplicaciones web necesita frecuentemente mantener el estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.**[20]**

El uso de campos de encabezados enviados en las transacciones HTTP le da gran flexibilidad al protocolo. Estos campos permiten que se envíe información descriptiva en la transacción, permitiendo así la autenticación, cifrado e identificación de usuario.**[20]**

4)- HTTPS

El protocolo de red HTTPS es la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de

cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. **[21]**

Cabe mencionar que el uso del protocolo HTTPS no impide que se pueda utilizar HTTP. Es aquí, cuando nuestro navegador nos advertirá sobre la carga de elementos no seguros (HTTP), estando conectados a un entorno seguro (HTTPS).**[21]**

Secure Sockets Layer (SSL)*:* Seguridad de la Capa de Transporte. Es un protocolo criptográfico que proporciona comunicaciones seguras en Internet. SSL proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de la criptografía.

Habitualmente, sólo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar; la autenticación mutua requiere un despliegue de infraestructura de claves públicas (o PKI) para los clientes. Los protocolos permiten a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir escuchas (eavesdropping), la falsificación de la identidad del remitente (phishing) y mantener la integridad del mensaje.**[22]**

SSL implica una serie de fases básicas:

- Negociar entre las partes el algoritmo que se usará en la comunicación.
- Intercambio de claves públicas y autenticación basada en certificados digitales.
- Cifrado del tráfico basado en cifrado simétrico.**[22]**

Las implementaciones actuales proporcionan las siguientes opciones:

- Para criptografía de clave pública: RSA, Diffie-Hellman, DSA (Digital Signature Algorithm) o Fortezza.
- Para cifrado simétrico: RC2, RC4, IDEA (International Data Encryption Algorithm), DES (Data Encryption Standard), Triple DES o AES (Advanced Encryption Standard).
- Con funciones hash: MD5 o de la familia SHA.**[23]**

SSL se ejecuta en una capa entre los protocolos de aplicación como HTTP, SMTP, NNTP y sobre el protocolo de transporte TCP, que forma parte de la familia de protocolos TCP/IP. Los protocolos HTTPS son utilizados por navegadores como: Safari, Internet Explorer, Mozilla Firefox, Opera, entre otros. El puerto estándar para este protocolo es el 443.

Aunque pueda proporcionar seguridad a cualquier protocolo que use conexiones de confianza (tal como TCP), se usa en la mayoría de los casos junto a HTTP para formar HTTPS. HTTPS es usado para asegurar páginas World Wide Web para aplicaciones de comercio electrónico, utilizando certificados de clave pública para verificar la identidad de los extremos.**[23]**

1.10.3 Información

1.10.3.1 Interrelacionar las Aplicaciones Web del Proyecto

¨Los Web Service son componentes de software que se comunican con otras aplicaciones codificando los mensaje en XML y enviando estos mensaje a través de protocolos estándares de Internet tales como el Hypertext Transfer Protocol (HTTP). Intuitivamente un Web Service es similar a un sitio web que no cuenta con un interfaz de usuario y que da servicio a las aplicaciones en vez de a las personas. Los Web Service, en vez de obtener solicitudes desde el navegador y retornar páginas web como respuesta, lo que hace es recibir solicitudes a través de un mensaje formateado en XML desde una aplicación, realiza una tarea y devuelve un mensaje de respuesta también formateado en XML¨. (Marcos Escobar.)

Microsoft y otras empresas líderes están promocionando SOAP como estándar de los mensajes para los Web Services. Un mensaje SOAP se parece mucho a una carta: es un sobre que contiene una cabecera con la dirección del receptor del mensaje, un conjunto de opciones de entrega (tal como la información de encriptación), y un cuerpo o body con la información o data del mensaje.**[24]**

Microsoft y otros líderes promocionan los Web Services como un modelo de programación para la comunicación entre aplicaciones. Estas compañías piensan que la conexión de aplicaciones a través de Internet mejorará la capacidad de las empresas para trabajar conjuntamente con sus socios de negocio, proveedores y clientes. Creando una capa de Web Services sobre una aplicación corporativa existente, las organizaciones podrán permitir que sistemas externos puedan invocar las funciones de la aplicación a través de Internet (o una intranet corporativa) sin tener que modificar la aplicación misma. **[24]**

Poner una capa de web services sobre las aplicaciones existentes es una solución muy interesante para integrar las aplicaciones desarrolladas por los diferentes departamentos y así reducir los costos de integración.

Un Servicio Web (en inglés Web Service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

1.10.3.2 Requisitos de un Web Service

- Interoperabilidad: Un servicio remoto debe permitir su utilización por clientes de otras plataformas.
- Amigabilidad con Internet: La solución debe poder funcionar para soportar clientes que accedan a los servicios remotos desde Internet.
- Interfaces fuertemente tipadas: No debería haber ambigüedad acerca del tipo de dato enviado y recibido desde un servicio remoto. Más aún, los tipos de datos definidos en el servicio remoto deben poderse corresponder razonablemente bien con los tipos de datos de la mayoría de los lenguaje de programación procedimentales.
- Posibilidad de aprovechar los estándares de Internet existentes: La implementación del servicio remoto debería aprovechar estándares de Internet existentes tanto como sea posible y evitar reinventar soluciones a problema que ya se han resuelto. Una solución construida sobre un estándar de Internet ampliamente adoptado puede aprovechar conjuntos de herramientas y productos existentes creados para dicha tecnología.
- Soporte para cualquier lenguaje: La solución no debería ligarse a un lenguaje de programación particular Java RMI, por ejemplo, esta ligada completamente a lenguaje Java. Sería muy difícil invocar funcionalidad de un objeto Java remoto desde Visual Basic o PERL. Un cliente debería ser capaz de implementar un nuevo servicio Web existente independientemente del lenguaje de programación en el que se halla escrito el cliente.
- Soporte para cualquier infraestructura de componente distribuida: La solución no debe estar fuertemente ligada a una infraestructura de componentes en particular. De hecho, no se bebería requerir el comprar, instalar o mantener una infraestructura de objetos distribuidos, solo construir un nuevo servicio remoto utilizar un servicio existente. **[25]**

1.10.3.3 Estándares Empleados

- Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- XML (Extensible Markup Language): Es el formato estándar para los datos que se vayan a intercambiar.

- SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Producer Call): Protocolos sobre los que se establece el intercambio.
- Otros protocolos: Los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- WSDL (Web Services Description Languages): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- UDDI (Universal Description, Discovery and Integration): Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.**[26]**

1.10.4 Tecnología

1.10.4.1 Servidores de Aplicaciones

En informática se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones. Proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación.

J2EE provee estándares que le permiten a un servidor de aplicaciones servir como "contenedor" de los componentes que conforman dichas aplicaciones.

Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones. Si bien el término es aplicable a todas las plataformas de software, hoy en día el término servidor de aplicaciones se ha convertido en sinónimo de la plataforma J2EE de Sun Microsystems.**[27]**

Estos componentes, escritos en lenguaje Java, usualmente se conocen como Servlets, Java Server Pages (JSPs) y Enterprise JavaBeans (EJBs) y permiten implementar diferentes capas de la aplicación, como la interfaz de usuario, la lógica de negocio, la gestión de sesiones de usuario o el acceso a bases de datos remotas. Los servidores de aplicación típicamente incluyen también *middleware* (o software de conectividad) que les permite intercomunicarse con variados servicios, para efectos de confiabilidad, seguridad y no-repudio. **[27]**

Los servidores de aplicación también brindan a los desarrolladores una Interfaz para Programación de Aplicaciones (API), de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación web moderna. Los servidores de aplicación también brindan soporte a una gran variedad de estándares, tales como HTML, XML, IIOP, JDBC, SSL, que les permiten su funcionamiento en ambientes web (como Internet) y la conexión a una gran variedad de fuentes de datos, sistemas y dispositivos.**[27]**

Un ejemplo común del uso de servidores de aplicación (y de sus componentes) son los portales de Internet, que permiten a las empresas la gestión y divulgación de su información, y un punto único de entrada a los usuarios internos y externos. Teniendo como base un servidor de aplicación, dichos portales permiten tener acceso a información y servicios (como servicios Web) de manera segura y transparente, desde cualquier dispositivo.

JBoss

JBoss es un servidor de aplicaciones Open Source muy popular en el mundo empresarial. Tiene una calidad y rendimientos realmente altos. JBoss es el servidor de aplicaciones de código abierto líder del mercado.

Ofrece grandes ventajas en lo que se refiere a escalabilidad del sistema y utilidades de funcionamiento, al mismo tiempo que proporciona un entorno robusto y rápido para la ejecución de las aplicaciones, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combina una arquitectura orientada a servicios con una licencia de código abierto.**[28]**

Una de las principales ventajas que ofrece es su carácter abierto, ya que todos los módulos que proporciona permiten su utilización dentro de otros servidores, con lo cual su manejo no está reñido con la independencia del servidor de aplicaciones. Como servidor J2EE permite crecer a la medida de las necesidades con instalaciones cluster (varias máquinas actuando como una sola).

Características de JBoss AS:

- JBoss incluye ya parte del estándar JAVAEE5 como EJB 3.0 y será certificado próximamente (JBoss AS 5).**[28]**

GlassFish

GlassFish es un servidor de aplicaciones que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito y de código libre, se distribuye bajo la licencia CDDL y la GNU GPL.**[29]**

Características GlassFish

- GlassFish es el Servidor OpenSource de Sun Microsystems, creador de JAVA y de los estándares J2EE.
- GlassFish cuenta con plugins para Eclipse y con excelente soporte en NetBeans 5.5.
- GlassFish incluye las nuevas librerías de Web Services (JAX-WS 2.0) y es la base de las nuevas plataformas SOA en JAVA
- GlassFish tiene licencia CDDL (tipo Mozilla) y GPL que permite su mezcla con otros proyectos Open Source y otros tipos de licencias como Apache, Mozilla, LGPL, GPL etc.

1.10.4.2 Servidor Web

Un servidor web es un programa que implementa el protocolo HTTP (HyperText Transfer Protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML (HyperText Markup Language), textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música**.[30]**

Servidor HTTP Apache

Apache es el Servidor Web más comúnmente utilizado en sistemas GNU/Linux. Los Servidores Web son usados para servir Páginas Web solicitadas por ordenadores clientes. El protocolo más utilizado para ver páginas Web es el Hyper Text Transfer Protocol (HTTP). **[31]**

Protocolos como el Hyper Text Transfer Protocol sobre Secure Sockets Layer (HTTPS), y File Transfer Protocol (FTP), un protocolo para subir y descargar archivos, también son soportados.

Los Servidores Web Apache a menudo se usan en combinación con el motor de bases de datos MySQL, el lenguaje de scripting PHP, y otros lenguajes de scripting populares como Python y Perl. Esta configuración se denomina LAMP (Linux, Apache, MySQL y Perl/Python/PHP) y conforma una potente y robusta plataforma para el desarrollo y distribución de aplicaciones basadas en la web.**[31]**

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.**[31]**

Ventajas:

- Modular.
- Open source.
- Multi-plataforma.
- Extensible.
- Popular (fácil conseguir ayuda/soporte) .
- Gratuito.**[31]**

Servidor HTTP Cherokee

Servidor HTTP Cherokee es Servidor web libre, multiplataforma, abierto bajo la licencia GPL. Apunta a ser un servidor web bastante rápido que también soporta las funcionalidades más comunes. Está escrito completamente en C, es escalable y puede usarse como un Sistema Integrado.**[32]**

Cherokee ofrece todo lo que se puede necesitar, debido a sus características de rapidez, flexibilidad, facilidad de configuración y su amplio soporte de tecnologías web como por ejemplo SCDI, PHP, CGI, TLS o SSL, además de hosts virtuales, autentificación, y compatibilidad total con los ficheros de log de Apache.**[32]**

1.10.5 Modelo de Redes

1.10.5.1 LAN

LAN son las siglas de Local Área Network, Red de área local. Una LAN es una red que conecta los ordenadores en un área relativamente pequeña y predeterminada (como una habitación, un edificio, o un conjunto de edificios).

Las redes LAN se pueden conectar entre ellas a través de líneas telefónicas y ondas de radio. Un sistema de redes LAN conectadas de esta forma se llama una WAN (Red de Área Ancha).**[33]**

Las estaciones de trabajo y los ordenadores personales en oficinas normalmente están conectados en una red LAN, lo que permite que los usuarios envíen o reciban archivos y compartan el acceso a los archivos y a los datos. Cada ordenador conectado a una LAN se llama nodo.**[33]**

Cada nodo (ordenador individual) en un LAN tiene su propia CPU con la cual ejecuta programas, pero también puede tener acceso a los datos y a los dispositivos en cualquier parte en la LAN. Esto significa que muchos usuarios pueden compartir dispositivos, como impresoras láser y datos.

1.10.5.2 WAN

Una red de área amplia, WAN (Wide Area Network), es un tipo de red de computadoras capaz de cubrir distancias desde unos 100 hasta unos 1000 Km., dando el servicio a un país o un continente. Son construidas por y para una organización o empresa particular y son de uso privado, otras son construidas por los proveedores de Internet (ISP) para proveer de conexión a sus clientes.

Internet proporciona WAN de alta velocidad, y la necesidad de redes privadas WAN se ha reducido drásticamente mientras que las VPN (Virtual Private Network) que utilizan cifrado y otras técnicas.

Las redes WAN pueden usar sistemas de comunicación vía satélite o de radio. Fue la aparición de los portátiles y los PDA´s la que trajo el concepto de redes inalámbricas.**[33]**

Su función fundamental está orientada a la interconexión de redes o equipos terminales que se encuentran ubicados a grandes distancias entre sí. Para ello cuentan con una infraestructura basada en poderosos nodos de conmutación que llevan a cabo la interconexión de dichos elementos, por los que además fluyen un volumen apreciable de información de manera continúa. **[33]**

Las redes WAN tienen carácter público, pues el tráfico de información que por ellas circula proviene de diferentes lugares, siendo usada por numerosos usuarios de diferentes países del mundo para transmitir información de un lugar a otro. A diferencia de las redes LAN, la velocidad a la que circulan los datos por esta suele ser menor que la que se puede alcanzar en las LAN.

La infraestructura de Redes WAN la componen además de los nodos de conmutación, líneas de transmisión de grandes prestaciones, caracterizadas por sus grandes velocidades y ancho de banda en la mayoría de los casos. Las líneas de transmisión (también llamadas circuitos, canales o troncales) mueven información entre los diferentes nodos que componen la red**.[33]**
Los elementos de conmutación también son dispositivos de altas prestaciones, pues deben ser capaces de manejar la cantidad de tráfico que por ellos circula. A estos dispositivos les llegan los datos por una línea de entrada, y este debe encargarse de escoger una línea de salida para reenviarlos. **[33]**

Una arquitectura empresarial proveerá una arquitectura sólida y de alto desempeño para lograr una verdadera integración empresarial.**[34]**

1.11 Análisis

1.11.1 Plataforma de desarrollo

1.11.1.1 LAMP

LAMP son las siglas de Linux, Apache, MySQL y PHP que son paquetes de software intermedio o middleware. A estos en conjunto se les denomina LAMP o plataforma LAMP.

La L en algunos casos suele reemplazarle por Windows o Mac, según sea el sistema operativo sobre el cual se instala y ejecuta el resto del software. La plataforma LAMP permite ejecutar aplicaciones Web eficientes con un mínimo de requerimientos. Cada elemento de este conjunto cumple una función específica así:

- Linux/Windows/Mac: Un sistema operativo que se encarga de comunicarse directamente con el hardware.
- Apache: Servidor Web, que se encarga de servir peticiones basadas en protocolo HTTP.
- MySQL: Motor de base de datos. Permite almacenar y gestionar información relacional.
- PHP: Lenguaje de programación similar en la sintaxis a C o Java, muy eficiente y comprensible.**[35]**

Para conseguir ejecutar una aplicación Web sobre esta plataforma, es necesario instalar y realizar algunas configuraciones en el servidor Apache, así este sabrá que tiene que dirigir flujos de código PHP hacia el compilador, tomar la salida y devolverla al cliente en un formato atendible. **[36]**

LAMP está considerada como una de las mejores herramientas pues emplean un servidor web versátil y potente. Aunque creados por separado, cada una de las tecnologías que lo forman dispone de una serie de características comunes. **[36]**

Estos cuatro productos funcionan con requerimientos relativamente pequeños sin perder estabilidad. Esto hace que sea una alternativa adecuada para pequeñas y medianas empresas.

