FACULTAD REGIONAL DE ARTEMISA

Sistema para la gestión del uso de las estaciones de trabajo del Centro de Desarrollo de Software de la Facultad Regional Mártires de Artemisa.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Leovan Peña Serrano

Tutora: Ing. Yuneisy Barrios Pérez

Cotutor: Lic. Javier G. Calvo Parapar

Artemisa

Julio de 2012

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter no exclusivo.

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

_____ _____

Yuneisy Barrios Pérez Leovan Peña Serrano

"La innovación es lo que distingue a un líder de los demás."

Steve Jobs

AGRADECIMIENTOS

Agradecer de forma especial a Juan Manuel por su colaboración y aporte al desarrollo de este trabajo.

A mi tutora por todo el apoyo recibido, sus críticas, sus recomendaciones y por su confianza en la realización de este trabajo.

A Marta por ser tan quisquillosa conmigo, por estar siempre al tanto de la marcha del trabajo, por todo su interés en verme convertido en ingeniero.

A Ivet, mi hermanita querida, por ser tan especial conmigo, por su preocupación y su entrega en todo momento.

Agradecerle a esta Revolución el haberme dado la posibilidad de formarme como profesional, en especial a su Comandante en Jefe, promotor principal de esta obra, con quien siempre estaré en deuda.

A mis amigos y amigas del eterno grupo 1, que siempre me ayudaron con sus ideas y su apoyo.

A todos aquellos que de una forma u otra contribuyeron a la realización de este trabajo.

DEDICATORIA

A mi padre y mi abuela, que donde quiera que estén, sé que están orgullosos de mi formación y mis resultados.

A mi madre, por su cariño, su comprensión, sus enseñanzas y por haber confiado en mí en todo momento.

A mis hermanos, por su apoyo.

A Ivet y Marta, mis hermanas de la universidad, por estar siempre a mi lado, por ser incondicionales.

A mis amigos Raimer, Yoandy y Lizbety, por todos sus consejos y sus preocupaciones.

A todos aquellos que de una forma u otra han formado parte de esta gran familia durante estos 5 años.

A Fidel por ser el protagonista principal de mi formación profesional.

RESUMEN

La principal misión de la Universidad de Ciencias Informáticas (UCI) es formar profesionales altamente calificados en la rama de la informática, así como la producción de software y soluciones informáticas que permitan avanzar en la informatización de la sociedad cubana, pues en la actualidad la informática juega un papel primordial en el desarrollo económico y social de cualquier país.

La presente investigación tiene como propósito fundamental contribuir a mejorar y perfeccionar el control que se realiza a la producción de software en el Centro de Desarrollo de Software (CDS) de la Facultad Regional Mártires de Artemisa (FRA), pues no existe un mecanismo eficaz que permita obtener toda la información necesaria en el desarrollo de las actividades productivas.

Teniendo en cuenta estos aspectos, la investigación entonces se basa en la implementación de un sistema de gestión para el control del uso de las estaciones de trabajo del CDS de la FRA. La aplicación es capaz de brindar una información detallada y lo más actualizada posible de toda la actividad productiva que se desarrolla en las computadoras del centro, a través del monitoreo constante de las mismas. El sistema permite obtener toda la información lo más detallada posible de la actividad de cada computadora, tales como procesos en ejecución, usuario que lo ejecuta, el tiempo de duración, el número de ip así como la hora de inicio de cada proceso, generando además un reporte lo más detallado posible según la elección que realice el usuario y los criterios de búsqueda establecidos en la aplicación.

Palabras claves:

Sistema de gestión, monitoreo, control.

ÍNDICE

INTRODUCCIÓN

En las últimas décadas el mundo ha estado transitando por la llamada era digital, post-industrial o de la información, que ha posibilitado un desarrollo continuo de las telecomunicaciones, las redes de computadoras y la información, lo que constituye el pilar fundamental de esta nueva revolución en la esfera del conocimiento. La aparición de las redes de computadoras y de la red de redes, conocida mundialmente como internet, amplió considerablemente las posibilidades de los humanos para procesar y conservar la información.

La informatización de cualquier país es un proceso clave en su desarrollo económico y social. El avance de la tecnología incluye a todos los sectores de las distintas ciencias existentes y la industria del software no está exenta a esta evolución. A pesar de su corto tiempo de vida, es visible el potencial adquirido hasta hoy y las perspectivas de cada día crecer más. La rivalidad entre las grandes empresas y transnacionales de la informática aumenta a cada instante, todas con el objetivo común de satisfacer a un cliente, que espera siempre una solución robusta, confiable, rápida y de calidad.

Ante el inminente desarrollo de la informática y su impacto en la sociedad, Cuba implementa también el denominado proceso de informatización de la sociedad cubana, que incluye todas las instituciones del territorio y está centrado fundamentalmente en el desarrollo de la informática y la creación de una infraestructura tecnológica que permita el avance de la isla en este ámbito. La informática se ha introducido en esferas de vital importancia para la sociedad como la salud, la educación y la economía, creando toda una red de instituciones que tienen la misión de preparar el personal necesario para enfrentar este complejo proceso.

Los Joven Club de Computación y Electrónica fueron los primeros centros con un plan de formación dirigido a fomentar el uso de la computación en todas las áreas

del desarrollo económico y social del país. Poco a poco se fueron creando las empresas de software que iniciaron la producción de sistemas informáticos que marcaron el comienzo de la informatización de la isla.

En el año 2002, después de la retirada del Centro de Exploración y Escucha Radioelectrónicas, conocido como "Base Lourdes", base militar y de monitoreo electrónico instalado en el país en 1964, que velaba por el cumplimiento de los acuerdos de desarme nuclear y la reducción de dichas armas, adoptados por Washington y Moscú, nuestro Comandante en Jefe, Fidel Castro Ruz, comenta la idea de convertir el territorio que ocupaba dicha instalación en la Universidad de las Ciencias Informáticas, centro de altos estudios con la misión de formar profesionales comprometidos con la Patria y altamente calificados en la rama de la informática, así como producir software y servicios informáticos a partir de la vinculación estudio trabajo como modelo de formación.

La UCI con la producción de software y soluciones informáticas, contribuye al crecimiento de la industria cubana del software y a la informatización del país. En sus inicios la universidad estuvo integrada por diez facultades, cada una con un perfil único a desarrollar. Posteriormente en marzo de 2005, el Comandante en Jefe anuncia la creación de las facultades regionales de la UCI en tres provincias diferentes del país. Bautizadas con el nombre de mini-uci, estas tres edificaciones se ubicarían en Artemisa, Ciego de Ávila y Granma.

Estas facultades forman parte de la extensión de la UCI por todo el país y desempeñan un papel trascendental en la informatización del territorio en el cual se encuentran ubicadas. En los cinco años que llevan de inauguradas, estas facultades han contribuido a la preparación y formación de los nuevos profesionales de la informática.

En las facultades de la universidad están los CDS que monitorean toda la actividad productiva, así como los diferentes procesos que en el marco investigativo se realizan. Estos centros forman parte de la red nacional de centros de la UCI,

integrada además por los Institutos Politécnicos de Informática y otras instituciones que se dedican a la producción y comercialización de software.

Actualmente suman más de 30 los CDS a lo largo de todo el país y superan la cifra de 200 los proyectos productivos tanto de carácter nacional, de exportación, como de interés para la propia universidad. Por concepto de exportación de servicios, la UCI factura anualmente más de 250 MM USD. Mantiene relaciones de intercambio con centros e instituciones de otros países como Brasil, España, México, Irlanda, Reino Unido, China y Venezuela, lo que posibilita aumentar el potencial de nuestra industria y ganar prestigio a nivel internacional en la fabricación y comercialización de productos informáticos.

Los CDS de las facultades son los encargados de dirigir el ciclo profesional de los estudiantes, quienes a partir del tercer año de la carrera se vinculan a proyectos reales, con la aplicación del nuevo modelo de formación del profesional. Este modelo de formación tiene como objetivo fundamental la integración de las actividades principales de la universidad, la docencia y la producción, en la preparación integral del futuro ingeniero en ciencias informáticas.

El CDS de la FRA está integrado por dos departamentos, Almacenes de Datos y Aplicaciones Informáticas e Integración. Cada departamento está desarrollando varios proyectos, donde se realizan actividades en función de la producción de software, en laboratorios destinados exclusivamente para estas labores. Los proyectos que en él se desarrollan están orientados esencialmente a la informatización de la provincia Artemisa, específicamente la dirección Provincial del Gobierno; para el Ministerio del Comercio Exterior de Cuba (MINCEX); así como enfocadas también a las soluciones de los procesos internos de la facultad que aún no se han informatizado.

La implementación en la FRA del nuevo modelo de formación del profesional, puesto en práctica a partir del segundo semestre del tercer año de la carrera, establece un número de horas a cumplir por cada estudiante en una semana, para

dar cumplimiento al programa de la práctica profesional y además aprovechar al máximo el tiempo en la producción de software.

En muchas ocasiones el horario establecido es violado por los propios estudiantes, no se realizan las actividades orientadas, no se aprovecha al máximo el tiempo de máquina, deficiencia que se observa en la práctica de otras actividades como es el caso de los juegos, las series y las películas, que no tienen relación con la producción de software y que influyen posteriormente en el incumplimiento de los cronogramas de trabajo de los proyectos.

En el CDS de la FRA no se puede llevar un control real del uso de las estaciones de trabajo. Son los tutores quienes realizan el control de las actividades de cada uno de sus estudiantes, reflejando esta información en un documento digital, en texto plano y en ocasiones por vía correo electrónico. Por consiguiente toda la información generada se encuentra dispersa y sin seguridad, teniendo entonces un bajo grado de consistencia de la información que se maneja.

Por otro lado está el profesor designado para el control de la asistencia a los laboratorios en el horario nocturno, método que no es muy eficiente, dado que es conocido con una buena precisión el horario de control del profesor, lo que provoca que no sea muy preciso el control de la asistencia, y aprovechamiento del tiempo de máquina de los estudiantes en las labores productivas. Todo esto provoca que las tareas orientadas por los tutores a los estudiantes no se cumplan, o se realicen con un mínimo de calidad, producto del finalismo, y por consiguiente en muchas ocasiones se tenga que volver a orientar las mismas, provocando un retraso considerable en la entrega inmediata de la información.

La situación problemática descrita con anterioridad conduce al siguiente **problema de la investigación:** ¿Cómo desarrollar el proceso de gestión del uso de las estaciones de trabajo para elevar la integridad y seguridad de la información en el CDS de la FRA?

Para darle solución al problema de investigación se presenta como **objeto de estudio:** los procesos de gestión del uso de las estaciones de trabajo y como **campo de acción:** la automatización de los procesos de gestión del uso de las estaciones de trabajo del CDS.

El **objetivo general** del trabajo es desarrollar un sistema de gestión de la información que contribuya a elevar la **integridad** y **seguridad** de la información en el control del uso de las estaciones de trabajo del CDS de la FRA.

Del objetivo general se derivan los siguientes **objetivos específicos:**

1. Elaborar la Fundamentación Teórica de la investigación.
2. Realizar el análisis y el diseño de la solución de software propuesta para el CDS de la FRA.
3. Implementar la solución propuesta para el CDS de la FRA.
4. Validar mediante pruebas funcionales los resultados obtenidos con la solución.

Teniendo en cuenta estos objetivos específicos las **preguntas científicas** que sustentan la investigación son:

1. ¿Cuáles son los fundamentos teóricos que sustentan el proceso de gestión del uso de las estaciones de trabajo?
2. ¿Cuál es el análisis y diseño de la solución de software propuesta para el CDS de la FRA?
3. ¿Cómo implementar la solución propuesta para el CDS de la FRA?
4. ¿Cuáles serán los resultados obtenidos con la solución mediante pruebas funcionales?

Para dar cumplimiento a los objetivos y a las preguntas científicas se presentan las **tareas de la investigación:**

1. Establecer los fundamentos teórico-metodológicos para el desarrollo de los procesos de gestión del uso de computadoras.
2. Caracterizar el proceso de gestión del uso de las estaciones de trabajo de la

FRA en lo relativo a integridad y seguridad de la información.

3. Análisis y diseño de la solución de acuerdo a los modelos realizados y los requerimientos identificados.
4. Desarrollo del sistema de gestión del uso de las estaciones de trabajo para el CDS de la FRA.
5. Validar el funcionamiento del sistema de gestión mediante las pruebas de aceptación y otros métodos.

