Universidad de las Ciencias Informáticas

Facultad 10

“Aplicación para el monitoreo de ventanas y notificación de tareas”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Rodolfo González Pelegrino

Tutor: Ing. Roberto Rodríguez Montoya

Ciudad de la Habana

Junio del 2009

Año del 50 aniversario del triunfo de la Revolución

“*Muchos no están de acuerdo con el software libre y les es útil atacarme a mí, porque no pueden hacerlo con la filosofía. Tienen el derecho. No me incomoda. Eso sí, me disgusta que en muchos artículos, por ejemplo, me llamen “evangelista del código abierto”, porque no apoyo sólo esa filosofía...*“

“*No apoyo solo el código abierto, o sea que la gente nada más conozca cómo funciona un programa, sino el software libre. Y la diferencia es que para nosotros es un asunto ético. El software privativo no es ético porque no respeta la libertad, y el código abierto no se interesa por esta cuestión. Si yo soy el “padre del código abierto” se hizo con “esperma” que me robaron sin mi consentimiento*“.

Richard Stallman

Declaración de autoría:

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 10 de la Universidad de las Ciencias Informáticas; así como a dicho centro a consultar cualquier cambio que se decida aplicar al mismo con su autor.

Para que así conste firmo la presente a los _____ días del mes de _____ del año 2009.

Rodolfo González Pelegrino	Roberto Rodríguez Montoya
Firma del autor	Firma del tutor.

Rodolfo González Pelegrino	Roberto Rodríguez Montoya
Firma del autor	*Firma del tutor*

Datos de contacto del tutor:

Graduado en la Universidad de Oriente en Ingeniería en automática, es profesor instructor, pertenece al departamento de Sistemas Digitales, e imparte las asignaturas de Teleinformática. Es miembro del polo Software Libre, tiene participación eventos nacionales e internacionales.

Agradecimientos:

Un sincero agradecimiento a mi familia en general, a mi novia por influir sobre mi formación profesional, a mis amigos por darme confianza cuando lo necesité, a mis compañeros por compartir durante tantos años de estudio, a Israel González Gutiérrez y Andrea Gutiérrez Rodríguez por la infinita preocupación y ayuda en cualquier cosa. Un eterno abrazo a todos por confiar y exigir de mí lo mejor.

Dedicatoria:

Dedico este trabajo a mi mamá Aides M.Pelegrino Vega, a mi papá Rodolfo González Gutiérrez, a mi hermana Bexi González Pelegrino, a mi abuela Bernarda Gutiérrez Rodríguez y a mi abuelo Israel González Arévalo a pesar de haber desaparecido físicamente. Estos han sido los autores principales que de haberme formado desde la cuna.

Resumen:

Con este trabajo se realiza una herramienta de supervisión y notificación de tareas productivas. Para lograr la solución se investiga sobre las aplicaciones controladoras de procesos y se analizan sus características y forma de funcionar. Su propósito está enmarcado en aumentar la producción, por lo que se integra a la gestión de proyecto.

Facilita la transmisión de tareas a los usuarios, evitando que estos pierdan tiempo en el cumplimiento de sus actividades, permite que el líder de proyecto conozca con datos específicos qué está fallando en el grupo de desarrollo, así como la preferencia de cada integrante de su equipo. Esto ayuda en gran medida a que las planificaciones del proyecto tengan éxito en su desempeño.

Índice de contenido

1.Introducción.....10
1.Capítulo 1. Fundamentación teórica de las aplicaciones de monitoreo de procesos.....14
1.1.Sistemas de monitoreo de procesos.....14
1.1.1.Funcionamiento de los sistemas de monitoreo.....14
1.1.2.Características de los sistemas de monitoreo.....15
1.2.Aplicaciones para el control de procesos.....16
1.2.1.Conky.....18
1.2.1.Gkrellm.....18
1.2.1.Gnome-system-monitor.....19
1.2.1.Ksysguard.....19
1.2.1.xfce4-taskmanager.....20
1.2.1.WhoWatch.....20
1.2.1.Spong.....21
1.2.1.Pandora FMS.....21
1.2.1.Monit.....22
1.2.1.Nagios.....22
1.2.1.Comandos para monitorear procesos.....23
1.1.Sistema de ventanas X Window System.....25
1.1.1.Arquitectura de X Window. 27
1.1.Modelo Cliente/Servidor.....28
1.1.1.Ventajas y desventajas de la arquitectura cliente/servidor.....30
1.1.Aplicación de monitoreo integrado a herramienta de gestión de proyecto.....31
1.2.Herramientas, lenguajes y tecnologías a utilizar.....32
1.2.1.Entorno integrado de desarrollo.....33
1.2.1.Lenguaje C/C++.....35
1.2.1.Libxml2.....37
1.2.2.Xlib 37
1.2.1.Hilos.....38
1.2.2.Colas de mensajes.....38
1.2.3.Socket.....39
1.2.4.XML.....40
1.2.1.Bash.....41
1.2.2.Dia.....41
1.2.3.Metodología ágil a utilizar.....41
1.1.Conclusiones.....43
2.Capítulo 2. Descripción y análisis de la solución propuesta.....45
2.1.Propuesta del sistema a implementar.....45

2.2.Reutilización de código....45
2.3.Por qué utilizar el sistema X Window System....46
2.1.Xlib y XCB....47
2.2.Arquitectura de la aplicación cliente....48
2.3.Conclusiones....49
3.Capítulo 3. Desarrollo ágil de la aplicación propuesta....51
3.1.Planificación del proyecto por roles....51
3.2.Modelo de Dominio....52
3.3.Lista de Reserva del Producto....53
3.4.Historias de usuarios....55
3.5.Plan de versiones....64
3.1.Tareas de Ingeniería....66
3.2.Diseño con metáforas....78
4.Capítulo 4. Validación de la solución propuesta....81
4.1.Casos de prueba de aceptación 81
4.1.1.Caso de prueba para la Historia de usuario: U-AMVEX-01....81
4.1.2.Casos de pruebas para la Historia de usuario: U-AMVEX-02....82
4.1.3.Casos de pruebas para la Historia de Usuario: U-AMVEX-03....84
4.1.4.Casos de pruebas para la Historia de Usuario: U-AMVEX-04....85
4.1.5.Caso de prueba para la Historia de Usuario: U-AMVEX-05....87
4.1.6.Casos de pruebas para la Historia de Usuario: U-AMVEX-06....87
4.1.7.Casos de pruebas para la Historia de Usuario: U-AMVEX-07 88
4.1.8.Casos de pruebas para la Historia de Usuario: U-AMVEX-08 89
4.1.9.Casos de pruebas para la Historia de Usuario: U-AMVEX-09....90
4.2.Resultados obtenidos....90
Conclusiones....92
Recomendaciones....93
Referencias Bibliográficas....94
Bibliografía....97
Anexos 99
Glosario de términos....114

1. Introducción

Las tecnologías de la información (TIC), fuente impulsora del desarrollo mundial, cada día aumentan aceleradamente. Nuestro país juega un papel importante en este punto, se ha introducido en la producción de software con el propósito de lograr un mayor desarrollo. La Universidad de las Ciencias Informáticas (UCI) es la principal portadora del desarrollo tecnológico para Cuba, la misma cuenta con varias facultades, cada una responde a un perfil determinado, y todas están orientadas a la formación de especialistas en informática, teniendo como principio del proceso docente educativo la formación desde la producción.

El estudiante para la adquisición de conocimientos y habilidades está vinculado a trabajos productivos, es decir, a un proyecto. Existen laboratorios condicionados para tareas especialmente dedicadas a la producción, cada uno de estos, cuenta con un líder, quien es la persona responsable de que el proyecto cumpla las tareas que la universidad le asigna. Cada proyecto tiene un número de estudiantes que generalmente es superior a la cantidad de computadoras del laboratorio que atiende.