Todos los elementos que forman LAMP son software libre, de modo que disfrutan de las siguientes ventajas propias del mismo:

- Libertad de copia y distribución: Se puede conseguir gratuitamente. Hay muchísimas fuentes donde conseguir cualquiera de las distribuciones. Si no tienes una conexión rápida, también regalan Linux en los CD-ROM de muchas revistas especializadas.
- Libertad de modificación: Junto a los programas ejecutables, se puede obtener su código fuente. Esto, si se tienen los conocimientos necesarios, permite verificar la seguridad y eficiencia de los mismos, además de modificar y/o añadir las características y comportamientos que deseemos.**[36]**

1.11.1.2 J2EE

J2EE es conocida como Java 2 Enterprise Edition. Versión avanzada de la plataforma Java de Sun Microsystems, destinada al desarrollo de aplicaciones empresariales.)

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de n niveles distribuidos, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. **[3]**

J2EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML y define cómo coordinarlos. También configura algunas especificaciones de componentes que incluyen Enterprise JavaBeans, Servlets, Portlets Java, Java Server Pages y varias tecnologías de servicios web. **[3]**

Esto permite crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores.

El servidor de aplicaciones puede manejar transacciones, seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de tareas de mantenimiento de bajo nivel.

1.11.2 Sistema Operativo

1.11.2.1 Linux

- Multitarea: Varios programas (realmente procesos) ejecutándose al mismo tiempo.
- Multiusuario: Varios usuarios en la misma máquina al mismo tiempo.
- Funciona en modo protegido 386.
- Tiene protección de la memoria entre procesos, de manera que uno de ellos no pueda colgar el sistema.
- Carga de ejecutables por demanda: Linux sólo lee de disco aquellas partes de un programa que están siendo usadas actualmente.
- Política de copia en escritura para la compartición de páginas entre ejecutables: Esto significa que varios procesos pueden usar la misma zona de memoria para ejecutarse.
- Memoria virtual usando paginación (sin intercambio de procesos completos) a disco: Una partición o un archivo en el sistema de archivos, o ambos, con la posibilidad de añadir más áreas de intercambio sobre la marcha (se sigue denominando intercambio, es en realidad un intercambio de páginas).
- La memoria se gestiona como un recurso unificado para los programas de usuario y para el caché de disco, de tal forma que toda la memoria libre puede ser usada para caché y éste puede a su vez ser reducido cuando se ejecuten grandes programas.
- Se realizan volcados de estado (core dumps) para posibilitar análisis post-mortem, permitiendo el uso de depuradores sobre los programas no sólo en ejecución sino también tras abortar éstos por cualquier motivo.
- Todo el código fuente está disponible, incluyendo el núcleo completo y todos los drivers, las herramientas de desarrollo y todos los programas de usuario; además todo ello se puede distribuir libremente.
- Emulación de 387 en el núcleo, de tal forma que los programas no tengan que hacer su propia emulación matemática. Cualquier máquina que ejecute Linux parecerá dotada de coprocesador matemático.

- Consolas virtuales múltiples: Varias sesiones de login a través de la consola entre las que se puede cambiar con las combinaciones adecuadas de teclas (totalmente independiente del hardware de video). Se crean dinámicamente y puedes tener hasta 64.
- Sistema de archivos de CD-ROM que lee todos los formatos estándar de CD-ROM.
- TCP/IP, incluyendo ftp, telnet, NFS, etc.
- Appletalk disponible en el actual núcleo de desarrollo.
- Software cliente y servidor Netware disponible en los núcleos de desarrollo**.[37]**

1.11.2.2 Comparación de Linux y Windows

Ver Anexo 5

1.11.3 Lenguaje de Programación

1.11.3.1 C++

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, como extensión del lenguaje de programación C. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. **[38]**
Existen también algunos intérpretes como ROOT (enlace externo). Las principales características del C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica (*templates*). Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución (*RTTI*)

C++ está considerado como un lenguaje potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.**[38]**

Ventajas:

- Herencia múltiple, Genericidad, Plantillas.
- Funciones virtuales, Excepciones, etcétera.**[38]**

Características

- Programación orientada a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además permite la reutilización del código de una manera más lógica y productiva.
- Portabilidad: Un código escrito en C++ puede ser compilado en diferentes ordenadores y sistemas operativos sin hacer apenas cambios.
- Brevedad: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguaje es preferible el uso de caracteres especiales que las "palabras clave".
- Programación modular: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Esta característica permite unir código en C++ con código producido en otros lenguajes.
- Velocidad: El código resultante de una compilación en C++ es muy eficiente, por su capacidad de actuar como lenguaje de alto y bajo nivel.**[39]**

1.11.3.2 Java

El objetivo principal de Java es conseguir un entorno de desarrollo de software que sea independiente de la plataforma de ejecución. Necesita 128 MB de memoria por sesión de usuario. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser similar a C++ y así facilitar un rápido y fácil aprendizaje. **[40]**

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el *garbage collector* (reciclador de memoria dinámica).

El reciclador se encarga de liberar memoria y como es un *thread* de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria. **[40]**

Java reduce un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- Aritmética de punteros.

- No existen referencias.
- Registros (struct)
- Definición de tipos (typedef)
- Necesidad de liberar memoria (free) **[41]**

El intérprete completo de Java que hay en este momento es muy pequeño, solamente ocupa 215 Kb de RAM.

1.11.3.3 Breve comparación de C++ y Java

Ver Anexo 6

1)- Sencillez

Java tiene una sencillez que no posee C++. Esto es debido a que una de las razones de la creación de Java es la de obtener un lenguaje parecido a C++ pero reduciendo los errores más comunes de la programación, algo que se logra con mucho éxito puesto que Java reduce un 50% los errores que se comenten en C++ entre los que destacan:

- Eliminación de la aritmética de punteros y de las *referencias*.
- Desaparecen los registros (*struct*), heredados del paradigma estructurado.
- No se permite ni la definición de tipos (*typedef*) ni la de macros (*#define*).
- Ya no es necesario liberar memoria (*free o delete*).

2)- Robustez

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución, por lo que se detectan errores muy rápido. Algunas de estas verificaciones que hacen que Java sea un lenguaje robusto son:

- Verificación del *código de byte*.
- Gestión de excepciones y errores.
- Comprobación de punteros y de límites de vectores.

Se aprecia una clara diferencia con C++ quién no realiza ninguna de estas verificaciones.

3)- Seguridad

En Java no se permite los accesos ilegales a memoria, algo que sí se permitía en C++. Esto es algo muy importante puesto que este tipo de problema puede ocasionar la propagación de virus y otras clases de programas dañinos por la red.

El código Java pasa muchos tests antes de ejecutarse en una máquina. El código se pasa a través de un verificador de *código de byte* que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal, código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto.

Algunos de los conocimientos que se pueden obtener de los códigos de byte si pasan la verificación sin generar ningún mensaje de error son:

El código no produce desbordamiento de operandos en la pila.

- El tipo de los parámetros de todos los códigos de operación es conocido y correcto.
- No ha ocurrido ninguna conversión ilegal de datos, tal como convertir enteros en punteros.
- El acceso a los campos de un objeto se sabe si es legal mediante las palabras reservadas *public*, *private* y *protected*.

Por todo esto, y por no permitirlo mediante Java la utilización de métodos de un programa sin los privilegios del núcleo (*kernel*) del sistema operativo, la obligación de autentificación por clave pública para la realización de modificaciones, se considera Java un lenguaje seguro. Todo esto no lo incorporan C++, por lo que Java es considerado como el más seguro.

4)- Lenguaje interpretado

Java es un lenguaje que puede ejecutar el código directamente. Esto es una característica que no posee C++. No obstante, y aunque en teoría se consumen menos recursos siendo los lenguajes interpretados, el actual compilador que existe es muy lento, unas 20 veces menos rápido que C++. Esto normalmente no es vital para la aplicación ni demasiado apreciable por el usuario, y además esta diferencia se está reduciendo con los nuevos compiladores JIT (*Just In Time).*

5)- Dinamicidad

Para la obtención de un mayor provecho de la tecnología orientada a objetos, Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Esta característica no es contemplada por C++, que enlaza todos los módulos cuando se compila.

6)- Portabilidad

Un programa Java puede ser ejecutado en diferentes entornos, algo imposible para C++.

7)- Neutralidad

Se dice que Java tiene una arquitectura neutra puesto que compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (*JRE* o *Java Runtime Enviroment*) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

No es así para C++, donde el código generado podrá ejecutarse únicamente en la plataforma en la que se generó.

8)- Recolección automática de basura (Garbage Colection)

Java modifica completamente la gestión de la memoria que se hace en C/C++. En C/C++ se utilizan punteros, reservas de memoria (con las ordenes *malloc*, *new*, *free*, *delete*...) y otra serie de elementos que dan lugar a graves errores en tiempo de ejecución difícilmente depurables.

Java tiene operadores nuevos para reservar memoria para los objetos, pero no existe ninguna función explícita para liberarla.

La recolección de basura (objetos ya inservibles) es una parte integral de Java durante la ejecución de sus programas. Una vez que se ha almacenado un objeto en el tiempo de ejecución, el sistema hace un seguimiento del estado del objeto, y en el momento en que se detecta que no se va a volver a utilizar ese objeto, el sistema vacía ese espacio de memoria para un uso futuro.

Esta gestión de la memoria dinámica hace que la programación en Java sea más fácil.

9)- Representación

Uno de los objetivos perseguidos en el desarrollo de Java era la obtención de programas con interfaces cómodas e intuitivas. Esto también se permite en C++, aunque con unos métodos más costosos, y en ningún caso con interfaces portables como los que Java crea.

1.11.3.4 Active Server Pages (ASP)

Active Server Pages (ASP) es el lenguaje scripting del lado servidor de Microsoft para páginas Web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).**[42]**

Con las ASP se pueden realizar muchas aplicaciones distintas. Permite acceso a bases de datos, al sistema de archivos del servidor y en general a todos los recursos que tenga el propio servidor. Tiene la posibilidad de comprar componentes ActiveX fabricados por distintas empresas de desarrollo de software que sirven para realizar múltiples usos, como el envío de correo y generar gráficas dinámicamente. **[42]**

ASP es una tecnología que trabaja en función de lenguajes privatizados y su mayor desventaja es que solo se puede implementar en los servidores web de su desarrollador Microsoft. Aunque puede trabajar en Linux el sistema operativo que se recomienda es Windows.**[42]**

Actualmente se ha presentado ya la segunda versión de ASP, el ASP.NET, que comprende algunas mejoras en cuanto a posibilidades del lenguaje y rapidez con la que funciona. ASP.NET tiene algunas diferencias en cuanto a sintaxis con el ASP, de modo que se ha de tratar de manera diferente uno de otro y para implementarlo es necesario montar el Servidor de la Plataforma .NET. **[42]**

1.11.3.5 PHP

Es un lenguaje de programación interpretado para la creación de páginas Web dinámicas. PHP (Personal Home Page Tools) posee mucha documentación, es rápido y está empapado en HTML. **[43]**

Actualmente también se puede utilizar para la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, PostgreeSQL, Oracle, ODBC, DB2, Microsoft SQL Server, entre otras.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX y Windows, y puede interactuar con los servidores web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI .**[43]**

PHP corre en 7 plataformas, funciona en 11 tipos de servidores, ofrece soporte sobre unas 20 Bases de Datos y contiene unas 40 extensiones estables, además:

- Es software libre y abierto, lo que implica menos costes y servidores más baratos que otras alternativas. Es muy rápido.
- Su librería estándar es realmente amplia, lo que permite reducir los llamados "costes ocultos", uno de los principales defectos de ASP.
- PHP tiene una de las comunidades más grandes en Internet, por lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos.
- Posee una potente variedad de extensiones para el acceso a la mayoría de los sistemas de gestión de bases de datos, por lo que una migración a otro sistema de gestión es mucho menos costosa que en otras plataformas. **[43]**

1.11.3.6 Breve comparación de PHP y Java

- Sintaxis: Java es el que mejor sintaxis tiene, PHP aún usa '::' y '->' y algunas funciones podrían ser usadas dentro de los objetos y no como procedimientos.
- Curva de aprendizaje: PHP es el más sencillo, aunque a veces hay que fijarse bien en el orden de los parámetros de algunas funciones porque a veces no siguen la misma lógica. y Java el más complicado de aprender.
- Velocidad de desarrollo: PHP es rápido si se usa algún framework. Java es el más lento.
- Plataforma: PHP trabaja mejor en LAMP, aunque funciona también en otras plataformas, Java trabaja bien en cualquier plataforma.
- Base de datos: Normalmente es MySQL para PHP, Oracle para Java.
- IDE (Integrated Development Environments): Java tiene varias herramientas comerciales, pero Eclipse es la mejor (incluso alguna de las comerciales como WASD está basada en Eclipse). Para PHP no existe una que destaque sobre las demás, aunque también se puede usar Eclipse.
- Soporte orientado a objetos: Java es el mejor, aunque PHP ha mejorado en las últimas versiones.
- Seguridad: Java parece el más seguro, aunque PHP tiene mala fama, pero es debido sobre todo a los desarrolladores, no al lenguaje de programación.
- Rendimiento: Suele ganar PHP en cuestión de velocidad y recursos. Java es más pesado.
- Servidor Web: PHP y Java tiene versiones comerciales y open source.

- Librerías y frameworks: Todos tiene muchas librerías y frameworks disponibles, siendo para PHP y Java las grandes mayorías gratuitas y open source.
- Soporte y comunidad: Para PHP y Java existen muchos grupos independientes.
- Coste: PHP es la alternativa totalmente gratuita, mientras que Java puede desarrollarse con herramientas gratuitas y de pago.**[45]**

NuSOAP

NuSOAP es un kit de herramientas (ToolKit) para desarrollar Web Services en PHP. Está compuesto por una serie de clases que hacen mucho más fácil el desarrollo de Web Services.

Provee soporte para el desarrollo de clientes (que consumen los Web Services) y de servidores (aquellos que los proveen). NuSOAP está basado en SOAP 1.1, WSDL 1.1 y HTTP 1.0/1.1**[46]**

NuSOAP no es el único soporte para Web Services en PHP, existen otros, pero se encuentra en una fase de desarrollo más avanzada. PHP a partir de su versión 5 comienza a dar soporte para SOAP, pero aún está en fase experimental. **[46]**

Características de NuSOAP:

- Está en una fase madura de desarrollo.
- No necesita módulos adicionales.
- Es muy fácil su instalación y uso.**[46]**

1.11.3.7 Lenguaje de Marcado Extensible (XML)

XML (eXtensible Markup Language) permite definir la gramática de lenguajes específicos, esto facilita declaraciones más precisas de contenido y resultados entre muchas plataformas.**[47]**

La tecnología XML (XSD) busca dar solución al problema de expresar la información estructurada de la manera más abstracta y reutilizable posible.

La información estructurada esta compuesta por partes bien definidas como un árbol de tramos de información, estas partes se llaman elementos, y se señalan mediante etiquetas.**[47]**

Una etiqueta consiste en una marca hecha en el documento, que señala un segmento de información con un sentido claro y definido. Para que un objeto de texto sea considerado documento XML debe estar "bien formado" de acuerdo con la especificación XML.

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente anidada. Los elementos con contenidos deben estar correctamente cerrados.
- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.
- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.
- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos entendibles por las personas. **[47]**

XML Schema

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma precisa, con normas sintácticas impuestas por el propio lenguaje XML, obteniendo un documento con un nivel alto de abstracción.**[47]**

XML Schema supera muchas de las limitaciones y debilidades de las DTDs (Document Type Definition). Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos. Fue diseñado completamente alrededor de namespaces y soporta tipos de datos de los lenguajes de programación.