Como **posible resultado** del trabajo se tiene un Informe detallado con toda la base teórico-práctico sobre la cual se sustenta la solución propuesta, así como el sistema para la gestión del uso de las estaciones de trabajo del CDS de la FRA.

En el desarrollo de la investigación se hace necesario el uso de los siguientes métodos científicos:

Métodos Teóricos:

Analítico-Sintético: Para el análisis de los documentos y la bibliografía con el objetivo de obtener información sintetizada y detallada relacionada con el objeto de estudio.

Análisis Histórico-Lógico: Este método posibilita el análisis histórico-lógico del proceso de gestión del uso de las estaciones de trabajo en la Universidad de las Ciencias Informáticas.

Modelación: Para modelar mediante diagramas los componentes del objeto de la investigación.

Métodos Empíricos:

Observación: Se emplea para comprobar cómo se realiza el proceso de gestión del uso de las estaciones de trabajo y así obtener un mejor entendimiento de este fenómeno.

El trabajo de diploma está estructurado de la siguiente manera: introducción, tres capítulos, conclusiones, recomendaciones, bibliografía, anexos y glosario de términos.

En el **Capítulo 1 Fundamentación Teórica** se hace un análisis del estado del arte, del objeto de estudio, se investiga acerca de los sistemas informáticos vinculados al campo de acción, se fundamentan las metodologías, tecnologías y herramientas utilizadas para el desarrollo del sistema de gestión. Algunos de los conceptos a tratar son: gestión, monitoreo y control, lo que posibilita una mejor comprensión de la investigación.

En el **Capítulo 2 Características, análisis y diseño del sistema** se define el negocio y se describe la solución propuesta para la situación problemática. Se presentan las características y funcionalidades del sistema a partir de los requisitos funcionales y no funcionales capturados. Además se realiza todo el diseño del sistema.

En el Capítulo 3 Implementación y validación del Sistema se incluye la programación realizada a partir de los requerimientos y los diagramas del diseño elaborados, así como las métricas y pruebas utilizadas para la validación de la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo aparece todo lo referente al estado de un conjunto de conocimientos que proporciona el marco teórico-conceptual referente al dominio de los sistemas de gestión y control de las computadoras en los laboratorios de producción, la descripción de las características fundamentales, estructura, componentes y funciones que cumple el objeto de estudio de interés para el investigador. Las relaciones de causa y efecto que genera la situación problemática, un análisis crítico a las soluciones dadas a problemas presentados en áreas y momentos diferentes, ofreciéndose un resumen de los principales aspectos tratados en los tópicos relacionados en este capítulo.

1.2 Conceptos asociados al dominio del problema

Existen un conjunto de conceptos asociados al problema planteado anteriormente, a los cuales se le dará una explicación en los siguientes acápites. Estos en su conjunto forman el marco teórico de la investigación.

Control

Todo estudio que se encamine a reducir las brechas existentes en los mecanismos de control de cualquier proceso en las organizaciones en Cuba debe tener como antecedentes la Resolución Económica del V Congreso del Partido Comunista de Cuba donde se consigna:"...En las nuevas condiciones que opera la economía..." se precisa, "...el control oportuno y eficaz de la actividad económica es esencial para la dirección a cualquier nivel..."[1]

Son numerosos los autores que han reflexionado sobre el control y gestión de distintos procesos, observándose una evolución hacia formas superiores que garanticen la eficiencia futura de los mismos. Para tener una mayor comprensión de lo planteado anteriormente se invita a dar un breve recorrido por los restantes

[1] Resolución Económica del V Congreso del Partido Comunista de Cuba.

acápites, de forma tal que permita formar un mayor juicio acerca del tema de estudio y darle la solución más apropiada al problema de investigación.

Muchas son las definiciones que han sido consultadas en distintas obras de diferentes autores entre ellas se cita: "control, es el proceso para asegurar que las actividades reales se ajustan a las actividades planificadas."[2]

El industrial francés Henry Fayol consideraba que el control consiste en verificar si todo marcha de acuerdo con el plan adoptado, con las instrucciones emitidas y con los principios establecidos. Tiene como fin señalar las debilidades y errores a fin de rectificarlos e impedir que se produzcan nuevamente.[3]

En otras palabras señala que el control es la medición y la corrección de las actividades de los subordinados para asegurar que los hechos se ajusten a los planes establecidos.

Luego de estos análisis se puede definir al control como la función administrativa que partiendo de las exigencias preestablecidas a la organización, y el conjunto de normas de especificaciones, vela por cumplimiento como medio para el logro de dichas exigencias.

Controlar en el sentido general es el término anglosajón que significa, dominar o conducir en la dirección deseada, asociándose a los vocablos regular, dirigir y verificar. El control entraña los siguientes elementos básicos que no deben ser omitidos para controlar un proceso:

- Establecer estándares de desempeño.
- Medir los resultados presentes.
- Comparar estos resultados con las normas establecidas.
- Tomar medidas correctivas cuando se detecten desviaciones.
- Premiar, felicitar, remunerar y disciplinar.

Proceso

[2] STONER, James; FREEMAN, R y GILBERT, D Administración. Sexta Edición. México, 1996.
[3] Fayol, Henry (1841-1925). 1946. Administración industrial y general. s.l. : El Ateneo, 1946. 950-02-3540-4.

El control surge por necesidad de los procesos de mantenerse y autogenerarse, pero primero se debe definir que es un proceso el cual según Peter Ferdinand Drucker, es el método sistemático para manejar actividades[4]. Otra definición de proceso es: "*Conjunto de actividades relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados*"[5], según lo establece la norma ISO 9000 del 2005.

Las actividades de cualquier organización pueden ser concebidas como integrantes de un proceso determinado. Desde este punto de vista, una organización cualquiera puede ser considerada como un sistema de procesos, más o menos relacionados entre sí.[6]

En otras palabras el proceso no es más que un conjunto de actividades que se relacionan entre sí, dependiendo de una o más entradas las convierte en salidas o resultados.

Gestión

Si se analiza esta última definición que aparece en las normas ISO se hace necesario intervenir con distintas acciones para que se obtengan resultados de salidas positivas y satisfactorias para el proceso, lo cual implica que no se obtienen por sí solo, si no hay que gestionar.

Gestión, proviene de la acepción latina gesti-onis, acción del verbo género que significa acción y efecto de gestionar, o sea, hacer diligencias que conduzcan al logro de un negocio o deseo cualquiera"[7], también se puede observar la gestión "como la actividad coordinada para dirigir y controlar una organización"[8], que puede ser utilizada en el dominio previsto porque se trata de controlar el proceso que las personas realizan en una actividad estructurada.

---

[4] Drucker, Peter Ferdinand (1909-2005). 2003. Gestión del Conocimiento. Bilbao: Deusto S.A. Ediciones, 2003. 9788423420230.
[5] Norma de Calidad ISO 9000/2005.
[6] Valentín Jiménez, José Manuel. 2008. *Gestión Empresarial.* [En línea] 2008. [Citado el: 12 de 02 de 2012.] http://www.gestionempresarial.info/VerItemProducto.asp?Id_Prod_Serv=28&Id_Sec=8.
[7] Llanes Delgado, W. 2004. Fundamentos de la dirección y gestión. La Habana: s.n., 2004.
[8] Norma de Calidad ISO 9000:2000.

La gestión consiste en seleccionar ciertas acciones, partiendo de diversas informaciones las cuales permiten tomar las decisiones más adecuadas.

Herramienta de Gestión

La gestión garantiza la efectiva y eficiente utilización de los recursos materiales, humanos y financieros puestos a disposición de la universidad para llevar a cabo su objetivo básico para el cual fue creada, para ello es necesario desarrollar una herramienta de gestión de control, la cual se define como una herramienta cuyo soporte es un conjunto de informes sistemáticos que permiten a la organización analizar la captación, transformación y utilización de los recursos.[9]

Para elaborar una herramienta de gestión es necesario establecer la relación existente entre las variables cuantitativas y cualitativas que conforman el modelo en cuestión, como resultado, sobre las variables claves elegidas permitirá observar la situación pasada, presente y futura.

- Monitorear las tendencias y cambios generales.
- Tomar decisiones oportunas y deseablemente acertadas.
- Adoptar las medidas correctivas.
- Controlar en el tiempo las principales variables.

La herramienta entonces permitirá analizar el cumplimiento de los objetivos y el desempeño de cada computadora en los laboratorios de producción. La herramienta de control moderna busca las principales vías y medios para alcanzar la eficiencia de los procesos en los laboratorios de producción, renunciando así a la forma tradicional de inspeccionar y corregir la no eficiencia.

La función de control surge como un requisito obligado para evaluar el resultado de las decisiones delegadas por los responsables de los proyectos y esto contribuye a mejorar la cultura y el entorno de gestión caracterizado por estimular y aunar los esfuerzos individuales de los estudiantes que participan en la ejecución del

[9] Hernández Regalado, Rafael. 2007. Las Mipimes en Latinoamerica.1ra. México: s.n., 2007. p18. 978-968-864-482-9.

proyecto. Este último aspecto pocas veces se asocia al control, sin embargo es uno de los aspectos fundamentales que promueven los sistemas modernos de gestión, reflejando el tránsito de personal controlado a involucrado, dando el éxito a la herramienta de gestión.

Monitoreo

Un elemento que se debe tener en cuenta es el proceso de monitoreo, el cual constituye un componente básico para cualquier herramienta de gestión. Monitoreo no es más que el proceso continuo y sistemático mediante el cual se verifica la eficiencia y la eficacia de un proyecto mediante la identificación de sus logros y debilidades y en consecuencia, se recomiendan medidas correctivas para optimizar los resultados esperados del proyecto. Es por tanto, condición para la rectificación o profundización de la ejecución y para asegurar la retroalimentación entre los objetivos, presupuestos teóricos y las lecciones aprendidas a partir de la práctica.[10]

Para que el proceso de monitoreo tenga éxito debe tener un mecanismo de recopilación de datos e información sobre las actividades que se desean monitorear, para ayudar a la toma de decisiones y realizar acciones al respecto. El modelo que se propone permite avanzar del enfoque retrospectivo al proactivo, orientado al futuro, de la poca implicación del estudiante a la alta implicación, del intensivo en mano de obra a la optimización de los recursos, para lograr finalmente salir de la información redundante y tediosa a la automatización de la información y servicios compartidos, consolidando así la imagen de la UCI como líder en la informatización de la sociedad cubana.

1.3 Análisis de otras soluciones existentes

Actualmente en la UCI el proceso de gestión del uso de las computadoras en los laboratorios de producción se realiza con un software implementado en la propia universidad que permite el monitoreo constante de las estaciones de trabajo, pero

[10] Centro Interamericano para el Desarrollo del Conocimiento en la Formación Profesional . 1996. CINTERFOR. Monitoreo y evaluación. [En línea] 1996. [Citado el: 14 de 02 de 2012.] http://temp.oitcinterfor.org/public/spanish/region/ampro/cinterfor/temas/rural/genero/modelo/monitor.htm.

con la limitante que solo funciona en software privativo (Sistema Operativo Windows), lo cual resuelve el problema, pero de forma parcial, pues la universidad se encuentra en el proceso de migración a tecnologías libres y es necesaria una aplicación que permita desarrollar estas mismas funciones en software libre.

Existen diversas aplicaciones que funcionan sobre software libre, que de una forma u otra monitorean la red y brindan información del funcionamiento de la misma. Ejemplo de algunas de ellas son las siguientes:

Nagios: Sistema de monitorización de redes de código abierto, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios de red (SMTP, POP3, HTTP, SNMP), la monitorización de los recursos de sistemas hardware (carga del procesador, uso de los discos, memoria, estado de los puertos), independencia de sistemas operativos, posibilidad de monitorización remota mediante túneles SSL cifrados o SSH, y la posibilidad de programar plugins específicos para nuevos sistemas. Se trata de un software que proporciona una gran versatilidad para consultar prácticamente cualquier parámetro de interés de un sistema, y genera alertas, que pueden ser recibidas por los responsables correspondientes mediante correo electrónico y mensajes SMS, cuando estos parámetros exceden de los márgenes definidos por el administrador de red.

Htop: Es una aplicación que muestra una lista de los procesos que están corriendo, la cual funciona bajo Consola/Terminal. La misma es muy útil en situaciones donde no se puede trabajar bajo entorno gráfico por una caída o cuelgue, permitiendo matar el proceso que está trayendo problemas. Dentro de su interfaz se puede visualizar información detallada de los procesos, uso de CPU, RAM y Swap.

