Universidad de las Ciencias Informáticas
FACULTAD 15

Título: “Subsistema de Gestión de Servidores y VLAN para la Producción en la Universidad de las Ciencias Informáticas”.

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas.

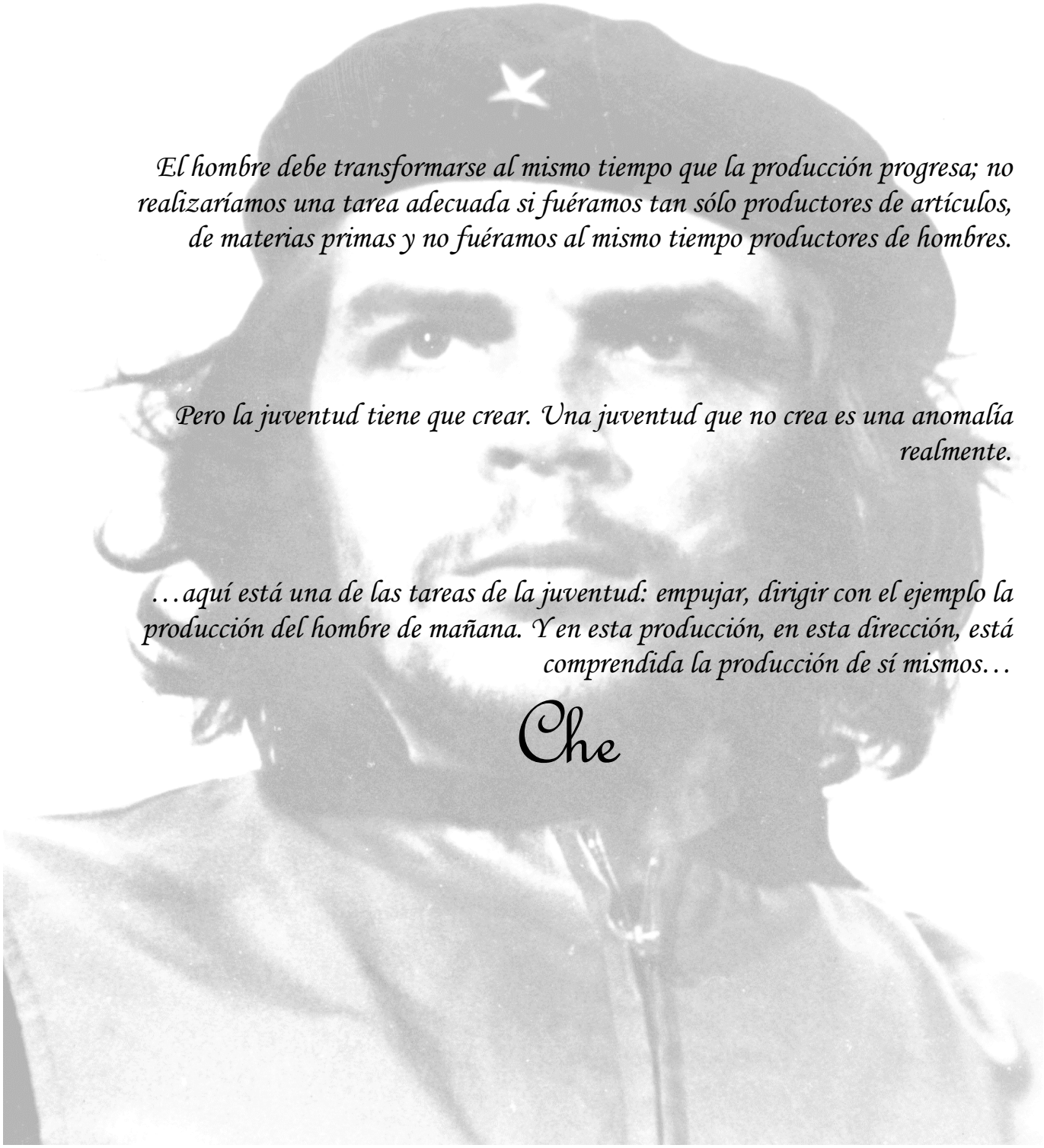
Autores: Omar Moreno Suárez

Juan Pérez Chirino

Tutores: Ing. Yulio Seriocha García Gallardo

Ing. Frank David Ávalos Palomo

Ciudad de La Habana, Junio 2010



El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres.

Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente.

…aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos…

Che

DEDICATORIA

A mis padres por todo el amor y dedicación que me han dado en estos 24 años de esfuerzo y sacrificio.

A mi hermano por el apoyo incondicional que me ha brindado.

A mi abuela por quererme como un hijo y dedicarse desde el momento en que nací a cuidar de mí y brindar lo mejor de ella para mi bienestar.

A mi tía Odelia por ser como mi segunda madre que siempre está conmigo aconsejándome y apoyándome.

A mis tíos de La Habana y mi primo Frank, les estoy agradecido eternamente por el apoyo y sacrificio que me han brindado en estos 5 años.

A mis tíos y primos de Holguín por preocuparse siempre por mí en especial a mi primo Mario.

A Any por ayudarme en todo lo que hiciera falta y por su buena voluntad.

A Laurent que a pesar de estar muy lejos está siempre presente en mi corazón y le estoy agradecido eternamente por ser el motor impulsor de mis actos y por proporcionarme la energía necesaria para continuar por el difícil pero hermoso camino de la vida.

A mis amigos de toda la vida que han estado conmigo en las buenas y las malas, muchas gracias hermanos.

Omar

A mis padres y abuelos por todo el amor y dedicación que me han dado en estos 23 años de esfuerzo y sacrificio.

A mis hermanos, primos y tíos por el apoyo incondicional que me ha brindado

A mis tutores Yulio y Frank por el apoyo incondicional que me ha brindado

A mis amigos de toda la vida que han estado conmigo en las buenas y las malas.

Juan

AGRADECIMIENTOS

A nuestro Comandante en Jefe y a la Revolución por proporcionarnos la oportunidad de estudiar en un centro de alto prestigio.

A la Universidad de las Ciencias Informáticas por formarnos como profesionales.

A nuestros tutores Ing. Yulio Seriocha García Gallardo e Ing. Frank David Ávalos Palomo por su dedicación.

A todos los profesores que de una forma u otra contribuyeron a formarnos como hombres de bien.

A todos los que de una forma u otra ayudaron, muchas gracias.

Omar y Juan

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Dirección Técnica y la Facultad 15 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Omar Moreno Suárez Juan Pérez Chirino

_____ _____

Yulio Seriocha García Gallardo Frank David Ávalos Palomo

_____ _____

Resumen

La Infraestructura Productiva (IP) de la Universidad de las Ciencias Informáticas (UCI) coordina la actividad productiva y brinda servicios para asegurar el correcto desarrollo y terminación de proyectos, productos o servicios.

Una de la las direcciones con que cuenta la IP es la Dirección Técnica (DT) que norma y controla el Entorno de Desarrollo de Software en la UCI y sus arquitecturas, así como el uso de Estándares y Arquitecturas de Información. Gestionando además la Infraestructura Tecnológica para el proceso productivo.

La mayoría de los servicios que presta la DT se encuentran automatizados a través del Sistema de Gestión de Servicios de la DT (SIGES-DT), este no cuenta con un subsistema para la gestión de la información de los servidores y las VLANs de forma eficaz.

Por lo antes expuesto esta investigación tiene como objetivo principal realizar el análisis, diseño, implementación y validación del subsistema para la gestión de la información de los servidores y las VLANs en los proyectos productivos en la UCI.

Para dar cumplimiento a este objetivo se siguió la línea de desarrollo del SIGES-DT, haciendo uso de metodologías y herramientas que se utilizaron en su desarrollo, además de analizar el negocio que tributa al levantamiento de requisitos funcionales y no funcionales que deberá tener el sistema, así como los artefactos propios del análisis y diseño del mismo. También se realiza la implementación según los artefactos resultantes de los flujos anteriores finalizando con la validación del resultado obtenido.

Palabras claves: sistema, gestión de servidores y VLAN.

Índice

Índice de Figuras.

Índice de Tablas.

Introducción

Actualmente la Industria de Desarrollo de Software es una de las más importantes que existe en el mundo, debido a que es de las más rentables. Esto se debe en gran medida al hecho de que la informatización se ha convertido en una necesidad para el progreso de la humanidad, ayudando a elevar la productividad y efectividad en cualquier rama que se aplica.

La Industria Cubana del Software está encaminada a ser uno de los eslabones principales en la economía del país, por esta razón es por lo que se han trazado varias estrategias con el objetivo de lograr la inserción del software cubano en el mercado mundial.

La idea de construir la Universidad de las Ciencias Informáticas (UCI) en Cuba ha sido un sueño del Comandante en Jefe Fidel Castro que se realiza día a día en el empeño de convertirla en un centro de excelencia. Desde los primeros pasos el líder de la Revolución ha guiado con su experiencia y visión de futuro la edificación de esta institución.

La UCI es una universidad productiva, cuya misión es producir software y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación.

La producción en la Universidad de Ciencias Informáticas se concentra en el desarrollo de proyectos y se destacan resultados en las esferas de salud, educación, software libre, teleformación, sistemas legales, realidad virtual, automatización, bioinformática, procesamiento de imágenes y señales, entre otras.

Promueve el desarrollo de productos y servicios informáticos en aquellas ramas donde Cuba tiene un reconocido prestigio en el mundo a través del concurso de los mejores especialistas del país para lograr una solución de calidad y de impacto internacional.

Además desarrolla programas de informatización de la sociedad cubana a través de la relación con entidades nacionales, los resultados alcanzados se extienden por todo el país.

Dentro de la Universidad de Ciencias Informáticas se encuentra Infraestructura Productiva (IP). Esta surge para coordinar la actividad productiva y brindar servicios para asegurar el correcto desarrollo y terminación de proyectos, productos o servicios.

Entre los servicios más importantes se brindan los de calidad de software, arquitectura y tecnología, servicios legales y diseño de comunicación visual.

La IP cuenta con varias direcciones entre ellas la Dirección Técnica (DT) de La Universidad de las Ciencias Informáticas que norma y controla el Entorno de Desarrollo de Software en la UCI y sus arquitecturas, así como el uso de Estándares y Arquitecturas de Información. Gestionando además la Infraestructura Tecnológica para el proceso productivo.

La DT está formada por un equipo de profesionales que se encargan de certificar los proyectos productivos de la UCI en cuanto a los Entornos de Desarrollo y Arquitectura de Software, así como en Arquitectura y Estándares de Información. En cuyas temáticas se brindan servicios de consultoría, vigilancia tecnológica y análisis de información a toda la comunidad productiva de la UCI. Cuenta con normas de estandarización establecidas, garantizando la soberanía tecnológica. Forma parte y se colabora con una Comunidad Mundial de Desarrollo de Software Libre.

Dentro de la DT se encuentra el Sistema de Gestión de Servicios que se encarga de la automatización de la mayor parte de los servicios que brinda la dirección.

En estos momentos el proceso de gestión de la información de los servidores y las VLANs es gestionado manualmente y no se cuenta con registros históricos actualizados de los mismos. Existen problemas con el conocimiento de la información referente a la ubicación y detalles de los servicios y aplicaciones que se ejecutan en los servidores de los proyectos productivos.

No existe un conjunto de condiciones extrínsecas, especializadas en la regulación y vigilancia del uso de los recursos de cómputo, necesarias para el proceso de producción de software. Esto conlleva a la fuga de información sensible en los proyectos productivos y se hace un mal uso de las computadoras destinadas a la producción.
La Universidad de las Ciencias Informáticas no posee un sistema informatizado, que permita llevar la gestión de la información de las VLAN o cercas lógicas en las diferentes áreas productivas.

Por las razones expuestas anteriormente dan lugar a que el **problema científico** a solucionar sea: ¿Cómo facilitar de forma eficaz el proceso de gestión de información relacionado con los servidores y VLANs en la Universidad de Ciencias Informáticas?
Donde se tiene como **objeto de estudio** el proceso de desarrollo de software y el **campo de acción** se enmarca en el proceso de gestión de la información relacionada con los servidores y VLANs en la UCI.
Para dar respuesta al problema planteado se define como **objetivo general:** Desarrollar un sistema que permita la informatización de los procesos que se llevan a cabo en la gestión de Servidores y las VLAN en los proyectos productivos en la UCI.
Los objetivos específicos de la investigación para dar cumplimiento a este objetivo general son:

- Identificar los conceptos fundamentales del negocio a informatizar.
- Describir los procesos del negocio identificados.

- Realizar el análisis y diseño e implementación de la aplicación.
- Evaluar la solución obtenida.

Inicialmente se parte de la siguiente **hipótesis**: Si se desarrolla una aplicación que informatice los procesos de Gestión de Servidores y las VLAN en los proyectos productivos en la UCI entonces se logrará un mayor control y eficacia de la Gestión de los Servidores y las VLAN para la producción.

Para dar cumplimiento a las tareas propuestas anteriormente se emplean métodos teóricos y empíricos como métodos científicos de la investigación.

Métodos teóricos

Analítico–sintético: Consiste en la reconstrucción de un todo, a partir del estudio de los elementos previamente descompuestos y revisados ordenadamente cada uno por separado.

Análisis histórico–lógico: El método histórico estudia la trayectoria de los fenómenos y acontecimientos en el devenir de su historia. El método lógico investiga la existencia o no de leyes generales del funcionamiento y desarrollo de los fenómenos. El método lógico requiere apoyo del método histórico para descubrir la existencia de leyes fundamentales de los fenómenos basados en los datos que va proporcionando este último, para despojarse de toda posibilidad de generar razonamientos especulativos.

Métodos Empíricos

Observación: Es el primer procedimiento de carácter empírico, en el cual pueden distinguirse el objeto, el sujeto, los medios y las condiciones de la observación. Además del sistema de conocimientos a partir del cual se formula la finalidad de la observación y se interpretan los resultados de ésta.

Entrevista: Se define como un método empírico complementario de investigación del cual se obtiene información amplia, abierta y directa de forma oral durante una conversación planificada entre el entrevistador y los entrevistados.

La investigación está formada por tres capítulos donde se abarcan los siguientes temas:

En el **capítulo 1:** Se expone la fundamentación teórica del trabajo, abordando todos los elementos teóricos que sustentan el problema científico, el objetivo del trabajo y se realiza un análisis de las tendencias de las tecnologías y metodologías actuales en el desarrollo de software. Se fundamentan las tecnologías y herramientas con las cuales se debe desarrollar el sistema, así como la metodología más adecuada para guiar el desarrollo del mismo.

En el **capítulo 2**: Se exponen los requisitos funcionales y no funcionales del sistema propuesto para la Gestión de Servidores, además de hacer referencia a los actores vinculados a los procesos que se definen. Se detallan además los artefactos generados durante las 4 fases que define RUP como metodología de desarrollo a utilizar, prestando especial atención en las descripciones de los casos de uso.

En el **capítulo 3:** Se desarrollan los artefactos pertenecientes a los flujos de trabajo análisis, diseño, implementación y pruebas, dígase diagramas de clases, diagramas de interacción, diseño de la base de datos, modelo de despliegue, diagrama de componentes y modelo de pruebas.

Capítulo 1. Fundamentación Teórica

1.1 Introducción

Durante el desarrollo del presente capítulo se abordarán diferentes temas como los sistemas de Gestión de Servidores y VLAN así como los principales conceptos asociados al dominio del problema. También se realizará un estudio de las Metodologías de Desarrollo de Software, Lenguajes de Programación, Sistemas Gestores de Bases de Datos, Estilos y Patrones Arquitectónicos, Entorno Integrado de Desarrollo y Lenguaje de Modelado más utilizados actualmente en el mundo y en nuestro país que permitirán hacer una correcta selección de los mismos para la posterior implementación del sistema. A partir de todos estos elementos se desarrollará la investigación, haciéndose más sencilla la comprensión de los temas que serán abordados en lo adelante.

1.2 Conceptos asociados al dominio del problema

Para mejor entendimiento y desenvolvimiento de las áreas temáticas que se abordarán en el presente trabajo, se mostrarán una serie de conceptos identificados durante la investigación realizada.

1.2.1 ¿Qué es un servidor?

Un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos. (1)

En una red, se denomina servidor a una computadora compartida por múltiples usuarios. Existen servidores de archivos, servidores de impresión, etc. Los servidores son computadoras de gran potencia que se encuentran a disposición de los usuarios. (2)

Un servidor es una computadora que forma parte de una red y que provee servicios a otras computadoras, que reciben el nombre de clientes.