1.11.4 Tecnologías

1.11.4.1 AJAX

AJAX, acrónimo de *Asynchronous Java Script And XML* (Java Script asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

Éstas se ejecutan en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.**[48]**

AJAX es una combinación de tres tecnologías ya existentes: **Ver Anexo 7**

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como Java Script y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XML Http Request para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XML Http Request para realizar dichos intercambios. **[48]**

XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Utilizando HTML, JavaScript y CSS es posible crear atractivos clientes Web de aplicaciones empresariales creadas en otros lenguajes (por ejemplo J2EE).

La principal ventaja de AJAX es que está basado en estándares abiertos y funciona en la mayoría de los browsers modernos sin necesidad de plug-in.

AJAX tiene una serie de inconvenientes:

- Solo se puede conectar al back-end utilizando HTTP.
- Las páginas HTML que utilizan AJAX pueden llegar a ser muy complejas y aún no existen buenos entornos de desarrollo para esta tecnología.

1.11.4.2 Protocolo de Acceso Simple a Objetos

SOAP (Simple Object Access Protocol) es un protocolo 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.**[49]**

Este protocolo usa el código fuente en XML. Esto es una ventaja ya que facilita su lectura por parte de cualquier persona, pero también es un inconveniente dado que los mensajes resultantes son más largos.**[49]**
El intercambio de mensajes se realiza mediante tecnología de componentes. El término Object significa que se adhiere al paradigma de la programación orientada a objetos.

Algunas de las ventajas de SOAP son:

- No esta asociado con ningún lenguaje: Se puede elegir desarrollar con el último y mejor lenguaje de programación que exista. SOAP no especifica una API, por lo que la implementación se deja al lenguaje de programación.
- No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- No está atado a ninguna infraestructura de objeto distribuido: La mayoría de los sistemas de objetos distribuidos se pueden extender, para que admitan SOAP.
- Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema donde ya están definidas en la especificación de esquema con XML. Además de no definir un medio de trasporte de los mensajes.
- Permite la interoperabilidad entre múltiples entornos: Se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensajes SOAP con aplicaciones que se ejecuten en otras plataformas. **[49]**

1.11.4.3 JSP

Java Server Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Permiten la utilización de código Java mediante scripts. Además es posible utilizar algunas acciones JSP predefinidas mediante etiquetas.

Estas etiquetas pueden ser enriquecidas mediante la utilización de Librerías de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas. JSP puede considerarse como una manera alternativa, y simplificada, de construir Servlets. Es por esto que una página puede hacer todo lo que un servlet puede hacer, y viceversa. Cada versión de la especificación de JSP está fuertemente vinculada a una versión en particular de la especificación de Servlets.**[50]**

El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se presentará en la pantalla del Navegador del usuario.

1.11.5 Sistemas Gestores de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un tipo de software muy específico o conjuntos de ellos que permite manejar de manera clara, sencilla y ordenada altos volúmenes de datos.

Actualmente existe una amplia gama de SGBD con características propias, no obstante todos deben tener en cuenta los siguientes aspectos de manera general: abstracción de la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia.**[51]**

Existen SGBD muy potentes tanto en la clasificación de software propietario como software libre.

Características de los Sistemas Gestores de BD:

- Abstracción de la información: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. Lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- Respaldo y recuperación: Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- Control de la concurrencia: En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.**[51]**

Como SGBD propietarios se encuentran Oracle y Microsoft SQL Server que lideran el mercado por sus altas prestaciones durante muchos años de experiencia mientras que como opción libre se encuentra, entre otros, MySQL, gestor muy utilizado en la web por su simplicidad de uso, y

PostgreeSQL que si bien no soporta alguno de los aspectos más avanzados, si representa una opción muy confiable y comprometedora.

1.11.5.1 Oracle

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

Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux. **[52]**

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

1.11.5.2 PostgreeSQL

PostgreeSQL Versión 8.2.x es un sistema gestor de base de datos de objetos relacionales basado en software libre. Ofrece una alternativa ante otros sistemas gestores de bases de datos comerciales. Similar a proyectos de software libres como Apache y GNU/Linux, PostgreeSQL no es controlado por ninguna compañía, pero responde al esfuerzo de una comunidad global de desarrolladores. **[53]**

Mediante un sistema denominado MVCC (Acceso Concurrente Multiversión) PostgreeSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

Cada usuario obtiene una visión consistente de lo último que se le hizo al *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando el uso de bloqueos explícitos.**[54]**

Ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: Clases,

herencia, tipos y funciones. Además de aportan potencia y flexibilidad adicional como son las restricciones (constraints), disparadores (triggers), reglas (rules) e integridad transaccional. **[53]**

Algunas características del PostgreeSQL 8.2.x

Extensible

- El código fuente está disponible para todos sin costo. Si necesita extender o personalizar PostgreeSQL pueden hacerlo con un mínimo esfuerzo, sin costos adicionales.

Multiplataforma

- PostgreeSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas.

Diseñado para ambientes de alto volumen

- PostgreeSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones. **[73]**

1.11.6 Framework

Framework Para Java

1.11.6.1 Spring

Spring es un framework de aplicación de código abierto que ayuda a hacer el desarrollo en J2EE mucho más fácil. Ayuda a estructurar aplicaciones completas de una manera consistente y productiva para crear arquitecturas coherentes. Es el más popular y ambicioso de todos los framework de peso ligero. Además está diseñado para facilitar una flexibilidad arquitectónica. **[40]**

A pesar de que Spring no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y sustituta del modelo de Enterprise JavaBeans.

Por su diseño, el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.**[77]**

Mientras que las características fundamentales de este framework pueden emplearse en cualquier aplicación hecha en Java, existen muchas extensiones y mejoras para construir aplicaciones basadas en web.**[77]**

Presenta varios módulos de los cuales los principales son:

- Contenedor de Inversión de Control: Permite una sofisticada administración de configuración para POJOs12 y trabaja con otras partes de Spring para proveer servicios como la administración de la configuración.
- Módulo para la Programación Orientada a Aspectos (AOP): Permite comportamientos que deberían ser esparcidos de otra manera a través de diferentes métodos para ser modularizados en un simple lugar. Spring usa la AOP para implementar importantes servicios tales como administración de transacciones declarativas, y además es usado para implementar códigos estándar que debería de otra manera ser esparcido entre las clases de la aplicación.
- Una abstracción de acceso a datos: Spring anima a un consistente acercamiento arquitectónico para acceso a datos, y posee una abstracción única y poderosa para implementarlo. Presenta una rica jerarquía de excepciones de acceso a datos, independiente de cualquier framework de persistencia.
- Administración de transacciones*:* Presenta una abstracción de transacciones que puede mover transacciones globales JTA (Java Transaction Administration) (manipuladas por un servidor de aplicaciones) o transacciones locales usando JDBC (Java Data Base Conectivity), Hibernate, JDO (Java Data Object) u otro API de acceso a datos. Estas abstracciones presentan un consistente modelo de programación en una variedad de ambientes y es la base de la administración de transacciones programáticas y declarativas usadas por Spring.
- Comunicación remota de peso ligero (Lightweight remoting): Spring presenta comunicación remota basada en POJOs(Plain Old Java Objects) sobre un rango de protocolos, incluyendo RMI(Java Remote Method Invocation), IIOP, Hessian, Burlap, y otros protocolos de servicios web.
- No es un framework agresivo*:* Este es el principal problema de los framework anteriores. Mientras que los framework tradicionales tales como EJB (Enterprise Java Beans) o Apache Avalon forzaban al código de las aplicaciones a ser dependientes del framework, implementando interfaces específicas de estos framework o heredando clases específicas; ayuda a reducir las dependencias del código de la aplicación sobre el framework.

- Ayuda a promover la reusabilidad de código*:* Ayuda a evitar la necesidad de tomar decisiones importantes y duras, como es si una aplicación alguna vez usará JTA o JNDI (Java Naming and Directory Interface); las abstracciones de Spring permitirán desarrollar el código en un diferente ambiente si se necesita alguna vez.
- Es consistente: En diferentes ambientes de ejecución y en diferentes partes del framework, Spring usa un consistente acercamiento, pues una vez que se aprende una parte del framework, se puede aplicar el conocimiento en muchas aplicaciones.
- Promueve la selección arquitectónica*:* Spring provee la columna vertebral arquitectónica, apunta para facilitar el reemplazo de cada capa. Por ejemplo, con una capa media de Spring, se pudiera ser capaz de cambiar de un framework de mapeo objeto relacional (ORM) **[44]**

Dentro de las ventajas que nos ofrece Spring, nos encontramos con que facilita la manipulación de nuestros objetos, reduce la proliferación de *Singletons*, elimina la necesidad de usar distintos y variados tipos de ficheros de configuración, mejora la práctica de programación, permite el uso o no de EJBs, realizando el mismo tipo de funciones sin ellos.**[77]**

1.11.6.2 Hibernate

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 de Java.

Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y gran número de tipos de datos. De una manera muy rápida y optimizada se puede generar una BD para cualquiera de los entornos soportados: Oracle, DB2, MySQL. **[40]**

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.
- Proporciona persistencia de una manera transparente para el desarrollador.
- Soporta el paradigma de la programación orientada a objetos de forma natural: herencia, polimorfismo, composición y el framework de colecciones de Java.

- Soporte para modelos de objetos con una granularidad muy fina y 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 las interacciones largas (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. **[40]**

En cuanto al manejo de consultas Hibernate saca una ligera ventaja ya que tiene su propio lenguaje “HQL (Hibernate Query Language)” que lo hace multi – motor de base de datos, eso es uno de los atractivos de Hibernate. **[74]**

El HQL, diseñado como una extensión mínima, orientada a objetos, de SQL, proporciona un puente elegante entre los mundos objeto y relacional. Hibernate ofrece facilidades para recuperación y actualización de datos, control de transacciones, repositorios de conexiones a bases de datos, consultas programáticas y declarativas, y un control de relaciones de entidades declarativas. **[76]**

Hibernate proporciona grandes beneficios como es la independencia de la base de datos, bajo acoplamiento entre negocio y persistencia, y un desarrollo rápido, con Hibernate podremos cubrir de manera sencilla y rápida el 80 - 90% de la persistencia de nuestra aplicación. Esto nos permite centrar nuestros esfuerzos en optimizar las consultas que realmente lo merecen. **[76]**

Hibernate es menos invasivo que otros marcos de trabajo de mapeo O/R. Se utilizan Reflection y la generación de bytecodes en tiempo de ejecución, y la generación del SQL ocurre en el momento de la arrancada. **[76]**

1.11.6.3 Symfony

Symfony versión 1.0.x es un framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. **[55]**

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreeSQL, Oracle y Microsoft SQL Server.

Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón MVC, soporta AJAX, plantillas y un gran número de bases de datos.**[55]**

Proporciona herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Algunos de los principales aspectos de Symfony son:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, y muy flexible en 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, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.**[55]**

1.11.6.4 CakePHP

CakePHP es un framework de desarrollo de aplicaciones web escrito en PHP, creado sobre los conceptos de Ruby on Rails.

CakePHP facilita al usuario la interacción con la base de datos mediante el uso de Active Record. CakePHP es un framework para php que nos permite programar más rápido evitándonos escribir código tedioso de tareas muy comunes.**[56]**

Además hace uso del patrón Modelo Vista Controlador.

- Compatible con PHP4 y PHP5.
- CRUD de la base de datos integrado.
- URLs amigables.
- Sistema de plantillas rápido y flexible.
- Helpers para AJAX, JavaScript, HTML, forms y más.
- Trabaja en cualquier subdirectorio del sitio.
- Validación integrada.
- Scaffolding de las aplicaciones.
- Access Control Lists.
- Sanitización de datos.
- Componentes de seguridad y sesión. **[56]**

1.11.7 Entorno de Desarrollo Integrado

1.11.7.1 Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.**[57]**

La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS.
- Integración con Ant.

- Asistentes (wizards): Para creación de proyectos, clases, tests, etc.
- Refactorización.

Asimismo, a través de "plugins" libremente disponibles es posible añadir:

- Control de versiones con Subversion.
- Integración con Spring.
- Integración con Hibernate. **[57]**

Eclipse es un completo IDE (Integrated Development Environment) de programación, fue desarrollado originalmente para construir proyectos Java aunque dada su gran evolución y creciente popularidad ahora se puede crear todo tipo de proyectos en numerosos lenguajes de programación. **[72]**

1.11.7.2 NetBeans

Se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un Entorno Integrado de Desarrollo (IDE) desarrollado usando la Plataforma NetBeans en su versión 6.0.

El soporte de Java Enterprise Edition es de importancia, en especial para los desarrolladores de JBuilder. Dado que cuenta con el mejor soporte a estándares industriales de la tecnología Java, el proyecto NetBeans ha hecho que el desarrollo de aplicaciones Java de tipo empresarial sea rápido y sencillo, su facilidad de uso, su cumplimiento de regulaciones, sus perfiles de rendimiento, además de su flexibilidad entre plataformas. **[58]**

Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos.**[58]**

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. Los desarrolladores acostumbrados a trabajar con herramientas basadas en la tecnología Java, tales como el conjunto de herramientas Borland JBuilder, descubren que la migración a NetBeans puede acelerar en forma significativa los esfuerzos de desarrollo.**[58]**

1.11.8 Herramientas

1.11.8.1 Herramienta CASE de Modelado

Visual Paradigm Versión 3.0

Esta herramienta CASE (Computer-Aided Software Engineering), ha sorprendido gratamente por el trabajo que se puede llegar a desarrollar básicamente. Para gestionar la persistencia y el mapeo de estas clases con la base de datos utiliza Hibernate para Java y NHibernate en el caso de un proyecto .Net.

Es capaz de desplegar todas las clases asociadas a las tablas (siguiendo el patrón de diseño Una Clase-Una Tabla). **[59]**

El Visual Paradigm es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto, genera la documentación del proyecto automáticamente en varios formatos como Web o .Pdf, además de permitir el control de versiones.

- Es robusto y portable.

Genera código y realiza ingeniería inversa para diez lenguajes de programación. Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue. **[59]**

Además exporta e importa los diagramas con estándar XML y como imágenes. Es gratis en su edición Community y es multiplataforma.

Visual Paradigm es una herramienta importante para desarrollar un proyecto importante. **[59]**

Rational Rose

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.**[60]**

El navegador UML de Rational Rose nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de casos de uso, lógica, de

componentes y de despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción. **[60]**

1.11.8.2 Software para el Control de Versiones

Subversion versión 1.4.6

Subversion es un software de sistema de control de versiones de código abierto y gratuito. Soporta el manejo de ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. **[61]**

El repositorio es como un servidor de ficheros ordinario, excepto que recuerda todos los cambios hechos a sus ficheros y directorios, permitiendo recuperar versiones antiguas de datos, o examinar el historial de cambios de los mismos. **[61]**

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Se le conoce también como svn por ser el nombre de la herramienta de línea de comandos. Una característica importante del Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. **[61]**

Características principales:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- Crear ramas y etiquetas es una operación más eficiente.
- Tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto). **[61]**

1.11.8.3 Software para la Gestión Proyectos

DotProject Versión 2.0.4

DotProject fue creado con el fin de construir una herramienta para la Gestión de Proyectos. Está construido por aplicaciones de Código Abierto y es mantenida por un pequeño pero dedicado grupo de voluntarios. Es una aplicación basada en web, multiusuario, soporta varios lenguajes y es software libre.**[62]**

Utiliza MySQL como base de datos (aunque otros motores como PostgreeSQL también pueden ser utilizados). La plataforma recomendada para utilizar DotProject se denomina LAMP (Linux + Apache + MySQL + PHP). De todas formas, existen binarios para instalar DotProject bajo otros sistemas operativos tales como Microsoft Windows (NT, 2000, XP) y Mac.**[62]**

El grupo que desarrolla DotProject basa su espíritu de trabajo en los siguientes puntos:

- Proveer a los usuarios de funcionalidad orientada a la Gestión de Proyectos.
- Construir una herramienta con una interfaz de usuario simple, claro y consistente.
- Código abierto, libre acceso y utilización.**[62]**

La última versión de DotProject, es la versión 2.0.4 y fue liberada en Junio del 2006. Existen dos tipos de distribuciones que dependen de la plataforma sobre la que se pretende instalar el producto (Linux o Windows).