OCS Inventory: Es un Software Open Source que permite realizar un inventario de todas las características de software y hardware de los equipos que se encuentran conectados en la red. Para ello cuenta con un servidor que es el encargado de almacenar y gestionar toda la información, y un agente instalado en cada equipo,

que envía toda la información recolectada al servidor, así como también realiza envió de paquetes a un número determinado de equipos. Estos paquetes pueden contener software a instalarse o una serie de instrucciones que se ejecutaran en el equipo.

Webmin: Webmin es una interfaz web para administrar y configurar un sistema Unix usando cualquier navegador que soporte tablas, formularios y Java para el módulo de administración de archivos. Webmin consta de un simple servidor web y una variedad de scripts Perl y no usa módulos externos. Con Webmin se pueden configurar infinidad de opciones, como cuentas de usuarios, el servidor Apache y DNS, los comandos programados, el arranque y apagado de la computadora, instalación de nuevos paquetes, copia de seguridad del sistema, el cortafuego, gestión de impresora, grabadora y GRUB. Permite además administrar el sistema desde la consola o de forma remota.

Ninguna de estas aplicaciones es capaz de brindar la información que detallada de los procesos en ejecución en un grupo de máquinas interconectadas. Es entonces necesario el desarrollo de un sistema informático con esta funcionalidad y que solucione la problemática existente en el CDS de la FRA.

1.4 Metodologías de desarrollo de software

Una metodología es un conjunto de procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software. Indica paso a paso lo que se debe realizar para obtener los distintos productos parciales y finales en el proceso de desarrollo de un software. Permite especificar las personas que van a participar en el proceso así como el papel o rol que van jugar en el mismo. Las metodologías de software se clasifican de la siguiente forma:

1. **Estructuradas.**

- Orientadas a procesos.
- Orientadas a datos.
- Mixtas.

2. **No estructuradas.**

- Orientadas a objetos.
- Sistemas de tiempo real.

Se puede clasificar además en metodologías pesadas y en metodologías ágiles. Las primeras realizan mayor énfasis en la planificación y control del proyecto, y requieren una extensa documentación. Una de las metodologías pesadas más conocidas y utilizadas es RUP[11], que divide el desarrollo en cuatro fases que definen su ciclo de vida.

Las metodologías ágiles surgen en febrero de 2001 en Estados Unidos y las mismas se encargan de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados. Hacen mucho más importante crear un producto de software que funcione, y no al hecho de escribir mucha documentación. El cliente está en todo momento colaborando en el proyecto, y es más importante la capacidad de respuesta ante un cambio realizado que el seguimiento estricto de un plan. En la actualidad estas metodologías se muy utilizadas por la sencillez y rapidez con que se manejan los proyectos, además de la constante interacción con el usuario, lo que permite un producto mejor elaborado y en menos tiempo.

Programación Extrema (XP)

Es la metodología ágil más utilizada de los procesos ágiles de desarrollo de software, centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define

[11] Rational Unified Process: en idioma español, Proceso Unificado de Rational.

como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

SCRUM

Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos diez años. Es una metodología ágil y flexible para gestionar el desarrollo de software. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

Permite tener un control continuo sobre el estado actual del software. Implica una filosofía de trabajo que no sólo involucra al desarrollador sino también al cliente, dando prioridad a los individuos y las interacciones sobre los procesos y las tareas, prefiriendo el software funcional sobre la excesiva documentación, promocionando la colaboración con el cliente, en lugar de la negociación de contratos y sobre todo teniendo capacidad de respuesta sobre los cambios en lugar de seguir estrictamente una planificación. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

SCRUM-XP (SXP)

Ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo. Está compuesta por la unión de dos metodologías ágiles de desarrollo SCRUM y XP.

Esta metodología consta de 4 fases principales, y de cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

Estas 4 fases son:

1. Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
2. Desarrollo, es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
3. Entrega, puesta en marcha.
4. Mantenimiento, donde se realiza el soporte para el cliente.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo.

Proceso Unificado de Software (RUP)

El proceso unificado conocido como RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto. Este proceso de desarrollo de software lo

conforman el conjunto de actividades necesarias para trasformar los requisitos funcionales de un usuario en un producto de software.

Lo que sustenta el proceso de desarrollo de software son: el proyecto, las personas, el producto y el proceso, existe una estrecha relación entre ellas. Es conocido como las cuatro P en el desarrollo del software. RUP define para cada etapa: el flujo de trabajo, los trabajadores que intervienen, las actividades que realizan y los artefactos que se necesitan o producen. Su meta es asegurar la producción de software con la más alta calidad, que cumpla con las necesidades de los usuarios dentro del cronograma planeado y la inversión prevista.

RUP se divide en 4 fases:

- Inicio: El objetivo de esta etapa es determinar la visión del proyecto.
- Elaboración: El objetivo es determinar la estructura óptima.
- Construcción: El objetivo es obtener la capacidad operacional inicial.
- Transición: El objetivo es obtener la primera versión del proyecto (release).

Además, el Proceso Unificado utiliza el Lenguaje Unificado de Modelado (UML) para expresar gráficamente todos los esquemas de un sistema software y los aspectos más importantes que se definen en esta metodología: es iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura.

1.5 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

Incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo.

Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos, además es el lenguaje de modelado de sistemas de software más utilizado y conocido en la actualidad. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar.

1.6 Sistema Gestor de Base de Datos

Un sistema de gestor de bases de datos (SGBD) se puede definir como un conjunto de programas cuyo objetivo es acceder a una colección de datos muy bien interrelacionadas. Permite la definición de Bases de Datos, la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Los sistemas de gestión de bases de datos, permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma. Las operaciones típicas que debe realizar un SGBD pueden resumirse en aquellas que afectan a la totalidad de los datos (o a todos los registros de un determinado tipo) y las que tienen lugar sobre registros concretos. Las funciones esenciales de un SGBD son las de descripción, manipulación y control.

MySQL

El sistema de base de datos operacional MySQL es hoy en día uno de los más importantes. Cuenta con millones de aplicaciones y aparece en el mundo informático como una de las más utilizadas por usuarios del medio. El programa MySQL se usa como servidor a través del cual pueden conectarse múltiples

usuarios y utilizarlo al mismo tiempo. Las plataformas que utiliza son de variados tipos y entre ellas se puede mencionar LAMP, MAMP, SAMP, BAMP y WAMP (aplicables a Mac, Windows, Linux, BSD, Open Solaris, Perl y Phyton entre otras). (Victoria, 2009).

MySQL es un software de código abierto, licenciado bajo la Licencia Pública General (GNU GPL)[12], aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL.

El lenguaje de programación que utiliza MySQL es Structured Query Language (SQL) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's[13] en gran cantidad de lenguajes de programación (C, C++, Java, PHP, etc.).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

Oracle

[12] La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

[13] Interfaz de programación de aplicaciones (API por sus siglas en inglés) Conjunto de funciones residentes en ficheros que permiten que una aplicación corra bajo determinado sistema operativo.

Es un sistema de gestión de base de datos objeto-relacional (ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation.

Es uno de los gestores de base de datos más confiables, seguros y difundido en el mundo entero, protege del fracaso del servidor, fracaso del sitio, el error humano, y reduce el tiempo fuera de servicio planeado, afianza los datos y habilita la complacencia con la única seguridad fila-nivelada, interviniendo de grano fino, encriptación de los datos transparente y llamada del total de datos. La principal dificultad que lo aparta de la presente investigación es que Oracle no es una herramienta libre.

Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

PostgreSQL

PostgreSQL es un gestor de bases de datos orientadas a objetos muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo o a un mejor nivel que muchos SGBD comerciales.

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. Es un Sistema Gestor de Base de Datos que ha sido desarrollado desde 1977. Comenzó como un proyecto en la Universidad Bekerley de California. Es un Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos, derivado de Postgres, desarrollado en la Universidad de California.

El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

PostgreSQL puede funcionar en múltiples plataformas (en general, en todas las modernas basadas en Unix) y, a partir de la versión 8.0 también en Windows de forma nativa.

Tiene amplias características, como son:

- DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.
- Altamente Extensible: PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.
- Soporte SQL Comprensivo: PostgreSQL soporta la especificación SQL99[14] e incluye características avanzadas tales como las uniones (joins) SQL92.
- Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

1.7 Herramienta CASE[15]

El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las

[14] Open Database Connectivity, es un estándar de acceso a Base de Datos.
[15] CASE es una sigla, que corresponde a las iniciales de: Computer Aided Software Engineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación.

capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.

También se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

Otras definiciones que nos acercan a CASE son las siguientes.

Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas.

Algunos de los componentes de las herramientas CASE permiten:

- Confeccionar la definición de requerimientos de los usuarios.
- Mejorar el diseño de los sistemas.
- Mejorar la eficiencia en la programación (por su generación automática de códigos).
- Otorgar a la administración un mejor soporte en la documentación.

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 con calidad, a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Algunas características de esta herramienta:

- Soporte de UML.
- Ingeniería de ida y vuelta.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Generación de bases de datos. Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos, desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

Rational Rose

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML, cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. Esta herramienta propone la utilización de cuatro tipos de modelos para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. A continuación se muestran algunas de las características que tiene Rational:

- **Desarrollo Iterativo:** Utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis

y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

- **Generador de Código:** Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.
- **Ingeniería Inversa:** Proporciona mecanismos para realizar la denominada Ingeniería Inversa, a partir del código de un programa, se puede obtener su diseño.
- **Trabajo en Grupo:** Permite varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

1.8 Lenguajes de Programación

La Programación es la codificación de las órdenes y datos que permiten la creación de un programa o aplicación.[16]

Desde el punto de vista informático es un proceso mediante el cual se puede crear programas mediante códigos generados y definidos a través de lenguajes de programación. Estos programas a su vez forman parte de otros más grandes conocidos como software. Por supuesto todo está en dependencia del tamaño del sistema que se quiera lograr. El software al final no son más que instrucciones que

[16] WORDREFERENCE.COM. Diccionario de la Lengua Española [Consultada, Marzo 21, 2012]. Disponible en: http://www.wordreference.com/definicion/programacion

va a seguir el hardware de una computadora determinada. De ahí que la programación sea una de las principales áreas en el mundo de la informática.

Un lenguaje de programación es un lenguaje artificial, diseñado para representar expresiones e instrucciones de forma inteligible para las computadoras.

Los lenguajes de programación son, en gran medida, comparables a los lenguajes naturales: sus símbolos básicos constituyen su alfabeto, y con ellos se construye el vocabulario del lenguaje, cuyos elementos se llaman tokens. Estos tokens se combinan de acuerdo con las reglas sintácticas del lenguaje, formando expresiones y sentencias cuyo significado viene dado por la semántica del lenguaje.

Los lenguajes de programación son sin embargo considerablemente más simples que los naturales en su sintaxis y, especialmente, en su semántica. En definitiva un lenguaje de programación es un conjunto predefinido de palabras y símbolos que se utilizan siguiendo unas reglas prefijadas (sintaxis) para expresar algoritmos.

PHP

Lenguaje interpretado, diseñado originalmente para la creación de Página web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

Principales características de PHP 5.

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados *ext's* o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones.
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

Perl

Es un lenguaje de programación diseñado por Larry Wall en 1987. Perl significa Practical Extraction and Report Language, algo así como lenguaje práctico de extracción y de informes. Es un lenguaje interpretado, aunque el intérprete de Perl, como todos los intérpretes modernos, compila los programas antes de ejecutarlos. Perl tiene características de muchos lenguajes de programación (buscando lo mejor de cada uno), pero al que más se parece es al C. Perl es un lenguaje que permite programar de forma muy rápida y es excelente para el tratamiento de cadenas.

Fue creado para simplificar tareas de administración de un sistema Unix, aunque hoy en día se ha convertido en una de las mejores herramientas para creación de scripts o de construcción de sitios Web. Es un lenguaje rápido pese a ser interpretado, multiplataforma y dispone de una gran cantidad de bibliotecas para el desarrollo de casi cualquier tipo de aplicación. Es software libre, no hay que pagarlo.

Dentro de las principales características de Perl se encuentran:

- Existen muy pocas cosas que no se pueden hacer con este lenguaje, por lo que se le puede considerar como un lenguaje que no tiene fronteras. Con Perl se puede programar cualquier necesidad que se tenga, ya que existen librerías y módulos para casi cualquier cosa que se requiera.
- Es rápido de crear, ya que no posee funciones que, aunque sean bastante interesantes, hagan disminuir la velocidad de desarrollo de una aplicación del lenguaje.
- Se puede utilizar en varios entornos, como puede ser Windows 95, OS/2, Linux, entre muchos otros, sin realizar cambios de código, siendo únicamente necesario la introducción del interprete Perl correspondiente a cada sistema operativo.
- Tiene características que soportan una variedad de paradigmas de programación, como la estructural, funcional y la orientada a objetos. Al mismo tiempo, Perl no obliga a seguir ningún paradigma en particular, ni obliga al programador a elegir alguna de ellas.