Actualmente se están presentando dificultades con la entrega de tareas en el proyecto Unicornios[1], ya sea por la realización de las mismas fuera del tiempo establecido o no entrega por parte de los estudiantes, lo que trae como consecuencia incumplimiento en el plan del proyecto; a raíz de esto surgen inconformidades con los clientes. En ocasiones el estudiante plantea que no existe tiempo para realizarlas (no verificó la herramienta de gestión de proyecto, o no hay máquina disponible). Sin embargo se han detectado algunas indisciplinas, por ejemplo, hay tendencias a ver vídeos, jugar y realizar otras actividades que entorpecen el desarrollo productivo, consumen gran tiempo en pérdida para el proyecto, y están fuera de lo establecido. Por el número de estudiantes que tiene un proyecto, a un líder se le hace difícil controlar si cada uno está utilizando la computadora con el objetivo de cumplir la tarea de producción, y además si este conoce la actividad que debe realizar en el momento que comienza a trabajar. Por otra parte, el jefe de proyecto no conoce las preferencias de su equipo, lo que trae como consecuencia que se realicen planificaciones sin éxito.

[1] Proyecto productivo de la Facultad 10 especializado en servicios de migración a software libre.

Introducción

Unicornios utiliza DotProject como herramienta para la gestión de proyecto, además ha desarrollado un módulo conocido como GENEST[2] para gestionar estadísticas sobre el uso de aplicaciones en las máquinas que los usuarios trabajan, pero el mismo no es capaz de monitorear el ordenador, recoger la información que necesita de cada cliente y llevarla hasta el servidor, por lo que su estado actual no le permite funcionar.

Problema científico: ¿Cómo supervisar el uso de aplicaciones y notificar tareas productivas de la herramienta de gestión de proyecto DotProject, de estudiantes y profesores en los laboratorios de producción de la facultad 10?.

Objeto de investigación: Sistemas para el monitoreo de tareas y notificación de eventos.

Campo de acción: Monitoreo de ventanas enfocadas para sistemas operativos GNU/Linux, en el proyecto Unicornios de la facultad 10.

Objetivo del trabajo: Implementar una aplicación cliente/servidor que permita supervisar procesos, notificar tareas productivas del DotProject e integrarse a GENEST, para optimizar el uso de los ordenadores en los laboratorios de producción.

Objetivos específicos:

- Analizar el funcionamiento de los sistemas de monitoreo de procesos y la distribución de software existentes.
- Implementar una aplicación cliente/servidor que permita monitorear los procesos y notificar tareas productivas del DotProject.
- Integrar la aplicación implementada al módulo GENEST.
- Validar el sistema implementado.

Idea a defender: Con el desarrollo y aplicación de una herramienta que permita obtener un entorno supervisado capaz de notificar tareas productivas de la herramienta de gestión de proyecto DotProject e integrarse a GENEST, se optimiza la producción en los laboratorios de proyecto de la facultad 10.

[2] Módulo del DotProject para la gestión de estadísticas del Home Compartido.

Las tareas fundamentales que se proponen para concretar los objetivos son las siguientes:

- Investigar y analizar las características, ventajas y desventajas de los sistemas de monitoreo.
- Sistematizar el funcionamiento de los sistemas de monitoreo.
- Analizar el funcionamiento del sistema X-window.
- Analizar el funcionamiento de flujos de comunicación a través de socket.
- Analizar la estructura de la base de datos del DotProject.
- Investigar el funcionamiento de GENEST.
- Implementar un software capaz de ser usado por diferentes tipos de arquitectura de hardware.
- Realizar pruebas de control para detectar y eliminar posibles errores que se detecten durante el ciclo de vida del producto.

Para dar cumplimiento a las tareas planteadas se emplean diversos métodos científicos, los cuales están constituidos por un conjunto de pasos o etapas bien establecidas que posibilitan dirigir el proceso de investigación de forma óptima, de modo que permita alcanzar su propósito, el conocimiento científico, de la manera más eficiente.

Los métodos teóricos utilizados en esta investigación son: el Histórico-Lógico y el Analítico-Sintético. El primero posibilita el estudio de los sistemas de monitoreo, permite conocer de forma general el funcionamiento y desarrollo de los mismos. El segundo facilita realizar un análisis sobre los diferentes componentes que conforman las aplicaciones de monitoreo, e integrar estos elementos para formar un todo.

Los métodos empíricos empleados son, la observación, medición y experimento. La observación científica es la percepción planificada, se utiliza para estudiar la interacción del usuario con las ventanas contenedoras del foco. La medición se usa con el objetivo de obtener información numérica acerca del tiempo que demora el estudiante o profesor con cada una de las aplicaciones, además para detectar programas en ejecución que tienen el foco, pero no están siendo usados. El experimento es aplicado para verificar teorías sobre el comportamiento de los integrantes del proyecto durante su horario productivo.

El documento se encuentra estructurado en 4 capítulos, incluyen toda la documentación teórica así como las pruebas realizadas al software. A continuación se hace una breve descripción de cada uno de ellos.

Capítulo 1: **Fundamentación teórica:** En este capítulo se realiza un análisis sobre de los diferentes sistemas de monitoreo, así como del funcionamiento del sistema de ventanas X Windows System. Además se describen las tecnologías, lenguajes y herramientas que dan solución al sistema propuesto y se justifica la metodología de desarrollo a utilizar.

Capítulo 2**: Análisis de la solución propuesta:** En este capítulo se hace un análisis de la solución propuesta. Se especifica y justifica la utilización de herramientas ya existentes que formarán parte de la solución del problema a resolver, así como la estrategia de integración del software.

Capítulo 3: **Desarrollo ágil del sistema propuesto:** En este capítulo se realiza el desarrollo ágil del software, utilizando como metodología de desarrollo SXP. Además se hace una descripción de la planificación del proyecto por roles, se explica el funcionamiento del software a través de modelos auxiliares, historias de usuarios y prototipos de interfaz de usuario.

Capítulo 4: **Validación de la solución propuesta:** En este capítulo se describen los casos de pruebas realizados a la aplicación en cada una de las iteraciones, se muestran los resultados obtenidos, se analizan y se presentan las funcionalidades alcanzadas hasta el período de desarrollo.

1. Capítulo 1. Fundamentación teórica de las aplicaciones de monitoreo de procesos

En este capítulo se analiza el funcionamiento de los sistemas de monitoreo de procesos, se estudia la situación actual a nivel mundial acerca del desarrollo de este tipo de sistemas, se identifica el mecanismo de monitoreo más apropiado para realizar la actividad de control de procesos y se analiza el funcionamiento del sistema de ventanas X Window, así como, herramientas, tecnologías y metodología de desarrollo a utilizar.

1.1. Sistemas de monitoreo de procesos

Un proceso es una instancia de un programa en ejecución. A los procesos frecuentemente se les refiere como tareas. Este contexto puede ser más procesos hijos que se hayan generado del principal (proceso padre), los recursos del sistema que esté consumiendo, sus atributos de seguridad, etc. Cada proceso que se inicia es referenciado con un número de identificación único, que es siempre un entero positivo.[1]

El monitoreo es un proceso continuo de recolección y análisis de datos cualitativos y cuantitativos. Permite realizar un conjunto de acciones periódicas y sistemáticas, de vigilancia, observación y medición de los parámetros relevantes de un sistema. Estos obtienen resultados acerca de las condiciones en tiempo real de una computadora, pueden registrar, por ejemplo, las estadísticas de los recursos utilizados por el equipo.

1.1.1. Funcionamiento de los sistemas de monitoreo

Los sistemas de monitoreo realizan tareas de control constantemente. Se comunican con procesos que supervisan el sistema operativo, que contienen información sobre el ordenador y para capturar los datos, utilizan funciones de bajo nivel. Por lo general estas aplicaciones están implementadas en el lenguaje de programación C/C++.

La monitorización de los sistemas puede realizarse a través de dos mecanismos, monitorización remota y

agentes locales instalados. La primera de estas, se realiza desde un ordenador hacia uno o varios de estos. Para establecer una conexión entre ambas partes, utilizan un protocolo[3] de red. Este tipo de monitoreo, puede representar de manera gráfica las acciones que se cometen en el ordenador que se está controlando.

El agente local, consiste en una aplicación que supervisa recursos del sistema operativo en el ordenador instalado. Su funcionamiento frente al monitoreo remoto básicamente es el mismo, la diferencia radica en que las herramienta locales no requieren de la conexión a una red, y por lo tanto la implementación es menos compleja y riesgosa.