Planner

El Planner es una herramienta para planear, programar y seguir proyectos para el escritorio GNOME. Es un proyecto de código abierto destinado a ser una alternativa ante las herramientas propietarias disponibles.

Es normal que todo este organizado, más que nada por la comodidad de tener todo bien estructurado, y saber exactamente donde se está situado en el proyecto, que falta y que se ha hecho hasta el momento.

Software para gestionar hay muchos y paras diversas plataformas, entre los más conocidos se encuentra el Microsoft Project para Windows y aunque no es muy conocido también existe el Planner para GNU/Linux-Gnome.**[63]**

Entre sus características se encuentran:

- Gestión de Calendarios.
- Gestión de Recursos.
- Seguimiento del avance del proyecto.
- Enlazar tareas.
- Exportación a diferentes formatos.**[63]**

También utiliza un formato de archivo basado en XML o una base de datos PostgreeSQL para almacenar los detalles del proyecto y cuenta con diagramas de Gantt y gestión de tareas y recursos.**[63]**

1.11.8.4 Software para la Gestión Documental

Trac ha sido concebido de forma modular donde se pueden añadir plugins que proporcionan distintas funcionalidades. (Casi todos los componentes estándar son módulos que pueden ser activados, desactivados o reemplazados o modificados por otros).**[78]**
Trac únicamente es una interfaz de lectura del repositorio, se puede integrar al Subversion usando múltiples medios (https: //, svn: //, svn+ssh: //, etc.).

Actualmente existen plugins para añadir funcionalidades al **Trac** como:

- Autenticación con formularios y usuarios en LDAP, BBDD o fichero.
- Uso de otros VCS como Bazaar, GIT, Mercurial o Monotone.
- Servicios adicionales como blogs, foros, etc.

Trac puede funcionar de dos formas:

- Mediante su propio servidor (tracd).
- Mediante un servidor estándar (lighttpd, apache2) que tenga soporte para ejecutar código Python usando scripts de CGI, FastCGI o mod_python.

1.11.8.5 Software para Salvas Automáticas

Hay muchas aplicaciones de copia de seguridad, pero quizás ninguna tan completa y a la vez tan sencilla como Genie Backup Manager.

Este programa, además de realizar copias de todos los ficheros y carpetas que selecciones, guarda también los mensajes de correo electrónico e incluso una copia de tu Escritorio.**[64]**

La copia de seguridad se puede guardar en el propio disco local, en unidades extraíbles, en una unidad conectada en red o incluso en un servidor FTP. Los datos se pueden guardar como un fichero EXE que permite recuperar la copia sin necesidad de tener Genie Backup Manager instalado.**[65]**

También se tiene la posibilidad de definir una serie de programas que se utilice con frecuencia y realizar una copia de seguridad especial de los mismos, ahorrando así futuras reinstalaciones. La interfaz del programa es muy intuitiva y sencilla de manejar, y además permite seleccionar entre dos métodos de copia, de dos o cuatro pasos.

Mejoras de la nueva versión: Cifrado de las copias de seguridad con algoritmos de codificación, división automática en varios CD, tamaño de copia ilimitado, opción de compresión, mejora del proceso de grabación.**[65]**

Características de Backup Manager v6.1:

- Backup Manager v6.1 es una herramienta GNU/Linux para realizar copias de seguridad a través de línea de comandos.
- Backup Manager v6.1 está escrito en bash y Perl, y puede crear archivos tar, tar.gz, tar.bz2, y Zip además de ser ejecutado en modo paralelo con diferentes archivos de configuración.
- Backup Manager v6.1 realiza backups incrementales y los archivos se mantienen por un número determinado de días y puede utilizarse FTP, SSH o RSYNC, para transferir los archivos generados a una lista de host remotos.
- El archivo de configuración que utiliza esta herramienta es muy simple y básico. **[65]**

Mejoras de la nueva versión: Cifrado de las copias de seguridad con algoritmos de codificación, división automática en varios CD, tamaño de copia ilimitada, opción de compresión y mejora del proceso de grabación.

Conclusiones

En este capítulo 1 se han analizado algunos aspectos teóricos que serán de gran ayuda para el desarrollo del capítulo 2 para proponer una solución. Se profundizó en el conocimiento de algunos conceptos necesarios para la compresión de este trabajo. Además se realizó un análisis completo de las tecnologías que serán utilizadas a lo largo del desarrollo del sistema propuesto, y se fundamentaron las elecciones del lenguaje, la arquitectura, el Sistema Gestor de Bases de Datos, y la metodología a utilizar. Además de aspectos de gran importancia que conforman la arquitectura empresarial.

2

CAPÍTULO 2
Solución de la Arquitectura de Software Propuesta para PNICG

Introducción

En este capítulo, se especifica la tecnología adecuada para llevar a cabo el desarrollo del Programa Nacional de Informatización del Conocimiento Geológico. En el PNICG se quiere implantar la arquitectura empresarial con énfasis en la reducción de costos y simplificación de los sistemas, donde primen procesos de consolidación a todo nivel y reutilización de servicios. Como tema indispensable se exponen las propuestas de lenguajes, tecnologías y herramientas a utilizar para el desarrollo del mismo, teniendo en cuenta el resultado de la investigación en el capítulo anterior. Además en algunos casos fue necesario hacer una comparación muy exhaustiva acerca de los determinados software libres que se utilizan para fundamentar la propuesta final.

2.1 Definición de la Arquitectura a Utilizar

2.1.1 Plataforma de Desarrollo

Una plataforma de desarrollo estándar para desarrollar aplicaciones empresariales es la J2EE que impacta fuertemente en el sector empresarial al establecer una infraestructura basada en estándares abiertos para aplicaciones distribuidas incluyendo transacciones, seguridad, componentes server-side, acceso a datos, persistencia, clustering y balanceo de carga. Aunque está en un proceso de madurez, la plataforma J2EE es una plataforma muy robusta, además de correr en múltiples sistemas operativos, máquinas hardware y cuyos usuarios pueden seleccionar la implementación que más le convenga.

2.1.1.1 J2EE

Una de las soluciones propuestas y más importantes es la plataforma de desarrollo, que investigando en varios documentos, libros y foros, además de investigar sobre importantes comparaciones, se llegó a la conclusión que la plataforma J2EE es la adecuada para el Programa

Nacional de Informatización del Conocimiento Geológico pues le permite al desarrollador crear una aplicación empresarial portable entre plataformas y escalable, a la vez integrable con tecnologías mencionadas anteriormente.

J2EE ofrece las siguientes ventajas:

- Soporte de múltiples sistemas operativos: Al ser una plataforma basada en el lenguaje Java, es posible desarrollar arquitecturas basadas en J2EE utilizando cualquier sistema operativo donde se pueda ejecutar una máquina virtual Java.
- Competitividad: Muchas empresas crean soluciones basadas en la plataforma J2EE para ofrecer características como rendimiento y precios muy diferentes. De este modo el cliente tiene una gran cantidad de opciones a elegir.
- Soluciones libres: En la plataforma J2EE es posible crear arquitecturas completas basadas en productos de software libre. Además los arquitectos normalmente disponen de varias soluciones libres para cada una de las partes de su arquitectura. **[68]**

J2EE tiene importantes desventajas.

- Depende de un único lenguaje: La plataforma J2EE depende exclusivamente del lenguaje Java. Sólo se puede utilizar este lenguaje para desarrollar aplicaciones lo que puede suponer un gran problema si el equipo no dispone de los conocimientos suficientes o tiene otras preferencias.
- Complejidad: Aunque no es una plataforma tan compleja, la creación de aplicaciones bajo J2EE requiere normalmente desarrolladores más experimentados que los necesarios para desarrollar bajo .NET
- Heterogeneidad: Existe una gran heterogeneidad en las soluciones de desarrollo. No existe en J2EE un similar a Visual Studio .NET. Las herramientas disponibles causa confusión dentro de los desarrolladores y puede crear dependencias dentro de las empresas. **[68]**

2.1.2 Sistema Operativo

Una de las decisiones más importantes a tomar en un proceso de automatización es el Sistema Operativo a utilizar.

Dentro del estudio realizado en el capítulo anterior se selecciona Linux por ser actualmente uno de los sistemas operativos más rápidos, robustos y estables ya que comprende una serie de características importantes como son:

- Es un software de código abierto, libre (gratuito), multitarea, multiproceso y confiable.
- Presenta gran seguridad porque aún no hay un virus que represente una amenaza para Linux. Esto significa que no se necesitará comprar una licencia de antivirus.
- Tiene gran flexibilidad porque cada distribución de GNU/Linux incluye el código fuente, el cual es libre de modificar a sus necesidades.
- Aprovecha el 100% del procesador.
- Linux aprovecha el 100% de la memoria RAM y asegura memoria física solo a programas que están corriendo en primer plano, además tiene un sistema de memoria virtual llamada swap.
- Incluye la mayoría de programas necesarios para el usuario**.**
- Incluye software para programación estructurada y orientada a objetos.
- Seguridad de datos e información por medio de autenticación.**[70]**

2.1.3 Lenguaje de Programación

Un lenguaje de programación adecuado requiere de amplia y sólida documentación, que haya sido aplicada con anterioridad y que cuente con un buen soporte para software. Además que elimine ciertos errores de la programación y muy comunes en los programadores. Además de ser un lenguaje robusto y orientado a objetos.

2.1.3.1 Java

Dentro de los lenguajes estudiados en el capítulo anterior y analizado sus características se selecciona Java, lenguaje de programación orientado a objetos, que va a tomar muchas de sus sintaxis de C y C++. Nuestra propuesta va a utilizar en la plataforma J2EE pues tiene como principal desventaja que exige como único lenguaje de programación Java.

Una de las principales características por las que Java es reconocido entre los programadores es por su plataforma independiente.

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. En general, cualquier aplicación que deseemos hacer con acceso a través web se puede hacer utilizando Java.**[71]**

2.1.4 Entorno Integrados de Desarrollo (IDE)

Los entornos de programación pueden variar desde la forma más rudimentaria, basada en una línea de comandos desde donde invocar al editor, al compilador y al probar el programa hasta los sofisticados entornos de desarrollo (IDE). Los programadores recurren a un Entorno de Desarrollo Integrado con el fin de facilitar su tarea de aprendizaje y desarrollo. La oferta de IDEs para el lenguaje Java es variada, abundante y es destacable pues muchos son productos libres. La elección de un IDE no es fácil, solo es necesario que posea las características obvias y generales que debe ofrecer como son editor sin corrección sintáctica y coloreo, herramientas gráficas, soporte integrado para la compilación, ejecución de programas, y opciones de debugging.

2.1.4.1 Eclipse

Dentro de los IDE estudiados en el capítulo anterior se tiene como propuesta el Eclipse versión 3.2, es un software de código abierto, independiente de la plataforma para desarrollar aplicaciones. Además es software libre, multi-lenguaje, extensible y práctico, no requiere instalación, ni gestión de proyectos.

Los requisitos fundamentales para instalar Eclipse son básicamente, contar con unos 200Mb de espacio en HD, una potente CPU, 512Mb de RAM y tener al menos el JRE (Java Runtime Environment) instalado para poder ejecutarlo correctamente. **[72]**

2.1.5 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos debe permitir la integridad de datos de la base de datos refiriéndose a la validez y la consistencia de los datos almacenados mediante restricciones o reglas que no se pueden violar. El SGBD es quien se debe encargar de mantenerlas y establecer medidas de seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. Requerir de gran capacidad para volúmenes de datos.

Los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo. Además debe ser OpenSource y multiplataforma.

2.1.5.1 PostgreeSQL

Como propuesta de SGBD se halla el PostgreeSQL Versión 8.2.x no es controlado por ninguna compañía en específico, basado en software libre, además de ofrecer muchas ventajas para su compañía o negocio respecto a otros sistemas de bases de datos.

Instalación Ilimitada:

- Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Se puede usar PostgreeSQL, pues no estaría violando acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.**[73]**

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.**[73]**

2.1.6 Framework

Los frameworks son la piedra angular de la moderna ingeniería del software y están ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente (source code).

Los frameworks son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados. La capacidad de reutilización del código y del diseño de frameworks permite una productividad mayor y un tiempo breve en el desarrollo de aplicaciones.

2.1.6.1 Hibernate

Hibernate es una herramienta para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Hibernate es una herramienta ORM (Object Relational Mapping) completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto OpenSource líder en este campo gracias a sus prestaciones, buena documentación y estabilidad. **[74]**

Utilizar un framework de ORM simplifica enormemente la programación de lógica de persistencia. Se trata de una idea completamente madura que cada vez se vuelve más popular. Hibernate es una buena herramienta en lo que se refiere a mapeo de clases en una base de datos relacional, pero en lo que se refiere al manejo de transacciones y conexiones le falta funcionalidad y capacidad.

2.1.6.2 Spring

El principal objetivo de Spring es el constituirse en una alternativa sencilla y fácil ante EJB. La simplificación del desarrollo de aplicaciones y pruebas (testing) es una de las claves del éxito de Spring. Este Framework se sustenta en su núcleo: Inversion de Control (Inversion of Control IoC) y la Programación Orientada a Aspectos (Aspect-Orient Programming).**[40]**

AOP es usada en Spring para:

- Proveer servicios de aplicación (Enterprise Services) declarativos. Ejemplo declarar el manejo de transacciones.
- Permitir a los usuarios la facilidad de implementar sus propios aspectos personalizados. **[68]**

Es un framework muy simple, conveniente y flexible, pero al mismo tiempo muy poderoso. Se recomienda el uso de Spring en los casos donde un contenedor de aplicaciones pesado no es necesario.

2.1.7 Sistema de Control de Versiones

El control de versiones se basa en una serie de acciones más o menos estándar de comunicación entre la copia de trabajo y el repositorio. Estas acciones son precisamente las que permite el cliente Subversion.

Cuando se analiza el tema de sistema de control de versiones se entendió que el mejor es el subversión que también es conocido como SVN, es un sistema de control de versiones que se ha popularizado bastante, dentro de los desarrolladores de software libre.

Dentro de las principales características están:

- Mantiene versiones no solo de archivos, sino también de directorios.
- Los cambios en el contenido de los documentos se mantiene pues guarda la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, pues las transacciones solo transmiten las diferencias y no los archivos completos.
- Mayor eficiencia en la creación de ramas y etiquetas**.[61]**

2.1.8 Herramienta de Gestión Documental

Existen paquetes de software que proporcionan las funcionalidades necesarias para la gestión documental, ya sea por separado o integrando múltiples funcionalidades en un único producto (generalmente de pago).

Alternativas Libres

Trac es un sistema web, multiplataforma, ligero y extensible que integra varios componentes con capacidades suficientes para la gestión documental de proyectos de desarrollo de software. Es un programa pensado para desarrolladores que necesitan mantener un proyecto.

Trac

- Un sistema para definir y visualizar el estado de los hitos de un proyecto (un hito incluye una descripción y una fecha y se usa como atributo de los tickets, que se asocian a hitos concretos).
- Un sistema de seguimiento de eventos en el sistema (histórico de cambios en el wiki, en el sistema de control de versiones, en el sistema de gestión de incidencias o vencimiento de un hito). **[78]**

2.1.9 Herramienta de Modelado

En los últimos años se han publicado diversos estudios y estándares en los que se exponen los principios que se deben seguir para la mejora de los procesos de software.

Una metodología para el desarrollo de un proceso de software es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de Sistemas Informáticos. Por ello escoger la metodología que va a guiar el proceso de desarrollo del sistema es un paso primordial.

Se decidió utilizar como metodología para controlar y planificar trabajo el Proceso Unificado de Modelado (RUP), por sus características y las facilidades que aporta a todo el proceso. Como herramienta de modelado de software se propone la herramienta Case Visual Paradigm.

2.1.9.1 UML

El UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo las unidades conceptuales, tales como procesos del negocio y funciones del sistema, como las unidades concretas, clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables**[79]**.