1.2.2 ¿Qué es una VLAN?

Una VLAN (red de área local virtual o LAN virtual) es una red de área local que agrupa un conjunto de equipos de manera lógica y no física. (3)

1.2.3 ¿Qué es un Sistema Informático?

Según Andrés Berdasco Blanco un Sistema Informático es el conjunto formado por elementos hardware y software que constituyen los recursos a los cuales llegan las peticiones de los usuarios para ser atendidas, en otras palabras se puede decir que es uno o varios ordenadores con un sistema operativo y con los programas necesarios por los usuarios. (4)

Es aquel sistema que se encarga del manejo de información en la computadora, a través de la cual el usuario controla las operaciones que realiza el procesador.

Los sistemas informáticos se deben mirar desde diferentes puntos de vista:

- Para una organización un sistema informático puede ser un departamento, con recursos, que está a disposición de la organización.
- En el mundo de la informática: es un conjunto de servidores, redes y terminales de trabajo para "hacer software".
- Para los usuarios es una herramienta más que les da la organización a la que pertenecen para mejorar el funcionamiento de sus tareas.

Después de una investigación sobre los sistemas informáticos; se define que un sistema informático es aquella unión que se concibe entre los programas y los componentes físicos, ofreciéndoles respuestas a aquellos usuarios que han hecho de una forma u otra cierto pedido.

1.3 Sistemas de Gestión de la Información

En la era de la información, de la explosión de sus tecnologías, se vive la etapa en la que la humanidad ha alcanzado un desarrollo imprevisible. Se habla constantemente sobre la sociedad de la información, es visible el paso de las sociedades industriales a las posindustriales y del conocimiento, donde el factor esencial de progreso es el conocimiento. Esta nueva sociedad, con organizaciones basadas en el aprendizaje, cuyo capital más preciado es el ser humano, se sustenta en un desarrollo tecnológico sin precedentes, es el punto en el cual las grandes compañías planifican sus productos en función de la gestión del conocimiento y de la viabilidad para su obtención.

En este contexto, debe entenderse que las tecnologías de información y las telecomunicaciones no son más que un medio para transmitir y gestionar datos, información y conocimiento, el conocimiento es factor fundamental para la creación de riquezas.

Una institución de información es una organización del conocimiento, que mediante un conjunto de procesos, gestiona las capacidades, provee a los equipos de trabajo con recursos para la solución de los problemas de forma eficiente en el menor tiempo posible, con el objetivo final de obtener ventajas competitivas sostenibles en el tiempo y de aumentar las ganancias. En este sentido, Gilberto Sotolongo expresa que "la gestión de la información se ocupa de los resultados

finales, no sólo de citas y localizaciones". (5) Fernández-Molina lo corrobora cuando afirma que los profesionales de la información y sus instituciones son un factor indispensable para la permanencia de la organización en el mercado. (6)

La información es un elemento fundamental para el desarrollo, con el paso de los años, la gestión de la información ocupa un espacio mayor en la economía de los países a escala mundial.

La gestión del conocimiento es la gestión de los activos intangibles que aportan valor a la organización al momento de obtener capacidades y competencias esenciales distintivas.

Las estrategias actuales para la gestión de la información y el conocimiento deben responder los nuevos tipos de demandas, resultantes de la aparición de tendencias gerenciales más modernas en las organizaciones. En la creación de los nuevos sistemas de gestión de la información es imprescindible considerar las fuentes de datos, documentales y no documentales, los sistemas informáticos, la cultura de información, los modelos de comunicación, entre otros elementos. Según los requerimientos de los procesos internos de trabajo y los flujos de información propios, todos ellos deben propiciar la gestión del conocimiento organizacional y la implementación de sistemas de gestión de la calidad para la evaluación de los resultados y los proyectos de la institución. Se requiere, además, de la incorporación de nuevos valores a los productos y servicios de información, así como de una diseminación muy bien dirigida, con el fin de que ellos lleguen a aquellos individuos y secciones cuya actividad de generación o aplicación del conocimiento y de toma de decisiones es más importante para la empresa.

La información es un agente importante en la modificación de las conductas existentes en la organización, su correcta gestión es una herramienta fundamental para la toma de decisiones, la formación del personal, la evaluación de los productos, la determinación de los errores y el control de los procesos. La información es un recurso vital para el desarrollo de la organización. El carácter intangible de la información ha hecho que muchos directivos de la organización no inviertan los recursos suficientes para las actividades de información.

1.4 Tendencias y tecnologías actuales

1.4.1 Estilos y patrones arquitectónicos

1.4.1.1 Estilos Arquitectónicos.

Según lo expuesto por Mark Klein y Rick Kazman en el año 1999, “un estilo arquitectónico es una descripción del patrón de los datos y la interacción de control entre los componentes, ligada a una descripción informal de los beneficios e inconvenientes aparejados por el uso del estilo”.

Los estilos arquitectónicos, afirman, “son artefactos de ingeniería importantes porque definen clases de diseño junto con las propiedades conocidas asociadas a ellos”. (7)

1.4.1.2 Patrones Arquitectónicos.

Se entiende por patrón de arquitectura de software la descripción de un problema particular y recurrente del diseño, este surge en un contexto específico representando un esquema genérico y probado de su solución. Existen numerosas clasificaciones y definiciones de diversos autores. A continuación se argumentan algunos de los patrones de arquitectura más utilizados en la actualidad.

1.4.1.3 Arquitectura Cliente-Servidor

Pertenece a la clasificación de estilos jerárquicos definida por Roy Fielding donde se consideran los estilos que incluyen alguna forma de distribución o topología de red. (8) Es un modelo para desarrollar sistemas de información donde las transacciones se dividan en procesos independientes cooperando entre sí para intercambiar recursos, información y servicios. El cliente es el proceso que inicia el diálogo y el servidor el que responde a las solicitudes. (Ver Figura 1.1)

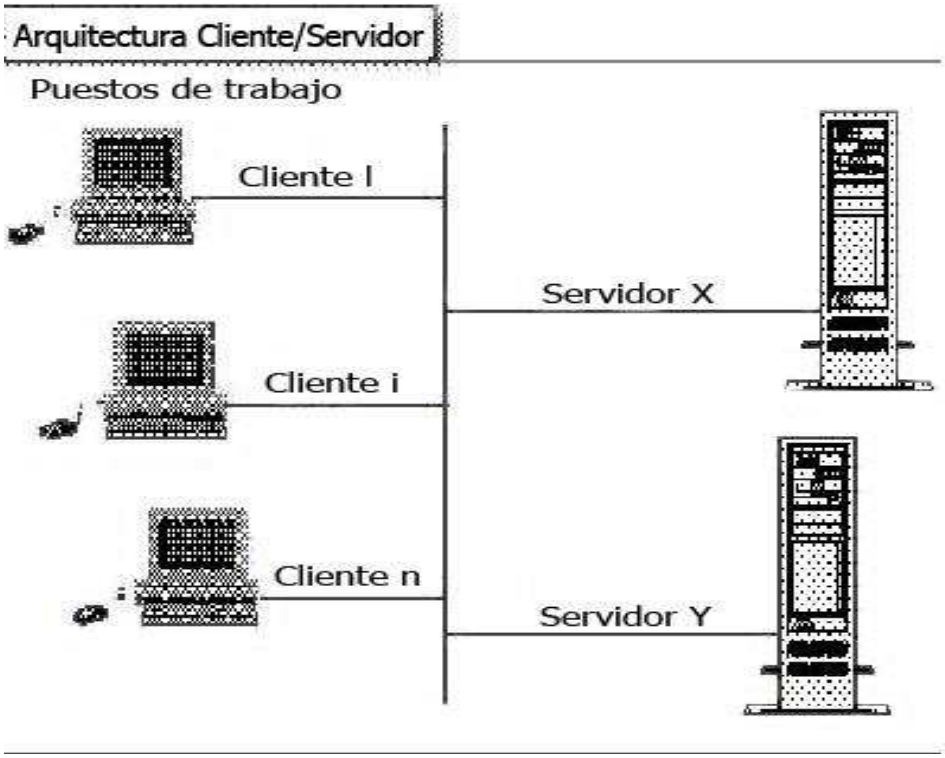


Figura 1.1 Arquitectura Cliente/Servidor

Existen numerosos aspectos que caracterizan a esta arquitectura:

- El Cliente y el Servidor pueden desempeñarse como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
- Las funciones de Cliente y Servidor pueden existir en plataformas separadas, o en la misma plataforma.
- Un Servidor puede suministrar servicio a numerosos Clientes en forma concurrente.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes.
- Los cambios realizados en las plataformas de los Clientes o de los Servidores se realizan de una manera transparente para el usuario final.

1.4.1.4 Arquitectura en Capas

La Arquitectura en Capas pertenece a una clasificación elaborada por Buschmann, Meunier, Rohnert, Sommerlad y Stal. Muchos sistemas de información en la actualidad se basan en este patrón de arquitectura en la modalidad de dos capas (Cliente/Servidor). Estas capas pertenecen al nivel de aplicación y nivel de base de datos. La utilización de este estilo acarrea inconvenientes como la recarga del nivel de aplicaciones lo que ha repercutido en la tendencia a utilizar arquitectura de tres capas. (Ver Figura 1.2) (9)

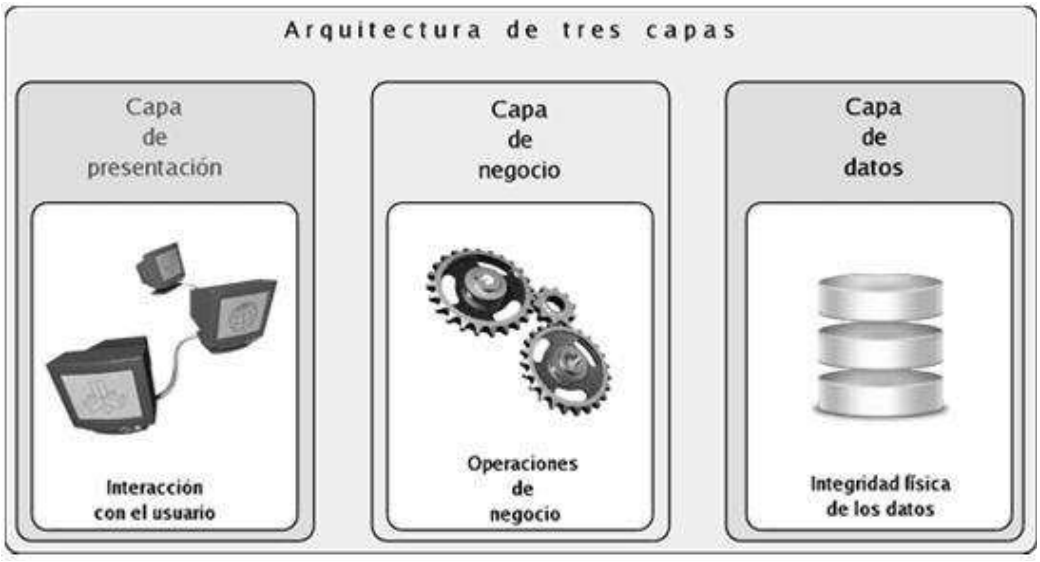


Figura #1.2 Arquitectura de tres capas.

Las tres capas propuestas por esta arquitectura son:

- Lógica de Presentación: Esta es la parte del código que interactúa con la PC o Terminal de autoservicio utilizada por el usuario. Se encarga de presentar el sistema, comunicar la

información a los usuarios y capturarla de la interacción usuario-sistema. Esta capa se encarga de tareas como la disposición de los elementos gráficos en la pantalla, escribir los datos en pantalla, manejo de ventanas y solamente se comunica con la capa de negocio.

- Lógica de Negocio: En esta capa es donde se establecen y codifican las reglas del negocio. Recibe las peticiones de los usuarios a través de la capa de presentación y mediante esta envía las respuestas solicitadas.
- Lógica del Procesamiento de los Datos: Esta capa se encarga del acceso y almacenamiento de los datos persistentes en la base de datos.

Ventajas del estilo en capas.

- Soporte de un diseño establecido en niveles de abstracción crecientes permitiendo a los implementadores del equipo de desarrollo el fraccionamiento de un problema complejo en una secuencia de pasos incrementales.
- Permite desarrollar acciones de optimización y refinamiento.
- Suministra amplia reutilización ya que permite utilizar implementaciones o versiones de una capa mientras soporten las mismas interfaces de cara a las capas adyacentes posibilitando definir interfaces de capa estándar.

Desventajas del estilo en capas.

- En numerosas ocasiones existe extrema dificultad para encontrar el nivel de abstracción correcto.
- Los cambios a realizar en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel, especialmente si se utiliza una modalidad relajada.
- La arquitectura en capas ayuda a controlar y encapsular aplicaciones complejas, pero complica en algunas ocasiones las aplicaciones simples.
- Pérdida de eficiencia.
- Trabajo innecesario por parte de capas más internas o redundante entre varias capas.

1.4.1.5 Modelo-Vista-Controlador (MVC)

Según Taylor y Medvidovic, es un estilo que pertenece a la familia de estilos de Llamada y Retorno que enfatiza la modificabilidad y la escalabilidad, pero otros autores como por ejemplo Buschmann, lo clasifica como patrón correspondiente al estilo de los sistemas interactivos. El patrón MVC aísla el modelado del dominio, la presentación y las acciones establecidas por los usuarios en el ingreso de información en tres clases diferentes:

- Modelo: El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista: Maneja la visualización de la información.
- Controlador: Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado. (10)

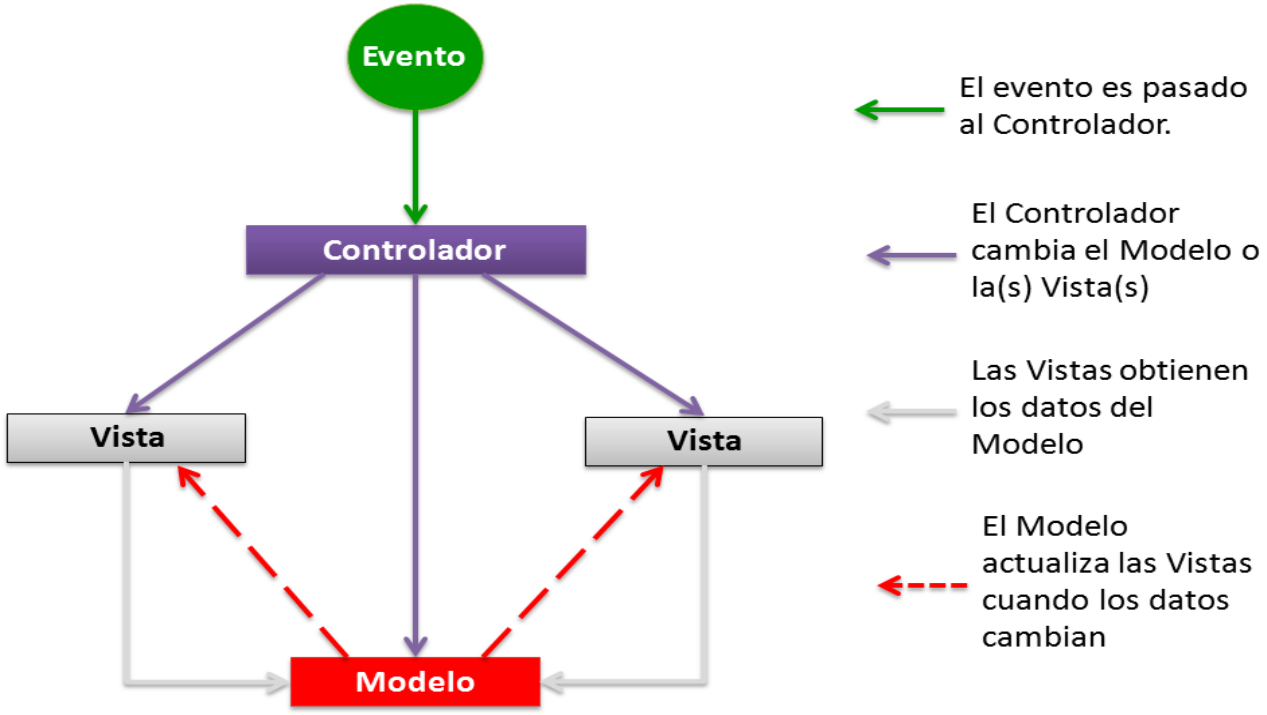


Figura #1.3 Funcionamiento del patrón MVC.