La actividad de monitoreo varía en cuanto a la necesidad de lo que se quiera controlar, puede ser, el estado de los servicios de red, procesos de un ordenador o parámetros físicos del hardware. Teniendo en cuenta el propósito para el que se controla, el sistema puede ser visible o no al usuario objeto de vigilancia.

1.1.2. Características de los sistemas de monitoreo

El funcionamiento de los sistemas remoto, exige alta velocidad de conexión para obtener un rendimiento óptimo cuando la distancia y la cantidad de equipos son considerables. Son muy usados por la comodidad con que reflejan sus datos y generalmente están orientados al control de redes. Se recomienda su uso cuando se cuenta con un servicio de red fiable, ya que dependen totalmente de esta, por lo que no garantizan el control total de los recursos del sistema permanentemente. Su empleo puede ser muy costoso.

Las aplicaciones locales son ligeras, escasamente requieren recursos del sistema operativo y aseguran el control del mismo mientras es usado. Son independientes de la conexión a una red y facilitan integrarse a esta. Se sugiere su empleo, cuando es indispensable que un ordenador sea chequeado constantemente. El uso de estas herramientas pueden ser aplicadas con tecnología barata.

[3] Método estándar que permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos)

Este trabajo está centrado en la supervisión de procesos y manejo de ventanas que están siendo usadas, en especial la contenedora del foco. Se fundamenta en aplicaciones instaladas en cada ordenador, puesto que se requiere un constante monitoreo. Se independiza de la conexión remota, para evitar que las fallas en el servicio de red interrumpan el control de las actividades que se llevan a cabo en el equipo. Incluye además la notificación de eventos.

Un evento es cualquier suceso que activa un proceso en un sistema automatizado. Este término es muy empleado en las herramientas de supervisión, para advertir la aparición o desaparición de nuevos procesos. Puede ser empleado para distintos fines en la programación y está estrechamente relacionado con el término notificación[4].

La notificación de eventos es un recurso que se utiliza en el monitoreo, para controlar las aplicaciones que se ejecutan y generar la información específica de los eventos, objeto de vigilancia dentro de un SO (sistema operativo).

1.2. Aplicaciones para el control de procesos

A nivel mundial se han desarrollados diferentes sistemas de monitoreo, ya sean para controlar redes o vigilar los movimientos que efectúan los usuarios en los ordenadores. Actualmente los sistemas operativos privativos y libres poseen gran variedad de software de este tipo.

La administración de procesos es un recurso importante que se emplea para conocer el rendimiento de la computadora, a continuación se muestra una tabla de los sistemas más importantes que controlan procesos en el ordenador, ya sea vía remota o local.

Tabla de aplicaciones que monitorizan procesos.

Nombre del Sistema	Sistemas Operativos	Licencia
conky	GNU/Linux	GPL
Gkrellm	GNU/Linux	GPL
gnome-system-monitor	GNU/Linux	GPL

[4] Es el acto de comunicar una tramitación o petición.

Ksysguard	GNU/Linux	GPL
xfce4-taskmanager	GNU/Linux	GPL
WhoWatch	GNU/Linux	GPL
Spong	GNU/Linux	GPL LAP
Pandora FMS	GNU/Linux, AIX, Solaris, HP-UX,BSD/IPSO, Windows	GPL
Monit	GNU/Linux	GPL
Nagios	GNU/Linux, Unix	GPL
dmachinemon	GNU/Linux	GPL
rnas-gtk	GNU/Linux	GPL
vtgrab	GNU/Linux	GPL
Zenoss	GNU/Linux , Windows, Unix	GPL
Big Brother	GNU/Linux, Unix, Windows NT, OpenVMS, Netware, Mac OS/9	GPL
Personal Task Manager	Windows	Freeware
AnVir Task Manager	Windows	Freeware
Extended Task Manager	Windows	Freeware
Psiloc Task Manager	Symbia	Shareware
DLL	Windows	Freeware
FileMon	Windows	Freeware
ProcessQuickLink	Windows	Freeware
Security Task Manager	Windows	Evaluación
¡MonitorPC Enterprise	Windows	Freeware

Estas aplicaciones son desarrolladas para que funcionen en sistemas operativos concretos. Además varía la forma de manejar los recursos del sistema para lograr un producto integral, por lo que es ineludible que presenten características distintas. Los programas privativos permiten conocer su funcionamiento de

forma gráfica sin poder acceder al código implementado. Debido a que el presente trabajo está soportado bajo GNU/Linux por la carencia de un sistema libre que controle ventanas enfocadas, y la necesidad de supervisar estas actividades, se hace un estudio sobre los sistemas libres que vigilan procesos. A continuación se describen algunos de estos sistemas.

1.2.1. Conky

Es un monitor de sistema ligero y altamente personalizable. Su utilidad facilita acceder a todo tipo de información sobre el sistema de forma rápida y directa. El programa se coloca en una ventana o se integra al escritorio, convirtiéndose en parte del fondo de pantalla y actualizándose cada cierto tiempo, el cual es definido por el usuario.

La configuración de Conky es el principal obstáculo para quienes lo desean usar, ya que no permite una interacción con el usuario de manera gráfica, obligando a que se realice a través de archivos, aunque existen numerosas configuraciones disponibles en Internet.

Principales características:

- Escasamente utiliza 1Mb (Megabit) de memoria RAM (Memoria de Acceso Aleatorio), lo que le coloca por debajo de casi cualquier aplicación.
- Es ideal para sistemas con pocos recursos.
- Lista los procesos que están siendo usados.
- Monitorea la temperatura y la carga de la Unidad Central del Proceso (CPU).
- Muestra la carga de la red.
- Muestra el uso de la RAM y la SWAP (Memoria de zona de intercambio).

1.2.1. Gkrellm

Es el más completo en cuanto a administración de procesos se trata. Está basado en el conjunto de bibliotecas GTK+, es usado con diversos fines, como monitorear el estado de la memoria principal, microprocesadores, discos duros, interfaces de red, entre otros.

Cuando se instala Gkrellm, se incluye otra utilidad llamada Gkrellmd; este programa no es más que un servidor que escucha solicitudes de clientes Gkrellm, para poder monitorizar los equipos a través de una red. En la actualidad es uno de los administradores de procesos más populares.

Principales características:

- Monitoriza el estado del sistema desde una única barra.
- Incorpora monitores gestionados por un único proceso para reducir la carga del sistema, como el de memoria, sistemas de archivo con opción de montar y desmontar, reloj, calendario, mediador de batería, temperatura del CPU, entre otros.
- Lista procesos que están siendo usados.
- Soporta plugins para múltiples tareas.

1.2.1. Gnome-system-monitor

Es una utilidad de monitoreo del entorno de escritorio GNOME[5]. Presenta una interfaz asequible que permite hacer un seguimiento detallado de los procesos activos, debido a su estructura en arborescencia.

Principales características:

- Monitoriza los procesos que están siendo usados.
- Controla el rendimiento del CPU y la memoria.
- Permite acceder a gráficas históricas, como del uso de la memoria y CPU.
- Posee funcionalidades básicas de monitoreo, como iniciar, detener y graficar los recursos que está usando el sistema.

1.2.1. Ksysguard

Ksysguard es el monitor del sistema estándar del entorno de escritorio KDE[6]. Su apariencia gráfica es relevante, y sus funciones son semejantes a gnome-system-monitor. Al abrir la aplicación, esta muestra dos hojas de trabajo: tabla de procesos y carga del sistema.

[5] Por sus siglas en inglés significa, GNU Network Object Model Environment.
[6] Por sus siglas en inglés significa, K Desktop Environment.

Principales características:

- Lista los procesos en ejecución.
- Permite ver los procesos en forma de árbol.
- Muestra la gráfica de la carga del CPU, memoria física, memoria de intercambio y carga promedio del sistema.
- Permite enviar señales para suspender, continuar, cortar, interrumpir y terminar un proceso.
- Facilita observar los procesos del sistema y de los usuarios por separados.