2.1.9.2 RUP

El Proceso Unificado es una propuesta de proceso para el desarrollo de software orientado a objetos que utiliza Unified Model Language *(UML)* para describir el proceso. Está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.

Sus características principales son:

1. Guiado/Manejado por casos de uso.
2. Centrado en arquitectura.
3. Iterativo e Incremental.
4. Desarrollo basado en componentes.
5. Utilización de un único lenguaje de modelación.
6. Proceso Integrado. **[79]**

Entre las características y facilidades que posee RUP se encuentran las siguientes:

- Es una plataforma flexible de procesos de desarrollo de software que ayuda ofreciendo guías consistentes y personalizadas de procesos para todo el equipo de proyecto.
- Se adapta a las necesidades de los proyectos: La plataforma **RUP** proporciona un framework de proceso configurable que permite seleccionar e implantar los componentes específicos de procesos necesarios para proporcionar consistencia para cada equipo y proyecto. **[79]**

2.1.9.3 Visual Paradigm

Es una potente herramienta CASE para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permita diseñar un producto con calidad de una forma rápida. **[59]**

Facilita la interoperabilidad con otras herramientas CASE como el Rational Rose y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE,

Debido a que el sistema operativo que se va a utilizar es Ubuntu 7.10 (distribución de GNU/Linux) se decidió utilizar el Visual Paradigm para visualizar y diseñar los elementos de software debido a que es multiplataforma y por las facilidades que brinda para el diseño de los diagramas necesarios y su documentación, además es mucho más fácil de utilizar que el Rational Rose pues genera los script de la Base de Datos e incluso posibilita crear Bases de Datos en PostgreeSQL y MySQL. **[59]**

2.1.10 Herramienta de Gestión de Proyecto

En el PNICG se necesita de una aplicación basada en web que sirva para solucionar problemas de gestión de proyectos.

El DotProject usa la interfaz intuitiva, presenta una estructura modular que soporta elementos de configuración, que pueden gestionarse a través del Entorno de Administración del Sistema.

El DotProject permite llevar un control de los costos del proyecto, calendario de entregas, reuniones, entregas de tareas, prototipos, gestión de eventos, agenda de contactos y foros. Su interfaz no es la más sencilla de utilizar pero actualmente, presume de ser una aplicación en expansión y que aporta muchas ventajas.

2.1.11 Sistema para Seguimiento de Errores

Un sistema de seguimiento de errores es una aplicación informática diseñada para asistir a los programadores y otras personas involucradas en el desarrollo y uso de sistemas informáticos en el seguimiento de los defectos de software.

Muchos sistemas de seguimiento de errores de Software libre permiten que los usuarios directamente den de alta la incidencia detectada, en muchas empresas de desarrollo de software se usan de manera estrictamente interna.**[80]**

Uno de los componentes principales de un sistema de seguimiento de errores es la base de datos donde se almacenan los hechos e historia de un fallo de software. Los hechos pueden ser una descripción detallada del fallo, la severidad del evento, forma de reproducirlo y los programadores que intervienen en su solución así como información relacionada al proceso de administración de la corrección del fallo como puede ser personal asignado, fecha probable de remedio y código que corrige el problema.**[81]**

Bugzilla y GNOME proponen la siguiente clasificación de severidad y prioridad de un problema de software:

- Bloqueador: Inhibe la continuidad de desarrollo o pruebas del programa.
- Crítico: Crash de la aplicación, pérdida de datos o fuga de memoria severa.
- Mayor: Pérdida de funcionalidad, como menúes inoperantes, datos de salida extremadamente incorrectos, o dificultades que inhiben parcial o totalmente el uso del programa.
- Normal: Una parte menor del componente no es funcional.
- Menor: Una pérdida menor de funcionalidad, o un problema al cual se le puede dar la vuelta.
- Trivial: Un problema cosmético, como puede ser una falta de ortografía o un texto desalineado.
- Mejora: Solicitud de una nueva característica o funcionalidad.**[81]**

Es inevitable, los programas tienen errores. No importa cuanto lo pruebes y el cuidado que se tenga en desarrollar la aplicación, siempre salen errores inesperados. Puesto que los errores forman parte del software y son bastante peligrosos, en cuanto a coste e imagen, es mejor tenerlos controlados**.**

2.2 Definición de Arquitectura Empresarial

En el capítulo anterior se abordaron detalles muy importantes para desarrollar una arquitectura reusable y flexible. Donde se destaca la arquitectura empresarial para disminuir las diferencias que existen hoy con el personal, los procesos de negocio, y las tecnologías. Esta investigación surgió para desarrollar una propuesta con la cual se logrará una uniformidad arquitectónica, adecuándose a las necesidades del Programa Nacional de Informatización del Conocimiento Geológico.

La Arquitectura Empresarial no es un proyecto de grandes dimensiones, sino una cultura de trabajo que permeará a todos los niveles de la empresa.

Frecuentemente, los equipos de desarrollo están en contra de la Arquitectura, considerándola como algo sin mucha importancia, ignorando el valor que genera su utilización. Logrando una adecuada alineación con el proceso de desarrollo, la Arquitectura mejora la calidad, los tiempos de desarrollo, además de garantizar consistencia al proceso. **[14]**

El concepto de Arquitectura Empresarial lleva varias distinciones ocultas, pero hay un solo objetivo implícito que es definir una forma ordenada de proveer a todos los niveles de la empresa un marco de trabajo claro, donde todos los integrantes estén involucrados, participando todos los niveles y finalmente apoyando las estrategias y metas del negocio.

La arquitectura empresarial no es una opción, existe en todas las organizaciones, algunas han sido diseñadas, mientras otras simplemente existen como resultado de la evolución de la organización, sin importar como impactan a la eficiencia de la organización.

2.2.1 ¿Por qué es importante la Arquitectura Empresarial?

Las organizaciones se enfrentan a un conjunto que situaciones ambientales e internas que las empujan a optimizar sus operaciones, de forma tal que sea capaz de responder con rapidez a los cambios estratégicos, amenazas externas o regulaciones de la industria, y lograr esto en la forma más costo-efectiva posible a fin de tener ventaja competitiva.**[15]**

Las organizaciones definen un conjunto de objetivos estratégicos del negocio de gran importancia, que deben ser soportados por las tecnologías de la información.

La razón primaria para desarrollar una arquitectura empresarial es soportar los objetivos del negocio proporcionando la tecnología fundamental y los procesos estructurados para una

estrategia de TI. Esto a su vez, hace que TI sea un activo capaz de responder a una estrategia de negocio moderna y exitosa.**[15]**

2.2.2 ¿Qué beneficios brinda una Arquitectura Empresarial?

Una arquitectura empresarial habilita a la organización para alcanzar el correcto balance entre eficiencia tecnológica e innovación del negocio.

La arquitectura empresarial asegura las necesidades de la organización permitiendo la mayor sinergia posible entre sus integrantes.**[15]**

Las ventajas tecnológicas resultantes de una buena arquitectura empresarial brindan beneficios importantes al negocio que son visibles en los resultados:

- Una operación de TI más eficiente.
    1. Menores costos de desarrollo, soporte y mantenimiento de software.
    2. Mayor portabilidad de aplicaciones.
    3. Interoperabilidad mejorada y administración de sistemas y redes más sencilla.
    4. Una mejor capacidad para atender asuntos que afectan toda la organización como la seguridad.
    5. Mayor facilidad para cambiar y actualizar componentes de sistemas.
- Mejor retorno en inversiones actuales y un menor riesgo en inversiones futuras.
    1. Reducción en la complejidad de la infraestructura de TI.
    2. Máximo retorno de inversión en la infraestructura existente.
    3. Flexibilidad para hacer, comprar o tercerizar soluciones de TI.
- Un proceso de adquisición más rápido, sencillo y económico
    1. Las decisiones de compra son más sencillas, dado que la información para gobernar este proceso está disponible a primera mano en un plan coherente.
    2. El proceso de adquisición es más rápido, maximizando la velocidad y flexibilidad para adquirir tecnología sin sacrificar la coherencia de la arquitectura.**[15]**

2.2.3 Significado de la "Arquitectura Empresarial de Tecnología" y la "Arquitectura Empresarial de Software"

La Arquitectura Empresarial es una herramienta conceptual que facilita a las organizaciones el entendimiento de su estructura y la forma en la que interactúan las diferentes soluciones y

servicios, proporcionando de esta forma un mapa de la entidad que facilita la dimensión ante cualquier cambio o evolución tecnológica y empresarial.

El enfoque de la Arquitectura Empresarial de Tecnología se enmarca en la perspectiva tecnológica que representa el hardware y las telecomunicaciones de una organización. Dentro de esta se encuentra: hardware de servidor y escritorio; sistemas operativos; componentes; telefonía; almacenamiento; impresoras; y, canales de comunicación, entre otros. **[82]**

La Arquitectura Empresarial de Software define las aplicaciones de la organización. Normalmente se incluye: automatización de los procesos empresariales; integración entre las diferentes aplicaciones y soluciones de la organización; estándares y políticas en lenguajes de programación, componentes e interfaces, entre otros. Debe proporcionar altos estándares de seguridad, disponibilidad y confiabilidad.**[82]**

2.3 Requerimientos de Software

- **Requerimientos de Software para el Servidor Web:**

  Sistema Operativo: Linux (Ubuntu 7.10).

  Servidor de Aplicaciones Web: Apache 2, para aplicaciones Java (Tomcat 5.5.17)

- **Requerimientos de Software para el Servidor de Bases de Datos:**

  Sistema Operativo: Linux (Ubuntu 7.10)

  Servidor de Bases de Datos: PostgreeSQL 8.2.

- **Requerimiento en los clientes:**

  Sistema Operativo: Cualquier sistema operativo + Navegador Web que soporte ejecutar scripts del lado del cliente como java script.

- **Requerimientos de software para las PC que contendrán aplicaciones de escritorio parte del sistema.**

  Sistema Operativo: Cualquier sistema operativo + Plataforma Java2.

2.4 Requerimientos de Hardware

- **Requerimientos de Hardware para el Servidor Web:**

  Velocidad del Procesador 3 GHz

Memoria RAM: 1 GB.

- **Requerimientos de Software para el Servidor de Bases de Datos:**

  Velocidad del Procesador 3 GHz

  Memoria RAM: 1 GB.

- **Requerimiento en los clientes:**

  Velocidad del procesador: 1 GHz

  Memoria RAM: 256 MB

En el caso de los clientes, estos son los requisitos que se plantean pues en las computadoras modernas estos son los requerimientos mínimos, un cliente en Internet con una PC que contenga un Navegador podrá acceder a las aplicaciones.

- **Requerimientos de software para las PC que contendrán aplicaciones de escritorio parte del sistema.**

  Velocidad del procesador: 1 GHz

  Memoria RAM: 256 MB

En caso de estas PC es importante señalar que la memoria RAM con 256 es el requerimiento mínimo donde quizás con la plataforma Java no se logre el rendimiento óptimo, sería ideal 512 MB de RAM, pero con esta descripción se puede perfectamente trabajar logrando un rendimiento aceptable.

2.5 Requerimientos de hardware para desarrollo

1. Cantidad de máquinas: 25.
2. Cantidad de Servidores: 1 PC que se usará como servidor del proyecto.
3. Velocidad del procesador: 2 GHz
4. Memoria RAM: 1 GB

2.6 Atributos de Calidad

La calidad de software se define como el grado en el cual el software posee una combinación deseada de atributos. Donde el software debe satisfacer los requerimientos no funcionales.

Los atributos de calidad se definen como las propiedades de un servicio que presta el sistema a sus usuarios. Pueden ser:

- **Observables vía ejecución:** Son aquellos atributos que se determinan el comportamiento del sistema en tiempo de ejecución.

- **No observables vía ejecución:** Son aquellos atributos que se establecen en el desarrollo del sistema. **[83]**

2.6.1 Disponibilidad

Es la medida de disponibilidad del sistema para su uso.**[83]**

Para lograr en el PNICG una disponibilidad continúa de datos y aplicaciones en toda una amplia gama de sistemas operativos, componentes de hardware y ubicaciones de centros de datos. Se puede crear, diseñar, implementar, optimizar y administrar una solución que sea compatible para alcanzar los niveles de disponibilidad requeridos por la empresa.

Desafíos y Soluciones.

Las aplicaciones deben ser accesibles, y los datos deben ser actuales, para obtener una alta disponibilidad de manera local o en toda una red de área amplia mundial. El uso de clústeres permite lograr la disponibilidad de las aplicaciones, al mismo tiempo que adquiere una mayor utilización del hardware.

Si bien las tecnologías de clústeres ayudan a garantizar la disponibilidad de la aplicación, y la aplicación es tan útil como los datos que ésta utiliza. Para estar preparado ante una interrupción en el sitio, las tecnologías de replicación de datos deben implementarse con tecnologías de clústeres a fin de proporcionar una solución integrada de recuperación ante desastres. De esta manera, se puede acceder tanto a la aplicación como a los datos desde un sitio secundario, sin limitaciones de distancia y con mínimas interrupciones.

2.6.2 Confidencialidad

Es la ausencia de acceso no autorizado a la información.**[83]**

En la Oficina Nacional de Recursos Minerales se trabaja con datos confidenciales los cuales deben mantenerse de forma segura y algunos de ellos transitan por la Red Nacional desde un extremo del país al otro. Este proceso requiere de un modelo de seguridad único, proporcionando una seguridad robusta para la autenticación y los modelos de autorización.
Para mitigar eficazmente la amenaza de que un intermediario que no es de confianza pueda manipular estos mensajes, las aplicaciones pueden proteger el mensaje en el remitente para que se detecte fácilmente cualquier intento de modificación. En este caso, dependiendo de la confidencialidad del contenido, el cifrado puede ser necesario.**[84]**

Las aplicaciones podrían requerir el seguimiento de cada mensaje mediante la autenticación en el nivel de mensaje. En estos tipos de aplicaciones, el rendimiento se puede sacrificar a cambio de un esquema de seguridad más fuerte.

Un mecanismo de autenticación basado en contraseña ayudará a proteger la capa de comunicación y al mismo tiempo continúa permitiendo la autenticación de mensajes.

La seguridad basada en contraseña es la configuración predeterminada más segura pues solo tienen acceso a la información aquellas personas que requieran del uso de la misma para desarrollar su trabajo.**[84]**

La información o datos nunca fluyen en un vínculo seguro (desde la perspectiva de la aplicación). Internamente, cada vínculo (un canal de transporte entre dos pares) se protege utilizando la seguridad de la capa de transporte (SSL). Esto significa que cuando un remitente crea y envía un mensaje, se envía sobre transporte seguro a cada uno de sus pares inmediatos, que tienen acceso al mensaje y, a su vez, envían el mensaje a sus pares inmediatos a través de conexiones seguras. Esta seguridad sólo funciona en el nivel de transporte y es independiente de los modelos de seguridad del mensaje.**[84]**

Para lograr la confidencialidad, las aplicaciones pueden emplear la seguridad de transporte mediante esquemas fuertes de pertenencia a grupos para evitar el acceso no autorizado a los mensajes.

2.6.3 Desempeño

Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. El desempeño de un sistema se refiere a aspectos temporales del comportamiento del mismo.

Se refiere a capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.**[85]**

La performance de una aplicación, es su desempeño en la plataforma real y es también, la capacidad que se tiene de lograr satisfacción. Para lograr esto en el PNICG solo es posible si se tiene en cuenta aspectos de correctitud (no existencia de errores), mantenibilidad (capacidad de resolución) y eficiencia (consumo de recursos), desde el comienzo de las etapas del desarrollo hay que integrar algunas prácticas específicas que incluyen poderosas herramientas que ayudan en el diseño y la documentación, integrando poderosos componentes para medir y auditar el código.

2.6.4 Confiabilidad

Es la habilidad de un sistema o equipo de mantenerse operativo con una probabilidad de operar bajo las condiciones preestablecidas sin sufrir fallas a lo largo del tiempo.**[85]**

En la Oficina Nacional de Recursos Minerales estas medidas de confiabilidad suelen utilizarse para calcular la probabilidad de que se produzca un error en un único componente de la solución. Una medida empleada para definir la confiabilidad de un componente o del sistema es el tiempo medio entre errores.