Ventajas de MVC

Esta arquitectura provee grandes ventajas, como la organización del código, la reutilización y la flexibilidad.

- Separa los datos de la representación visual de los mismos.
- Crea independencia de funcionamiento.
- Ofrece maneras sencillas de probar el correcto funcionamiento del sistema.
- Permite el escalamiento de la aplicación en caso requerido.
- Soporte de vistas múltiples, debido a que no existe una dependencia directa entre la vista y el modelo, la interfaz de usuario posibilita mostrar varias vistas de la misma información de forma simultánea.

- Adaptación al cambio, ya que al existir nuevas necesidades por parte de los usuarios la interfaz de usuario se ve sometida a cambios con frecuencia. Como el modelo no depende de las vistas, facilita la adición de nuevas opciones de presentación generalmente sin afectar al modelo.

Desventajas de MVC

- Su estructura agrega complejidad al sistema.
- Costos de actualizaciones frecuentes.

Patrón de arquitectura a utilizar: MVC.

De los patrones arquitectónicos antes tratados se ha podido comprobar que son útiles en la construcción de sistemas complejos, presentando cada uno de ellos sus ventajas y desventajas. Para el desarrollo del sistema se seleccionó el Modelo-Vista- Controlador. Esta elección está basada en las ventajas resultantes de su uso que en todo momento inclinan la balanza a su favor aunque no se deben ignorar sus desventajas.

- Este patrón es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios.
- Está implementado por la mayoría de los frameworks web de la actualidad.
- A pesar de que el número de archivos a mantener y desarrollar crece de forma considerable, la separación entre modelo y vista consiguen un mantenimiento más sencillo de las aplicaciones.

1.5 Lenguajes de programación

Los lenguajes de programación y los entornos de desarrollo integrados tienen suma importancia para cualquier proyecto de software pues con estos se han creado innumerables programas y herramientas que han ayudado al hombre a controlar de una manera más sencilla a los ordenadores, así como de realizar múltiples tareas y actividades. Un lenguaje de programación es aquel que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. En la actualidad existe un numeroso conjunto de lenguajes que se utilizan en el desarrollo de aplicaciones web. Estos se clasifican en lenguajes del lado del Servidor y del lado del Cliente según la correspondencia que exista entre su ubicación y la arquitectura Cliente/Servidor.

Lenguajes del lado del cliente: Son aquellos que pueden ser directamente procesados por el navegador y no necesitan tratamiento previo.

Lenguajes del lado del servidor: Son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible.

1.5.1 Lenguajes del lado del cliente

Entre los lenguajes del lado del cliente se encuentran JAVA, JavaScript, Applets de Java, Flash, HTML y CSS, los dos últimos organizan a los demás dentro de la página web. A continuación se presentan las características más importantes de algunos de ellos.

1.5.1.1 JavaScript

Este lenguaje permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web mayor dinamismo. Es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario.

El advenimiento de Java Script ha resuelto de manera fácil y elegante la mayoría de los problemas con que se enfrenta el diseñador de páginas web referidos a la programación. Sus requerimientos son relativamente sencillos, es un lenguaje cuyos códigos se interpretan en el navegador del cliente, sin tener que ir y venir del cliente al servidor actualizando la información.

1.5.1.2 Hyper Text Markup Language (HTML)

Es un lenguaje de marcado diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Fue desarrollado en 1990 por Tim Berners-Lee con el objetivo de facilitar el acceso a investigaciones científicas. Los documentos HTML contienen dos tipos de información: la que se muestra en la pantalla (texto, imágenes...) y los códigos (tags o etiquetas), transparentes al usuario, que indican cómo se debe mostrar esa información.

1.5.1.3 Cascading Style Sheets (CSS)

El origen de las Hojas de Estilo en Cascada del inglés (Cascading Style Sheets) data alrededor del año 1970 con el objetivo de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos. Permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web de forma simultánea. Funciona a través de reglas, que se dividen en dos partes: un selector y la declaración que a su vez, está compuesta por una propiedad y el valor que se le asigne.

Ventajas de las hojas de estilo:

Control del diseño: Al definirse el formato de un documento electrónico mediante CSS se puede cambiar dicho formato modificando solamente la hoja de estilo. Esto permite reducción de tiempo y uniformidad en los procesos de diseño.

Redefinición de etiquetas: Se declara mediante una sola línea de código el formato deseado especificando color y tamaño a las etiquetas deseadas. En caso de no utilizar CSS este proceso sería muy engorroso.

Personalización: Permite definir aspectos concretos de un documento facilitando su diseño.

Maquetación y accesibilidad: Mayor precisión al colocar los elementos en el diseño al exceptuar el uso de tablas para la maquetación. Esto permite un documento con un formato más exquisito a pesar de que el navegador no soporte CSS.

Tamaño: Se reduce el tamaño de los ficheros ya que todos los requerimientos que guardan relación con el formato de todas las páginas estarán en un solo archivo que pertenece a la CSS.

1.5.2 Lenguajes del lado del servidor

1.5.2.1 Active Server Pages (ASP)

ASP del inglés (Active Server Pages) es una tecnología creada por Microsoft para el desarrollo web. La misma estaría incluida dentro de los lenguajes **ISS (Include Server Side)**[1] de 2ª generación, por tanto se cataloga como una evolución de los **CGI's (Common Gateway Interface).** (11)

Se pueden definir como archivos que se ejecutan en el servidor cuyo contenido puede ser tanto HTML como código de script. Soporta VBScript, Java y JavaScript. ASP no define un lenguaje de programación sino que define una serie de objetos de servidor, constituidos por una serie de métodos que se pueden utilizar en actividades como acceso a bases de datos, lectura de ficheros entre otras.

1.5.2.2 HypertextPreprocessor (PHP)

PHP del inglés (Hypertext Preprocessor), es un leguaje de alto nivel embebido en páginas HTML y ejecutado en el servidor. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí misma. La meta del lenguaje es permitir rápidamente a la comunidad de desarrolladores la generación dinámica de páginas Web. (12)

[1] Del inglés (Server Side Includes): Son directivas insertadas en páginas HTML que permiten inserción de contenido generado dinámicamente en las páginas web.

Dispone de múltiples herramientas que permiten acceder a bases de datos de forma sencilla, por lo que es ideal para crear aplicaciones para Internet.

Características del lenguaje

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierte en la herramienta ideal para la creación de páginas Web dinámicas: (13)

- Soporte para una gran cantidad de bases de datos entre las que se encuentran: MySQL, PostgreSQL y Oracle.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas de la Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- El código se pone al día continuamente con mejoras y extensiones del lenguaje para ampliar las capacidades de PHP.

Ventajas que proporciona la utilización del lenguaje PHP

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y Postgres.
- Posee una amplia documentación en Internet, incluyendo una gran variedad de ejemplos y de ayudas.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la Web.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel, facilitando un poco la programación.
- En PHP los scripts del lado del servidor se insertan dentro del código HTML. Es gratuito y multiplataforma, y está ampliamente difundido en el mundo entre la comunidad de programadores.

Frameworks en PHP

Un framework es una estructura de software que está formada por diversos componentes personalizables que se pueden configurar y adicionar elementos para el desarrollo de una aplicación. Su utilización fomenta la reutilización de código, promueve buenas prácticas de desarrollo y proporciona una reducción de tiempo en los procesos de desarrollo.

Symfony

Symfony es un framework que optimiza el desarrollo de aplicaciones web. Su objetivo es acelerar la creación y mantenimiento de aplicaciones web sustituyendo las repetitivas tareas de codificación para lo cual proporciona varias herramientas y clases. Es fácil de instalar en cualquier configuración, sólo es necesario Unix o Windows con un servidor web y PHP instalado. Es compatible con la mayoría de los sistemas de bases de datos. Está desarrollado en PHP 5 y publicado bajo una licencia de software libre. (14)

Principales características

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Es independiente del sistema gestor de bases de datos.
- Fomenta las mejores prácticas y patrones de diseño para la web.
- Código fácil de leer que permite un mantenimiento muy sencillo.
- Fácil de extender, permitiendo su integración con librerías desarrolladas por terceros.

Entorno de desarrollo y herramientas: Este framework contiene diferentes entornos de desarrollo y varias herramientas que posibilitan la automatización de las tareas que son comunes a la ingeniería de software para desarrollar aplicaciones:

- Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.
- El framework de desarrollo de pruebas unitarias y funcionales proporciona las herramientas ideales para el desarrollo basado en pruebas ("test-drivendevelopment").
- La barra de depuración web simplifica la depuración de las aplicaciones, ya que muestra toda la información que los programadores necesitan sobre la página en la que están trabajando.
- La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.
- Es posible realizar cambios "en caliente" de la configuración (sin necesidad de reiniciar el servidor).
- El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación.

¿Por qué utilizar Symfony?

- Es adecuado para proyectos de gran envergadura y que contengan algún grado de complejidad.
- Posee calidad en el código fuente y cuenta con gran documentación disponible.
- Sus herramientas y clases proporcionan una reducción del tiempo en el desarrollo de una aplicación web compleja.
- Implementa el MVC.
- Implementa las tareas que son comunes en el desarrollo web fomentando la reutilización.

1.6 Entorno Integrado de Desarrollo (IDE).

1.6.1 Zend Studio.

Zend Estudio es un programa de la compañía Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda, permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. (15)

Si se desea aumentar la productividad en los desarrollos PHP no cabe duda que este programa puede resultar útil y prestar gran ayuda en la realización de los mismos ofreciendo un entorno agradable y funcionalidades que simplifican el trabajo y optimizan el resultado. Todas las opciones que dispone están pensadas con acierto por personas capacitadas que conocen a fondo la tecnología. Zend Studio incorpora suficientes ayudas como para que su utilización sea confiable y resulte idóneo para desarrollar aplicaciones Web.

1.6.2 Eclipse

En la web oficial de eclipse, se define como "An IDE foreverything and nothing in particular"(un IDE para todo y para nada en particular). Eclipse es en el fondo, únicamente un armazón sobre

el que se puede montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados.

La arquitectura de plugins de Eclipse permite además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros. Existen versiones de Eclipse instalables para cualquier plataforma que incluyen el código fuente y los plugins más habituales.

Una de las características más curiosas del IDE Eclipse es el modo en que se compilan los proyectos. No existe en Eclipse ningún botón que permita compilar individualmente un fichero concreto. La compilación es una tarea que se lanza automáticamente al guardar los cambios realizados en el código.

La principal diferencia entre un simple editor y un buen entorno de desarrollo es que este se integre con una buena herramienta visual para depurar los programas escritos. Eclipse incluye un depurador potente, sencillo y muy cómodo de utilizar. Permite de una forma muy simple, generar automáticamente la documentación del propio proyecto.

Eclipse ofrece servicios que permiten el trabajo entre desarrolladores de forma más eficiente y sincronizada, pues cuenta con las tareas, que posibilitan una comunicación directa entre los miembros del equipo, permitiendo dar órdenes de trabajo de manera online en forma de correo electrónico. Esta opción hace de Eclipse un IDE ideal para el trabajo en equipo.

1.6.3 NetBeans

En la web oficial de NetBeans se define como un entorno de desarrollo para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo.

NetBeans permite crear aplicaciones Web con PHP 5, un potente debugger integrado y además viene con soporte para Symfony un gran framework MVC escrito en PHP. Al tener también soporte para AJAX, cada vez más desarrolladores de aplicaciones LAMP o WAMP, están utilizando NetBeans como IDE.

1.7 Metodologías de Desarrollo del Software.

Los procesos de desarrollo son el modo de trabajar eficientemente para evitar catástrofes que llevan a que un gran porcentaje de proyectos se terminen sin éxito. El objetivo de un proceso de desarrollo es elevar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso.

Grady Rumbaugh[2] dio la siguiente definición:

"Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo". (16)

Para tener un proceso de producción de software con el menor número de fallos, adecuado a las necesidades del cliente y entregar a tiempo el producto, la producción de software debe convertirse en un proceso disciplinado.

Una metodología de desarrollo de software define qué, cuándo, y cómo alcanzar un determinado objetivo. Esta debe ser capaz de servir como guía para todos los participantes (clientes, usuarios, desarrolladores y directores ejecutivos), evolucionar durante muchos años permitiendo limitar su alcance en un momento del tiempo dado a las realidades de la tecnología, herramientas, personas y patrones de organización.

Se puede decir que, una metodología de desarrollo de software guía a los desarrolladores, a los que les brinda métodos y herramientas, proporcionándoles una ayuda muy importante e indispensable para que el producto final posea las funcionalidades requeridas por el cliente y que cumpla con las necesidades del mismo y del usuario final, es una secuencia de actividades organizadas y bien pensadas que transforman los requisitos del cliente en el producto final.

En el mundo existen diferentes metodologías que son usadas para realizar el análisis de un software, entre ellas se encuentran las metodologías tradicionales y las metodologías ágiles. Las metodologías tradicionales se centran en el control de los procesos estableciendo actividades involucradas como: los artefactos que se deban producir y las herramientas que se deben usar; sin embargo las metodologías ágiles se centran en otras dimensiones como el factor humano o

[2] Científico de la computación y metodologista que escribió varios libros sobre UML y RUP junto a Ivar Jacobson y Grady Booch.

el producto de software dándose mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones cortas.

Algunas de las metodologías más usadas en la actualidad son:

- Rational Unified Process.
- Extreme Programming.
- SCRUM.
- Crystal.

1.7.1 Rational Unified Process (RUP)

La metodología RUP es una metodología pesada, está basada en una notación gráfica, la cual permite especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos. El proceso unificado de desarrollo, RUP, es el resultado de la evolución e integración de diferentes metodologías de desarrollo de software. Permite sacar el máximo provecho de los conceptos asociados a la orientación a objetos y al modelado visual. Cuenta con las mejoras prácticas del modelo de desarrollo de un software en particular. (17)

Sus principales características son:

- **Guiado por casos de uso**: Los casos de uso son el instrumento para describir el comportamiento del software y extraer los casos de prueba con los que se valida el sistema.
- **Centrado en la arquitectura**: Los modelos son proyecciones del análisis y el diseño, describe la arquitectura del producto a desarrollar.
- **Iterativo e incremental**: Durante todo el proceso de desarrollo se producen versiones superiores.

RUP divide en 4 fases el desarrollo del software:

- **Inicio**: Determinar la visión del proyecto.
- **Elaboración**: Determinar la arquitectura óptima.
- **Construcción**: Obtener la capacidad operacional inicial.
- **Transición**: Obtener el release del proyecto.

Flujos de Trabajo:

Disciplina de Desarrollo

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

- **Requerimientos**: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño**: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas, por lo que indica con precisión lo que se debe programar.
- **Implementación**: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba**: Busca los defectos a lo largo del ciclo de vida.
- **Instalación o despliegue**: Produce el release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, entre otras.) para entregar el software a los usuarios finales.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un alto grado de certificación en el desarrollo del software. La metodología RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios.

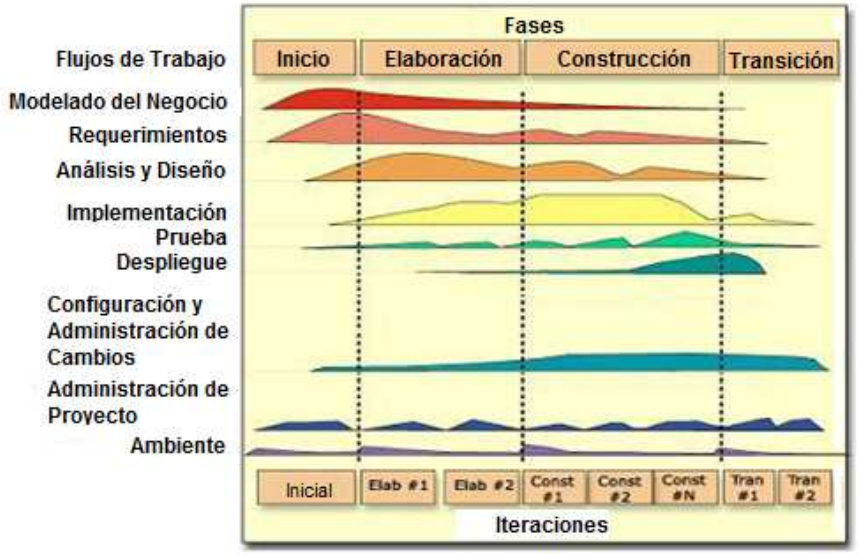


Figura #1.4 Fases y flujos de trabajo de la metodología RUP.