1.2.1. xfce4-taskmanager

Es un simple administrador de procesos para el entorno de escritorio Xfce4. El diseño gráfico es amigable y fácil de comprender. Está pensado para sistemas con pocos recursos.

Principales características:

- La principal características es su ligereza.
- Lista los procesos que se ejecutan en el sistema.
- Muestra la carga y la memoria RAM del CPU.
- Muestra la prioridad de cada proceso y su identificador[7].
- Permite detener y terminar un proceso.

1.2.1. WhoWatch

Es una aplicación de consola que supervisa lo que se realiza en el sistema operativo. Cuenta con diversas opciones, las cuales son accedidas a través de combinaciones de teclas que son mostradas en la parte inferior de la consola. A pesar de ejecutarse en consola, presenta un diseño que ayuda al usuario a comprender su funcionamiento.

Principales características:

- Muestra información a los usuarios sobre los procesos utilizados, en forma de árbol.
- Visualiza la cantidad de procesos que se están usando.
- Identifica cada proceso con su usuario.

[7] El identificador o PID, es un número entero usado por el kernel para identificar un proceso de forma unívoca.

- Muestra la memoria utilizada, carga del CPU, fecha actual y el directorio desde donde se ejecuta una aplicación.
- Permite suspender un proceso.
- Detecta conexiones al ordenador, mostrando el usuario y la dirección de la computadora conectada.

1.2.1. Spong

Es un simple sistema de supervisión de red de paquetes. Ha sido desarrollado en el lenguaje de programación Perl. La monitorización se lleva a cabo de forma remota o local, según el fin para el que es usado y se ejecuta en consola.

Principales características:

- Cliente basado en el seguimiento (CPU, disco, procesos, registros, etc).
- Supervisa servicios de red.
- Muestra resultados a través de texto o de interfaz.
- Cuenta con servicio de mensajería basada en normas cuando se producen problemas.
- Es fácil de configurar.

1.2.1. Pandora FMS

Es un software de código abierto (Open Source) de monitoreo. Vigila aplicaciones de un sistema y permite conocer el estado de cualquier elemento de estos. Funciona en múltiples sistemas operativos, con agentes específicos para cada plataforma. Pandora FMS también monitoriza servicios TCP/IP.

Principales características:

- Supervisa la carga del procesador, disco, el uso de la memoria, los procesos en ejecución, archivos y factores ambientales como la temperatura.
- Detecta nuevos sistemas en red.
- Controles de disponibilidad o rendimiento.

- Alerta cuando detecta algún problema.
- Genera gráficas e informes en tiempo real.
- Conserva datos de meses.
- Arquitectura modular y escalable.

1.2.1. Monit

Es una herramienta desarrollada para sistemas Unix, es una utilidad para la gestión y el control de procesos, archivos, directorios y sistemas de ficheros. Puede hacer ping (Rastreador de Paquetes Internet) a un ordenador remoto, comprobar el puerto TCP/IP, realizar conexiones e iniciar o finalizar un proceso si este consume demasiados recursos. Está diseñado como un sistema autónomo y no depende de plugins ni bibliotecas especiales.

Principales características:

- Fácil de instalar y configurar.
- Permite ver el estado del servidor en el navegador web.
- Activa o desactiva el servicio de vigilancia.
- Inicia, para y reinicia los procesos y servicios.
- Configura dependencias entre los servicios.
- Informa de alerta de error en el servicio y la recuperación.
- Controla el rendimiento del CPU y la memoria.
- Chequea y monitorea archivos, directorios y dispositivos.
- Extensible, flexible y configurable para la notificación de alertas.
- Prueba las conexiones de red a los distintos servidores.
- Permite configuración de las normas preventivas para actuar antes de que se produzca un error.

1.2.1. Nagios

Está escrito en el lenguaje de programación C y es un sistema Open Source de monitorización de red.

Fue creado por Ethan Galsta. Es considerado por muchas personas como el padre de todos los sistemas de monitorización de GNU/Linux.

Principales características:

- Monitorización de servicios de red (SMTP,POP3,HTTP,ICMP,SNMP).
- Monitorización de los recursos de un ordenador (carga del procesador, uso de los discos, archivos del sistema) en varios sistemas operativos.
- Monitorización remota, a través de túneles SSL cifrados, o SSH.
- Diseño simple de plugins, que permiten a los usuarios desarrollar sus propios chequeos de servicios dependiendo de sus necesidades, usando diferentes herramientas como (Bash, C++, Perl, Ruby, Python, PHP, C#).
- Chequeo de servicios paralizados.
- Posibilidad de definir la jerarquía de la red, permitiendo distinguir entre ordenadores caídos inaccesibles.
- Notifica a los contactos cuando ocurren problemas en servicios, así como cuando son resueltos (vía correo electrónico, mensajería instantánea , o cualquier método definido por el usuario junto con su correspondiente complemento).
- Posibilidad de definir manejadores de eventos que se ejecuten al ocurrir un evento de un servicio.
- Interfaz web opcional, para observar el estado de la red actual, notificaciones, historial de problemas, archivos de registros, etc.

1.2.1. Comandos para monitorear procesos

En los sistemas Unix se llama proceso a un programa en ejecución y al objeto abstracto que crea el sistema operativo para manejar el acceso a los recursos del sistema (memoria, Unidad Central de Procesamiento, dispositivos de E/S). Pueden coexistir varias instancias de un mismo programa ejecutando en forma simultánea. Cada una de ellas es un proceso diferente.

Unix es un sistema multiproceso por tiempo compartido. A los ojos de un usuario en un momento dado

hay múltiples programas en ejecución, cada uno de ellos avanzando en su tarea. Sin embargo en una máquina con un solo procesador hay en cada instante solo un proceso en ejecución. Es el sistema operativo el que va rotando el uso del procesador a intervalos breves(algunas decenas de milisegundos) entre los procesos definidos en el sistema, de forma que se crea la ilusión que todos avanzan simultáneamente.

Para cada proceso definido en el sistema, el kernel[8] del sistema operativo almacena y mantiene al día varios tipos de información sobre el proceso. Esta información podemos ordenarla de la siguiente forma:

1- Información general. Identificadores de proceso, usuario y grupo.
2- Ambiente (variables, directorio actual, etc.).
3- Información de E/S.
4- Información de estado.

Los sistemas Unix cuentan con diferentes comandos que admiten visualizar los procesos en ejecución, estos son:

- ps: Imprime la lista de procesos en ejecución. Los campos de información más importantes desplegados por ps para cada proceso son: usuario, identificadores de proceso, uso de recursos reciente y acumulado, estado del proceso y comando invocado.
- top: Muestra los procesos usados en forma de tabla. Actualiza un reporte similar al generado por ps a intervalos periódicos. Normalmente top ordena los procesos por uso de CPU en forma decreciente. Al comando top se le ha desarrollado versiones gráficas para entornos de escritorio, como gtop para Gnome y ktop para KDE.
- pstree: Muestra los procesos en forma de árbol.
- w y uptime: Estos comandos reportan tres números que indican la carga promedio del sistema en un intervalo de 1, 5 y 15 minutos respectivamente. El parámetro reportado es el promedio de la cantidad de procesos listos para correr. Este valor es un indicador rápido de la actividad del

[8] Facilita a los programas el acceso seguro al hardware y administra los recursos del sistema.

sistema y suele vigilarse para detectar sobrecargas en el mismo, o registrarse periódicamente para un análisis posterior en caso de problemas.

- vmstat: Reporta varias estadísticas que mantiene el núcleo (kernel) sobre los procesos, la memoria y otros recursos del sistema. Alguna de la información reportada por vmstat es la siguiente:
  1-Cantidad de procesos en diferentes estados.
  2- Valores totales de memoria asignada a procesos y libre.
  3- Estadísticas sobre paginado de memoria.
  4- Uso de CPU reciente clasificado en inactivo, ejecutando en modo usuario y ejecutando en modo kernel.