- Tiene incorporado un poderoso sistema de procesamiento de texto y una enorme colección de módulos disponibles.

HTML

HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla de HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto.[17]

Permite describir la estructura y el contenido en forma de texto, además de complementar el texto con objetos tales como imágenes. Este lenguaje se escribe mediante etiquetas, que aparecen especificadas por corchetes angulares.

Por otra parte, HTML permite incluir scripts (por ejemplo, de Javascript), códigos que pueden modificar el comportamiento de los navegadores web y de otros procesadores de HTML. Es el lenguaje de marcado predominante para la elaboración de páginas web.

JavaScript

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Con JavaScript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

JavaScript es el siguiente paso, después del HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con

[17] Definición. DE. Disponible en: http://definicion.de/html/

rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.[18]

Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, JavaScript pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

CSS

El nombre hojas de estilo en cascada viene del inglés *Cascading Style Sheets*, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.[19]

1.9 Justificación de las tecnologías y herramientas utilizadas

Teniendo en cuenta lo anteriormente expuesto se propone utilizar:

- **Metodología de desarrollo SXP:** Es una metodología ágil creada en la universidad a partir de lo mejor de las metodologías XP y SCRUM, proporcionando un mayor contacto con el cliente, entrega rápida de la información y orientada a proyectos pequeños con requisitos imprecisos y cambiantes.
- **Sistema Gestor de Bases de Datos PostgreSQL:** Es el SGBD de código abierto más avanzado del mundo, es multiplataforma y orientado a objetos. Utiliza control de versionado concurrente y soporte para la implementación

[18] Álvarez, Miguel Ángel. 2001. Una introducción meramente conceptual al potente lenguaje de script del lado del cliente.
[19] Eguíluz Pérez Javier. ¿Qué es CSS? En línea: http://www.librosweb.es/css/capitulo1.html

de consultas complejas. Con este SGBD se podrá mantener una base de datos actualizada, confiable y configurable.

- **Herramienta CASE Visual Paradigm:** Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. Además es libre.
- **Lenguajes de Programación PHP, Perl, HTML, CSS, JavaScript:** Son lenguajes altamente utilizados en el diseño y desarrollo de sitios Web, por su potencial, características y sus facilidades de trabajar con PostgreSQL en caso de Perl y PHP. Permiten el diseño fácil y rápido de estos sitios, con interfaces amigables para el usuario, y con un diseño. Permiten además el despliegue de los sistemas en la mayoría de los servidores y sistemas operativos sin coste alguno.

1.10 Conclusiones Parciales

El capítulo Fundamentación Teórica, ofrece información general de cómo ocurre el proceso descrito en el objeto de estudio, visualizando el posible problema y variables a estudiar. También permite la búsqueda y estudio de la información referente al problema, consulta a expertos y revisión bibliográfica en todas las fuentes disponibles adquiriendo así mayor dominio del problema científico a investigar. El establecimiento de un marco teórico ayuda a prevenir y a minimizar posibles errores que pueden presentarse durante el estudio, al formular este capítulo queda orientado el modo de conducir la investigación.

CAPÍTULO 2: CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

En el presente capítulo se muestra una propuesta de la solución. Se realiza una descripción general de la propuesta y su funcionamiento. Se definen los requerimientos funcionales y los requerimientos no funcionales, descritos mediante las historias de usuarios. Se realiza también el plan de entrega en el cual se indican las historias de usuario que se crearán para cada versión de la aplicación. Además se muestra el plan de iteraciones donde se muestran las historias de usuarios que se realizarán en cada iteración, según la prioridad en el negocio.

2.2 Concepción del Sistema

Se desea desarrollar una aplicación para la gestión del uso de las estaciones de trabajo del Centro de Desarrollo de Software de la Facultad Regional de Artemisa, donde se pueda controlar todo el trabajo que se realiza en cualquier computadora del centro, durante el tiempo que esté encendida.

Propuesta del Sistema

En el presente trabajo se propone un sistema que brinde la posibilidad de monitorear el trabajo que se realiza en las estaciones de trabajo del Centro de Desarrollo de Software de la Facultad Regional de Artemisa. Para ello el sistema define los usuarios por roles: Administrador y Usuario sin privilegios, cada uno con permisos diferentes sobre el sistema en cuestión, lo cual le proporciona al sistema seguridad en la información que maneja. El sistema debe brindar la posibilidad de mostrar las informaciones de monitoreo que se realicen en las computadoras de la facultad. La interfaz visual del sistema debe ser diseñada de forma tal que permita a los usuarios una fácil y rápida interacción con la aplicación.

Planificación del proyecto por roles

Tabla 1. Planificación del proyecto por roles.

Rol	Responsabilidad	Nombre
Líder del Proyecto	Toma las decisiones finales acerca de los estándares y convenciones a seguir durante el proyecto.	Yuneisy Barrios Pérez.
Cliente	Participa en las tareas que involucran la lista de reserva del producto.	CDS de la FRA
Programador	Produce el código y escribe las pruebas unitarias.	Leovan Peña Serrano
Analista	Escribe las historias de usuario y las pruebas funcionales para validar su implementación.	Leovan Peña Serrano
Diseñador	Diseña el sistema y los prototipos de interfaces. Es el máximo responsable del diseño de la metáfora y supervisa el proceso de construcción.	Leovan Peña Serrano
Encargado de Pruebas	Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas periódicamente, y es responsable de las herramientas de soporte para las pruebas.	Leovan Peña Serrano
Arquitecto	Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas.	Leovan Peña Serrano.

Modelo de Dominio

Dentro de las actividades más importantes definidas en la metodología SXP se encuentra generar el Modelo de Dominio. En el mismo se representan las clases conceptuales u objetos del mundo real permitiendo entender el problema en cuestión. Este modelo ayuda a comprender conceptos con los que los usuarios trabajan y deben ser parte de la solución final.

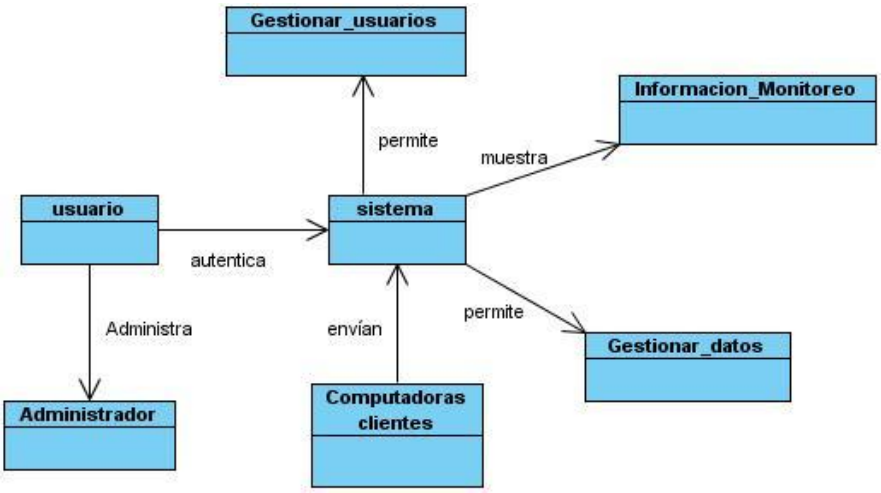


Figura 1. Modelo de Dominio.

Conceptos del Modelo de Dominio

Usuario: Profesor, Tutor o directivo del centro de desarrollo que gestiona toda la información de monitoreo, según el rol que tenga definido en el sistema.

Administrador: Usuario con acceso al sistema y que tiene privilegios administrativos. Puede modificar la información del sistema según las necesidades propias del CDS.

Gestionar usuarios: Permite las operaciones de insertar, modificar, eliminar y mostrar los usuarios con acceso al sistema. Estas funcionalidades solo están disponibles para los usuarios administradores del sistema.

Información monitoreo: Muestra la información que se obtiene del monitoreo de las máquinas clientes. La misma se muestra según los criterios seleccionados por el usuario.

Gestionar datos: Permite insertar, mostrar y eliminar los datos de la BD, que se obtienen del monitoreo de las máquinas clientes. La funcionalidad de eliminar solo está permitida para aquellos usuarios que sean administradores.

Computadoras clientes: Son las máquinas a las cuales se les realiza el monitoreo.

2.3 Listas de Reserva del Producto

Este es el primer artefacto que se genera en la etapa de captura de requisitos, se conforma por el cliente y el analista. En él se recoge en una lista priorizada todo el trabajo a desarrollar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para una iteración.

Esta lista puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto y del cliente. Con la restricción de que sólo puede cambiarse entre iteraciones. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

Tabla 2. Lista de reserva del producto.

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
Alta				
	1	Realizar Petición	3 semanas	Analista

	2	Insertar usuario	1 semana	Analista
	3	Definir niveles de acceso	1 semana	Analista
	4	Modificar usuario	1 semana	Analista
	5	Eliminar usuario	1 semana	Analista
	6	Mostrar usuarios	1 semana	Analista
	7	Insertar datos	2 semanas	Analista
	8	Mostrar datos	1 semana	Analista
	9	Eliminar datos	1 semana	Analista
Media				
Requisitos no funcionales Baja				
Usabilidad	10	El sistema podrá ser usado por cualquier persona con conocimientos básicos de informática.		
Seguridad	11	El sistema tendrá establecido un sistema de restricciones en correspondencia con los permisos establecidos para cada nivel de usuario.		
Confiabilidad y disponibilidad	12	El sistema debe estar disponible todo el tiempo que esté en uso la estación de trabajo.		
Rendimiento	13	La velocidad de procesamiento de la información y el tiempo de respuesta no debe exceder de los 10 segundos.		
Software	14	Tanto las estaciones de trabajo cliente como el servidor de aplicaciones poseerán como Sistema Operativo Ubuntu con		

		una versión igual o superior a la 10.04.		
Hardware	15	Las estaciones de trabajo cliente deberán contar con un procesador con una velocidad de procesamiento superior a 1 GHZ, y los servidores con una velocidad de procesamiento superior a 2 GHz. Ambas estaciones de trabajo deben poseer 1GB de memoria RAM y poseer una tarjeta de red.		

Personas relacionadas con la aplicación

Son aquellas personas que interactúan con el sistema y que obtienen un resultado de los procesos que en él se ejecutan. Estas personas son:

- Profesores.
- Tutores.
- Jefes de Departamentos.
- Junta Directiva del Centro de Desarrollo.

2.4 Historias de usuario

Las historias de usuarios describen las tareas que el sistema debe hacer, estas tareas son definidas según lo que necesite el cliente. Se escriben en un lenguaje natural y con palabras concisas para no exceder su tamaño en unas pocas líneas de texto. Estas van a ser la guía para posteriormente construir las pruebas de aceptación comprobando de esta manera la correcta implementación de las historias de usuario. En la presente investigación fueron identificadas las siguientes historias de usuario:

1. Realizar Petición.
2. Gestionar Usuarios.

3. Definir niveles de acceso.
4. Gestionar Datos.

En el artefacto Historia de usuario están registradas todas las historias de usuarios con sus respectivos prototipos de interfaces.

Tabla 3. Historia de usuario 1.

Historia de Usuario	
Número: HU_1	**Nombre Historia de Usuario:** Realizar Petición.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leovan Peña Serrano	**Iteración Asignada:** 2
Prioridad en Negocio: Alta.	**Puntos Estimados:** 3 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 3 semanas
Descripción: El sistema le solicita a la máquina cliente la información.	
Observaciones: La máquina cliente debe tener instalada la herramienta de monitoreo.	
Prototipo de interface:	

Tabla 4. Historia de usuario 2.

Historia de Usuario	
Número: HU_2	**Nombre Historia de Usuario:** Gestionar Usuarios.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leovan Peña Serrano	**Iteración Asignada:** 2
Prioridad en Negocio: Alta.	**Puntos Estimados:** 4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 4 semanas
Descripción: Permite insertar, modificar, eliminar y mostrar los usuarios de la Base de Datos.	

Observaciones:
Prototipo de interface: (Ver Anexo 1, Figura 1, Figura 2, Figura 3, Figura 4).

Tabla 5. Historia de usuario 3.