Otras Características de la Metodología RUP. (18)

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, entre otros.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

1.7.2 Extreme Programming (XP)

La Programación Extrema es una de las metodologías de desarrollo de software más exitosas en la actualidad, se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado; utilizada para proyectos de corto plazo para su entrega y equipo de desarrolladores pequeños. La misma consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, que puede ser fundamental para el éxito del proyecto.

A diferencia de RUP como metodología pesada, la Programación Extrema (XP) es una metodología ágil que intenta reducir la complejidad del software por medio de un trabajo orientado al objeto, basado en las relaciones interpersonales y la velocidad de reacción. Intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Este representante debe estar en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones. (19)

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. (20)

Características de XP, la metodología se basa en: (21)

- **Pruebas Unitarias**: Se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelanta en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

- **Re fabricación**: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares**: Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

La Programación Extrema empieza en pequeño y añade funcionalidad con retroalimentación continua. El manejo del cambio se convierte en parte particular del proceso. El costo del cambio no depende de la fase o etapa. No introduce funcionalidades antes de que sean necesarias. El cliente o el usuario se convierten en miembro del equipo. Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Fases de XP

- **Planificación:** Historias de Usuario, Plan de Entregas, Velocidad del Proyecto, Proyecto, Rotaciones, Reuniones.
- **Diseño:** Metáfora del Sistema, Tarjetas CRC, Soluciones Puntuales, Funcionalidad mínima, Reciclaje.
- **Desarrollo:** Disponibilidad del cliente, Unidad de Pruebas, Programación por parejas, Integración.
- **Pruebas:** Implantación, Pruebas de aceptación.

Ventajas de XP

- Los programadores inexpertos aprenderán de los expertos al emparejarse con ellos.
- Al trabajar de dos en dos, el código será de mayor calidad desde el mismo momento de crearlo y tendrá menos fallos.
- El estilo de programación tiende a unificarse.

Desventajas de XP

- Para un programador experto puede resultar tedioso tener a un inexperto a su lado permanentemente.

- El programador experto no aprende y su trabajo se ve ralentizado.
- La mejora o cambios en el estilo de programación puede resultar más complejo.

Análisis Comparativo.

RUP constituye una metodología formal, ampliamente configurable, con facilidad para la construcción de componentes reutilizables, que puede ser aplicada a muchos proyectos de desarrollo de software, permitiendo adaptar sus artefactos, roles, y modelos a las especificidades del proyecto que esté siendo desarrollado. Extreme Programming es una metodología ágil, que requiere interacción con el cliente en todo momento, que es utilizada principalmente en proyectos donde se automaticen procesos con requerimientos inestables, y que tiene como meta entregar el producto al cliente lo más rápido posible.

Metodología	Dirigido por casos de uso	Desarrollo iterativo e incremental	Construcción de componentes reutilizables	Participación activa del cliente
XP	No	No	No	Si
RUP	Si	Si	Si	No

Tabla # 2.1: Comparación entre metodologías.

1.8 Lenguaje de Modelado.

Lenguaje Unificado de Modelado (UML).

Desde los inicios de la ingeniería de software el hombre necesitó modelar mediante dibujos sus conceptos. Con ello se facilita el análisis y comprensión de sus ideas. Con el desarrollo de esta industria y los procesos de desarrollo de software en equipo surge la necesidad de crear lenguajes visuales que estandarizarán el modelado. Este debía cumplir con una premisa: la modelación de un sistema debe ser comprendido por desarrolladores de cualquier parte del mundo. Para dar solución a lo mismo es creado el Lenguaje Unificado de Modelado por Grady Booch, Jim Rumbaugh e Ivar Jacobson.

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Esta herramienta de modelado ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos

como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.(22)

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.
UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema (Figura 1.6). Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre ellos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos.

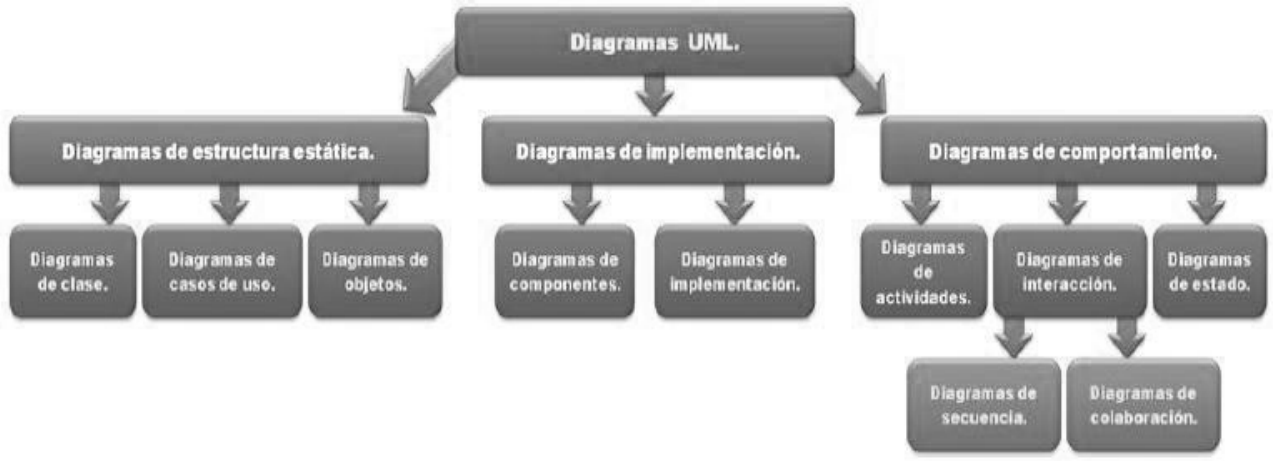


Figura #1.5 Diagramas UML

1.9 Herramientas CASE.

Las herramientas CASE[3] son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de

[3] Case: **C**omputer **A**ided **S**oftware **E**ngineering, en español (Ingeniería de Software Asistida por Computadoras).

tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Las Herramientas CASE tienen como objetivo:

1. Mejorar la productividad en el desarrollo y mantenimiento del software.
2. Aumentar la calidad del software.
3. Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
4. Mejorar la planificación de un proyecto
5. Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
6. Automatizar el desarrollo del software como: documentación, generación de código, pruebas de errores y gestión del proyecto.
7. Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
8. Gestión global en todas las fases de desarrollo de software con una misma herramienta.
9. Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.9.1 Visual Paradigm.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos. (23)

El Modelado Visual es el proceso que permite representar gráficamente el sistema software, permitiendo resaltar los detalles más importantes. Un buen modelo:

- Identifica requisitos y comunica información.
- Se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos.
- Permite ver las relaciones entre los componentes del diseño.
- Mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.

Visual Paradigm es una Herramienta CASE que da soporte al modelado visual con UML 2.0 como mínimo, entre sus principales características se encuentran:

- Entorno de creación de diagramas para UML 2.0
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

1.9.2 Rational Rose.

IBM Rational Rose Enterprise es uno de los productos más completos de la familia Rational Rose. Todos los productos de Rational Rose dan soporte a Unified Modeling Language (UML), pero no son compatibles con las mismas tecnologías de implementación. Rational Rose Enterprise es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Al igual que todos los productos de Rational Rose, ofrece un lenguaje de modelado común que agiliza la creación del software. (24)

Dos características populares de Rational Rose son su capacidad de proporcionar el desarrollo iterativo e ingeniería ida-vuelta. Rational Rose permite que los diseñadores se aprovechen del desarrollo iterativo porque el nuevo uso se puede crear en etapas con la salida de una iteración que se convierte en la entrada al siguiente. Entonces, como el revelador comienza a entender cómo los componentes obran recíprocamente y hacen modificaciones en el diseño, Rational Rose puede realizarse qué es llamada “ingeniería ida-vuelta” yendo detrás y la puesta al día del resto del modelo para asegurar el código sigue siendo constante.

Rational ofrece diferentes funcionalidades:

- Diseño dirigido por modelos que redundan en una mayor productividad de los desarrolladores, admitiendo UML, COM, OMT y Booch.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

- Capacidades de ingeniería inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Soporte OLE.
- Disponibilidad en múltiples plataformas.

1.10 Sistemas Gestores de Bases de Datos

1.10.1 PostgreSQL

PostgreSQL ofrece muchas ventajas para su compañía o negocio respecto a otros sistemas de bases de datos:

- Este software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características, estabilidad y rendimiento.
- En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad.
- El código fuente está disponible para todos sin costo.
- PostgreSQL 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.

PostgreSQL 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.

Manejador de bases de datos. PgAdmin.

Todas las bases de datos construidas necesitan ser administradas y qué mejor para cumplir con dicho objetivo que un programa con interfaz gráfica. Para PostgreSQL existen muchas herramientas de propósito general que administran sus bases de datos entre las que se encuentra PgAdmin una de las más utilizadas por la comunidad de desarrollo del software libre. PgAdmin diseña, mantiene y administra fácilmente las bases de datos construidas en PostgreSQL, funciona en distintos sistemas operativos como Windows 95/98, NT, XP, así como en plataformas libres. Es la herramienta más popular y completa, diseñada para responder a las necesidades de los usuarios permitiéndole escribir desde simples consultas y sentencias SQL hasta diseñar complejas bases de datos. Algunas de las características de PgAdmin son:

- Entradas SQL aleatorias.

- Pantallas de información y ayudas para bases de datos, tablas, índices, secuencias, vistas, programas de arranque, funciones y lenguajes.
- Preguntas y respuestas para configurar usuarios, grupos y privilegios.
- Control de revisión con mejora de la generación de script.
- Ayudas para importar y exportar datos.
- Ayuda para migrar bases de datos.

1.10.2 Oracle

Oracle es un Sistema de Gestión de Bases de Datos Relacional (SGBDR). La base de datos Oracle es una herramienta muy confiable y segura, tiene opciones de auditoría, copia de seguridad y aplicaciones para la toma de decisiones que lo diferencian de sus competidores libres y propietarios. Además se basa en la tecnología cliente/servidor. El acceso a los datos se otorga según privilegios concedidos por el administrador y es posible acceder a datos de Oracle usando software de otros fabricantes. Es una de las bases de datos más utilizadas actualmente.
Oracle está disponible en múltiples plataformas como Windows y Linux. La naturaleza multiplataforma de Oracle, lo convierte en una verdadera solución empresarial.

1.11 Conclusiones Parciales

En este capítulo se realizó un estudio de los conceptos fundamentales existentes en los procesos de negocio, realizando una breve explicación de cómo se llevan a cabo. Además se analizaron los sistemas de gestión de la información.

Después de haber analizado las características fundamentales de las tecnologías, lenguajes de programación y sistemas gestores de bases de datos candidatos para la implementación de la propuesta de este trabajo y teniendo en cuenta que éste formará parte de un sistema ya desarrollado se tuvo como principal elementos a considerar en cada selección la posibilidad de usar las mismas herramientas y tecnologías usadas en el anterior desarrollo del Sistema de Gestión de Servicios por ser éstas más conocidas y documentadas. También se analizó que fuesen herramientas y tecnologías libres, o de código abierto, motivada por las características que presentan, que las hacen factibles para países subdesarrollados como Cuba.

A partir de estas consideraciones y el estudio de las características fundamentales de los lenguajes de programación se usará PHP como lenguaje del lado del servidor por todas las características antes mencionadas y por ser el lenguaje usado en el desarrollo del Sistema de Gestión de Servicios, además PHP es un lenguaje orientado a objetos, simple, elegante y con

seguridad en el tratamiento de tipos de datos, cuenta con todas las funciones suficientes y necesarias para brindar una solución óptima. También es el lenguaje de scripting más famoso del mundo debido a muchas razones, pero principalmente por su amplia flexibilidad y simpleza, a esto se suma que es muy usado en la UCI y existe mucha documentación y experiencia en su uso.

Como lenguaje del lado del cliente se propone el uso de JavaScript, por su universal aceptación y la limitante de portabilidad de Visual Basic Script al ser compatible solamente con Internet Explorer.

Se ha seleccionado además la tecnología NetBeans como editor de PHP. La elección se ha basado fundamentalmente en la experiencia con su uso en la implementación del Sistema de Gestión de Servicios. También en su soporte, posibilidades de depuración y pruebas de PHP con un kit muy completo de herramientas para la creación de aplicaciones altamente fiables como lo requiere el mercado; en su alto rendimiento y escalabilidad; la seguridad mejorada sobre otras tecnologías web existentes y la posibilidad de hacer despliegue de sistemas desarrollados con esta tecnología en ambiente tanto Windows como Linux, además de ser de código abierto, objetivos específicos en la presente investigación.

Como framework a utilizar, Symfony deviene candidato ideal porque además de haber sido usado en todo el anterior desarrollo del Sistema de Gestión de Servicios separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web; proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja; así como automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Como Servidor Web se hace necesario utilizar el Apache debido a que el Sistema de Gestión de Servicios utiliza Apache como Servidor Web. También es conveniente utilizarlo por su aceptación universal y gran potencialidad.

Como sistema de gestión de base de datos, se seleccionó PostgreSQL en su versión 8.1. PostgreSQL en un potente gestor de Base de Datos. Es software libre, corre en un gran número de sistemas operativos, siendo multiplataforma. Tiene todas las características de los SGBD modernos, como son las llaves foráneas, vistas, procedimientos almacenados, varios tipos modernos de datos, entre otros.

Postgres 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. Otras características aportan potencia y flexibilidad adicional como son las restricciones (constraints), disparadores (triggers), reglas (rules) e integridad transaccional.

Como ya se ha explicado en epígrafes anteriores de este capítulo, la correcta selección de la metodología de desarrollo a utilizar en un proyecto es crucial para el buen desarrollo del mismo. Después de analizar las características fundamentales de las metodologías de desarrollo RUP y XP se concluyó que para un mejor desarrollo del proyecto, era necesario utilizar RUP.
A pesar de que pudiese parecer un proyecto de corto plazo, es decir pequeño, representa un sub-sistema de un gran proyecto que necesita de una metodología robusta para su desarrollo, y además de que debe existir uniformidad entre todos los sub-sistema, por lo que no puede utilizarse una metodología ligera en algunos módulos y una pesada en otros.

Con la comparación de las herramientas de modelado CASE se llegó a la conclusión de que se utilizaría Visual Paradigm, esta herramienta soporta hasta la fecha UML 2.1 completo como mínimo y BPMN, entre otros. Visual Paradigm también ha sido usada en el Sistema de Gestión de Servicios por lo cual es conveniente su uso debido a que existe un mayor dominio en su utilización. Además la UCI cuenta con la licencia para el uso de esta herramienta.

Luego de haber seleccionado las herramientas y tecnologías necesarias para el desarrollo de la presente investigación, se está en condiciones de hacer una presentación del sistema propuesto.

Capítulo 2: Descripción de la Solución Propuesta

2.1 Introducción

En el presente capítulo se describen los procesos del negocio que tienen que ver con el objeto de estudio, primeramente se modela el negocio propuesto y se identifican los actores, trabajadores, los casos de uso correspondientes y la descripción de los mismos. Se especifican las reglas asociadas al dominio del problema.
También se enumeran los requisitos tanto funcionales como no funcionales que debe cumplir el sistema que se propone (La aplicación Web a desarrollar), permitiendo así hacer una mejor concepción de forma general del sistema, se identificará mediante un Diagrama de Casos de Uso las relaciones de los actores que interactúan con el sistema y las secuencias de acciones con las que interactúan.

El desarrollo de la aplicación Web que se propone se centra en la puesta en práctica del Proceso Unificado de Desarrollo de Software (RUP), con el apoyo del Lenguaje Unificado de Modelado (UML) para la modelación (generación) de los distintos artefactos que involucran el desarrollo del trabajo y haciendo uso de la herramienta case Visual Paradigm para darle cumplimiento a la solución propuesta.

2.2 Modelo del negocio propuesto

Para describir los procesos del negocio que se relacionan con el campo de acción de este trabajo es necesario centrarse en los procesos de gestión de servidores y VLANs de los proyectos productivos. Para un mayor y mejor entendimiento de cómo se lleva actualmente a cabo estos procesos es mejor hacer uso de las técnicas de modelado que propone UML. El paso más importante dentro de dicho modelado es la identificación de los procesos del negocio. La obtención de los procesos del negocio es crucial ya que establece las metas y límites de modelado del negocio.