Estos programas tienen un grupo de opciones que son utilizadas para mostrar datos como la carga de la Unidad Central de Procesamiento, el uso de la memoria RAM, el usuario y el identificador correspondiente a cada proceso, entre otros.
En el universo del software libre, existen múltiples herramientas que controlan procesos, sin embargo la cantidad de programas que se especializan en controlar atributos de ventanas es muy pobre, y estos para mostrar los resultados requieren de la colaboración del usuario, las aplicaciones son, xprop y xwininfo. Estas se ejecutan desde la consola y quedan en espera hasta que se de click en alguna parte de la pantalla. Xprop muestra los atributos de una ventana, al igual que xwininfo, como el identificador, la posición en pantalla, el tamaño, etc.

1.1. Sistema de ventanas X Window System

Los programas controladores, descritos anteriormente, permiten conocer el nombre de la aplicación que se ejecuta, pero no las características de la ventana que responde a un proceso.
Una ventana contiene los siguientes atributos:[2]

- Atributos asociados a la geometría de la ventana: posición, altura, ancho del borde y orden de apilamiento.
- Atributos no configurables (fijados en la creación): profundidad, clase (de entrada/salida o sólo de

entrada), visual (tipo de funcionamiento del hardware gráfico).

- Atributos configurables: fondo, nombre de la ventana, borde, gravedad de bit, gravedad de ventana, contenido de la ventana mientras la visibilidad es parcial, total o ninguna, manejo de eventos, mapa de color y la posición del cursor.

El presente trabajo está enmarcado en supervisar ventanas y conocer las características de esta, como el nombre y la aplicación a la que pertenece. Para obtener esta información, es imprescindible estudiar el funcionamiento del programa que las crea, e interactuar con este.

X-Window es un potente y complejo sistema de ventanas desarrollado en el Instituto Tecnológico de Massachusetts a mediados de 1980. Se ha convertido en el sistema de ventana estándar de los sistemas Unix, y en especial de GNU/Linux. Es distribuido independiente de la arquitectura de la máquina, del sistema operativo utilizado y del tipo de hardware empleado por la computadora. Fue creado con el objetivo de facilitar el desarrollo de programas con interfaz gráfica de manera sencilla, numerosas aplicaciones requieren de este recurso.

X Window está orientado a display[9]. En principio sólo se suele disponer de un display (el 0, ya que estos van numerados) y una pantalla. Cada posición en pantalla puede determinarse a partir de dos valores numéricos que representan las coordenadas del píxel[10], por ejemplo, (0,0) es la esquina superior izquierda de la pantalla y (1023,767) es la esquina inferior derecha en un display de una anchura y altura de 1024x768 píxel de resolución. Las ventanas de cada pantalla están organizadas en forma de árbol con una ventana raíz que ocupa toda la pantalla.
Con el display, la pantalla, y las coordenadas, X Window crea la ventana con los atributos solicitado en la posición de indicada. Para comprender como se realiza la petición de creación de ventana al sistema X Window, es indispensable conocer la arquitectura que soporta y su funcionamiento.

[9] Un display puede definirse como una o más pantallas y sus dispositivos de entrada.

[10] Es la menor unidad en color que forma parte de una imagen.

1.1.1. Arquitectura de X Window.

El sistema de ventanas X Window está basado en la arquitectura cliente/servidor, donde el servidor se encuentra en cualquier ordenador de la red, y el cliente en cualquier computadora conectada a esta. El servidor es un programa dedicado a suministrar servicios de display en una terminal gráfica, a petición del usuario. Además controla la pantalla y maneja el teclado y el ratón(u otros dispositivos de entrada). Igualmente, es responsable de la salida sobre la pantalla, el mapeado de colores, la carga de fuentes y el mapeado del teclado. Típicamente, los servidores se ejecutan en máquinas y terminales de trabajo de tipo gráfico y de alto rendimiento.

Los programas clientes que realizan peticiones al servidor son: (Ver Anexo #1)

Aplicaciones: Programas que facilitan al usuario la realización de un determinado tipo de trabajo.

Gestor de ventanas: El gestor de ventanas, es un programa cliente encargado de gestionar la manera en que son presentadas las ventanas en pantalla. Cuenta con algunos privilegios que le permiten mostrar la interfaz; para lograrlo, establece una comunicación con el servidor X y este construye y mapea la ventana que es solicitada. La comunicación en este esquema, se realiza mediante cuatro mensajes:[3]

- Petición: son mensajes de solicitud de servicio que envía el cliente al servidor, tales como: peticiones de creación, dimensionado o movimiento de una ventana y dibujo de primitivas gráficas. Estos mensajes son atendidos en el mismo orden en que son solicitados.
- Respuesta: posible mensaje de respuesta, en caso de que el cliente necesite una información proveniente del servidor.
- Evento: es un mensaje enviado del servidor al cliente cuando ocurre un evento en el ordenador que constituye el servidor, como pulsación de teclado o botón del mouse, movimiento de una ventana, etc.
- Error: Es enviado por el servidor al cliente en caso de error.

El intercambio de estos mensajes se realiza a través del protocolo X. Este se ejecuta sobre la conexión de red y permite que se efectúen solicitudes y respuestas entre cliente y servidor. Existen librerías que

permiten desde la programación, interactuar con los eventos que se producen entre un cliente y el servidor, a través de funciones comprensibles por ambos.

1.1. Modelo Cliente/Servidor

El presente trabajo requiere de centralizar los reportes producidos por la aplicación local instalada. Para ello se hace imprescindible buscar una arquitectura, esta es un diseño estructural que gobierna la interconexión y la interacción de los componentes que se conectan. Actualmente existen numerosas arquitecturas de este tipo, entre las más aplicadas están:

1. Arquitectura Redes de Pares (Peer to Peer (P2P)).
2. Arquitectura Cliente-Cola-Cliente.
3. Arquitectura Multi-capas.
4. Arquitectura Cliente /Servidor.

Las redes de pares se basan en la filosofía de compartir ficheros directamente de forma descentralizada y distribuida. Es muy utilizada cuando se trata de compartir datos, ya que todas las computadoras interconectadas gozan de la misma categoría y cada una brinda a la red, los recursos que su propio usuario le asigne. La arquitectura Cliente-Cola-Cliente, en la cual las redes P2P se basó inicialmente, habilita todos los nodos de la red para tratarlos como clientes simples. Esta arquitectura, al igual que la P2P no es factible para la solución de este trabajo, porque la información recopilada no tiene como propósito viajar hacia los demás ordenadores de la red para ser conocimiento de cualquier usuario. El modelo multi-capas, a diferencia de la arquitectura cliente/servidor, posee en algunos casos tres tipos de nodos en la red.

- Clientes que interactúan con los usuarios finales.
- Servidores de aplicación que procesan los datos para los clientes.
- Servidores de la base de datos que almacenan los datos para los servidores de aplicación.

Puesto que en este trabajo no se procesan datos para los clientes, sino que solo se almacena la información, no se necesita este modelo. Se utiliza la arquitectura cliente/servidor, ya que la información

coleccionada tiene que estar centralizada. Cliente/Servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí, para intercambiar información, servicios o recursos. (Ver Anexo #2)

El cliente es un programa que realiza peticiones e inicia un requerimiento de servicio.

Principales funciones del cliente:

- Generar consultas.
- Mantener y procesar el diálogo con el usuario.
- Manejar posibles errores.
- Permitir entrada de datos y validación.
- Gestionar ayuda.

El servidor es cualquier recurso de un cómputo conectado en una red dispuesto a proveer múltiples servicios a los clientes.

Principales funciones del servidor:

- Almacenar y organizar de datos.
- Actualizar los datos almacenados.
- Administrar recursos compartidos.
- Ejecutar toda la lógica para procesar una transacción.
- Procesar los elementos del servidor (datos, capacidad de CPU, almacenamiento en disco, capacidad de impresión, manejo de memoria y comunicación).
- Gestionar periféricos compartidos.
- Controlar el acceso concurrente a bases de datos compartidas.
- Permitir enlazar las comunicaciones con otras redes.

Entre las principales características de la arquitectura cliente/servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor.

- El cliente no depende de la ubicación física del servidor, del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Esta arquitectura reparte la capacidad de proceso entre el cliente y el servidor. La asignación de responsabilidades y la organización de sus componentes constituyen ventajas en un sistema. La división de estas dos partes, cliente y servidor, es conocido como separación de tipo lógico, debido a que el servidor no necesariamente se ejecuta en una sola máquina. Es recomendable que el servidor sea una máquina bastante potente, ya que necesita de más recursos para responder las solicitudes.