Historia de Usuario	
Número: HU_3	**Nombre Historia de Usuario:** Definir niveles de acceso.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leovan Peña Serrano	**Iteración Asignada:** 2
Prioridad en Negocio: Alta.	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1 semana
Descripción: Define el rol de cada usuario que se crea en la base de datos y los permisos que tendrá dentro del sistema.	
Observaciones:	
Prototipo de interface (Ver Anexo 1, Figura 1)	

Tabla 6. Historia de usuario 4.

Historia de Usuario	
Número: HU_4	**Nombre Historia de Usuario:** Gestionar Datos.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leovan Peña Serrano	**Iteración Asignada:** 2
Prioridad en Negocio: Alta.	**Puntos Estimados:** 4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 4 semanas
Descripción: Permite las operaciones de insertar, mostrar y eliminar los datos que están en la Base de Datos.	
Observaciones:	
Prototipo de interface (Ver Anexo 2).	

2.5 Tareas de Ingeniería

Tabla 7. Tarea de ingeniería 1.1.

Tarea de Ingeniería	
Número Tarea: 1.1	**Número Historia de Usuario:** 1
Nombre Tarea: Implementación de la funcionalidad realizar petición.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Fecha de Inicio:14/3/2012	**Fecha Fin:**4/4/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permita al sistema realizar la petición de la información a la máquina cliente.	

Tabla 8. Tarea de Ingeniería 2.1.

Tarea de Ingeniería	
Número Tarea: 2.1	**Número Historia de Usuario:** 2
Nombre Tarea: Implementación de la funcionalidad realizar usuario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/4
Fecha de Inicio:11/1/2012	**Fecha Fin:**18/1/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permita insertar los usuarios en la Base de Datos.	

Tabla 9. Tarea de Ingeniería 2.2.

Tarea de Ingeniería

Número Tarea: 2.2	**Número Historia de Usuario:** 2
Nombre Tarea: Implementación de la funcionalidad modificar datos del usuario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/4
Fecha de Inicio:18/1/2012	**Fecha Fin:**25/1/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permita modificar los datos de los usuarios de la Base de Datos.	

Tabla 10. Tarea de Ingeniería 2.3.

Tarea de Ingeniería	
Número Tarea: 2.3	**Número Historia de Usuario:** 2
Nombre Tarea: Implementación de la funcionalidad eliminar usuario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/4
Fecha de Inicio: 25/1/2012	**Fecha Fin:**1/2/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permita eliminar un usuario de la Base de Datos.	

Tabla 11. Tarea de Ingeniería 2.4.

Tarea de Ingeniería	
Número Tarea: 2.4	**Número Historia de Usuario:** 2
Nombre Tarea: Implementación de la funcionalidad mostrar usuarios.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/4
Fecha de Inicio: 1/2/2012	**Fecha Fin:**8/2/2012

Programador Responsable: Leovan Peña Serrano.
Descripción: Implementar las funcionalidades necesarias que permita mostrar los usuarios existentes en la Base de Datos.

Tabla 12. Tarea de Ingeniería 3.1.

	Tarea de Ingeniería
Número Tarea: 3.1	**Número Historia de Usuario:** 3
Nombre Tarea: Implementación de la funcionalidad definir niveles de acceso.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha de Inicio: 8/2/2012	**Fecha Fin:** 15/2/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permita definir los niveles de acceso de los usuarios en el sistema.	

Tabla 13. Tarea de Ingeniería 4.1.

	Tarea de Ingeniería
Número Tarea: 4.1	**Número Historia de Usuario:** 4
Nombre Tarea: Implementación de la funcionalidad insertar datos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3/4
Fecha de Inicio: 15/2/2012	**Fecha Fin:** 29/2/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permitan insertar los datos obtenidos de las estaciones cliente en la Base de Datos del Sistema.	

Tabla 14. Tarea de Ingeniería 4.2.

Tarea de Ingeniería	
Número Tarea: 4.2	**Número Historia de Usuario:** 4
Nombre Tarea: Implementación de la funcionalidad mostrar datos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/4
Fecha de Inicio: 29/2/2012	**Fecha Fin:** 7/3/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permitan mostrar los datos que se encuentran almacenados en la Base de Datos del Sistema.	

Tabla 15. Tarea de Ingeniería 4.3.

Tarea de Ingeniería	
Número Tarea: 4.3	**Número Historia de Usuario:** 4
Nombre Tarea: Implementación de la funcionalidad eliminar datos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1/4
Fecha de Inicio: 7/3/2012	**Fecha Fin:** 14/3/2012
Programador Responsable: Leovan Peña Serrano.	
Descripción: Implementar las funcionalidades necesarias que permitan eliminar los datos que se encuentran almacenados en la Base de Datos del Sistema.	

2.6 Plan de releases

El cliente y los desarrolladores acuerdan el orden en que deberán implementarse las Historias de Usuario y asociadas a estas las entregas y el resultado de esta fase es un Plan de Entregas o Plan de Releases.

En el plan de releases se definen las iteraciones a realizar, las entregas intermedias y la entrega final. Tiene como entrada la relación de Historias de Usuario definidas previamente. Para colocar una historia en cada iteración se tiene en cuenta la prioridad que definió el cliente para dicha historia.

Tabla 16. Plan de releases.

Release	Descripción de la iteración	Orden la HU a implementar	Duración total.
Iteración 2	En esta iteración se desarrollarán las historias de usuario de prioridad alta y el sistema será capaz de gestionar toda la información que se encuentre en la base de datos, así como la información que se genera en cada una de las estaciones de trabajo.	HU_2 HU_3 HU_4 HU_1	12 semanas

2.7 Diagrama de paquetes

SXP está basada en la metodología XP, que define un término llamado metáfora, la cual según Martin Fowler es una historia compartida que describe como debería funcionar el sistema y define que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema.

Se debe diseñar la solución más simple que pueda funcionar y ser implementado en un momento dado del proyecto; lo cual genera el artefacto conocido como Modelo de Diseño, que a su vez está compuesto por un diagrama de paquetes, el cual expone dicho diseño.

Los diagramas de paquetes describen los elementos físicos del sistema y sus relaciones. Muestran las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados. Además muestran la organización y las dependencias lógicas entre un conjunto de componentes de software, sean éstos componentes de código fuente, librerías, binarios o ejecutables.

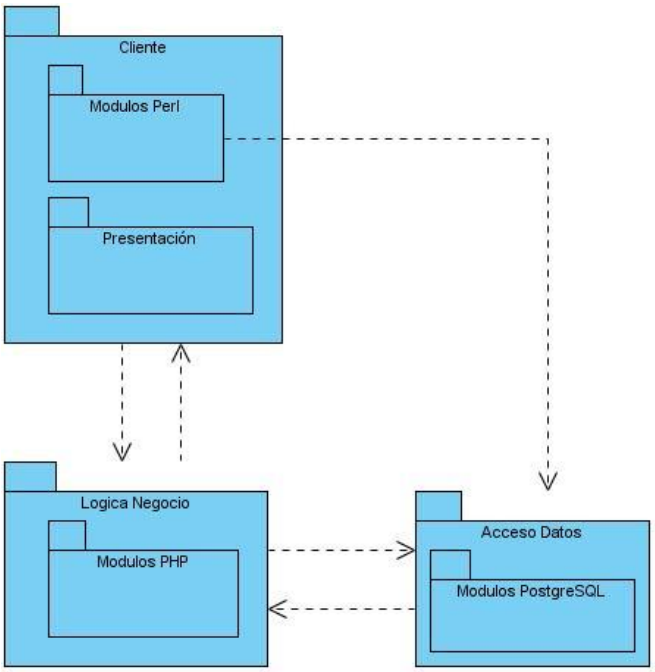


Figura 2: Diagrama de paquetes.

Descripción

Cliente: Es la estación de trabajo que está siendo monitoreada. Dentro de ella se localiza el script de Perl que realiza el monitoreo, y desde esta estación se accede a la capa de presentación del sistema.

En la **capa de presentación** se tienen los componentes HTML, CSS y JavaScript, los cuales son los lenguajes que se utilizan en la interfaz visual de la aplicación. Esta capa de presentación depende de la lógica de negocio para mostrar los datos.

En la capa de **lógica de negocio** se encuentra el componente PHP 5.3, que no es más que el módulo principal de la aplicación, que engloba toda el funcionamiento del negocio.

La capa de **acceso a datos** la conforman los paquetes módulos de PostgreSQL. Dentro de este paquete se tiene el módulo libdbi-perl que permite la conexión de los clientes con la Base de Datos y el envío de la información de monitoreo hacia ésta.

Dentro del paquete Módulos de PostgreSQL se encuentra Php5-PostgreSQL que es el módulo de PHP utilizado para la conexión a la base de datos PostgreSQL, el mismo depende del componente PostgreSQL-Client que no es más que el cliente del gestor de bases de datos PostgreSQL.

Diagrama de Componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

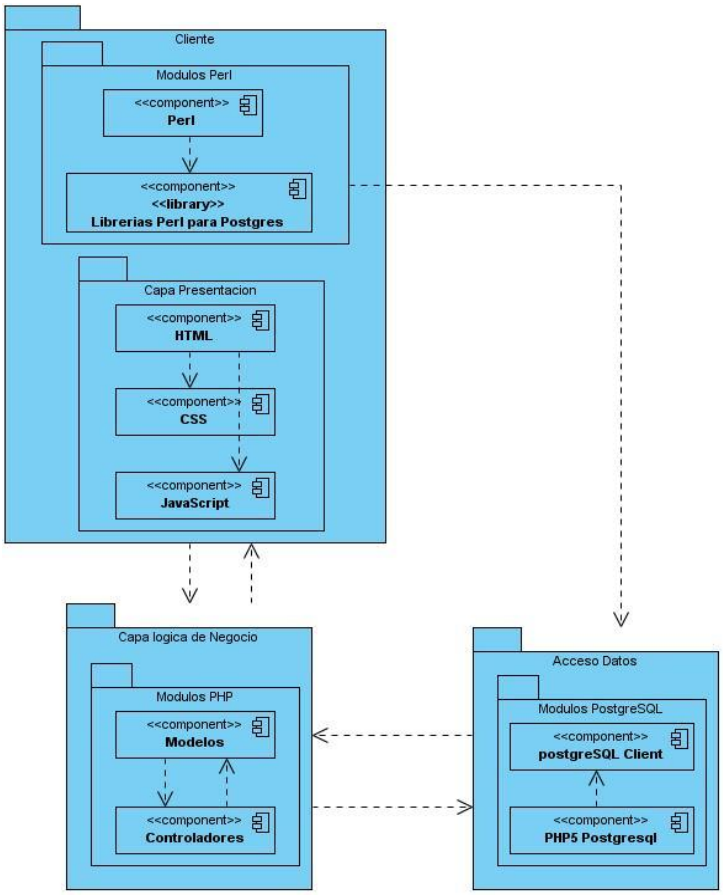


Figura 3. Diagrama de componentes.

2.8 Principios de Diseño

Para lograr una mayor aceptación por parte de los usuarios es necesario crear una aplicación con interfaces amigables que permitan mejorar su usabilidad. Para tener las interfaces de comunicación con el usuario de una manera estándar y sencilla se utilizaron colores suaves y sin sobrepasar la suma de 4, para que el usuario sea capaz de trabajar con la aplicación sin problemas.

Se diferenciaron las interfaces a partir de los roles de cada usuario en el sistema, con el propósito de no presentar elementos innecesarios a los diferentes usuarios que existen en el sistema., de esta forma solo se muestra la información referente al rol que presenta el usuario autenticado. La interfaz del sistema presenta una sencilla organización que posibilita un fácil entendimiento de la aplicación.

Patrones de Diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones durante el desarrollo de un software y otros ámbitos referidos al diseño de interfaces. Forman parte de una solución a un problema de diseño y expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

Los patrones de diseño se clasifican de acuerdo a las siguientes nominaciones:

- **Patrones creacionales:** Inicialización y configuración de datos.
- **Patrones estructurales:** Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- **Patrones de Comportamiento:** Más que describir objetos o clases, describen la comunicación entre ellos.

En el desarrollo de este sistema se utiliza el patrón creacional Modelo Vista Controlador (MVC)[20]. Este patrón plantea la separación del problema en tres capas: la capa modelo, que representa la realidad; la capa controladora **,** que conoce los métodos y atributos del modelo, recibe y realiza lo que el usuario quiere hacer; y la capa vista, que muestra un aspecto del modelo y es utilizada por la capa anterior para interaccionar con el usuario.