Dada la estructuración del negocio que se está estudiando se propone un Modelo del Negocio ya que este permite de forma visual mostrar a los usuarios los principales conceptos que se manejan. Esto se hace con el objetivo de ayudar a los usuarios, clientes, desarrolladores y otros a tener un mejor entendimiento de los procesos de negocio y a utilizar un vocabulario común entre todos para un mejor entendimiento del contexto en que se pone el sistema. Para una mejor captura de los requisitos y así construir un sistema correcto se necesita tener un seguro conocimiento de las actividades y del funcionamiento del objeto de estudio.

2.2.1 Reglas generales del negocio

A continuación se definirán las políticas que deben cumplirse o condiciones que deben satisfacerse para que se ejecuten los casos de uso, de manera que regulen los procesos de negocio de acuerdo a las especificidades de cada uno.

- El Jefe de Proyecto es el encargado de brindar toda la información referente a los servidores.
- El Especialista de la DT es el encargado de hacer el levantamiento de toda la información de los servidores.
- Al Especialista de la DT debe brindársele toda la información referente a los servidores y sus servicios.
- El Especialista de Redes utiliza la información recogida para la creación de las VLANs

2.2.2 Actores del negocio

Los actores del negocio son aquellas personas o sistemas que obtienen un resultado observable de gran valor de los procesos del negocio. A continuación se definen los actores del negocio estudiados en la siguiente tabla.

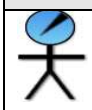
Actor	Descripción
 Jefe de proyecto	Brinda toda la información referente a los servidores.

Tabla #2.1: Descripción de los Actores del Negocio

2.2.3 Trabajadores del negocio

Los trabajadores del negocio son aquellas personas o sistemas que están vinculadas en uno o más procesos del negocio, participando en ellos, pero que a diferencia de los actores no obtienen valor alguno. Los trabajadores del negocio estudiado se definen en la siguiente tabla.

Trabajador	Descripción
 Especialista de la DT	Procesa y registra toda la información
 Especialista de redes	Utiliza la información registrada para la creación de las VLANs

Tabla #2.2: Descripción de los Trabajadores del Negocio

2.2.4 Diagrama de Casos de Uso del Negocio

El Diagrama de Caso de Uso del Negocio describe los procesos de negocio de una empresa en términos de casos de uso y actores, que se corresponden con los procesos del negocio y los clientes, respectivamente. Este modelo describe los procesos relacionados con el trabajo de gestionar la información de los servidores y VLANs do los proyectos productivos.

Figura #2.1 Diagrama de Casos de Uso del Negocio

2.3 Descripción de los procesos del negocio propuesto

Se expondrán a continuación las descripciones de cada caso de uso, en vista a una mayor comprensión del desarrollo de los procesos, puesto que los mismos describen en detalle los pasos y estados por los que transitan los casos de uso.

Diagramas de Actividades del Caso de Uso: Gestionar Servidor

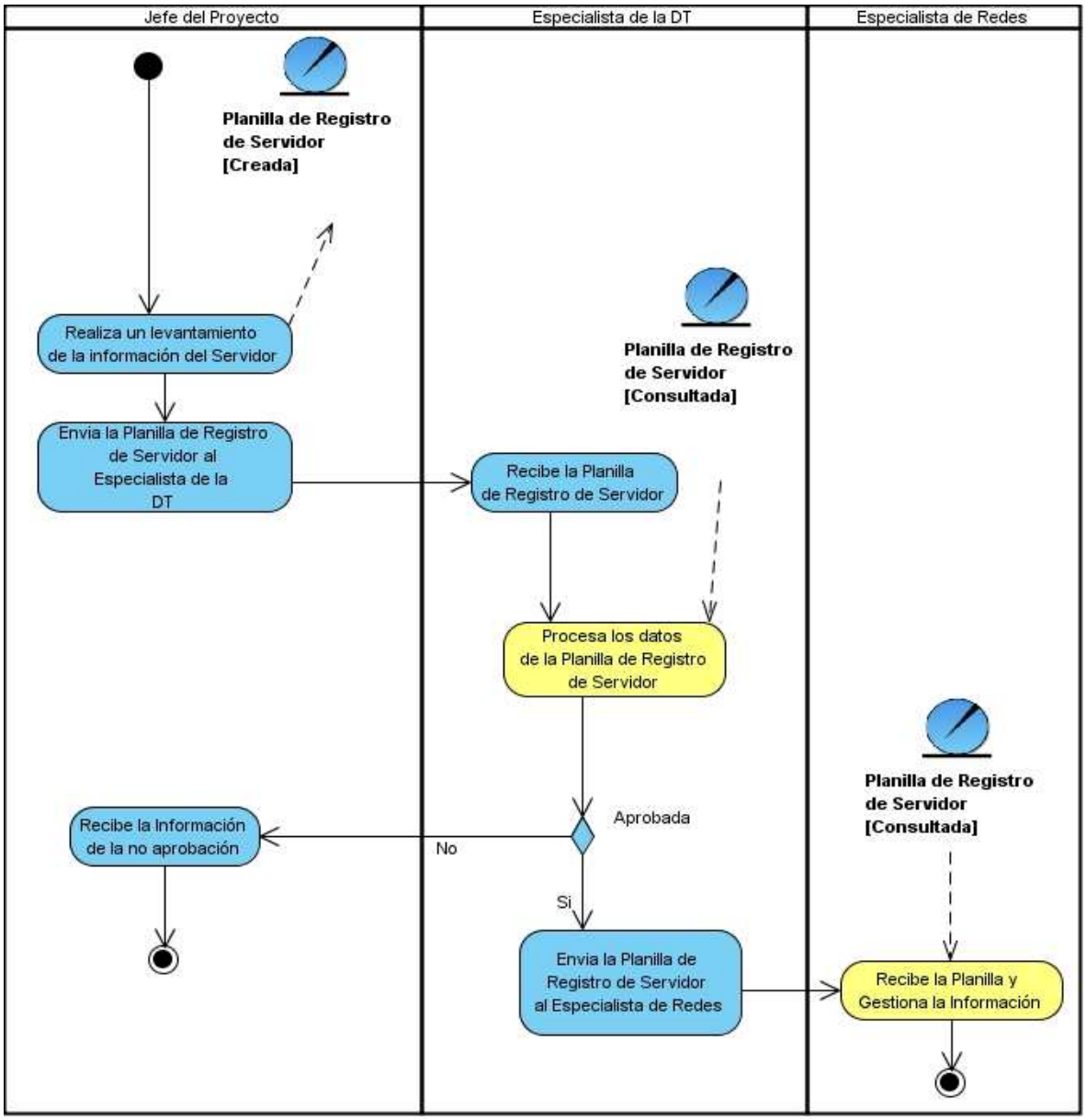


Figura #2.2 Diagrama de Actividades del Caso de Uso del Negocio: Gestionar Servidor

2.3.1 Descripción del Caso de Uso de Negocio: Gestionar Servidor

Caso de Uso:	**Gestionar Servidor.**
Actores:	Jefe de Proyecto
Trabajadores	Especialista de la DT

Resumen:	El caso de uso se inicia cuando el actor Jefe de Proyecto realiza un levantamiento de la información del Servidor y la envía al Especialista de la DT que la procesa y envía al Especialista de Redes que termina así el caso de uso.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El caso de uso se inicia cuando el actor Jefe de Proyecto realiza un levantamiento de la información del Servidor	2. El Especialista de la DT recibe la planilla con los datos.
	3. Verifica que todos los datos de la planilla estén correctos.
	4. Envía los datos al Especialista de Redes
	5. El especialista de Redes recibe la información y termina el caso de uso.
Prioridad:	Crítico

Tabla #2.3: Descripción textual del Caso de Uso: Gestionar Servidores

2.4 Modelo de Objetos

Figura #2.3 Modelo de Objetos del Caso de Uso del Negocio: Gestionar Servidor

2.5 Requisitos del Sistema

Desde ahora, ya conocido los conceptos asociados al objeto de estudio del problema se comenzará a modelar el sistema que se va construir, y de paso se analizará qué debe hacer el sistema para que se cumplan los objetivos planteados al inicio de este trabajo. Para lo mismo se identifican los requisitos Funcionales (RF) y No Funcionales (RNF), y modelan los RF en representaciones de Casos de Uso del sistema. De acuerdo con los objetivos planteados el sistema debe ser capaz de:

2.5.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, por tal motivo a continuación se presenta un conjunto de ellos:

RF1. El sistema debe ser capaz de: Autenticar Usuario.

RF2. El sistema debe ser capaz de: Gestionar Usuario.

- Insertar usuario.
- Eliminar Usuario.
- Buscar Usuario.
- Modificar Usuario

RF3. El sistema debe ser capaz de: Gestionar Infraestructura.

- Insertar Infraestructura.
- Eliminar Infraestructura.
- Modificar Infraestructura.
- Mostrar cantidad de Infraestructuras de la UCI.

RF4. El sistema debe ser capaz de: Gestionar Área Productiva.

- Insertar Área Productiva.
- Eliminar Área Productiva.
- Modificar Área Productiva.
- Mostrar cantidad de Áreas Productivas de la UCI.

RF5. El sistema debe ser capaz de: Gestionar Centro Productivo.

- Insertar Centro Productivo.
- Eliminar Centro Productivo.

- Modificar Centro Productivo.
- Mostrar cantidad de Centros Productivos de la UCI.

RF6. El sistema debe ser capaz de: Gestionar Proyecto.

- Insertar Proyectos Productivos.
- Eliminar Proyectos Productivos.
- Modificar Proyectos Productivos.
- Mostrar cantidad de Proyectos Productivos UCI.

RF7. El sistema debe ser capaz de: Gestionar Laboratorio

- Insertar Laboratorios.
- Eliminar Laboratorios.
- Modificar Laboratorios.
- Mostrar cantidad de Laboratorios de la UCI.

RF8. El sistema debe ser capaz de: Gestionar Servidor

- Insertar los datos del Servidor.
- Insertar los servicios que brinda el Servidor.
- Insertar los servicios que se le prestará al servidor.
- Modificar los datos del Servidor.
- Eliminar los datos del Servidor.
- Mostrar cantidad de servidores dedicados a la producción en la UCI.

RF9. El sistema debe ser capaz de: Gestionar VLAN

- Insertar VLAN.
- Eliminar VLAN.
- Modificar VLAN.
- Mostrar cantidad de VLANs en la UCI.

RF10.El sistema debe ser capaz de: Exportar Reportes.

- Exportar los reportes generados a un formato PDF.

RF11. El sistema debe ser capaz de: Generar Reportes.

- Generar reportes gráficos de los datos de los servidores existentes en la UCI.

2.5.2 Requisitos No Funcionales

Para que el sistema que se propone logre ser óptimo y eficaz en el campo de acción en que se utilice, no solo es importante definir las capacidades que debe cumplir sino también aquellas cualidades que lo harán más atractivo al cliente, usable, rápido y confiable.

A continuación se expondrán los requisitos no funcionales que el sistema propuesto debe cumplir.

2.5.2.1 Usabilidad

RNF1. El sistema debe estar disponible el mayor tiempo posible.

RNF2. El sistema debe ser accesible desde todos los puntos donde exista una máquina conectada a la red.

RNF3. El sistema debe tener una interfaz que le sea familiar al usuario para aprovechar sus conocimientos en el manejo de herramientas de software

RNF4. El sistema debe tener una interfaz de fácil aprendizaje para que usuarios inexpertos puedan familiarizarse lo más pronto posible y le sea cómodo el manejo del software.

RNF5. El sistema debe diferenciar las interfaces gráficas y opciones para los usuarios que accedan al sistema con diferentes roles.

2.5.2.2 Confiabilidad y Seguridad

RNF6. El sistema debe tener la capacidad de identificar con certeza a los diversos usuarios o entidades que interactúan con él.

RNF7. El sistema debe tener la capacidad de darle seguridad al usuario, por lo que las informaciones solo serán vistas por quien esté capacitado para esto.

RNF8. El tiempo entre fallos debe ser breve o cero, haciendo lo posible para que esto no ocurra.

RNF9. La base de datos debe estar fraccionada sobre varios esquemas, dividiendo así de una forma lógica las funcionalidades, evitando así la pérdida total de la información en caso de algún accidente o ataque.

RNF10. Permitir autenticarse a través de un Servidor de Dominio.

2.5.2.3 Rendimiento

RNF11. El sistema debe ser capaz de mantener el mismo rendimiento y estabilidad a medida que aumenta la cantidad de datos a gestionar.

RNF12. El Sistema debe ser capaz de correr al ser montado en una misma PC todos sus componentes tales como Gestor de Bases de Datos, Aplicación Web, etc.

2.5.2.4 Soportabilidad y Operabilidad

RNF13. El software podrá operar en cualquier sistema operativo debido a que se desarrollará en PHP y herramientas de software libre como es el caso de PostgreSQL. En cuanto al soporte debe tenerse en cuenta en el sistema operativo en que se esté trabajando y buscar la tecnología necesaria para las herramientas que requiere el software en el mismo.

2.5.2.5 Mantenimiento y Actualización

RNF14. El ambiente de desarrollo donde se implementará dicho software será: NetBeans.

RNF15. Gestor de Base de Datos: PostgreSQL

RNF16. Lenguaje de Programación: PHP

RNF17. El sistema debe ser construido en un código estándar, cada procedimiento debe estar comentado.

2.5.2.6 Funcionalidad

RNF18. Capacidades de búsqueda con velocidad apropiada.

RNF19. El sistema debe ser Multiplataforma.

2.5.2.7 Desempeño y Escalabilidad

RNF20. El tiempo de respuestas debe ser corto con respuestas rápidas y eficientes.

RNF21. El sistema podrá soportar un gran número de clientes online.

RNF22. El sistema debe tener un rendimiento óptimo debido a que presta servicios a un gran número de usuarios.

2.5.2.8 Requisitos de Software del Sistema

RNF23. Las computadoras clientes deben contar con un navegador.

RNF24. La resolución de pantalla debe ser de 800 x 600 px o mayor.

RNF25. La computadora servidor debe tener instalado el SGBD PostgresSQL 8.2.

RNF26. La computadora servidor debe tener instalado el Servidor Apache 2.2

RNF27. La computadora servidor debe tener instalado el Framework Symfony y PHP 5

RNF28. El software del sistema debe ser operado por personas debidamente preparadas. Además debe tener revisiones y mantenimientos en el tiempo realizadas por personal capacitado.

2.5.2.9 Requisitos de Hardware del Sistema

RNF29. Las computadoras clientes deben estar conectadas en red.

RNF30. El servidor debe tener un microprocesador con frecuencia superior a 2GHz, memoria RAM mayor 1GB y Disco duro mayor de 40GB.

RNF31. El servidor debe tener tarjeta de red Fast Ethernet con velocidad de 100Mbps o superior

2.6 Propuesta de Solución

Haciendo uso de las habilidades y facilidades que brinda UML, se capturarán los requisitos funcionales (RF) del sistema y así se representará a través de un Diagrama de Casos de Uso. Para lo mismo se tendrá en cuenta lo definido anteriormente y se definirán cuáles serán los actores que van a interactuar con el sistema, y a la vez los Casos de Uso que van a representar las diferentes funcionalidades.

Un Caso de Uso es una operación o tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso. Por otra parte un Actor no es como tal parte del sistema, sino que es un Rol de un usuario, donde este puede intercambiar información y representa a un ser humano, software o a una máquina que interactúa con el sistema. En este caso interactúan cuatro actores.

2.6.1 Actores del Sistema

Actor	Descripción
Especialista de la DT	Es el actor encargado de configurar los perfiles de los usuarios del sistema y gestionar la información de infraestructuras, áreas productivas, centros productivos y laboratorios.
Especialista de Redes	Es el actor encargado de Interactuar con el sistema con el objetivo de gestionar las VLANs.
Jefe de proyecto	Es el actor encargado de Interactuar con el sistema con el objetivo de gestionar los datos referentes a su proyecto productivo y los servidores que hay en este.
Usuario	Es el actor encargado de generalizar el acceso dentro

	del sistema y debe autenticarse para entrar al sistema.

Tabla #2.4: Descripción de los Actores del sistema a automatizar

2.6.2 Diagrama de Casos de Uso del Sistema (DCUS)

Un DCUS representa gráficamente a los procesos y su interacción con los actores, permite comprender qué actor interviene en qué proceso.