Los modos de falla son causados por:

- Desgaste y deterioro.
- Errores humanos en la ejecución de las tareas de Mantenimiento, y/o en la operación del equipo.
- Problemas de diseño.**[86]**

Los aspectos de confiabilidad señalados en esta metodología van dirigidos a centrar al alcance del proyecto de una manera más eficiente ya que permite identificar alcances innecesarios que

llevaría a identificar áreas incompletas del diseño conceptual y un mejor entendimiento de los requerimientos de mantenimiento de las instalaciones (mantenimiento en diseño) lo que augura en el futuro en una reducción de costes de mantenimiento, una mejor aplicación de las actividades de mantenimiento donde sea requerido y para la reducción de la tasa de fallos.

Un sistema es más confiable si es tolerante a errores. La tolerancia a errores es la capacidad de un sistema para seguir funcionando cuando se produce un error en parte del sistema. Para conseguir tolerancia a errores hay que diseñar el sistema con un alto grado de redundancia de hardware. Si se produce un error en un único componente, el componente redundante asumirá su función sin que se produzca un tiempo de inactividad apreciable. **[86]**

2.6.5 Seguridad Externa

Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.**[83]**

Se refiere a la protección que en toda institución se requiere establecer, previniendo posibles ataques desde el exterior, esta delimitada entre otras cosas por: marcas visibles que señalan impedimentos al libre acceso de personas y vehículos que lleguen del exterior y que por algún motivo pretenden entrar a la institución. Contempla toda el área circunvecina a la institución como zona de circulación prohibida o restringida. Para cumplir con esta indicación se requiere la implementación de vigilancia permanente. Además de la prevención contra incendios y desastres naturales.

Para ello se desea en el PNICG coordinar el establecimiento de prácticas, controles de implantación de seguridad, protección de la información, activos de tecnología de información, detectar la incidencia de eventos de acceso no autorizado, autenticación, confidencialidad, negación del servicio, integridad de la información, y no repudio de operaciones. Al mismo tiempo supervisar el monitoreo constante de las herramientas tecnológicas de seguridad y del inventario de activos informáticos.

Un previo control de la interacción con sistemas de información externos y los procesos de diagnóstico de vulnerabilidades tecnológicas en las aplicaciones.

2.6.6 Seguridad Interna

Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.**[85]**

El PNICG se basa principalmente en la gestión y administración de toda la información geológica del país, información en ocasiones sensible en cuestiones políticas o económicas. Debido a esto es necesario definir políticas de seguridad para mantener el control estricto sobre esta información, garantizando que este disponible solo para aquellas personas que tengan acceso a la misma.

La información esta dividida en dos grandes grupos, información contenida en bases de datos, la cual mediante el gestor de Base de Datos, se definen los niveles de seguridad para el acceso y modificación de la misma. Solo debe modificar aquella persona catalogada como administrador, el mismo debe tener un usuario y contraseña de acceso único.

Estos niveles deben ser definidos en cada aplicación, desglosando detalladamente que información esta restringida y cual es pública. Por otro lado se encuentra la información que se obtiene mediante otros servicios (FTP, HTTP, EMAIL), donde se deben establecer mecanismos para su control y seguridad, cada módulo o aplicación igualmente debe clasificar la información pública de la restringida y la transferencia de datos se hará con el protocolo HTTPS, encriptando los datos para así hacerlos más seguros.

Con respecto a las aplicaciones que requieran de una política de acceso, se definirán los niveles de acceso de manera que garantice el mayor nivel de seguridad posible.

2.6.7 Integrabilidad

Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. Además de la ausencia de alteraciones en la información.**[88]**

Debido a que todas las aplicaciones que se desarrollan en el PNICG se realizan sobre la misma base o plataforma, y con la misma estructura de datos, todos los objetos que se creen pueden ser utilizados en cualquier aplicación. Esta es la capacidad de escalabilidad e integración. Se pueden

crear aplicaciones independientes y luego hacer que interactúen de manera sencilla con menos esfuerzo de desarrollo.

Mecanismos de Control

- Autenticación. Identificar correctamente los clientes de la aplicación, entre los que pueden figurar usuarios finales, servicios, procesos o equipos. Autorización. Definir lo que pueden ver y hacer en la aplicación los clientes autenticados.
- Comunicaciones seguras. Garantizar la privacidad y la integridad de los mensajes cuando pasan de una red a otra.
- Suplantación. Se trata de la técnica utilizada por una aplicación de servidor para obtener acceso a recursos en nombre de un cliente. El contexto de seguridad del cliente se utiliza para los controles de acceso realizados por el servidor.
- Delegación. Una forma ampliada de suplantación que permite a un proceso de servidor que trabaja en nombre de un cliente el acceso a recursos de un equipo remoto.
- Contexto de seguridad. Es un término genérico usado para referirse a la colección de configuraciones de seguridad que afectan al comportamiento relativo a la seguridad de un proceso o subproceso. Los atributos de la sesión de inicio y el testigo de acceso de un proceso conforman el contexto de seguridad del proceso.
- Identidad. Hace referencia a una característica exclusiva de un usuario o servicio que lo identifica. Suele tratarse, por ejemplo, de un nombre de visualización, que a menudo adopta el formato autoridad/nombre de usuario.

Principios básicos

- Adoptar el principio de privilegios mínimos.
- Usar las defensas a fondo.
- No confiar en los datos introducidos por el usuario.
- Utilizar opciones predeterminadas seguras.
- No depender de la seguridad por medio de la oscuridad.
- Realizar controles en la puerta.
- Asumir que los sistemas externos no son seguros.
- Reducir el área expuesta.

- Cometer errores de forma segura.
- No olvidar que el alcance de la seguridad lo define su punto más débil.
- Si no lo utiliza, deshabilítelo.

2.6.8 Interoperabilidad

Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de Integrabilidad**.[88]**

El PNICG se debe enfrentar al reto de tener que hacer funcionar conjuntamente una gran variedad de aplicaciones desarrolladas en varios lenguajes de programación o el poder integrar tecnologías heterogéneas para que operen conjuntamente.

Los distintos sistemas deben comunicarse entre sí e intercambiar datos unos con otros. La interoperabilidad es el intento de hacer compatibles todos los sistemas a nivel de código fuente, centrándose exclusivamente en añadir nuevas capas de middleware que intentan darle a todos los sistemas un aspecto y unas funciones similares, o buscar la forma de hacer que todos los sistemas sean intercambiables.

En la Oficina Nacional de Recursos Minerales la interoperabilidad, es una pieza importante pues todas las aplicaciones deben conectarse e intercambiar datos y así es como se puede resolver la diversidad y heterogeneidad que ofrece el código.

El XML (eXtensible Markup Language), hace posible que el software comparta información de manera eficiente, y abre la puerta a un mayor grado de "interoperabilidad por diseño" entre distintos tipos de software. Nuestro objetivo es aprovechar al máximo el software y permitir a todas las aplicaciones trabajar conjuntamente.

2.6.9 Modificabilidad

Es la habilidad de realizar cambios futuros al sistema.**[87]**
El código ha de ser fácil de mantener, o sea, ha de permitir que se realicen cambios para adaptarlos a nuevas situaciones.

En la Oficina Nacional de Recursos Minerales muchos datos están almacenados de forma estática pues no varían, pero algunos minerales como el petróleo están en continúo movimiento, debido a algunos descubrimientos que se hacen, y esta información hay que actualizarla y modificarla, esto revela que las aplicaciones deben estar implementadas de una forma que se pueda efectuar una modificación en cualquier momento debido a algún cambio.

2.6.10 Mantenibilidad

Capacidad de modificar el sistema de manera rápida y a bajo costo.**[87]**

La Mantenibilidad está inversamente relacionada con la duración y el esfuerzo requerido por las actividades de mantenimiento. Puede ser asociada de manera inversa con el tiempo que se toma en lograr las acciones de mantenimiento, en relación con la obtención del comportamiento deseable del sistema.

Hay que hacer las aplicaciones extremadamente fiables y consecuentemente costosas. Además de que cuando falle sea fácil de poner en funcionamiento de nuevo. Si todo es así y construido muy fiable y fácil de reparar, la información estará más segura. Así mismo sí surge cualquier problema debe ser rápidamente resuelto, este es el principal logro de un alto nivel de disponibilidad operativa de una aplicación. La disponibilidad del equipo tanto como la seguridad, es de mucha importancia, porque no se puede tolerar tener un equipo fuera de servicio.

Todo esto es muy necesario en la Oficina Nacional de Recursos Minerales pues una sustitución innecesaria cuesta prácticamente igual que un fallo real, cuando el componente investigado es desmontado y reemplazado. La disminución de estas situaciones sería un gran reductor de costo. En general cuanto más simple es el diseño, más favorable es la mantenibilidad.

2.6.11 Portabilidad

Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.**[85]**

La Oficina Nacional de Recursos Minerales necesita que las aplicaciones sean portátiles, que puedan ser utilizadas en cualquier ordenador que posea el sistema operativo para el que fue programada sin instalación previa; esto significa que no es necesaria la instalación de bibliotecas adicionales en el sistema para su funcionamiento. En un inicio necesita que las aplicaciones

clientes funcionen en varias plataformas (Windows, Linux). Partiendo de esta idea, las aplicaciones que se implementen, deben garantizar su funcionamiento sobre las dos plataformas antes mencionadas.

Para que las aplicaciones del PNICG sean portátiles, los desarrolladores deben conseguir que la aplicación deje el ordenador donde se ha ejecutado completamente "limpio". Esto implica que la aplicación no debe usar el registro, ni guardar ficheros en ningún lugar que no sea su directorio de instalación.

2.6.12 Escalabilidad

Es el grado con el que se pueda ampliar el diseño arquitectónico, de datos o procedimientos. También es la propiedad deseable de un sistema, una red , que indica su habilidad para manejar el crecimiento continuo de trabajo de manera fluida o para estar preparado a hacerse más grande sin perder calidad en los servicios ofrecidos.**[91]**

Se desea que las aplicaciones del PNICG sean escalables de tal forma que al aumentar la carga del sistema se le pueda añadir servidores o ampliar los existentes sin que sea necesario realizar modificaciones.

Se podría definir como la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.**[92]**

2.7 Seguridad

Acegi

Acegi es un gestor de seguridad que está diseñado fundamentalmente para ser usado con Spring y en el que destaca su versatilidad. Acegi proporciona una capa que envuelve diversos estándares de seguridad presentes en Java y ofrece una forma unificada de configuración a través de un descriptor en XML. Acegi es una herramienta bajo la licencia de Apache que permite incorporar en nuestros desarrollos Spring seguridad de una manera simple, práctica y confiable.

Cubre la capa web y la de negocio. A nivel Web captura todas las peticiones mediante la implementación de un filtro y a nivel de métodos mediante interceptación a través de AOP. En

ambos casos permite aplicar los criterios de seguridad que trae de serie o añadir nuevas opciones de forma sencilla implementando los interfaces diseñados a tal fin.

Acegi se ha elegido como opción frente a los sistemas propietarios de los diferentes vendedores por su universalidad de uso, además engloba los API's de seguridad de Java. Todos los problemas se resuelven con la implantación correcta de políticas y prácticas de seguridad.

Acegi es un framework de seguridad, es open source y permite mantener la lógica de negocio libre de código de seguridad. ACEGI proporciona cuatro opciones principales de seguridad:

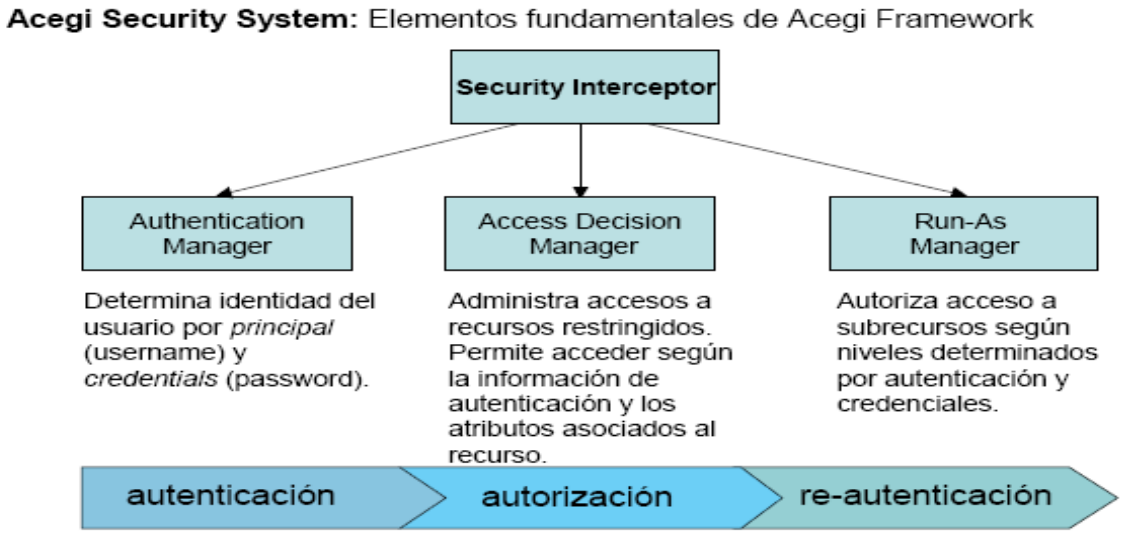
1. Listas de control de acceso (ACL) web basadas en esquemas URL.
2. Protección de métodos y clases Java usando AOP.
3. Single sign-on (SSO).
4. Seguridad proporcionada por el contenedor Web.

Acegi Security System para Spring proporciona la funcionalidad necesaria para adoptar mecanismos de seguridad en aplicaciones Java utilizando características de programación orientado a aspectos, de forma trasparente para el desarrollador, sin necesidad de desarrollar código, utilizando para ello el soporte prestado por el framework Spring.

Spring brinda un módulo especial para poder hacer esta integración y poder llevar a cabo una mejora en la cuestión de seguridad del sistema utilizando las funcionalidades que brinda Acegi, como la autenticación de usuarios, seguridad de peticiones web, seguridad en la capa de servicios, entre otros. Todo esto con el propósito de que la seguridad del sistema este respaldada, no solo por técnicas tradicionales y validaciones como lo está actualmente, sino por un framework que garantice y brinde las características necesarias para no comprometer la seguridad del sistema.

Acegi Security System:

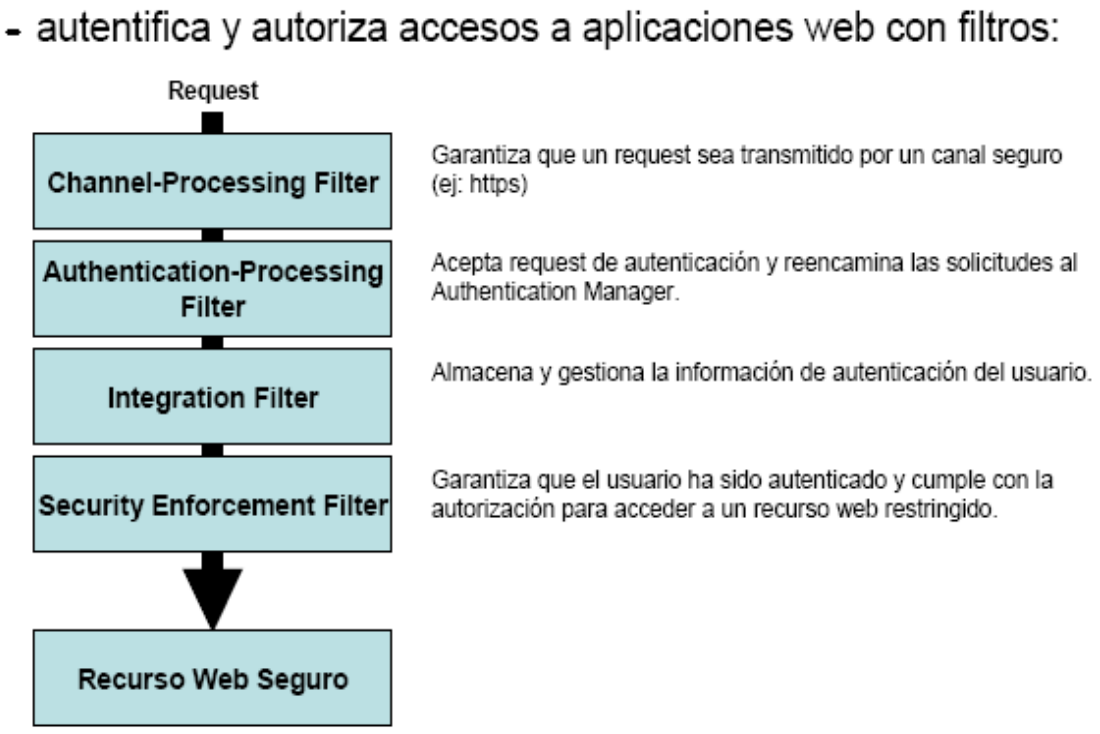
- Seguridad declarativa para aplicaciones basadas en Spring Framework.
- Sub proyecto de Spring Framework.
- Provee una colección de beans configurables en el contexto Spring.
- Basado en inyecciones de dependencia y AOP.
- Autentifica y autoriza accesos a aplicaciones web con filtros.