Diseño de la Base de Datos. Diagrama Entidad Relación

El correcto funcionamiento de la aplicación depende también del buen diseño y funcionamiento de la base de datos a utilizar. En este epígrafe se muestran el Diagrama de Entidad Relación (DER), y el Diagrama de Clases Persistentes (DCP), correspondiente al diseño de la BD.

[20] En inglés Model View Controller.

El DER representa la realidad de la problemática identificada a través de las entidades y de los enlaces que rigen la unión de las mismas y que constituyen la relación del modelo. Es el modelo conceptual más utilizado en el diseño de BD, capaz de mostrar cómo funcionará el sistema y la información que almacenará. Los DER son una necesidad para diseñar y mantener una buena BD relacional, son una forma gráfica de representarla.

Para el presente sistema se representaron un total de 6 tablas, interrelacionadas unas con otras. Para el diseño del DER se estableció un estándar para organizar el trabajo, representando los nombres de las tablas y sus atributos en minúsculas y en el caso de los nombres compuestos, su unión se realiza con el guión bajo.

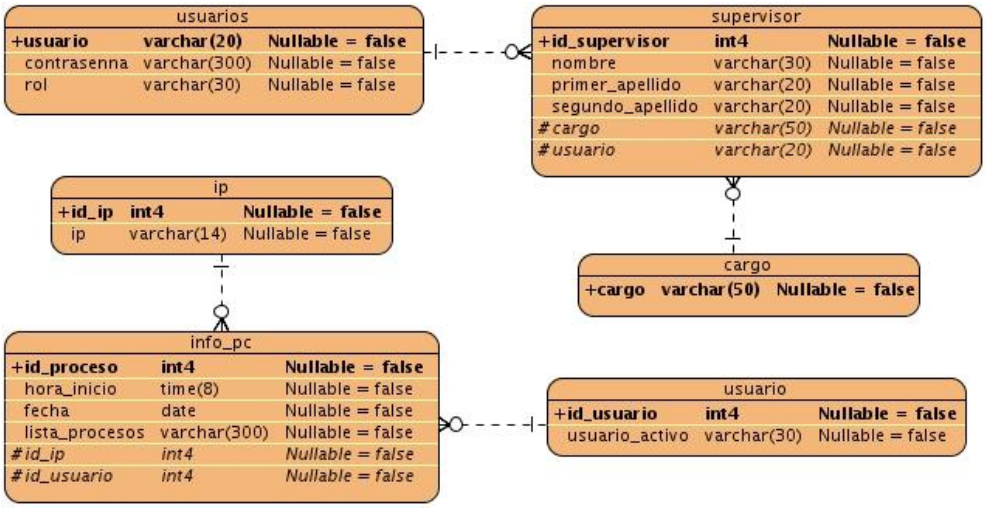


Figura 4. Diagrama Entidad Relación.

Clases Persistentes

Estas clases contienen toda la información que se almacena en la BD. El diagrama de clases persistentes es utilizado en el modelado de la estructura lógica de la BD, donde las tablas son representadas por clases y las columnas por atributos de éstas, haciendo referencia directamente a las entidades lógicas y sus atributos.

Las clases persistentes son las clases que necesitan ser capaz de guardar su estado en un medio permanente. Son aquellas que contienen toda la información valiosa e importante que necesariamente debe ser almacenada en la BD.

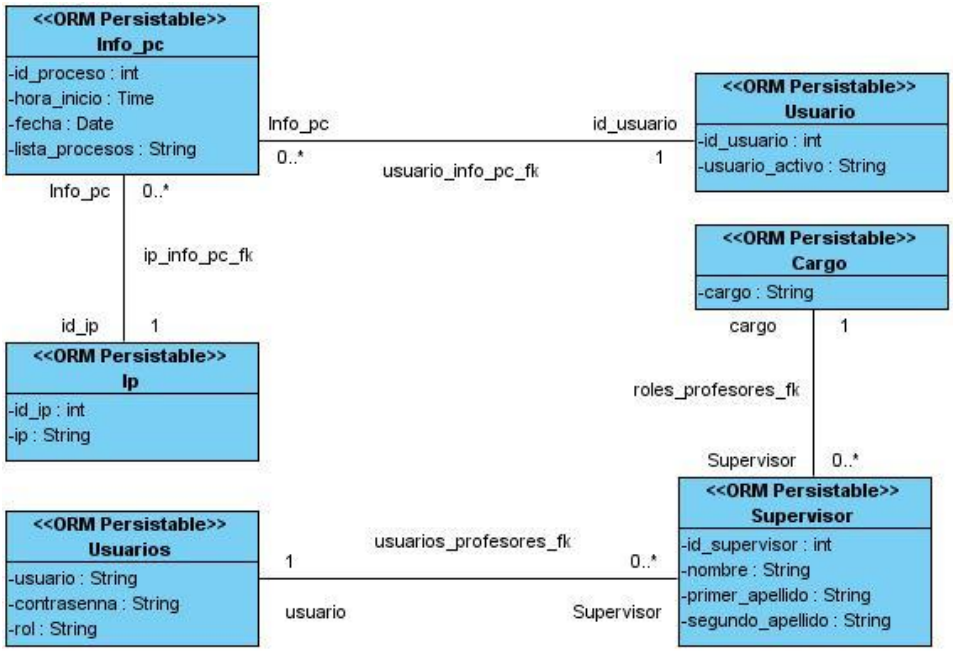


Figura 5. Diagrama de Clases Persistentes

2.9 Conclusiones Parciales

En el capítulo se ha detallado la propuesta de solución como aspecto inicial para lograr un mejor entendimiento entre el cliente y los desarrolladores. Se realizó la planificación del proyecto por roles, para distribuir las tareas y de esta forma garantizar la realización de un trabajo organizado. Además se identificaron las necesidades del cliente, plasmándolas mediante requerimientos, los cuales se describieron a través de las Historias de Usuario. Por último, tomando en cuenta la prioridad para el negocio de las Historias de Usuario se creó el plan de iteraciones, estimando el tiempo de desarrollo en semanas para cada una de ellas.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

3.1 Introducción

En el presente capítulo se exponen los casos de pruebas o *test* de aceptación a las que fue sometida la aplicación que fueron realizadas en cada una de las iteraciones, las cuales fueron necesarias para avanzar hacia la próxima iteración, pues para lograr un producto con calidad es necesario implementar un plan de pruebas desde el principio y así darle seguimiento a los cambios y desarrollar iterativamente. En este capítulo además de las pruebas se dan a conocer los resultados obtenidos hasta el momento.

3.2 Pruebas de software. Casos de prueba

Las pruebas de software son los procesos que permiten verificar la calidad de un producto de software. Se realizan con el fin de identificar posibles fallos de implementación, calidad o usabilidad de un programa. Es una fase en el desarrollo de un software, y consiste en probar las aplicaciones construidas.

Para lograr un producto con calidad es necesario realizar un plan de pruebas desde el principio. Darle seguimiento a los cambios y desarrollar iterativamente.

Pruebas unitarias

Consisten en probar o testear piezas de software pequeñas; a nivel de secciones, procedimientos, funciones y módulos; aquellas que tengan funcionalidades específicas. Dichas pruebas se utilizan para asegurar el correcto funcionamiento de secciones de código, mucho más reducidas que el conjunto, y que tienen funciones concretas con cierto grado de independencia.

El objetivo principal de las pruebas unitarias es detectar errores en el trabajo realizado para posteriormente corregirlos. Ejemplos de pruebas unitarias se encuentran las pruebas de caja blanca y las pruebas de caja negra.

Prueba de caja blanca

Se denomina prueba de caja blanca a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo.

En las pruebas de caja blanca se conoce y se tiene en cuenta el funcionamiento interno de un sistema. Las pruebas se planifican observando la estructura del algoritmo del sistema, sus condicionales, bucles, casos especiales.

Prueba de caja negra

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.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

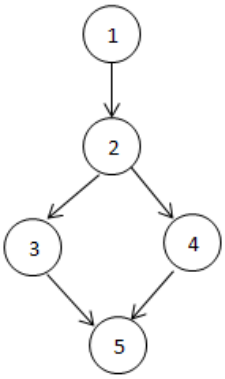
1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En esta investigación se realizaron las pruebas de caja blanca, utilizando el método del camino básico y determinando la complejidad ciclomática del algoritmo.

A continuación se exponen las pruebas realizadas a algunos algoritmos que se consideran la espina dorsal de la aplicación.

Insertar Supervisor

```
1  $nombre = $_POST['nombre'];
   $primer_apellido = $_POST['primer_apellido'];
   $segundo_apellido = $_POST['segundo_apellido'];
   $cargo = $_POST['cargo'];
   $usuario = $_POST['usuario'];
   $contrasenna = $_POST['contrasenna'];
   $rol = $_POST['rol'];
   $CSupervisor = new
   Supervisor($nombre,$primer_apellido,$segundo_apellido,$cargo,$usuari
   o,$contrasenna,$rol);
   $control->InsertarSupervisor($CSupervisor);
   $mensaje  = "Hubo error al insertar";
2  if($control){
3  $mensaje = "Insertado satisfactoriamente";}
   header("location:../Vista/adicionar_supervisor.php?mensaje=$mensaje"
   );}
4  else {
   header("location:../Vista/eliminar_supervisor.php?mensaje=$mensaje")
   ;}
5  }´
```



Complejidad Ciclomática: 2

Caminos:

C1: 1-2-3-5

C2: 1-2-4-5

Tabla 17. Caso de Prueba del Camino Básico para Insertar Supervisor. Variante 1.

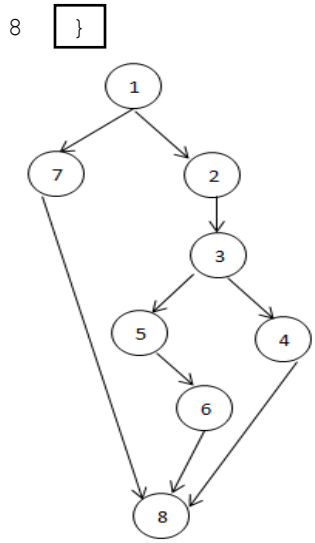
	Caso de Prueba para C1
$control	True
$mensaje, header	El usuario se insertó correctamente
Resultado	El método ha recorrido el camino correcto indicando en un mensaje que el usuario ha sido adicionado satisfactoriamente.

Tabla 18. Caso de Prueba del Camino Básico para Insertar Supervisor. Variante 2.

	Caso de Prueba para C2
$control	False
header	El usuario no se insertó
Resultado	El método ha recorrido el camino incorrecto indicando en un mensaje que el usuario no ha sido adicionado.

Eliminar Supervisor

```
1  if (!empty($_POST['campos'])) {
2  $var = "'" . implode("','", $_POST['campos']) . "'";
   $control->EliminarSupervisor($var);
3  if ($control) {
4  $mensaje = "Usuarios eliminados correctamente";
   header("Location:../Vista/eliminar_supervisor.php?mensaje=$mensaje")
   ;}
5  else if (!$control) {
6  $mensaje = "No se pudo establecer la conexi&oacute;n con la base de
   datos";
   header("Location:../Vista/eliminar_supervisor.php?mensaje=$mensaje")
   ;}
7  else{
   header("location:../Vista/eliminar_supervisor.php?mensaje=$mensaje"
   );}
```



Complejidad ciclomática: 2

Caminos:

C1: 1-7-8

C2: 1-2-3-4-8

C3: 1-2-3-5-6-8

Tabla 19. Caso de Prueba del Camino Básico para Eliminar Supervisor. Variante 1.

	Caso de Prueba para C1
$control	True
$mensaje, header	El usuario se eliminó correctamente
Resultado	El método ha recorrido el camino correcto indicando en un mensaje que el usuario ha sido eliminado satisfactoriamente.

Tabla 20. Caso de Prueba del Camino Básico para Eliminar Supervisor. Variante 2.

	Caso de Prueba para C1
$control	False
$mensaje, header	El usuario no se eliminó correctamente
Resultado	El método ha recorrido el camino correcto indicando en un mensaje que no existe conexión con la BD.

Tabla 21. Caso de Prueba del Camino Básico para Eliminar Supervisor. Variante 3.

	Caso de Prueba para C1
$control	
header	El usuario se eliminó correctamente
Resultado	El método ha recorrido el camino correcto indicando en un mensaje que se debe seleccionar un criterio para eliminar.