El servidor al iniciarse espera a que lleguen solicitudes de los clientes, por lo que adopta una posición pasiva en la comunicación. Al recibir una petición, la procesa y luego responde a los clientes. Tienen la capacidad de atender a varios clientes, aunque se puede restringir a un número limitado. Los clientes se mantienen activo, debido a que inician la comunicación con el servidor. Normalmente pueden conectarse a varios servidores y permanecer en espera hasta que se responda su demanda.

1.1.1. Ventajas y desventajas de la arquitectura cliente/servidor

Ventajas:

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor.
- La escalabilidad de este modelo facilita el aumento de la capacidad de clientes sin provocar daños en el servidor.
- La responsabilidad de la administración de los datos está centralizada en el servidor.
- El esquema cliente/servidor facilita la integración entre sistemas diferentes.
- Favorece la adaptación a cambios en la tecnología, pues facilita la migración de las aplicaciones a otras plataformas y al aislar claramente las diferentes funciones de una aplicación, hace más fácil incorporar nuevas tecnologías.
- Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es

posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio.

Desventajas:

- Las características del hardware y el software son generalmente determinantes.
- El servidor exige recursos para responder a los pedidos de los clientes.
- El servidor es el único eslabón débil en la red de cliente/servidor, debido a que toda la red está construida en torno a él.
- Es importante que los clientes y los servidores utilicen el mismo mecanismo (por ejemplo sockets), lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas.
- El desempeño es otro de los aspectos que se deben tener en cuenta en el esquema cliente/servidor. Problemas de este estilo pueden presentarse por congestión en la red, dificultad de tráfico de datos, etc.
- Montar esta arquitectura constituye un gasto mayor por parte del servidor.

1.1. Aplicación de monitoreo integrado a herramienta de gestión de proyecto.

El sistema de monitoreo propuesto, incorpora funcionalidades para aumentar el rendimiento de la producción en el laboratorio de proyecto. Para ello, se integra a un sistema de gestión de proyecto.

La gestión de proyecto es una disciplina de la Ingeniería de Software, que comprende un conjunto de herramientas y técnicas que una organización emplea para administrar recursos, de modo que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definido.

En la actualidad existen múltiples herramientas que se utilizan en la gestión de proyectos, entre ellas están: DotProject, eGroupWare, Basecamp, activeCollad, Gantt PV, GanttProject, Teamwork, Planner, trac, TUTOS, ToDoList, X-Man, entre otras. Estas son de vital importancia para darle cumplimiento a las actividades que se orientan en un grupo de desarrollo.

Unicornios usa DotProject como aplicación para la gestión. Los usuarios para colocar o revisar una tarea deben acceder a esta herramienta y operar de acuerdo a lo deseado. En muchas ocasiones cuando se coloca una actividad a una determinada persona que integra el proyecto, este no se da cuenta en el momento que se le asigna la tarea, debido a que la herramienta de gestión no notifica al usuario la nueva acción. Lo que trae como consecuencia incumplimientos en el plan de desarrollo de un producto. Para muchos usuarios estos sistemas de gestión son muy tediosos por la frecuencia con que deben revisar la herramienta para conocer la nueva tarea.

Esta investigación propone una solución a esta polémica. Consiste en integrar al sistema de monitoreo de procesos especializado en el manejo de ventanas enfocadas, un módulo que notifique eventos producidos por la incorporación de tareas en la herramienta de gestión. La notificación de eventos se lleva a cabo a través de mensajes de texto.

También DotProject cuenta con un módulo nombrado GENEST, este tiene como objetivo mejorar el proceso de gestión, para esto es se encarga de dibujar gráficos estadísticos sobre el uso de aplicaciones. Está compuesto por un sistema de navegación por pestañas que permite el acceso a cada uno de los diferentes grupos de información clasificadas en las siguientes: vista general, de usuarios, de proyectos y de reportes pdf. Actualmente su estado no le permite funcionar porque requiere de una aplicación que supervise las máquinas de producción utilizando tecnología XML y traslade los datos hacia el servidor del DotProject. Por esta razón se decide realizar la herramienta de supervisión orientada a hacer funcionar el módulo descrito.

1.2. Herramientas, lenguajes y tecnologías a utilizar

Para lograr un producto es importante el uso de tecnologías, lenguajes de programación y herramientas que ayudan a la solución del problema. Para solucionar el problema planteado se tiene en cuenta la necesidad de un desarrollo que controle ventanas en el sistema operativo GNU/Linux y el uso de herramientas libres, para construir un programa de código abierto. Esto facilita que a la aplicación implementada se le agregue nuevas funcionalidades, y sea más robusto.

1.2.1. Entorno integrado de desarrollo

Para implementar la aplicación, es ventajoso utilizar un entorno integrado de desarrollo (IDE) que facilite la programación. Este es un conjunto de herramientas útiles para el programador. Los componentes más comunes de un IDE son: editor de texto, compilador, intérprete, depurador y los plugins que cada IDE contenga como característica especial. Entre los más conocidos en el software libre están:

Eclipse:

Es una excelente herramienta desarrollada por IBM (International Business Machines), es multiplataforma y soporta lenguajes de programación como C, C++, Java, Perl, entre otros. El principal obstáculo para utilizarlo es su lentitud, ya que requiere de un ordenador con amplios recursos.

Kdevelop:

Es una aplicación que requiere de pocos recursos, soporta múltiples lenguajes y facilita el completado automático del código en C y C++. Está pensada para el entorno de escritorio KDE, aunque también funciona en otros entornos.

Anjuta:

Es un entorno integrado de desarrollo (IDE) desarrollado por Naba Kumbar en 1999. Fue desarrollado para programar en C y C++ en sistemas GNU/Linux aunque actualmente admite lenguajes como C#, Perl, Python y Bash. Su principal objetivo es trabajar con GTK[11] y en el escritorio GNOME, además ofrece un gran número de características avanzadas de programación. Anjuta es un software libre, liberado bajo la licencia GPL. [4]

Anjuta incluye:

- Administrador de proyectos.
- Asistentes.
- Plantillas.
- Depurador interactivo.

[11] Biblioteca que contiene objetos y funciones para desarrollar interfaces gráficas de usuarios.

- Editor de texto; que verifica y resalta las sintaxis escritas.
- Extensión para Subversion (control de versiones).
- Interprete de comandos propio.

A pesar de incluir abundantes utilidades y tener una amplia variedad de funcionalidades es un sistema bastante estable y muy ligero en el consumo de recursos. Por sus características es usado para la realización de script[12] en el lenguaje Bash.

Code::Blocks:

Es el IDE empleado para la implementación de la herramienta que se propone, soporta los lenguajes de programación C y C++. Proporciona a los usuarios una completa interfaz de trabajo para escribir su código con facilidad, diseñado para ser muy extensible y totalmente configurable. Es una herramienta de software libre basada en la plataforma de interfaces gráficas WxWidgets, lo que permite correr libremente en diversos sistemas operativos.

Code::Blocks detecta los compiladores instalados en el sistema, facilitando elegir el más adecuado para compilar. Facilita el uso de una serie de librerías populares para desarrollar aplicaciones con interfaz gráfica para usuarios (GUI), como GTK+, QT4, FLTK y wxWidgets, siendo estas las librerías más usadas en los sistemas libres.

Características generales: [5]

- Es open source, con licencia GPLv3.
- Es multiplataforma, funciona en los sistemas operativos: GNU/Linux (Ubuntu y Debian), Windows 2000 / XP / Vista y Mac OS X 10.4+.
- Es extensible mediante plugins.
- Importa proyectos desarrollados en Dev-C++ y MSVC.
- Consta de espacios de trabajo para combinar múltiples proyectos.
- Facilita el completamiento de código.
- Vista de proyectos en forma de árbol.

[12] Conjunto de instrucciones que permiten automatizar una tarea.

- Desensamblado de código.
- Volcados de memoria.
- Múltiples hilos (threads).