Para ver la descripción de cada uno de estos elementos ver los anexos 8, 9 10.

1- Autentifica y autoriza accesos a aplicaciones web con filtros:

- Filtros interceptan request entrantes y aplican procesos de seguridad antes de ser evaluados por la aplicación.
- Acegi provee un conjunto de filtros específicos a diferentes etapas del proceso request.
- Interceptan las solicitudes y reencaminan el flujo de la aplicación a los correspondientes *Authentication* y *Access Decision Managers*.



Conclusiones

En este capítulo 2 se ha definido la arquitectura de software para la propuesta arquitectónica del Programa Nacional de Informatización del Conocimiento Geológico abordando aspectos muy importantes por los que se rige la Arquitectura Empresarial al igual que sus beneficios. Esta solución esta basada en un profundo análisis de las Plataformas de Desarrollo, Sistema Operativo, Lenguajes de Programación, Tecnologías que serán utilizadas, además de las herramientas estipuladas que hacen que la arquitectura de software sea robusta. Conjuntamente se plasmaron los Requisitos No Funcionales del PNICG, con los cuales no sería posible llevar a cabo el desarrollo del Programa.

Conclusiones

- Se cumplieron todos los objetivos trazados en esta investigación y a la vez se cumplió con el problema científico a partir de definir una propuesta arquitectónica para el PNICG.

- Se propone el uso de la arquitectura empresarial para alcanzar la eficiencia tecnológica deseada e innovación en el negocio.

- El desarrollo de la arquitectura de software es una de las etapas fundamentales en el desarrollo de software, pues es aquí donde los profesionales aportan todos sus conocimientos, creatividad y experiencia para crear la mejor propuesta de solución que se dará al cliente que cumpla con los requerimientos funcionales y no funcionales establecidos para el sistema en desarrollo, así como sus preocupaciones principales de lo que esperan del sistema.

- Es de vital importancia que todo sistema de software esté respaldado por una arquitectura sólida que facilite el entendimiento del mismo, que sea capaz de organizar el desarrollo, fomentar la reutilización y hacer evolucionar el sistema.

- Se realizó un estudio profundo acerca de varios elementos arquitectónicos que existen en la actualidad, para hacer uso de las mejores técnicas de diseño arquitectónico.

- Se desarrolló una arquitectura de tres capas que facilitó el desarrollo paralelo del sistema, el mantenimiento y soporte de la aplicación al tratarse cada capa de forma individual.

- Se definieron componentes y funcionalidades reusables que fueron empleados en cada uno de los módulos permitiendo acortar el tiempo de desarrollo de la aplicación.

Recomendaciones

Con el objetivo de mejorar y ampliar la documentación que se presenta en este trabajo se recomienda.

- Efectuar un continuo refinamiento de la arquitectura de software.
- Aumentar el número de arquitectos a la hora de desarrollar otro proyecto, logrando así una mejor división del trabajo, la cual ayude a realizar un trabajo más rápido, profundo y abarcador sobre la arquitectura de un sistema.
- Valorar la posibilidad de hacer extensiva esta propuesta arquitectónica a otros proyectos.

Bibliografía

1. Arquitecturas de Aplicaciones en N-capas con .NET. Noviembre 2002, Madrid.

2. Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschman, PATTERN-ORIENTED SOFTWARE ARCHITECTURE. Volumen 1.

3. Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschman, PATTERN-ORIENTED SOFTWARE ARCHITECTURE. Volumen 2.

4. Carlos E. Cuesta. Arquitecturas de Software. Ingeniería del Software I, Universidad Rey Juan Carlos. Society, S. E. S. C. o. t. I. C. (2000). "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems."

5. Bass, L., Clements, P., y Kazman, R. (1998) Software Architecture in Practice. Addison Wesley.

6. Bosch, J. (2000). Design & Use of Software Architectures". Addison Wesley.

7. Brown, A. W. y. W., K. C. (1998) The Current State of CBSE. IEEE Software

8. Carlos Reynoso, N. K. (2004) Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.

9. Clements, P. (1996). A Survey of Architecture Description Languages. Proceedings of the International Workshop on Software Specification and Design.

10. Dewayne E. Perry, A. L. W. (1992). "Foundations for the study of software architecture."

11. Gamma, E. (2002). Patrones de Diseño, Addison-Wesley.

12. Jacobson, I. (1999). The Unified Software Development Process. Addison Wesley.

13. Kruchten, P. (1995). The 4+1 View Model of Architecture. IEEE Software.

14. Len Bass, P. C., Rick Kazman (2003). Software Architecture in Practice (2nd Ed.), Addison Wesley.

15. Mary Shaw, D. G. (1996). Software Architecture. Perspectives on an Emerging Discipline, Prentice Hall.

16. Robert Monroe, A. K., Ralph Melton y David Garlan (1997) Architectural Styles, design patterns, and objects. IEEE Software.

17. Rumbaugh, J. (1991). Object-Oriented Modeling and Design, Prentice Hall.

18. Rumbaugh, J. (1999). The Unified Modeling Language Reference Manual, Addison Wesley. SEI (1990) Workshop on Domain-Specific Software Architectures. Software Engineering Institute.

19. Ingeniería del Software, Un enfoque práctico. Sexta Edición. Roger S Presuman. 2005.

Referencias Bibliográficas

1. Shaw, D.G.y.M., *An introduction to software architecture*. 2000.

2. Zapata Sánchez, A.F., *Arquitectura de Software*.

3. Touris, A.M., *Arquitectura empresarial y software libre, J2EE*. 2002.

4. Ross, J.w., *Enterprise architecture as strategy*. 2006.

5. Kruchten, P., *Architectural Blueprints: The "4+1" View Model of Software Architecture*. 1995.

6. F. Buschman, R.M., H. Rohnert, P. Sommerlad y M. Stal. *Pattern- Oriented Software Architecture: Patterns for Concurrent and Networked Objects*. Vol. 2. 2000.

7. Garlan, R.A.y.D., *A formal approach to software architectures*. 1992.

8. Doberkat, E.-E., *Pipes and filters: Modelling software architecture through relations*. 2002.

9. Garlan, M.S.y.D., *Software Architecture: Perspectives on an emerging discipline*. 1996.

10. Martin, R.C., *Design Principles and Design Patterns*.

11. *Microsoft Patterns & Practices*. 2004.

12. Shen, D.Y. *Integración de JSF, Spring e Hibernate para crear una Aplicación Web del Mundo Real*. [Cited; Available from: http://www.programacion.net/tutorial/jap_jsfwork/3/.

13. Fowler, M., *Patterns of Enterprise Application Architecture*.

14. *Arquitectura Empresarial*. 2005 [cited; Available from: http://www.cutter.com.mx/easummit/conferencias.html.

15. *Arquitectura Empresarial*. 2006 [cited; Available from: http://www.tuobra.unam.mx/publicadas/040702105342-Arquitec.html.

16. Soto, M.A. *Protocolo TCP/IP*. [Cited; Available from: http://usuarios.lycos.es/janjo/janjo1.html.

17. *Servicio de transferencia de ficheros FTP*. 2005 [cited; Available from: http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/lpractico/node132.html.

18. *Red Hat Enterprise Linux 4: Manual de referencia*. [Cited; Available from: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ftp.html.

19. Juszkiewicz, L. *Manual de Ftp*. [Cited; Available from: http://www.monografias.com/trabajos29/manual-ftp/manual-ftp.shtml.

20. Daccach, J.C. *HTTP (Hypertext transfer protocol)*. [Cited; Available from: http://www.gestiopolis.com/delta/term/TER187.html.

21. *El protocolo HTTPS*. [Cited; Available from: http://www.programacionweb.net/articulos/articulo/?num=411.

22. Abed, M.H., *APLICACIÓN DE LOS ALGORITMOS DE SEGURIDAD EN UN SISTEMA*

23. *Protocolo SSL*. [Cited; Available from: http://www.pedroximenez.com/ssl.htm.

24. Morales, J.H.G., *Web Services*. 2004.

25. Castejón Garrido, J.S., *Arquitectura y diseño de sistemas web moderno*.

26. Gaytán, I., *Introducción a Web Services con .NET y FLEX*. 2007.

27. *Servidor de aplicaciones*. 2004 [Cited; Available from: http://es.desarrolloweb.org/Servidor_de_aplicaciones.

28. *JBoss*. 2005 [cited; Available from: http://www.osmosislatina.com/jboss/basico.htm.

29. *JBoss vs. Glassfish*. [Cited; Available from: http://www.javamexico.com/blogs/laliux/jboss_vs_glassfish.

30. *Servidor Web*. [Cited; Available from: http://es.desarrolloweb.org/Servidor_web.

31. *Apache*. [Cited; Available from: http://es.wikipedia.org/wiki/Servidor_HTTP_Apache.

32. *Servidor HTTP Cherokee*. [Cited; Available from: http://es.wikipedia.org/wiki/Servidor_HTTP_Cherokee.

33. *Las Redes*. [Cited; Available from: http://www.monografias.com/trabajos15/redes-clasif/redes-clasif.shtml.

34. *Aplicación Empresarial*. 2006 [Cited; Available from: http://www.aplicacionesempresariales.com/empresa/secondlife-como-aplicacion-

35. *Introducción al desarrollo de software para LAMP*. [Cited; Available from: http://cafedechucho.blogspot.com/2007/07/introduccin-al-desarrollo-de-software.html.

36. *LAMP*. [Cited; Available from: http://www.deltaasesores.com/prof/PRO152.html.

37. Johnson, M.K. *Características principales de Linux* 2006 [cited; Available from: http://www.wikilearning.com/tutorial/caracteristicas_principales_de_linux/20536.

38. *C++*. [Cited; Available from: http://arco.inf-cr.uclm.es/~david.villa/pensar_en_C++/products/vol1/pr02s05.html.

39. *Introducción a C++*. [Cited; Available from: https://belenus.unirioja.es/~creguia/introduccion.html.

40. Rod Johnson, J.H., Alef Arendsen, Thomas Risberg, Colin Sampaleanu *Professional Java Development with the Spring Framework*. 2005.

41. *Características de Java*. [Cited; Available from: http://www.manual-java.com/manualjava/caracteristicas-java.html.

42. Pey, X. *características de ASP*. [Cited; Available from: http://www.netveloper.com/contenido2.aspx?IDC=41_0&IDP=3&P=39.

43. Stig Sæther Bakken, A.A., Egon Schmid y Jim Winstead, *Manual de PHP*. 2001.

44. Jesus Castagnetto, H.R.y.S.S., *Professional PHP Programming*. 1999.

45. *Comparación de J2EE y PHP*. [Cited; Available from: http://sentidoweb.com/2007/07/24/comparativa-entre-j2ee-php.php.

46. Brea, O.F. *Trabajando con NUSOAP*. 2005 [Cited; Available from: http://www.desarrolloweb.com/articulos/1884.php.

47. Romero, A.R., *Introducción al XML en Castellano*. 2000.

48. VV.AA. y FAWCETT, J.y.M., JEREMY, *AJAX*. 2006.

49. *Artículo de SOAP*. [Cited; Available from: http://www.desarrolloweb.com/articulos/1557.php.

50. Guervos, J.J.M. *Programando con JSPs* 2004 [Cited; Available from: http://geneura.ugr.es/~jmerelo/JSP/.

51. *Qué es un Sistema Gestor de Base de Datos*. [Cited; Available from: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

52. *Qué es Oracle*. 2007 [cited; Available from: http://www.desarrolloweb.com/articulos/840.php.

53. Lockhart, T., *The PostgreeSQL Tutorial Introduction*. 1998.

54. Stinson, B., *PostgreeSQL Essential Reference*. 2001. 400.

55. Potencier, F.Z.y.F. *Symfony, la guía definitiva* [cited; Available from: http://www.librosweb.es/symfony/.

56. Leopoldo, C. *CakePHP*. 2007 [cited; Available from: http://www.carlosleopoldo.com/post/cakephp/.

57. *Eclipse*. [Cited; Available from: http://www.eclipse.org/.

58. *NetBeans*. 2005 [cited; Available from: http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html.

59. *Visual Paradigm*. 2007 [cited; Available from: http://www.visual-paradigm.com/product/vpuml/.

60. *Rational Rose*. 2005 [cited; Available from: http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci516025,00.html.

61. Ben Collins-Sussman, B.W.F.a.C.M.P. *Version Control with Subversion:*. 2003 [Cited; Available from: http://svnbook.red-bean.com/en/1.0/svn-book.pdf.

62. Molino, G.D. *DotProject, gestor de proyectos para la empresa.* 2007 [cited; Available from: http://camyna.com/2007/12/12/dotproject-gestor-de-proyectos-para-la-empresa-libre/.

63. *Planner*. 2007 [cited; Available from: http://chicheblog.blogspot.com/2007/02/planner-el-planificador-para-gnome.html.

64. *Genie Backup Manager* 2007 [cited; Available from: http://www.genie-soft.com/products/gbm/default.html.

65. *Genie Backup Manager* 2004 [cited; Available from: http://genie-backup-manager.softonic.com/.

66. Lee Ackerman, L.O.y.B.P., *Building SOA Solutions Using the Rational SDP*. 2007. 636.

67. Jacobson, I., Grady Booch, and James Rumbaugh., *El Proceso Unificado de Desarrollo de Software*. 1999.

68. R. Johnson, J.H., A. Arendsen. *"Spring. Java/J2EE Application Framework"*. 2004-2005 [cited; 290]. Available from: http://www.springframework.org/documentation.

69. Deepak Alur, J.C., and Dan Malks, *Core J2EE Patterns*.

70. *Características de los Sistemas Linux* 2005 [Cited; Available from: http://xml.cie.unam.mx/xml/Linux/glinux-2.html.

71. *Manual de Java*. [Cited; Available from: http://www.desarrolloweb.com/manuales/57/.

72. *Eclipse*. 2005 [cited; Available from: http://unreal-hosting.net/lampyeclipse.pdf.

73. *PostgreeSQL*. 2004 [cited; Available from: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.

74. *Hibernate*. [Cited; Available from: http://mundogeek.net/archivos/2007/01/27/hibernate/.

75. Glögls, M. *Persistencia de Objetos Java: El Camino hacia Hibernate*. 2007 [cited; Available from: http://www.programacion.com/java/tutorial/hibernate/.

76. *Definición de Hibernate*. 2006 [cited; Available from: http://www.unife.edu.pe/ingenieria/desarrollo.doc.

77. Lambert, R. *"An Introduction to the Spring Framework"*. 2005 [cited; Available from: http://cjug.org/presentations/2005/June21/Spring-Framework-Intro-Rob-Lambert.ppt.

78. [cited; Available from: http://www.uv.es/sto/charlas/GPICSL/GPICSL.html.

79. Larman, C., *UML y Patrones*. 1999.

80. *Seguimiento de errores*. [Cited; Available from: http://producingoss.com/es/bug-tracker.html.

81. Gracia, J. *Seguimiento y gestión de errores, Bug Tracking*. 2004 [cited; Available from: http://www.ingenierosoftware.com/pruebas/gestionerrores.php.