Pruebas de Aceptación

La programación extrema define entre iteración e iteración un conjunto de casos de pruebas o tests de aceptación para poder avanzar a una iteración superior. Durante el desarrollo del sistema de gestión del uso de las estaciones de trabajo un conjunto de casos de prueba a las que fue sometido el sistema para comprobar el funcionamiento de acuerdo a las Historias de Usuario.

Se definieron casos de prueba de aceptación para todas las historias de usuario. Fueron diseñadas nueve pruebas de aceptación, con un total de veintiséis iteraciones, hasta obtener los resultados esperados para cada prueba según los requisitos definidos en las Historias de Usuario.

Caso de Prueba de Aceptación, Historia de Usuario: Realizar monitoreo.

Esta sección cubre el conjunto de pruebas funcionales realizadas con la historia de usuario Realizar monitoreo.

En esta HU se intenta comprobar que los scripts de monitoreo son capaces de obtener las informaciones necesarias de las estaciones clientes, a partir de las especificaciones descritas en ellos.

Esta prueba se desarrolló en cinco iteraciones debido a las inconformidades arrojadas, las cuales responden a diferentes errores encontrados, por lo que fue necesario iterar hasta obtener los resultados requeridos.

Tabla 22. Prueba de Aceptación HU Realizar monitoreo.

	Caso de Prueba de Aceptación
Código Caso de Prueba: SGET-1-1	**Nombre Historia de Usuario**: Realizar monitoreo
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo comprobar la realización del	

monitoreo de las estaciones clientes.
Condiciones de Ejecución: Deben estar instaladas las herramientas necesarias para el monitoreo de las estaciones clientes.
Entrada / Pasos de Ejecución: 1. Programar el cron para que los script necesarios para el monitoreo se ejecuten en intervalos de 5 minutos.
Resultado Esperado: Se obtienen los procesos en ejecución de las estaciones de trabajo, la hora de inicio de los mismos, el usuario que los ejecuta y la dirección ip donde se están ejecutando.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Aceptación, Historia de Usuario: Gestionar Usuarios.

Esta sección cubre el conjunto de pruebas funcionales realizadas con la historia de usuario: Gestionar usuarios.

Para esta historia de usuario se comprueba que el sistema permita la inserción y eliminación de los usuarios que podrán interactuar con el mismo, teniendo en cuenta que no podrán repetirse los usuarios en la Base de Datos. Además con esta prueba se comprueba que los datos de los usuarios puedan ser modificados en cualquier momento, así como ser mostrados en un listado.

Para esta HU se realizaron cuatro pruebas en ocho iteraciones debido a las inconformidades arrojadas, las cuales responden a diferentes errores encontrados, por lo que fue necesario iterar hasta obtener los resultados requeridos.

Tabla 23. Prueba de Aceptación HU Gestionar usuarios (Insertar usuarios).

	Caso de Prueba de Aceptación
Código Caso de Prueba: SGET-2-1	**Nombre Historia de Usuario**: Gestionar usuarios
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	

Descripción de la Prueba: La prueba tiene como objetivo comprobar la inserción de los usuarios que tendrán acceso al sistema, en la Base de Datos.
Condiciones de Ejecución: Debe estar autenticado en el sistema.
Entrada / Pasos de Ejecución: 1. Abrir la aplicación. 2. Acceder al sistema. 3. Acceder a la vista de insertar usuarios. 4. Llenar los datos de los usuarios. 5. Enviar los datos.
Resultado Esperado: Se insertan los datos correctamente en la Base de Datos.
Evaluación de la Prueba: Satisfactoria.

Tabla 24. Prueba de Aceptación HU Gestionar usuarios (Modificar usuarios).

Caso de Prueba de Aceptación	
Código Caso de Prueba: SGET-2-2	**Nombre Historia de Usuario**: Gestionar usuarios
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo comprobar que se pueden modificar los datos de los usuarios de la Base de Datos.	
Condiciones de Ejecución: Debe estar autenticado en el sistema.	
Entrada / Pasos de Ejecución: 1. Abrir la aplicación. 2. Acceder al sistema. 3. Acceder a la vista de modificar datos de usuarios. 4. Seleccionar el usuario que se desea modificar. 5. Modificar los datos del usuario seleccionado. 6. Aceptar los cambios.	
Resultado Esperado: Se modifican los datos de los usuarios en la Base de Datos.	

Evaluación de la Prueba: Satisfactoria.

Tabla 25. Prueba de Aceptación HU Gestionar usuarios (Eliminar usuarios).

Caso de Prueba de Aceptación	
Código Caso de Prueba: SGET-2-3	**Nombre Historia de Usuario**: Gestionar usuarios
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo comprobar que se pueden eliminar usuarios de la Base de Datos.	
Condiciones de Ejecución: Debe estar autenticado en el sistema.	
Entrada / Pasos de Ejecución: 1. Abrir la aplicación. 2. Acceder al sistema. 3. Acceder a la vista de eliminar usuarios. 4. Seleccionar el usuario a eliminar. 5. Enviar la solicitud.	
Resultado Esperado: Se eliminan los usuarios de la Base de Datos.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 26. Prueba de Aceptación HU Gestionar usuarios (Mostrar usuarios).

Caso de Prueba de Aceptación	
Código Caso de Prueba: SGET-2-4	**Nombre Historia de Usuario**: Gestionar usuarios
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo verificar que el sistema muestra un listado con los usuarios que tienen acceso a la aplicación.	
Condiciones de Ejecución: Debe estar autenticado en el sistema.	

Entrada / Pasos de Ejecución: 1. Abrir la aplicación. 2. Acceder al sistema. 3. Acceder a la vista mostrar usuarios.
Resultado Esperado: Se muestra el listado de los usuarios con acceso a la aplicación.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Aceptación, Historia de Usuario: Definir niveles de acceso.

Esta sección cubre el conjunto de pruebas funcionales realizadas con la historia de usuario: Definir niveles de acceso.

Para esta historia de usuario, se comprueba que el usuario una vez autenticado, solamente pueda realizar las tareas asignadas a él, según el rol que tiene dentro del sistema, asegurando de esta manera un nivel de seguridad adecuado dentro del sistema.

La prueba se desarrolló en tres iteraciones debido a las inconformidades arrojadas, las cuales responden a diferentes errores encontrados, por lo que fue necesario iterar hasta obtener los resultados requeridos.

Tabla 27. Prueba de Aceptación HU Definir niveles de acceso.

	.Caso de Prueba de Aceptación
Código Caso de Prueba: SGET-3-1	**Nombre Historia de Usuario**: Definir niveles de acceso.
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo definir los niveles de acceso para cada usuario, según el rol que tenga en el sistema.	
Condiciones de Ejecución: Debe estar autenticado en el sistema.	
Entrada / Pasos de Ejecución:	

1. Abrir la aplicación. 2. Acceder al sistema. 3. Realizar las tareas permitidas para el usuario según el rol.
Resultado Esperado: Se definieron correctamente los niveles de acceso para cada tipo de usuario.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Aceptación, Historia de Usuario: Gestionar datos.

Esta sección cubre el conjunto de pruebas funcionales realizadas con la historia de usuario: Gestionar datos.

Para esta historia de usuario se comprueba que los scripts instalados en las estaciones clientes capturen la información necesaria para el monitoreo y que sean capaces de enviarla al Servidor de Base de Datos. Se comprueba además que estos datos puedan ser mostrados, siguiendo para ello diferentes criterios de selección y que además permita puedan ser eliminados aquellos datos que se deseen.

Se realizaron tres pruebas para esta HU en diez iteraciones debido a las inconformidades arrojadas, las cuales responden a diferentes errores encontrados, por lo que fue necesario iterar hasta obtener los resultados requeridos.

Tabla 28. Prueba de Aceptación HU Gestionar datos (Insertar datos).

	Caso de Prueba de Aceptación
Código Caso de Prueba: SGET-4-1	**Nombre Historia de Usuario**: Gestionar datos
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo comprobar que se insertan los datos obtenidos en el monitoreo de las estaciones clientes en la Base de Datos.	
Condiciones de Ejecución: Deben estar instaladas las herramientas necesarias para el monitoreo de las estaciones clientes.	

Entrada / Pasos de Ejecución: 1. Programar el cron para que los script necesarios para el monitoreo se ejecuten en intervalos de 5 minutos.
Resultado Esperado: Se insertan los datos de monitoreo en la Base de Datos.
Evaluación de la Prueba: Satisfactoria.

Tabla 29. Prueba de Aceptación HU Gestionar datos (Mostrar datos).

Caso de Prueba de Aceptación	
Código Caso de Prueba: SGET-4-2	**Nombre Historia de Usuario**: Gestionar datos
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo comprobar que la aplicación muestra los datos obtenidos en el monitoreo de las estaciones clientes.	
Condiciones de Ejecución:	
Entrada / Pasos de Ejecución: 1. Abrir la aplicación. 2. Acceder al sistema. 3. Acceder a la vista obtener información de pc.	
Resultado Esperado: Se muestran los datos del monitoreo.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 30. Prueba de Aceptación HU Gestionar datos (Eliminar datos).

Caso de Prueba de Aceptación	
Código Caso de Prueba: SGET-4-3	**Nombre Historia de Usuario**: Gestionar datos
Nombre de la persona que realiza la prueba: Leovan Peña Serrano	
Descripción de la Prueba: La prueba tiene como objetivo comprobar que se pueden eliminar los datos obtenidos en el monitoreo de las estaciones clientes, existentes en la	

Base de Datos.
Condiciones de Ejecución: Debe estar autenticado en el sistema.
Entrada / Pasos de Ejecución: 1. Abrir la aplicación. 2. Acceder al sistema. 3. Acceder a la vista eliminar información de pc. 4. Seleccionar el criterio por el cual se desea eliminar la información.
Resultado Esperado: Se eliminan los datos.
Evaluación de la Prueba: Satisfactoria.

3.3 Validación teórica del diseño. Análisis de la integridad de los datos

La integridad de los datos y la redundancia de la información son aspectos importantes a tener en cuenta en el diseño de las BD. La integridad conserva la seguridad en un sistema de bases de datos que permite el acceso a múltiples usuarios en tiempos paralelos y se refiere además a la corrección y exactitud de la información que contiene. Es el término utilizado para decir que la información almacenada tiene calidad.

Reglas de integridad

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina *restricciones de dominio*. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la de integridad de entidades y la de integridad referencial. Antes de definirlas es preciso conocer el concepto de *nulo* y *dominio*.

Nulo: Es un indicador que le dice al usuario que el dato falta o no es aplicable. Por conveniencia, un dato que falta normalmente se dice que tiene valor Nulo, pero el

valor de Nulo no es un valor de dato real. En vez de ello es una señal o un recordatorio de que el valor falta o es desconocido.

Dominio: Posibles valores que puede tener un campo. Un dominio no es más que un tipo de dato; posiblemente un tipo simple definido por el sistema o por el usuario. El Dominio de un atributo define los valores posibles que puede tomar este atributo. Además de los Dominios "naturales", usados como tipos de datos, el administrador del sistema puede generar sus propios dominios definiendo el conjunto de valores permitidos. Esta característica, usada en forma correcta, se convierte en mecanismo de control, restricción y validación de los datos a ingresar.

Existen básicamente dos reglas de Integridad asociadas con el modelo relacional: la Integridad de Entidad y la Integridad Referencial. Estas dos reglas son generales y tienen relación con las llaves primarias y foráneas.

Integridad de Entidad: Las restricciones de entidades aseguran la integridad de las entidades que son modeladas por el sistema. En el nivel más simple, la existencia de una clave principal es una restricción de entidad que impone la regla "cada entidad debe estar identificada de forma única". En esta no está permitido que algún componente de la clave primaria acepte valores nulos. Esta regla sólo se aplica a las relaciones base y a las claves primaria, no a las claves foráneas ni alternativas.

Este tipo de integridad se cumple en la BD propuesta pues cada tabla tiene definida su clave primaria. Las llaves primarias de las tablas son campos secuenciales o valores que representan un código único, de esta forma este campo nunca será nulo ni se repetirá. Es de vital importancia especificar la clave principal para cada registro, ya que esta representa la herramienta fundamental que utiliza el servidor de BD para seleccionar la información que se necesite, evitando así la duplicidad de los registros.

Integridad Referencial: La regla de Integridad referencial define que la base de datos no debe contener valores de claves foráneas sin concordancia.

Esta regla se aplica a las claves foráneas. Si en una relación hay alguna clave foránea, entonces sus valores deben coincidir con los valores de la clave primaria a la que hace referencia, o bien, debe ser completamente nulo.

Esta regla impide que, por ejemplo, en una base de datos académica, exista un profesor en un departamento inexistente, o un curso impartido por un profesor inexistente.