Soporte de compiladores

Una de sus ventajas es la de poder enlazar una variedad de compiladores para realizar su trabajo. Entre los múltiples compiladores soportados están:[6]

- MSVC++
- GNU Compiler Collection (MingW / GNU GCC).
- Borland C++ 5.5.
- Digital Mars Compiler.
- Open Watcom.

Si estos compiladores están instalados pueden ser detectados automáticamente al iniciar Code::Blocks. De estos se utiliza el GCC, ya que es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr.

Característica de compilación:[7]

- Compila directamente o con archivos compuestos por un conjunto de dependencias y reglas.
- Proyectos con múltiples objetivos.
- Estadísticas y resumen de código.
- Soporte para compilación en paralelo.
- Dependencias entre proyectos dentro del espacio de trabajo.

1.2.1. Lenguaje C/C++

El lenguaje C nace en los Laboratorios Bell de AT&T y originalmente está muy ligado al sistema operativo Unix, ya que su desarrollo se realizó en este sistema. C está inspirado en el lenguaje Basic (B), escrito por

Ken Thompson en 1970 con intención de recodificar a Unix, que en la fase de arranque está escrito en ensamblador[13], en vistas a su transportabilidad a otras máquinas. B era un lenguaje evolucionado e independiente de la máquina, inspirado en el Lenguaje de Programación Básico Combinado (BCPL) concedido por Martin Richard en 1967.

En 1972, Dennis Ritchie modifica el lenguaje B, creando el lenguaje C y reescribiendo a Unix en dicho lenguaje. La novedad que proporcionó el lenguaje C sobre el B fue el diseño de tipos y estructuras de datos.

Una de las peculiaridades de C es su riqueza de operadores. Puede decirse que prácticamente dispone de un operador para cada una de las posibles operaciones en código máquina. C es un lenguaje de programación de propósito general. Sus principales características son: [8]

- Programación estructurada.
- Economía de las expresiones.
- Abundancia en operadores y tipos de datos.
- Codificación en alto y bajo nivel simultáneamente.
- Reemplaza ventajosamente la programación en ensamblador.
- Utilización natural de las funciones primitivas del sistema.
- Producción de código objeto altamente optimizado.
- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.
- Los argumentos de las funciones se transfieren por su valor.

C++ es un lenguaje diseñado a mediados de los años 1980, por Bjarne Stroustrup. Es un superconjunto de C que tiene como característica principal, el soporte para programación orientada a objetos y de plantillas o programación genérica. Se creó para añadirle cualidades y características de las que carecía C. Los paradigmas que abarca C++ son:

- Programación estructurada.

[13] Lenguaje de programación de bajo nivel muy cercano al código máquina.

- Programación genérica.
- Programación orientada a objetos.

Este lenguaje mantiene la potencia para programar a bajo nivel y se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

Una de las razones de programar en C++ es su increíble versatilidad. Con él pueden programarse desde las aplicaciones más simples, a las más complejas como sistemas operativos. Su portabilidad permite que un programa escrito en C++, pueda compilarse en cualquier sistema sin necesidad de cambiar apenas el código.

Este lenguaje facilita el uso de dos librerías (Libxml2 y Xlib) muy útiles para la solución de este trabajo. Puesto que la información recopilada por la herramienta de control es enviada en un documento de manera organizada, se necesita integrar el sistema de supervisión a la generación de documentos a través del lenguaje de marcas extensible (XML).

1.2.1. Libxml2

Libxml2 es una biblioteca y un parser en lenguaje C que facilita el manejo de los documentos XML. La librería permite el acceso para lectura y escritura. También se puede utilizar en otros lenguajes como Java, C++, Perl, Python, Ruby, PHP, C# y Pascal.

1.2.2. Xlib

Para monitorear ventanas y obtener los datos deseados, es imprescindible un intermediario en el sistema X Window, que proporcione la información pedida, esta es la librería Xlib. Esta es una biblioteca de bajo nivel, que aparece en 1985 y está basada en la filosofía de eventos. Constituye la Interfaz de Programación de Aplicaciones (API) básica de X Window y tiene un conjunto de funciones y macros realizadas en C. Es la librería de más bajo nivel que controla el protocolo X. Entre las bibliotecas que utilizan Xlib están:

- Intrinsics(Xt).

- Xaw
- Motif
- GTK+
- Qt(versión para X11)

El conocimiento de las funciones de bajo nivel hacen lo mismo que otras librerías de una forma más adaptada a las necesidades del desarrollador, proporciona más control al programador, produce programas pequeños, rápidos y eficientes. Este trabajo utiliza la librería xlib, para conocer las peticiones de creación de ventanas que hacen los diferentes clientes (aplicaciones y gestor de ventanas), los procesos que son creados o eliminados, y los eventos producidos por los dispositivos de entrada. Con el manejo de estos recursos, la aplicación es capaz de monitorear ventanas.

1.2.1. Hilos

Un hilo de ejecución es una unidad de procesamiento. Este, a través de la programación, permite escribir programas más eficientes al facilitar a una aplicación realizar varias tareas concurrentemente. Los hilos dentro de un proceso comparten el mismo espacio de memoria. Esta técnica posibilita que aquellos procesos que pueden requerir un tiempo considerable en llevarse a cabo, trabajen paralelamente al proceso principal.

1.2.2. Colas de mensajes

Una cola de mensajes es una estructura de datos gestionada por el kernel, en la cual van a poder escribir y leer los procesos que se ejecuten en el sistema. Los mecanismos de sincronismo para que no se produzca colisión son responsabilidad del kernel.

Los datos que se escriben en la cola deben tener formato de mensaje y son tratados como un todo indivisible. La estructura de datos que forma la cola de mensajes tiene una forma definida: un identificador, un campo que marca el tipo de mensaje y un área de datos. A una cola de mensajes puede acceder cualquier proceso que conozca su identificador y tenga los permisos necesarios.

El kernel gestiona las colas de mensajes como un mecanismo FIFO (first input, first output), es decir, el primer mensaje escrito en la cola será el primero en salir de ella al realizar una lectura. Sin embargo, para darle más flexibilidad, se pueden hacer peticiones de lectura que extraigan mensajes de un tipo determinado, con lo que se rompe la gestión de tipo FIFO, aunque ésta se sigue manteniendo para todos aquellos mensajes que son de un mismo tipo. Esta clasificación de los mensajes por tipos permite distinguir cuáles son los mensajes que van destinados a cada uno de los procesos lectores.

1.2.3. Socket

La comunicación entre la aplicación de supervisión y el servidor se realiza utilizando socket. Este es un método para la comunicación entre dos programas que corren sobre ordenadores distintos o en el mismo ordenador. Este se define como el punto final en una conexión. Se crean y se utilizan con un sistema de peticiones o de llamadas de función. Es usado para intercambiar flujos de datos. Un socket queda definido por una dirección IP (Es un número que identifica un ordenador dentro de una red que utilice el protocolo de Internet), un protocolo y un número de puerto.

Los sockets permiten implementar una arquitectura cliente/servidor. Uno de los programas debe estar ejecutado y en espera de que otro quiera conectarse a él, el primero es el cliente y el segundo el servidor. Existen básicamente dos tipos de canales de comunicación o sockets, los orientados a conexión y los no orientados a conexión.

En el primer caso ambos programas deben conectarse entre ellos con un socket y hasta que no esté establecida correctamente la conexión, ninguno de los dos puede transmitir datos. Esta es la parte del Protocolo de Control de Transmisión (TCP) del Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP), y garantiza que todos los datos van a llegar de un programa a otro correctamente. Se utiliza cuando la información a transmitir es importante, no se puede perder ningún dato y no importa que los programas se queden bloqueados esperando o transmitiendo datos. Si uno de los programas está atareado en otra cosa y no atiende la comunicación, el otro quedará bloqueado hasta que el primero lea o escriba los datos.[9]

En el segundo caso, no es necesario que los programas se conecten. Cualquiera de ellos puede transmitir datos en cualquier momento, independientemente de que el otro programa esté escuchando o no. Es el llamado Protocolo de Datagrama de Usuario (UDP), garantiza que los datos que lleguen sean correctos, pero no que todos lleguen a su destino. Se utiliza cuando es muy importante que el programa no se quede bloqueado y no importa que se pierdan datos. [10] Por esta razón se utiliza socket orientado a conexión.