82. *Arquitectura Empresarial*. 2007 [cited; Available from: http://www.acis.org.co/index.php?id=543.

83. Barbacci, M., Klein, M., Longstaff, T., & Weinstock, C. *Quality Attributes*. 1995 [cited; Available from: http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.021.html.

84. *Seguridad* [cited; Available from: http://msdn.microsoft.com/es-es/library/aa751919.aspx.

85. Kazman, R., Clements, P., Klein, M. *Evaluating Software Architectures. Methods and case studies. Addison Wesley*. 2001 [Cited; Available from: ftp://ftp.sei.cmu.edu/pub/sati/Papers_and_Abstracts/ISAW-2.ps.

86. *Descripción de la disponibilidad, la confiabilidad y la escalabilidad*. [Cited; Available from: http://technet.microsoft.com/es-es/library/aa996704(EXCHG.65).aspx.

87. Booch, G., Rumbaugh, J., & Jacobson, I. *The UML Modeling Language User Guide. Adisson-Wesley*. 1999 [cited.

88. Bass, L., Clements, P., & Kazman, R. *Software Architecture in practice. Addison-Wesley*. 1998 [cited.

89. *Crear software interoperable por diseño*. 2004 [cited; Available from: http://www.microsoft.com/spain/interop/02-03interoperability.mspx.

90. *Diseño Web Centrado en el Usuario: Reusabilidad y Arquitectura*. [Cited; Available from: http://www.dirinfo.unsl.edu.ar/~profeso/PagProy/articulos/cacic%202004_1.pdf.

91. Pressman, R.S., *Ingeniería del Software, Un enfoque práctico*. Sexta ed. 2002.

92. *Escalabilidad*. [Cited; Available from: http://es.wikipedia.org/wiki/Escalabilidad.

Anexos

Anexo 1: Resultados de los Métodos Teóricos utilizados para la investigación.

Entrevista realizada a personal específico miembro de cada proyecto, sobre mejores Lenguajes de Programación, Tecnologías más usadas, Sistemas Gestores de Base Datos, Framework, IDEs y Herramientas para la Propuesta de Arquitectura del Programa Nacional de Informatización del Conocimiento Geológico.

La entrevista tiene como objetivo principal conocer diferentes aspectos de la arquitectura de software empleada en el Programa Nacional de Informatización del Conocimiento Geológico antes de hacer la propuesta arquitectónica para dicho programa.

Nombre del Encuestado _____.
Cargo que ocupa en el Proyecto: _____.

1. ¿Cúal es la plataforma de desarrollo utilizada?._____.
2. ¿En qué lenguaje se están programando las aplicaciones?._____.
3. ¿Qué tecnologías se usan?._____.
4. ¿Cúal es la arquitectura usada para el desarrollo de las aplicaciones?._____.
5. ¿Mediante que SGBD se gestiona la información?._____.
6. ¿Qué framework se utiliza?._____.
7. ¿Qué Entorno de Desarrollo Integrado (IDE) se emplea?._____.
8. ¿Qué herramientas son las utilizadas para garantizar el Modelado?._____.
9. ¿Qué herramientas son las utilizadas para garantizar el control de versiones?._____.
10. ¿Qué herramientas son las utilizadas para garantizar la gestión de proyectos?._____.
11. ¿Qué herramientas son las utilizadas para garantizar la gestión documental y el control de errores?._____.

12. ¿Qué herramientas son las utilizadas para garantizar la gestión de Salvas Automáticas?._____.

Anexo 2:

Figura 1: Vistas de la Arquitectura Propuesta por RUP.

Anexo 3:

Figura 2: Estilo arquitectónico tuberías y filtros aplicado al servidor.

Anexo 4:

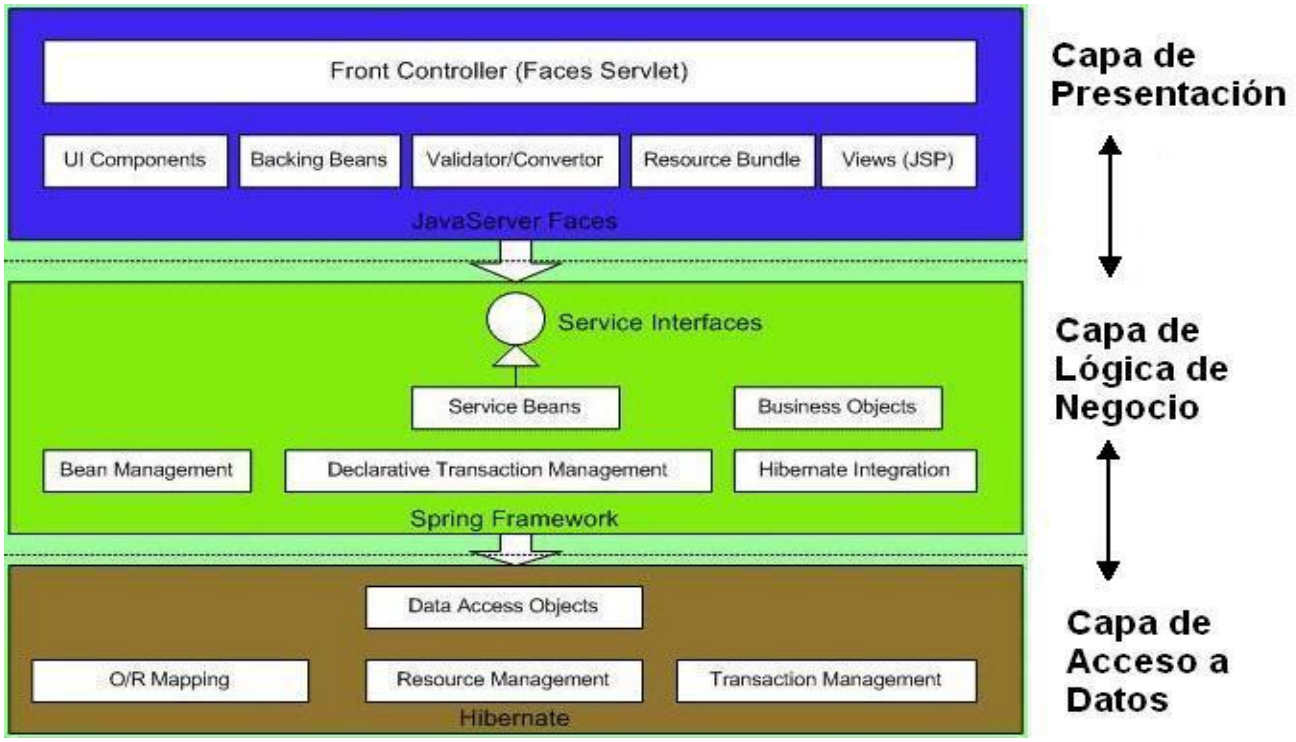


Figura 3: Distribución de la Arquitectura Multi-capa.

Anexo 5:

Características	Linux	Windows
Seguridad	1- Si alguien descubre un agujero de seguridad en cualquier versión de Linux, cualquier programador habilidoso lo puede subsanar puesto que es de código abierto. 2- La propia estructura lógica sobre la que funciona Linux es más segura que la de Windows.	3- Microsoft no le concede a la seguridad toda la importancia que se merece porque si hiciera sus Sistemas Operativos más robustos y fiables, serían más difíciles de manejar para los que no tienen idea de Informática. 4- Se descubren muchos más agujeros de seguridad en Windows; sobre todo en lo referente a la seguridad cuando se navega por Internet.
Memoria RAM	1- Aprovecha 100% de la memoria RAM utilizando la memoria virtual Swap mejorando la velocidad.	1-Windows le proporciona memoria física a todas las aplicaciones estén corriendo o no, haciendo el sistema muy lento.
Procesador	1- Lo utiliza al 100%	1- No lo hace
Multitarea	1- Si	1- No
Multiproceso	1- Si	1- No

Tabla 1: Comparación de los Sistemas Operativos Linux y Windows

Anexo 6:

Características	**Java**	**C++**
Sencillez	Sí	No
Robustez	Sí	No
Seguridad	Sí	No
Interpretado	Sí	No
Dinamicidad	Sí	No
Portabilidad	Sí	No
Neutralidad	Sí	No
Garbage Colection	Sí	No
Excepciones	Sí	Algunas
Representación	Alta	Alta

Tabla 2: Comparación de los lenguajes de programación Java y C++

Anexo 7:

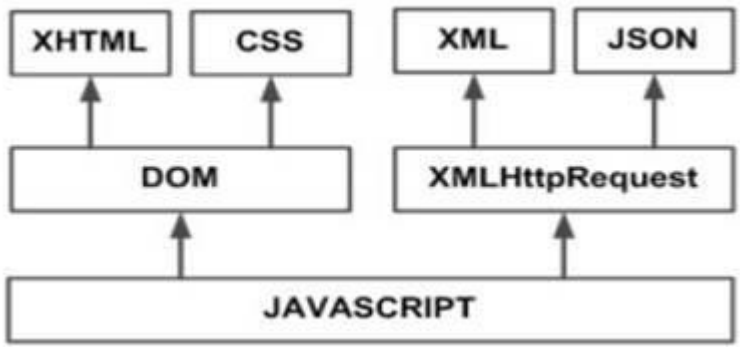


Figura 4: Tecnologías agrupadas bajo el concepto de AJAX.

Anexo 8:

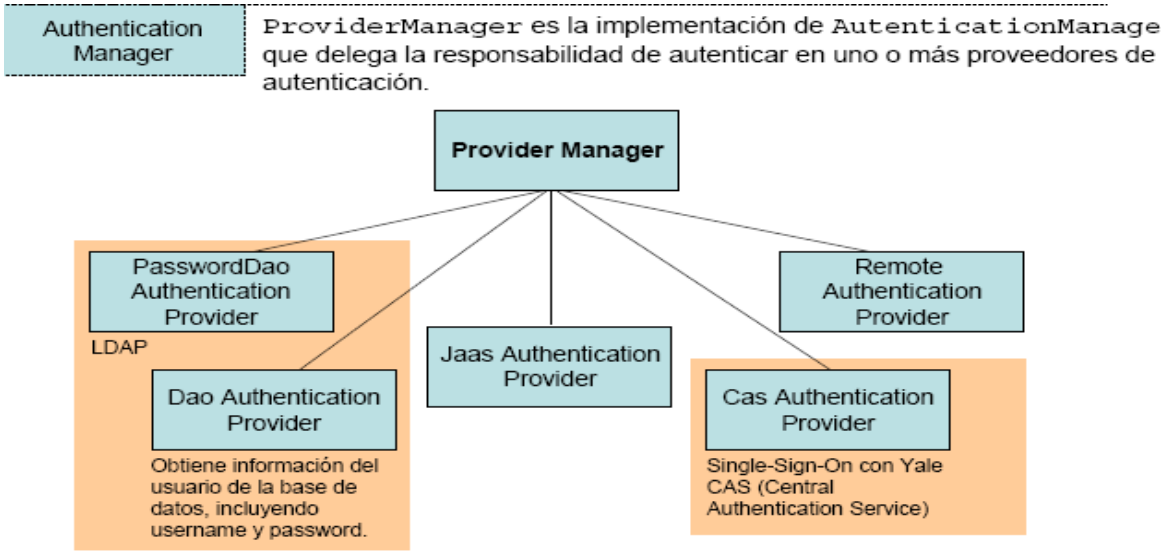


Figura 5: Descripción del elemento Authentication Manager

Anexo 9:

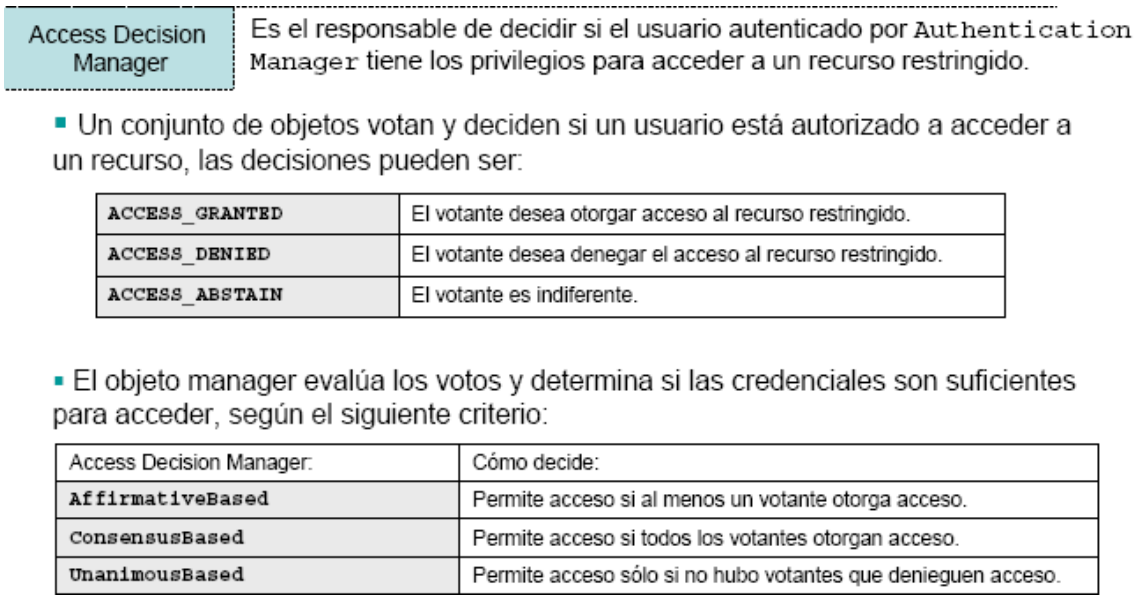


Figura 6: Descripción del elemento Access Decision Manager

Anexo 10:

Run-As Manager	Autoriza acceso a subrecursos según niveles determinados por autenticación y credenciales.

- `AbstractSecurityInterceptor` puede reemplazar de forma temporal el objeto de autenticación siempre que haya sido procesado exitosamente por el `AuthenticationManager` y por `AccessDecisionManager`.
- Permite a un usuario hacer llamadas a objetos que requieran diferentes credenciales de autenticación y autorización.
- Esta característica es particularmente útil para llamar Web Services remotos (según documentación de referencia del framework).

Figura 7: Descripción del elemento Run-As Manager

Opiniones y Avales

UCI Universidad de las Ciencias Informáticas

Se otorga el presente

RECONOCIMIENTO

A: Darais Santos Pérez

Por haber resultado

Mención

Con la Presentación del Trabajo:

"Propuesta de la arquitectura de software para el Proyecto Nacional de Informatización del CG"

En la VI edición de la

Jornada Científica Estudiantil

"Seamos realistas y hagamos lo imposible"

Decano Fac. 9

Producción FEU

Presidente FEU Fac. 9

Consejo FEU Facultad 9

Glosario de Términos

API: Application Programming Interface (Interfaz de Aplicación para Programación).

Capas: Es uno de los patrones de diseño más utilizado para cualquier tipo aplicaciones, en este, básicamente se divide los elementos de diseño en paquetes. Se describen las diversas capas del sistema y su contenido, detallan los límites entre las capas y las reglas preestablecidas para cada una.

FTP: Protocolo para la transferencia de archivos. Este protocolo es el designado para intercambiar archivos en Internet. HTML: Lenguaje de marcado de hipertexto, es el lenguaje autoritario para crear documentos en la World Wide Web. Define la estructura de un documento web usando etiquetas y atributos.

Hyper Text Transfer Protocol (HTTP): Protocolo de transferencia de hipertexto, es el protocolo usado en cada transacción de la Web. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido. También sirve el protocolo para enviar información adicional en ambos sentidos, como formularios con campos de texto.

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

Servicios Web: Un servicio Web (Web Service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, a través de la Web.

Servidor Web: Un servidor Web es un programa que implementa el protocolo HTTP (Hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML (Hypertext Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. Un servidor Web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita.

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

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

Wrapped: Término en inglés que significa envoltura y que en el contexto del texto denota la acción de enmascarar u ocultar una implementación.

WSDL: Web Services Description Language (Lenguaje de descripción de los Servicios Web.)