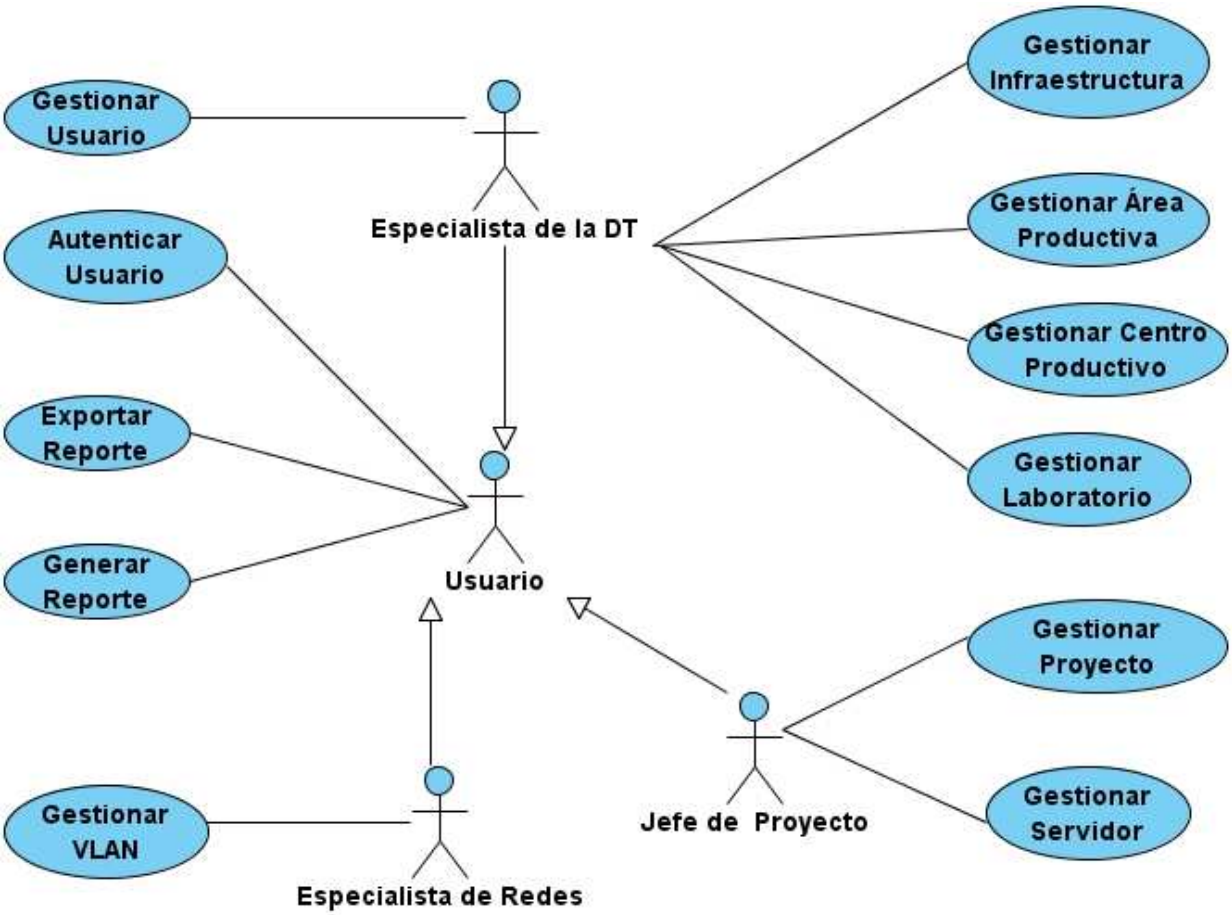


Figura #2.4: Diagrama de Casos de Uso del Sistema

2.7 Descripción de Casos de Uso del Sistema

Para una mayor comprensión de los casos de uso se explicara en detalle el flujo que describen, en el expediente de proyecto propuesto por la universidad en el Modelo del Sistema.

2.8 Conclusiones Parciales

En este capítulo a partir del análisis de los procesos de negocio, se identificaron actores, entidades, trabajadores y casos de uso del negocio. También se analizaron las funcionalidades del sistema a través de los requerimientos funcionales y a partir de ellos se determinaron los casos de uso del sistema, que son los que guiarán el proceso de desarrollo hasta la implementación del sistema.

Capítulo 3: Construcción de la Solución

3.1 Introducción

El análisis de un sistema se centra en la investigación del problema y no en la manera de definir una solución, mientras que el diseño pone de relieve una solución lógica: cómo el sistema satisface los requerimientos funcionales, requerimientos de calidad y las restricciones, es decir, en esta parte del proceso de desarrollo del software se decide cómo se va a llevar a cabo el mismo.

En este capítulo se realiza el análisis, diseño, implementación y pruebas de la propuesta de solución, se presentan los diagramas de clases del análisis y del diseño de los diferentes casos de usos definidos en el capítulo anterior, además de los diagramas de interacción correspondientes a cada uno de ellos, se muestra el modelo de datos, se realiza el modelo de despliegue donde se representan los nodos en los que se distribuye la aplicación y el de componentes para una mejor descripción de la solución propuesta. Finalmente se realiza el modelo de pruebas para verificar que el software satisface los requisitos establecidos por los clientes y futuros usuarios del mismo.

3.2 Modelo de Análisis

En la construcción del Modelo de Análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas, es el resultado de la actividad de analizar los casos de uso. Entre las principales ventajas que aporta al proceso de desarrollo de un software es que suaviza la transición al diseño, sirve para tener una visión general de la propuesta de sistema, y entre otras cosas apoya el cambio a otra plataforma de programación puesto que no está ligado a un lenguaje en particular.

El objetivo principal de este flujo son los diagramas de clases de análisis, los cuales muestran qué clases participan en las realizaciones de los distintos casos de usos, constituyendo la base sobre la cual debe realizarse el Modelo de Diseño.

3.3 Diagramas de Clases del Análisis

Estos diagramas se encuentran en el expediente de proyecto propuesto por la universidad en el Modelo de Análisis

3.4 Diagramas de Interacción del Análisis

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si consideramos el “interior” del sistema,

un objeto de interfaz recibirá este mensaje del actor. El objeto de interfaz enviará a su vez un mensaje a algún otro objeto, y de esa forma los objetos implicados interactuarán para llevar a cabo el caso de uso. (26)

Los diagramas de interacción están constituidos por dos tipos: los diagramas de Colaboración y los diagramas de Secuencia. Ambos expresan información Similar, pero en una forma diferente.

3.4.1 Diagramas de Secuencia

Muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones.

3.4.2 Diagramas de Colaboración

Muestran las interacciones entre un conjunto de objetos, ordenadas según el Tiempo en que tienen lugar. Representa una forma de indicar el período durante el que un objeto está desarrollando una acción directamente o a través de un Procedimiento.
Los diagramas de interacción del Análisis del sistema quedan reflejados en el expediente de proyecto propuesto por la Universidad en el Modelo de Análisis.

3.5 Modelo de Diseño

En la fase de diseño se modela el sistema de manera que soporte todos los requisitos, tanto funcionales como no funcionales, creándose así una entrada apropiada para las actividades de implementación.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar, constituyendo una entrada principal en la actividad de implementación.

Para un mayor entendimiento del diseño de las clases realizado, es conveniente hacer una breve descripción del funcionamiento del framework que se utilizó, pues al estar el diseño completamente ligado al lenguaje de programación como ya se ha dicho, sin conocer cómo funciona al menos de manera general el Symfony es muy difícil poder entender el flujo que describen los diagramas.

Descripción general del funcionamiento del Symfony

Symfony organiza el código fuente en una estructura de tipo proyecto y almacena los archivos del proyecto en una estructura estandarizada de tipo árbol. Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones. Cada aplicación a su vez está formada

por uno o más módulos y un módulo normalmente representa a una página web o a un grupo de páginas con un propósito relacionado, los módulos además almacenan las acciones, que representan cada una de las operaciones que se puede realizar dentro de él.

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. En primer lugar, el controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno de cada tipo. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática, es además el único punto de entrada a la aplicación, carga la configuración y determina la acción a ejecutarse. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita por el usuario.

Las acciones son el corazón de la aplicación, puesto que contienen toda la lógica de la misma. Utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición. Son métodos con el nombre executeNombreAccion de una clase llamada nombreModuloActions que hereda de la clase sfActions y se encuentran agrupadas por módulos. La clase que representa las acciones de un módulo se encuentra en el archivo actions. class.php, en el directorio actions/ del módulo.

Symfony maneja automáticamente las sesiones del usuario y es capaz de almacenar datos de forma persistente entre peticiones. Utiliza el mecanismo de manejo de sesiones incluido en PHP y lo mejora para hacerlo más configurable y más fácil de usar. Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida.

Las clases de la capa del modelo también se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario. Cuando Propel encuentra restricciones de claves foráneas o cuando encuentra datos de tipo fecha, crea métodos especiales para acceder y modificar esos datos.

La abstracción de la base de datos es completamente invisible al programador, ya que la realiza otro componente específico llamado Creole. Así, si se cambia el sistema gestor de bases de

datos en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración.

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

Normalmente se trabaja con las plantillas y con el layout. Estas partes están formadas por código HTML que contiene pequeños trozos de código PHP, que normalmente son llamadas a los diversos helpers disponibles. Los helpers son funciones de PHP que devuelven código HTML y que se utilizan en las plantillas.

El Layout, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en él, o si se mira desde el otro punto de vista, decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado “decorator”. Normalmente se utiliza el layout para mostrar la navegación, el logotipo del sitio, entre otros. Incluso es posible definir más de un layout y decidir en cada acción cuál se va a utilizar.

3.5.1 Diagramas de Clases del Diseño

Los diagramas de clases del diseño quedan reflejados en el expediente de proyecto propuesto por la Universidad en el Modelo de diseño

3.6 Diseño de la Base de Datos

El diseño de la Base de Datos ocupa un lugar fundamental dentro del proceso de desarrollo del software, con el auge de las tecnologías los sistemas informáticos en casi su totalidad se apoyan en bases de datos para el manejo y almacenamiento de la información. Luego de haber definido las clases del diseño que intervienen en los casos de uso, corresponde analizar cuales poseen un carácter permanente y a partir de ello realizar el diagrama de clases persistentes, para luego desarrollar el modelo de datos del sistema.

3.6.1 Diagrama de Clases Persistentes

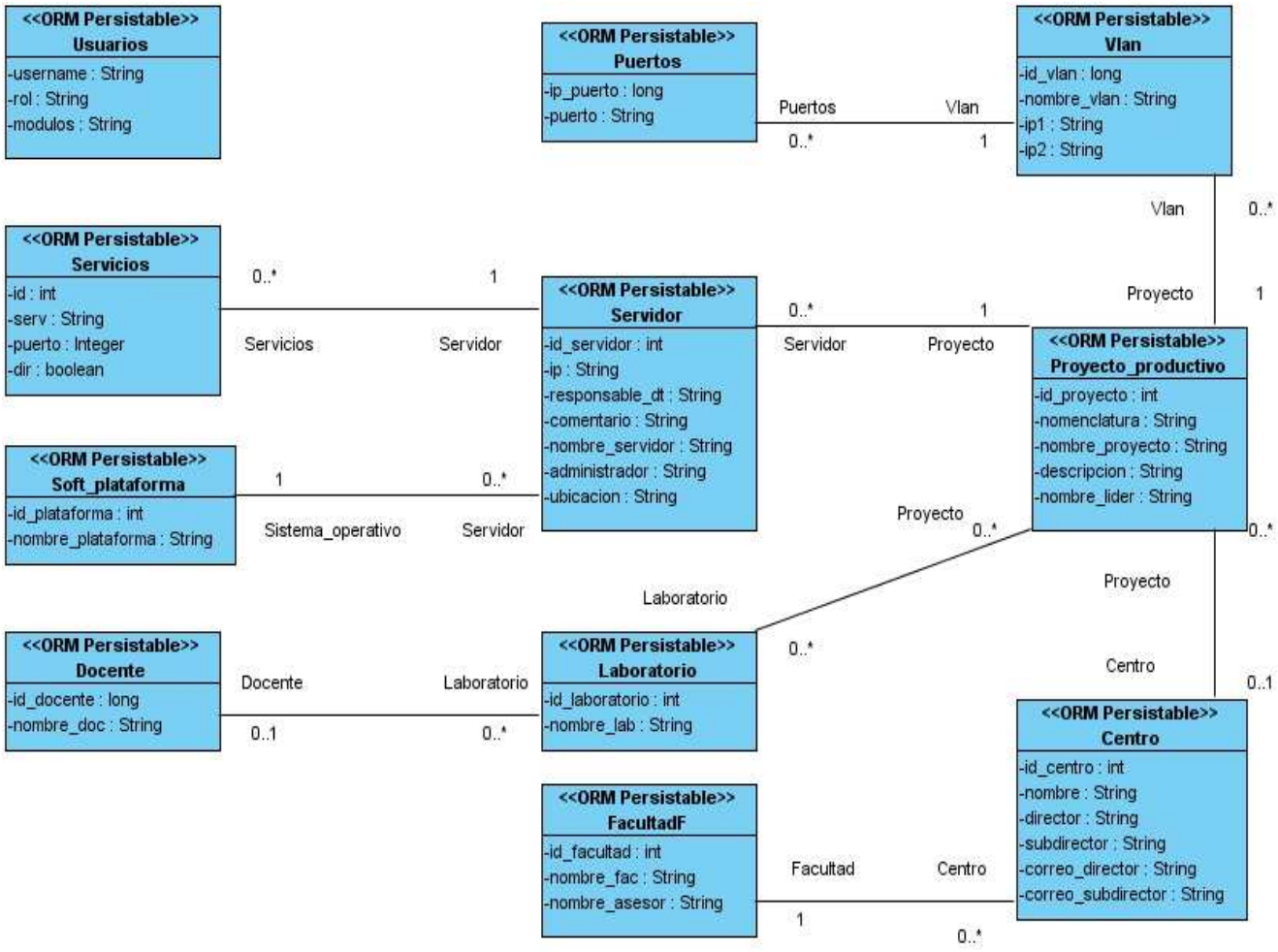


Figura #3.1: Diagrama de Clases Persistentes

3.6.2 Diagrama Entidad Relación

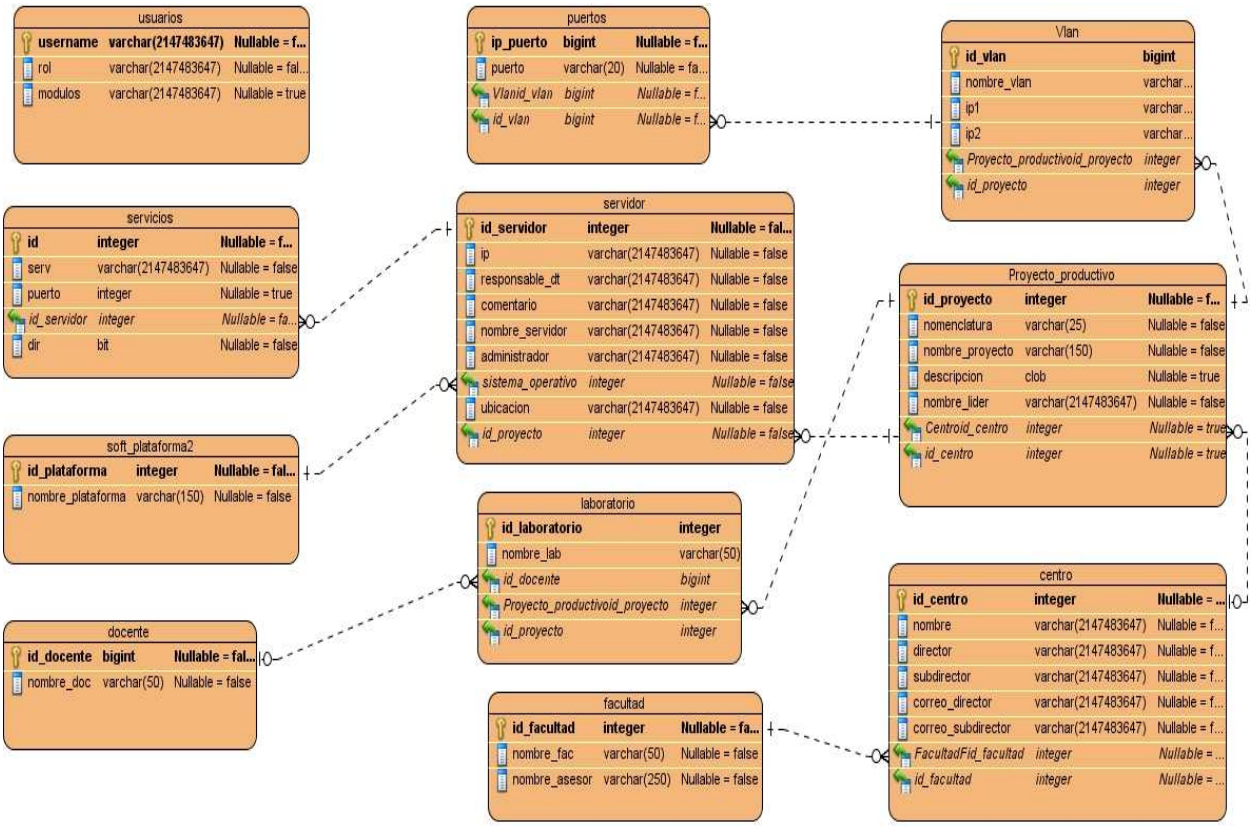


Figura #3.2: Diagrama Entidad Relación

3.7 Diagrama de Despliegue

El modelo de despliegue describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura software y la arquitectura hardware.