La integridad referencial controla que se cumplan las siguientes reglas:

- No se podrá introducir un valor en la tabla relacionada si antes no ha sido introducida en la tabla principal.
- No se puede eliminar un registro de una tabla principal si existen registros coincidentes en la tabla relacionada.
- No se puede cambiar un valor de la clave primaria en la tabla principal si el registro tiene registros relacionados.

3.4 Redundancia de la información

La redundancia de datos es aquella información duplicada o almacenada varias veces en una misma BD, que dificulta la modificación de los datos y crea inconsistencia entre los mismos. Por otra parte ocupa un mayor espacio de almacenamiento e influye de forma negativa en el tiempo de acceso a los datos, aumentando el mismo.

La BD diseñada para el sistema de gestión del uso de las estaciones de trabajo del CDS está libre de redundancias, pues se eliminan casi completamente los datos repetidos innecesariamente.

3.5 Resultados Obtenidos

Como resultado de este trabajo queda disponible el Sistema de Gestión del uso de las estaciones de trabajo en su versión 1.0, siendo posible esperar otros resultados en versiones posteriores.

3.6 Sobre el tiempo de desarrollo

El correcto control del uso de las estaciones de trabajo en el CDS de la FRA es una problemática que viene dada desde el curso 2010-2011, por las razones anteriormente expuestas; por lo que en el presente curso escolar se comenzó a dar los primeros pasos en el transcurso del mes de noviembre en el desarrollo de esta herramienta. Primero fue necesario realizar labores de investigación para obtener conocimientos sobre el tema en cuestión.

Posteriormente, en enero de 2012 se comenzó a estudiar las herramientas, lenguajes y tecnologías necesarias para el desarrollo de esta aplicación, así como fue necesario en ese mes aprender a trabajar con ellas, sobre todo con Perl, lenguaje de vital importancia en el desarrollo del sistema. En febrero se inició la etapa de desarrollo del producto, y a finales del mes de marzo se tiene el sistema con el 40% de las funcionalidades planificadas. El desarrollo continuó y ya en mayo estaba el sistema de gestión con las funcionalidades requeridas implementadas, y con un funcionamiento estable.

Si se tiene en cuenta que el equipo de desarrollo no poseía experiencia previa de los elementos y conceptos más importantes para desarrollar el sistema de gestión, y que además el período de tiempo empleado fue relativamente corto, se puede plantear entonces que se superaron las expectativas de la planificación inicial.

3.7 Funcionalidades obtenidas

Al concluir el desarrollo de la aplicación se obtiene el Sistema de Gestión del uso de las estaciones de trabajo con varias funcionalidades que constituyen la esencia del mismo. A continuación un pequeño resumen de las principales funcionalidades:

- Monitoreo de las computadoras conectadas en la red.
- Muestra la información de monitoreo de las computadoras, atendiendo a diferentes criterios de selección.
- Gestiona los usuarios que tendrán acceso al sistema.
- Gestiona los permisos de los usuarios, atendiendo a los roles definidos en el sistema.

3.8 Aporte social y económico

El desarrollo de este trabajo ha posibilitado realizar un notable aporte al Centro de Desarrollo de la Facultad Regional Mártires de Artemisa proveyéndola de una herramienta en óptimas condiciones para desarrollar una correcta gestión del aprovechamiento del tiempo de máquina en las estaciones de trabajo. Además es notable el ahorro en divisas por concepto de hojas y tóner si se tiene en cuenta que un paquete de hojas tiene un valor de $4.00 CUC y un tóner un valor de $12.00 CUC y $16.00 CUP.

No es necesaria tampoco la presencia de un técnico de laboratorio o tutor que controle la asistencia al tiempo de producción. Se logra controlar el trabajo de todo un día, en los diferentes laboratorios del centro. Elimina la adquisición de un software en el mercado que realice estas tareas, y por consiguiente el pago de una licencia por el mismo, ahorrando al país dinero, que se puede invertir en otras actividades de mayor prioridad.

3.9 Conclusiones Parciales.

En el presente capítulo se elaboraron y aplicaron los casos de pruebas de aceptación a cada historia de usuario para dar validez y seguridad a la propuesta de solución. Se validó la integridad de los datos en la BD, haciendo un análisis de las categorías existentes. Se arriba a la obtención de un software altamente calificado para su uso, posibilitando así al CDS de la FRA la obtención de información fiable y segura del trabajo que realizan sus integrantes.

CONCLUSIONES

A partir de los objetivos planteados y el trabajo realizado en esta investigación donde se desarrolló una herramienta para la gestión del uso de las estaciones de trabajo del CDS de la FRA, se arribó a las siguientes conclusiones:

- Se caracterizaron las soluciones de monitoreo existentes en la actualidad, evidenciándose la necesidad de crear una solución propia para estos fines.
- Se seleccionaron las herramientas, tecnologías y metodología más apropiadas para el desarrollo de la solución.
- Se implementó la solución.
- Se validó el eficaz funcionamiento del sistema mediante pruebas funcionales.

RECOMENDACIONES

- Desarrollar una nueva versión del sistema, con nuevas funcionalidades, y/o mejoras a las actuales, para mejorar la calidad y robustez del producto.
- Validar el funcionamiento de la herramienta en otras instituciones que utilicen software libre.
- Desarrollar nuevas funcionalidades que permitan no solo el monitoreo en estaciones con software libre, sino también en aquellas que posean software propietario.

BIBLIOGRAFÍA

1. [En línea] [Citado el: 16 de 02 de 2012.] http://www.lcc.uma.es/~eat/services/html-js/manual14.html.
2. *Scribd.* [En línea] Universidad Politécnica de Valencia. [Citado el: 19 de 03 de 2012.] http://es.scribd.com/doc/19317161/57/Diagrama-de-Componentes.
3. **2012.** ¿Qués es Software Libre? [En línea] 6 de 02 de 2012. [Citado el: 17 de 2 de 2012.] http://www.gnu.org/philosophy/free-sw.es.html.
4. **Alvarez, Miguel Angel. 2001.** desarrolloweb.com. [En línea] 01 de 06 de 2001. [Citado el: 16 de 02 de 2012.] http://www.desarrolloweb.com/articulos/25.php.
5. **2010.** BALUART. *BALUART.NET.* [En línea] 19 de 07 de 2010. [Citado el: 18 de 03 de 2012.] http://www.baluart.net/articulo/introduccion-a-los-patrones-de-diseno-con-php.
6. **Castro Ruz, Fidel. 1997..** *Resolución Económica del V Congreso del Partido Comunista de Cuba.* La Habana. : Editora Política, 1997.
7. **Centro Interamericano para el Desarrollo del Conocimiento en la Formación Profesional . 1996.** CINTERFOR. *Monitoreo y evaluación.* [En línea] 1996. [Citado el: 14 de 02 de 2012.] http://temp.oitcinterfor.org/public/spanish/region/ampro/cinterfor/temas/rural/genero/modelo/monitor.htm.
8. Definición.DE. [En línea] [Citado el: 15 de 02 de 2012.] http://definicion.de/html/.
9. **Delgado Expósito, Erly.** Metodologías de desarrollo de software. ¿Cuál es el camino? [En línea] [Citado el: 23 de 02 de 2012.] http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml.
10. **Drucker, Peter Ferdinand (1909-2005). 2003.** *Gestión del Conocimiento.* Bilbao : Deusto S.A. Ediciones, 2003. 9788423420230 ..
11. **Eguíluz Pérez, Javier.** librosweb.es. [En línea] [Citado el: 16 de 02 de 2012.] http://www.librosweb.es/css/.
12. *Estado actual del diagnóstico de transformadores de potencia en las centrales eléctricas cubanas.* **Montané García, Jorge Juan, y otros. 2011.** 1, Habana : Cujae, 2011, Ingeniería Energética, Vol. XXXII, pág. 1. disponible en http://dialnet.unirioja.es/servlet/listaarticulos?tipo_busqueda=EJEMPLAR&revista_busqueda=15656&clave_busqueda=278602. 1815-5901.

13. **Fayol, Henry (1841-1925). 1946.** *Administración industrial y general.* s.l. : El Ateneo, 1946. 950-02-3540-4.
14. **Hernández Regalado, Rafael. 2007.** *Las Mipimes en Latinoamerica.* 1ra. México : s.n., 2007. pág. 18. 978-968-864-482-9.
15. IE Gaviota. [En línea] [Citado el: 15 de 02 de 2012.] http://www.juntadeandalucia.es/averroes/iesgaviota/informatica/html.html.
16. **2012.** Introducción a las metodologías de Desarrollo de Software con UML. [En línea] 04 de 03 de 2012. [Citado el: 22 de 02 de 2012.] http://www.ati.es/spip.php?article1136.
17. **Lamarca Lapuente, María Jesús.** Hipertexto. [En línea] [Citado el: 15 de 02 de 2012.] http://www.hipertexto.info/documentos/html.htm.
18. **Llanes Delgado, W. 2004.** *Fundamentos de la dirección y gestión.* La Habana : s.n., 2004.
19. maestros del web. [En línea] [Citado el: 16 de 02 de 2012.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.
20. **melinkoff, Ramón. 1990.** *Los procesos administrativos.* Caracas : Editorial Panapo, 1990.
21. Metodología Scrum. [En línea] [Citado el: 22 de 02 de 2012.] http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html.
22. Metodologías de desarrollo de software. [En línea] [Citado el: 22 de 2 de 2012.] http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html.
23. **Monzón Quintana, Iliana, Prendes Gutiérrez, Ramón y Diéguez Glez, Mayelín.** Implantación de los Sistemas de Gestión de Calidad ISO 9000. [En línea] [Citado el: 20 de 1 de 2012.] http://www.monografias.com/trabajos27/implantacion-sistemas/implantacion-sistemas.shtml.
24. **Oré B, Ing. Alexander. 2009.** CalidadySoftware.com. [En línea] 2009. [Citado el: 21 de 02 de 2012.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.
25. **Ortiz, Carlos, y otros. 2007.** El Control como fase del proceso administrativo. [En línea] 18 de 9 de 2007. [Citado el: 20 de 1 de 2012.] http://www.monografias.com/trabajos12/cofas/cofas.shtml.
26. **Paz, Ariel. 2011.** *Estaciones de trabajo: ¿qué son y en qué se utilizan?* [En línea] 08 de 06 de 2011. [Citado el: 14 de 02 de 2012.] http://tecnologiaymas.over-blog.es/article-estaciones-trabajo---que-que-utilizan-85830173.html.

27. Programación Extrema XP. [En línea] [Citado el: 22 de 02 de 2012.] http://www.ecured.cu/index.php/Programaci%C3%B3n_Extrema_%28XP%29.
28. Proyecto GNU. [En línea] [Citado el: 22 de 2 de 2012.] http://www.ecured.cu/index.php/Proyecto_GNU.
29. **2010.** SCRUM como metodología de desarrollo. [En línea] 07 de 03 de 2010. [Citado el: 17 de 02 de 2012.] http://www.omitsis.com/scrum-como-metodologia-de-desarrollo.
30. **2008.** SCRUM, Metodología ágil de desarrollo. [En línea] 14 de 02 de 2008. [Citado el: 22 de 02 de 2012.] http://arturoweb.wordpress.com/2008/02/14/scrum-metodologia-agil-de-desarrollo/.
31. **Stephen y De Cenzo, David. 1996.** *Fundamentos de administración, concepto y aplicaciones.* México : s.n., 1996.
32. **Stones, James, Freeman, R y Gilbert, D. 1996.** *Administración.* México : Sexta Edición, 1996.
33. **Tedeschi, Nicolás .** MSDN. [En línea] Microsoft. [Citado el: 18 de 03 de 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.
34. Utilizar Nagios en Linux para Monitorizar Redes. [En línea] [Citado el: 20 de 1 de 2012.] http://www.linux-party.com/modules.php?name=News&file=article&sid=6079/utilizar-nagios-en-linux-para-monitorizar-redes.
35. **Valentín Jiménez, José Manuel. 2008.** *Gestión Empresarial.* [En línea] 2008. [Citado el: 12 de 02 de 2012.] http://www.gestionempresarial.info/VerItemProducto.asp?Id_Prod_Serv=28&Id_Sec=8.
36. **Valverde, David. 2007.** Introducción a la Programación Extrema (XP). [En línea] 06 de 09 de 2007. [Citado el: 22 de 02 de 2012.] http://www.davidvalverde.com/blog/introduccion-a-la-programacion-extrema-xp/.