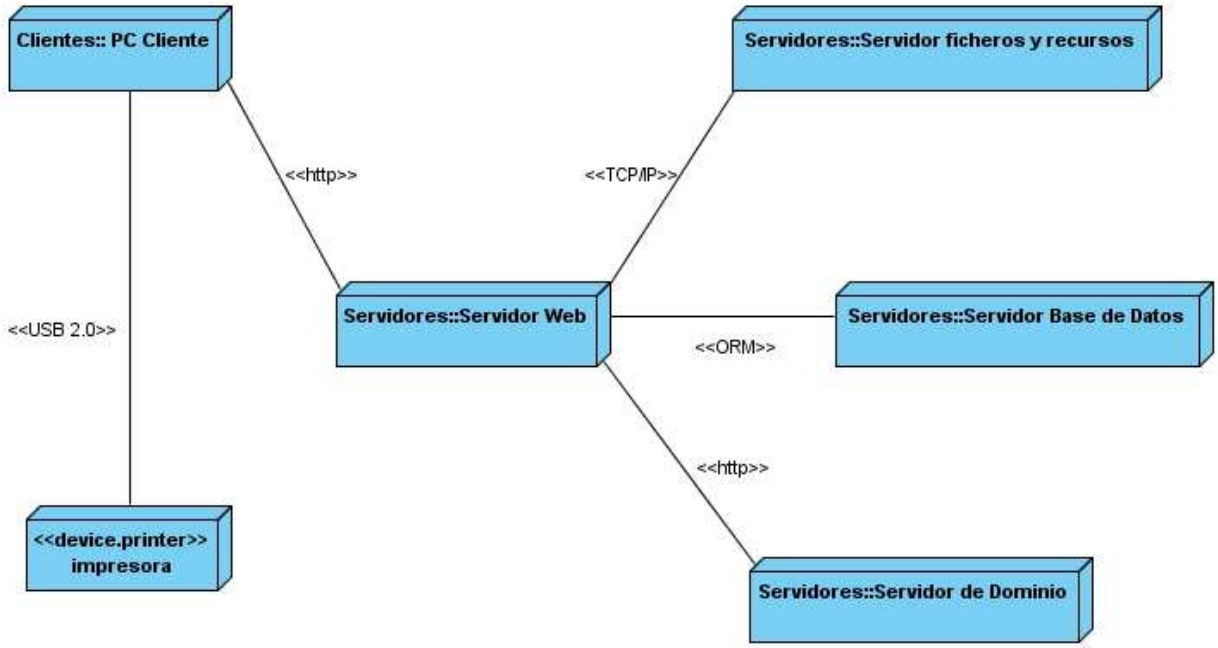


Figura #3.3: Diagrama de Despliegue

3.8 Modelo de Implementación

3.8.1 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, entre otros.

Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas.

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. A continuación se muestra el diagrama de componentes correspondiente al sistema que se propone en la presente investigación.

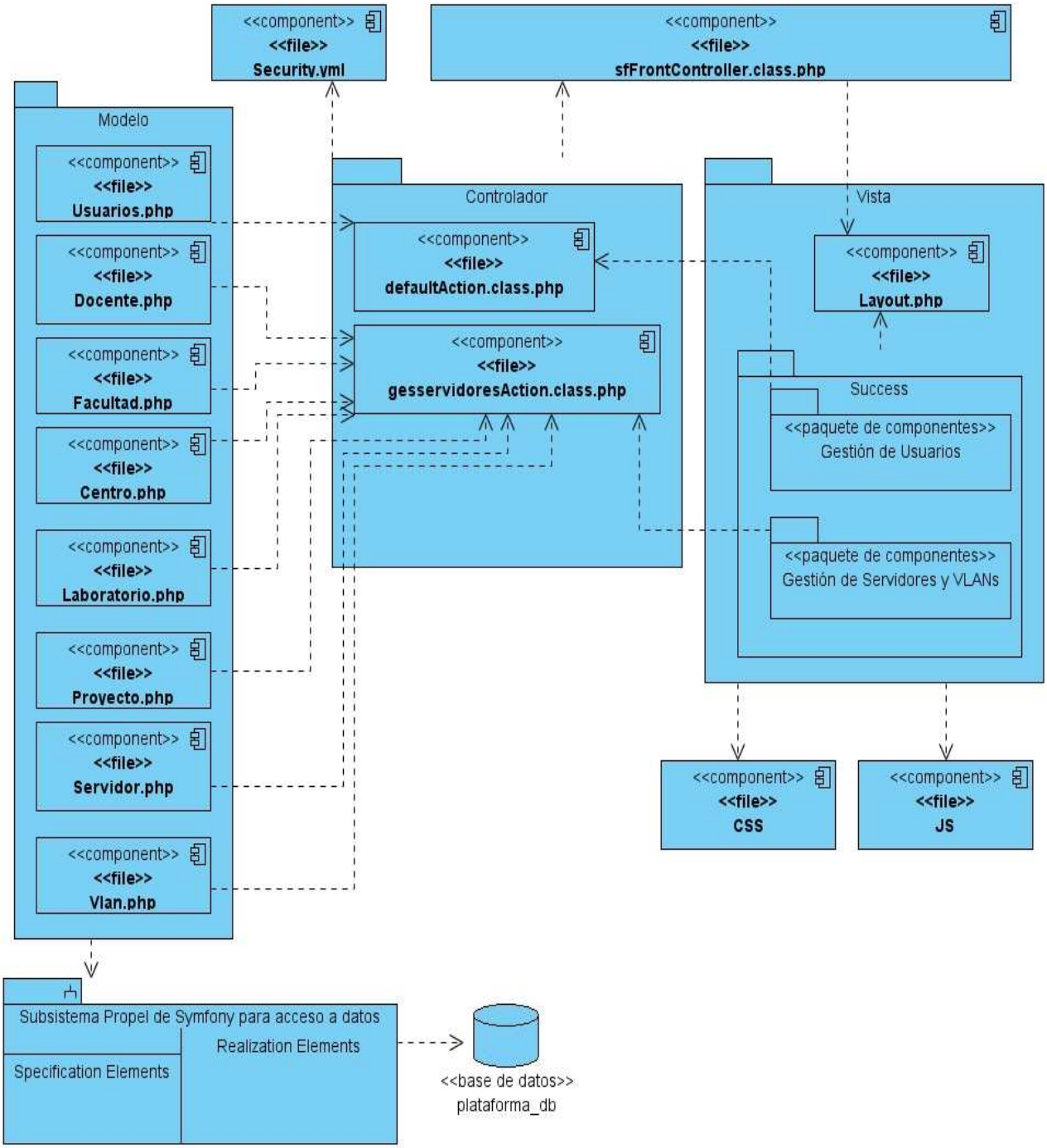


Figura #3.4: Diagrama de Componentes

Para una mayor comprensión del Diagrama de Componentes, se expondrá una vista detallada de los paquetes Gestión de Usuarios y Gestión de Servidores y VLANs.

Vista Detallada: *Paquete Gestión de Usuarios:*

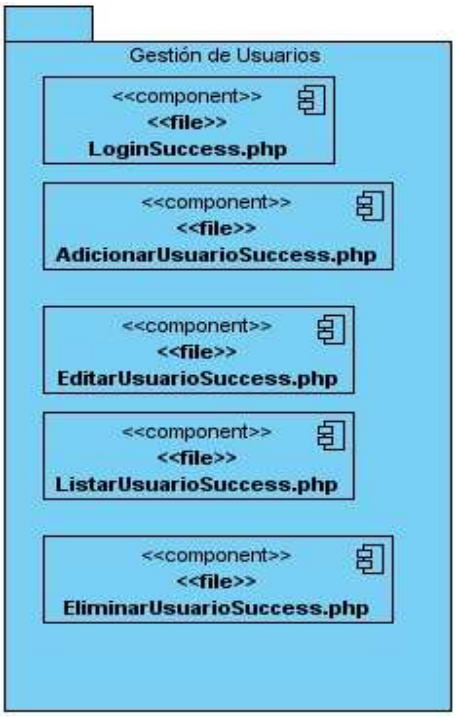


Figura #3.5: Paquete Gestión de Usuarios

Vista Detallada: *Paquete Gestión de Servidores y VLANs:*

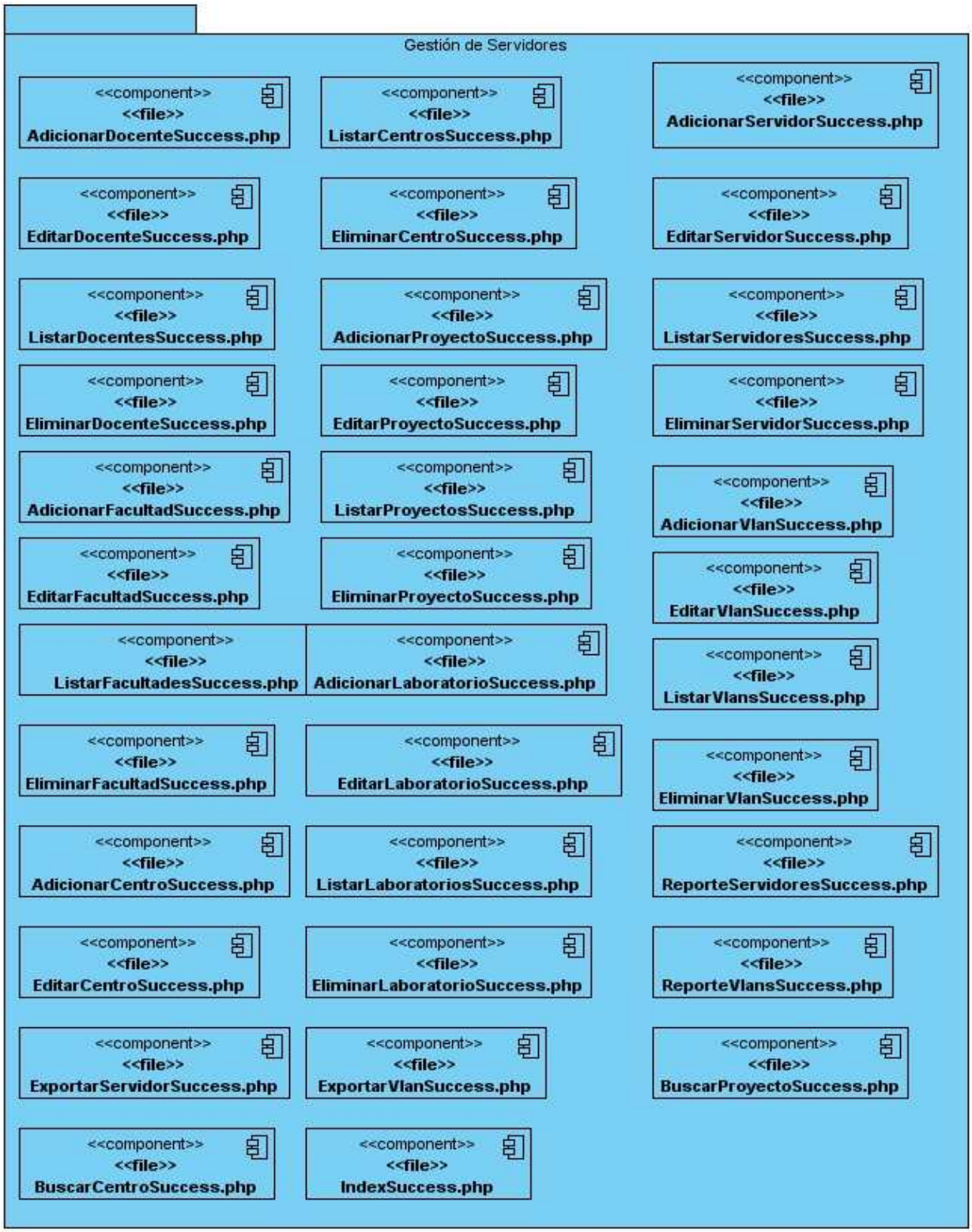


Figura #3.6: Paquete Gestión de Servidores y VLANs

3.9 Modelo de Pruebas

Uno de los objetivos de la fase de pruebas del sistema es verificar software satisface los requisitos establecidos por los clientes y futuros usuarios del mismo. Dentro de esta fase pueden desarrollarse varios tipos distintos de pruebas entre los que se encuentran las pruebas de caja blanca y caja negra. La prueba de caja blanca se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. (25)

3.9.1 Prueba de Caja Negra.

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

- Funciones incorrecta o ausente.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

A continuación se muestra el modelo de pruebas para los casos de uso más relevantes, la realización para los demás casos de uso se encuentra reflejado en el expediente de proyecto propuesto por la universidad en el Modelo de Pruebas.

3.9.1.1 Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Gestionar VLAN	EC 1.1: Adicionar VLAN	En este escenario se comprueba la respuesta del sistema cuando se adiciona una VLAN	1. El Especialista de la DT escoge la opción Adicionar del menú de Gestionar VLAN 2. El sistema le muestra un formulario para la captura de los datos necesarios para la nueva VLAN. 3. El Especialista de la DT llena los datos de la VLAN. 4. El sistema valida los datos. 5. El sistema muestra el listado de VLANs.
	EC 1.2: Adicionar VLAN con los datos incorrectos	En este escenario se comprueba la respuesta del sistema cuando se adiciona una VLAN con los datos	1. El Especialista de la DT escoge la opción Adicionar del menú de Gestionar VLAN. 2. El sistema le muestra un formulario para la captura de los datos necesarios para la nueva VLAN. 3. El Especialista de la DT llena los datos de la VLAN. 4. El sistema valida los datos. 5. El sistema muestra un mensaje informando que los datos no están correctos

	EC 1.3: Modificar VLAN	En este escenario se comprueba la respuesta del sistema cuando se modifica una VLAN	1. El Especialista de la DT escoge la opción Editar de la VLAN que desea en la lista de laboratorios. 2. El sistema le muestra un formulario con los datos de la VLAN. 3. El Especialista de la DT edita los datos de la VLAN. 4. El sistema valida los datos. 5. El sistema muestra el listado de VLANs.
	EC 1.4: Modificar VLAN con los datos incorrectos	En este escenario se comprueba la respuesta del sistema cuando se modifica una VLAN con los datos incorrectos.	1. El Especialista de la DT escoge la opción Editar de la VLAN que desea en la lista de VLANs. 2. El sistema le muestra un formulario con los datos de la VLAN. 3. El Especialista de la DT edita los datos de la VLAN. 4. El sistema valida los datos. 5. El sistema muestra un mensaje informando que los datos no están correctos.
	EC 1.5: Eliminar VLAN	En este escenario se comprueba la respuesta del sistema cuando se elimina una VLAN.	1. El Especialista de la DT escoge la opción Eliminar de la VLAN que desea en la lista de VLANs. 2. El sistema le muestra un mensaje para que el Especialista de la DT confirme que si desea eliminar la VLAN. 3. El Especialista de la DT confirma que desea eliminar la VLAN. 4. El sistema muestra el listado de VLANs.
SC 2: Gestionar Servidor	EC 2.1: Adicionar Servidor	En este escenario se comprueba la respuesta del sistema cuando se adiciona un servidor.	1. El Especialista de la DT escoge la opción Adicionar del menú de Gestionar Servidor 2. El sistema le muestra un formulario para la captura de los datos necesarios para el nuevo servidor. 3. El Especialista de la DT llena los datos del servidor. 4. El sistema valida los datos. 5. El sistema muestra el listado de servidor.

	EC 2.2: Adicionar Servidor con los datos incorrectos	En este escenario se comprueba la respuesta del sistema cuando se adiciona un servidor con los datos incorrectos.	1. El Especialista de la DT escoge la opción Adicionar del menú de Gestionar Servidor. 2. El sistema le muestra un formulario para la captura de los datos necesarios para el nuevo servidor. 3. El Especialista de la DT llena los datos del servidor. 4. El sistema valida los datos. 5. El sistema muestra un mensaje informando que los datos no están correctos
	EC 2.3: Modificar Servidor	En este escenario se comprueba la respuesta del sistema cuando se modifica un servidor.	1. El Especialista de la DT escoge la opción Editar del servidor que desea en la lista de servidores. 2. El sistema le muestra un formulario con los datos del servidor. 3. El Especialista de la DT edita los datos del servidor. 4. El sistema valida los datos. 5. El sistema muestra el listado de servidores.
	EC 2.4: Modificar Servidor con los datos incorrectos	En este escenario se comprueba la respuesta del sistema cuando se modifica un Servidor con los datos incorrectos.	1. El Especialista de la DT escoge la opción Editar del laboratorio que desea en la lista de servidores. 2. El sistema le muestra un formulario con los datos del servidor. 3. El Especialista de la DT edita los datos del servidor. 4. El sistema valida los datos. 5. El sistema muestra un mensaje informando que los datos no están correctos.
	EC 2.5: Eliminar servidor	En este escenario se comprueba la respuesta del sistema cuando se elimina un servidor.	1. El Especialista de la DT escoge la opción Eliminar del servidor que desea en la lista de servidores. 2. El sistema le muestra un mensaje para que el Especialista de la DT confirme que si desea eliminar el servidor. 3. El Especialista de la DT confirma que desea eliminar el servidor. 4. El sistema muestra el listado de servidores.

Tabla. #3.1: Secciones a probar.

3.9.1.2 SC 1: Gestionar VLAN

Valores:

V. Válido

I. Inválido

N/A. No es necesario

Id del escenario	Escenario	Proyecto	Nombre de la VLAN	IP Inicial	IP Final	Puerto	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Adicionar VLAN.	V	V	V	V	N/A	El sistema adiciona una VLAN con los datos introducidos.	Se obtuvo el resultado esperado.
EC 1.2	Adicionar VLAN con los datos incorrectos.	Alguno de los datos toma valor I					El sistema emite un mensaje para que corrija los datos.	Resultado satisfactorio.
EC 1.3	Modificar VLAN.	V	V	V	V	N/A	El sistema modifica la VLAN.	Se obtuvo el resultado esperado.
EC 1.4	Modificar VLAN con los datos incorrectos.	Alguno de los datos toma valor I					El sistema emite un mensaje para que corrija los datos.	Resultado satisfactorio.
EC 1.5	Eliminar VLAN.						El sistema elimina la VLAN.	Se obtuvo el resultado esperado.

Tabla. #3.2: Sección Gestionar VLAN.

3.9.1.3 SC 2: Gestionar Servidor

Variables:

V1. Área Productiva

V2. Centro Productivo

V3. Sistema Operativo

V4. Proyecto

V5. IP del Servidor

V6. Responsable de la DT

V7. Administrador

V8. Servicios Propios

V9. Servicios de la DT

V10. Ubicación Física

V11. Observaciones

Id del escenario	**Escenario**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**	**V10**	**V11**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 2.1	Adicionar Servidor.	V	V	V	V	V	V	V	V	V	V	V	El sistema adiciona un servidor con los datos introducidos.	Se obtuvo el resultado esperado.
EC 2.2	Adicionar Servidor con los datos incorrectos.	Alguno de los datos toma valor I											El sistema emite un mensaje para que corrija los datos.	Resultado satisfactorio.
EC 2.3	Modificar Servidor.	V	V	V	V	V	V	V	V	V	V	V	El sistema modifica el servidor.	Se obtuvo el resultado esperado.

EC 2.4	Modificar Servidor con los datos incorrectos.	Alguno de los datos toma valor I	El sistema emite un mensaje para que corrija los datos.	Resultado satisfactorio.
EC 2.5	Eliminar Servidor.		El sistema elimina el servidor.	Se obtuvo el resultado esperado.

Tabla. #3.3: Sección Gestionar Servidor.

3.9.2 Prueba de Carga.

Las Pruebas de Carga fueron realizadas al sistema para determinar el tiempo de respuesta de la del servidor web ante peticiones concurrentes hechas por los usuarios. En este caso se le realizó la prueba a las secciones “Autenticar Usuario”, “Adicionar Servidor” y “Adicionar VLAN”, dando como resultado el tiempo Promedio de respuesta y el tiempo Máximo que se puede demorar el sistema ante peticiones concurrentes hechas por 100 usuarios. Esta prueba se realizó con la herramienta de carga JMeter que permite llevar a cabo simulaciones sobre cualquier recurso de software. Posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de mayor envergadura simulando la aplicación en tiempo de ejecución. Para la realización de la prueba fue usada una computadora servidor con un microprocesador Intel Core 2 Duo (frecuencia: 2.6GHz), memoria RAM de 4GB y disco duro de 500GB y como cliente fue usada una computadora con microprocesador Intel Dual Core (frecuencia: 2.0 GHz), memoria RAM de 2GB y disco duro de 250GB conectadas a través de tarjetas de red Fast Ethernet con velocidad de 100Mbps.

En la tabla 3.4 se muestran los parámetros evaluados por la herramienta JMeter:

- Página a la que se accede
- Número de muestras
- Tiempo medio en milisegundos
- Mediana en milisegundos
- Línea de 90% en milisegundos
- Tiempo mínimo en milisegundos
- Tiempo máximo en milisegundos
- Porcentaje de Error
- Rendimiento
- Kb/segundo

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Iniciar Sesión	100	15818	16768	27179	2307	30114	0.00%	29.7/min	5463.8
Adicionar Servidor	100	17370	17362	24383	6084	29328	0.00%	24.1/min	2667.5
Adicionar Vlan	100	12145	11996	20389	1700	27862	0.00%	27.8/min	13024.3
TOTAL	300	15111	15351	22994	1700	30114	0.00%	1.2/sec	17832.1

Tabla. #3.4: Parámetros de rendimiento evaluados por la herramienta JMeter

A partir de los valores obtenidos en la tabla se llega a la conclusión que la aplicación tuvo el comportamiento esperado en las condiciones en que se realizó la prueba. No hubo porcentaje de errores y se obtuvieron buenos tiempos y rendimiento para 100 peticiones concurrentes utilizando los equipos descritos anteriormente.

La gráfica mostrada en la figura 3.7 muestra el tiempo medio para las secciones a las que se les realizó la prueba, a partir de los datos obtenidos en la tabla anterior.

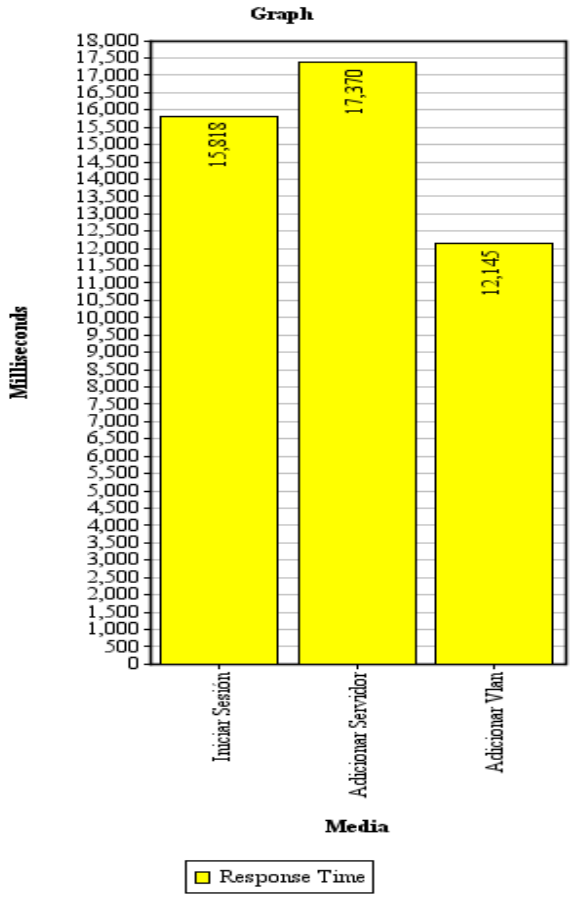


Figura #3.7: Gráfica del tiempo medio para las secciones a las que se les realizó la prueba

3.10 Conclusiones Parciales

Con el desarrollo de este capítulo han quedado expuestos los principales artefactos de los flujos de trabajo: Análisis y Diseño, Implementación y Prueba. Quedaron evidenciadas las características principales del sistema desarrollado como solución al problema científico planteado mediante la representación de sus funcionalidades en el diagrama de Clases del Diseño y el modelo de componentes. Se ha arribado a la conclusión, mediante las pruebas hechas a la aplicación, que cumple de manera satisfactoria con las funcionalidades identificadas durante el proceso de desarrollo del software llevado a cabo en como parte de la presente investigación. Los patrones de diseño utilizados ayudaron eficientemente y con resultados positivos, a lograr un adecuado rendimiento del sistema desarrollado.

Conclusiones Generales

Con la realización del presente trabajo se arriba a las siguientes conclusiones.

- Fueron analizados los sistemas de gestión de información.
- Las investigaciones realizadas posibilitaron comprender el entorno del negocio a informatizar y recopilar la información necesaria para identificar los requisitos.
- Las herramientas, metodología de desarrollo y lenguajes utilizados, brindaron el soporte necesario para lograr un producto con los requerimientos deseados, además de proporcionarle al mismo calidad y rendimiento.
- De acuerdo a las condiciones en que se desarrolló este subsistema, el cual sigue algunos requerimientos establecidos por los sistemas ya implementados en el SIGES-DT, se seleccionó la plataforma de desarrollo, apoyándose además en un estudio breve de plataformas existentes.
- Los resultados obtenidos a partir de los análisis del diseño y la implementación del sistema son positivos, tomando como argumento la aplicación de pruebas de caja negra y carga que permitieron evaluar el nivel de calidad que presenta el software.

Recomendaciones

Cumplido el principal objetivo, la realización del análisis y diseño e implementación del subsistema de gestión de servidores y VLAN para la producción en la Universidad de las Ciencias Informáticas, se recomienda:

- Que este material sea tomado como consulta por el personal (dígase técnicos o profesionales) que vayan a desarrollar un sistema similar.
- Documentar el framework empleado para facilitar el uso de otros desarrolladores.
- Continuar desarrollando el sistema con el objetivo de hacer su uso más flexible, posibilitando su aplicación en otras instituciones del país donde existan infraestructura productiva de software.
- Integrar finalmente el módulo Gestión de Servidores y VLANs a los restantes módulos del Sistema de Gestión de Servicios de la Dirección Técnica para corroborar su buen funcionamiento e integración total.

Bibliografía Referenciada

1. masadelante.com. [En línea] http://www.masadelante.com/faqs/servidor.

2. LORENZO, servidor. [En línea] http://www.lorenzoservidor.com.ar/info01/diccio-s-v.htm.

3. Kioskea.net. [En línea] http://es.kioskea.net/contents/internet/vlan.php3.

4. **Blanco, Andrés Berdasco.** 2000.

5. **Sotolongo Aguilar, Gilberto.** *Derroteros de la gestión de información y documentación en las organizaciones.* 1992.

6. **Fernández-Molina.** *La responsabilidad de los profesionales de la documentación en la prestación de servicios de información.* 1995.

7. **Kazman, Rick; Klein, Mark; Clements, Paul;.** Software Engineering Institute| Carnegie Mellon. [En línea] 2009. http://www.sei.cmu.edu/publications/documents.

8. **Fielding, Roy Thomas .** Architectural Styles and the Design of Network-based Software Architectures. [En línea] 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. 08.

9. **Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael;.** *Pattern Oriented Software Architecture.* 1996. Vol. vol. 1.

10. **Taylor, Richard; Medvidovic, Nenad; Anderson, Kenneth; Whitehead, James; Robbins, Jason; Nies, Kari; Oreizy, Peyman; Dubrow, Deborah;.** *A Component- and Message-Based Architectural Style for GUI Software.* Seattle : s.n., 1995. págs. 295-304.

11. Microsoft ASP.net. [En línea] http://www.asp.net/.

12. adrformacion.com. [En línea] 2004. http://www.adrformacion.com/cursos/php/leccion1/tutorial1.html#1_2.

13. **Rodas Hinostroza, Raul;.** LinuxCentro.net. [En línea] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

14. symfony.es. [En línea] http://www.symfony.es/que-es-symfony/.

15. Mestrosdelweb.com. [En línea] http://www.maestrosdelweb.com/editorial/zendstudio/.

16. **Jacobson, Booch, Ivar y Rumbaugh, Grady.** *El proceso Unificado de desarrollo de software.* 16.

17. **Pressman, Roger.** *Ingeniería de Software: Un enfoque Práctico.*

18. Metodologia XP Vs. Metodologia Rup. [En línea] http://metodologiaxpvsmetodologiarup.blogspot.com/.

19. **Molpeceres, Alberto;.** *Procesos de desarrollo: RUP, XP y FDD.* 2003.

20. **Letelier Penadés, Patricio.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* 2006.

21. **Mendoza Sanchez ,María A.** *Metodologías De Desarrollo De Software.* 2004.

22. Clikear.com. [En línea] http://www.clikear.com/manuales/uml/introduccion.aspx.

23. freedownloadmanager.org. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/..

24. IBM. [En línea] http://www-01.ibm.com/software/awdtools/developer/rose/.

25. MODELOS DE PRUEBAS PARA PRUEBAS DEL SISTEMA. [En línea] http://www.lsi.us.es/~javierj/publications/MDA14.pdf.

26. **RUMBAUHG, J.; I. JACOBSON.** *El Lenguaje Unificado de Modelado. Manual de referencia.*

Bibliografía Consultada

1. **Ciudad Ricardo, Febe Angel.** *¿CÓMO CONFECCIONAR UN ESTADO DEL ARTE?* Ciudad de La Habana : s.n., 2008.

2. **Hernández León, Rolando Alfredo y Coello González, Sayda.** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de La Habana : EDUNIV, 2002. 959-16-0343-6.

3. Aja Quiroga, Lourdes. **Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones.** **http://bvs.sld.cu/revistas/aci/vol10_5_02/aci04502.htm,2002**

4. Potencier Fabien, Zaninotto François**. Symfony, la guía definitiva.** **http://www.librosweb.es/symfony_1_0/**

Glosario de Términos

“A”

ASP: Active Server Pages (Son archivos que se ejecutan en el servidor cuyo contenido puede ser tanto HTML como código de script)

“C”

CSS: Cascading Style Sheets (Hojas de estilo en cascada).

CGI: (Common Gateway Interface).

“D”

DT: Dirección Técnica: equipo de profesionales que se encargan de certificar los proyectos productivos de la UCI en cuanto a los Entornos de Desarrollo y Arquitectura de Software, así como en Arquitectura y Estándares de Información.

“F”

Framework: Es una estructura de software que está formado por diversos componentes personalizables que se pueden configurar y adicionar elementos para el desarrollo de una aplicación.

“H”

HTML: HyperText Markup Language (lenguaje de marcas de hipertexto).Es el lenguaje que define el formato estándar de los documentos que circulan en la Web.

Herramientas CASE: (Computer Arded Software Engineering, Ingeniería de Software Asistida por Ordenador): son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

“I”

IP: Infraestructura Productiva

ISS: Include Server Side

IDE: Entorno Integrado de Desarrollo

“M”

MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario.

MVC: Modelo Vista Controlador

“P”

PHP: HypertextPreprocessor (Es un leguaje de alto nivel embebido en páginas **HTML** y ejecutado en el servidor.)

“R”

RUP: Rational Unified Process : en español Proceso Unificado de Desarrollo :es una metodología pesada , que entre sus principales características se encuentran que es guiado por casos de uso, centrado en la arquitectura e iterativo e incremental..

UCI: Universidad de las Ciencias Informáticas

UML: Unified Modeling Language en español, Lenguaje Unificado de Modelado). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

"V"

VLAN: Red de área local virtual

"X"

XP: Extreme Programming, en español Programación Extrema: es una metodología ágil que intenta reducir la complejidad del software por medio de un trabajo orientado al objeto, basado en las relaciones interpersonales y la velocidad de reacción.