1.2.4. XML

XML son las siglas de Extensible Markup Language, es un metalenguaje que permite a los diseñadores crear sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones. Es una tecnología muy útil y valiosa para la representación y el intercambio de datos. XML es una versión de SGML (Structured Generalized Markup Language) y un conjunto de reglas que sirven para definir etiquetas semánticas y organizar un documento. Permite manejar datos complejos y proporcionar diferentes vistas.

La estructura de un XML se compone por un conjunto ilimitado de etiquetas, cada una de estas debe abrirse y cerrarse respectivamente luego del contenido adicionado entre ambas. Este metalenguaje a diferencia del Lenguaje de Marcas de Hipertexto (HTML), separa el contenido de la presentación.

Los fundamentos del XML son muy sencillos y en principio, las únicas herramientas que nos hacen falta son:[11]

- Un editor para escribir los documentos XML, por ejemplo el notepad en el sistema operativo Windows y el emacs en GNU/Linux.
- Un procesador o parser XML.

Es además un estándar internacional que presenta una estructura jerárquica y su análisis sintáctico es fácil debido a las estrictas reglas que rigen la composición del documento. Es de vital importancia la aplicación de esta tecnología en el desarrollo de la implementación, ya que permite organizar la información que es enviada al servidor central.

1.2.1. Bash

Bash es un intérprete de comandos (shell) de tipo Unix escrito para el proyecto GNU. Su nombre es un acrónimo de Bourne-Again Shell. Es el shell por defecto en la mayoría de las distribuciones de GNU/Linux en la actualidad. Es el encargado de leer los caracteres tecleados por los usuarios, los interpreta y los ejecuta.

Para el desarrollo de este trabajo, se necesita conocer el valor de algunas variables del sistema que posee el intérprete de comandos, por ejemplo, el usuario que está trabajando en el sistema, además Bash facilita ejecutar comandos para crear y borrar copias de seguridad generada por la aplicación cliente, esto ocurre cuando el cliente no puede enviar la información al servidor porque el ordenador es reiniciado o apagado inesperadamente.

1.2.2. Dia

Dia es un programa de creación de diagramas basado en GTK+ bajo la licencia GPL. Está inspirado en el programa comercial de Windows ‘Visio’, y puede ser usado para dibujar diferentes tipos de diagramas. Dispone de una serie de extensiones para ayudar en la elaboración de diagramas entidad interrelación, UML, flujo de datos, diagramas de red, entre otros. Dia incluye una herramienta para generar código a partir de los diagramas realizados. Puede cargar y guardar los diagramas personalizados a un formato XML, exportar diagramas a una serie de formatos, incluyendo EPS, SVG, XFig, WMF y PNG, e imprimir los mismos.

1.2.3. Metodología ágil a utilizar

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayuda a la documentación para el desarrollo de programas. En la actualidad es imprescindible para obtener alta calidad en los resultados de un programa.

Las metodologías clásicas tradicionales, no facilitan el desarrollo rápido de aplicaciones, pues requiere de una excesiva cantidad de documentación, además no se adaptan a los cambios, por lo que no son

métodos adecuados cuando se trabaja en un entorno donde los requisitos pueden variar. Las metodologías no ágiles, es aplicable cuando el equipo de desarrollo es numeroso, no así en las ágiles, que son usadas en equipos pequeños (se recomienda para menos de diez personas).

Entre las metodologías ágiles están:

- Extreme Programming (XP).
- Scrum .
- Familia de Metodologías Crystal.
- Feature Driven Development.
- Dynamic Systems Development Method.
- Adaptive Software Development.
- Open Source Software Development.

XP (Programación Extrema)

La metodología XP empieza con cuatro valores: Comunicación, Retroalimentación, Simplicidad y Coraje. En esta plataforma XP construye un proceso de diseño evolutivo que se basa en refactorizar un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, lo que es más, combina la disciplina con la adaptabilidad de una manera que indiscutiblemente la hace la más desarrollada de entre todas las metodologías adaptables.

SXP

SXP es la metodología empleada, está compuesta por las metodologías SCRUM y XP que 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.

SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos, de forma que sepamos por dónde andamos.

XP más bien es una metodología encaminada para el desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Consta de 4 faces principales, Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto. Desarrollo, es donde se realiza la implementación del sistema hasta que esté listo para ser entregado. Entrega, puesta en marcha; y por último mantenimiento, donde se realiza el soporte para el cliente.

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 nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

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. (Ver Anexo #3) [12]

1.1. Conclusiones

En este capítulo se ha mostrado al lector de forma general los aspectos teóricos a tener en cuenta para el desarrollo de una herramienta de supervisión y notificación de eventos. Se describió el monitoreo remoto y a través de agentes locales, y se tomó este último por la manera en que se desea controlar las aplicaciones. Además se hizo un estudio del sistema de ventanas X Window System, y la forma de obtener resultados a partir de este. Para desarrollar la aplicación se analizaron diferentes Entornos de Desarrollo Integrado (IDE) que participan en la implementación que se quiere realizar y se tomaron, Anjuta para la realización de script en Bash y Code::Block para la implementación de la herramienta propuesta.

2. Capítulo 2. Descripción y análisis de la solución propuesta

En este capítulo se hace un análisis de la solución propuesta. Se especifica y justifica la utilización de herramientas ya existentes que formarán parte de la solución del problema a resolver, así como la estrategia de integración al software.

2.1. Propuesta del sistema a implementar

Luego de un estudio de las herramientas existentes para el control de procesos, se llegó a la conclusión de que ninguna representa una solución integral a la problemática presente en Unicornios, laboratorio de proyecto de la facultad 10. Las aplicaciones que controlan procesos en el sistema operativo GNU/Linux, no permiten conocer las características de las ventanas construidas por el servidor a petición de un programa cliente, ni tampoco notificar eventos de la herramienta de gestión de proyecto DotProject. Producto a esto se decide comenzar, a partir del estudio realizado, el desarrollo de la aplicación que supervisa y notifica tareas.

La aplicación funciona la mayor parte del tiempo en modo invisible, supervisando las ventanas enfocadas que el usuario utilice durante su estancia en el ordenador. Los datos son recopilados utilizando la tecnología XML y al producirse el cierre de sesión son enviados a un servidor central.

La herramienta cuenta con archivos de configuración que permiten definir las aplicaciones que se desea monitorear. Constantemente realiza una salva de lo que controla, lo que facilita que en caso de producirse un cierre forzoso del sistema, no se pierda la información.

El envío del documento XML hacia el servidor se realiza utilizando socket orientado a conexión, para asegurar que los datos que viajan desde un ordenador a otro lleguen completos, para esto el programa se queda bloqueado hasta que se realiza la operación.

2.2. Reutilización de código

La reutilización permite un aprovechamiento más eficiente del código ya construido. El software

reutilizable reduce los costes de diseño, codificación y comprobación, porque se amortiza el esfuerzo a lo largo de varios años. Al reducirse la cantidad de código se simplifica también la comprensión, lo cual hace que aumente la probabilidad de que el código sea correcto.

En la implementación de la aplicación se utiliza esta técnica. Se aplican componentes ya existentes para establecer la comunicación entre el cliente y el servidor, el código reutilizable posee funciones de apertura de conexiones, escritura y lectura de datos, y cierre de comunicación.

También se reutiliza código en la aplicación de la tecnología XML para generar la información de forma organizada, se usa un parser implementado, el cual es un módulo encargado de analizar la estructura gramatical del documento.

En la transmisión de datos entre los procesos de monitoreo y la generación de documentos XML, se emplea componentes reutilizables, logrando una eficiente transportación de la información. Con el aprovechamiento de este recurso se logra ahorrar tiempo y esfuerzo en el proceso de desarrollo del software. Es además una excelente técnica para implementar sistemas que funcionen correctamente.

2.3. Por qué utilizar el sistema X Window System

X Window Systems es un sistema de ventanas orientado a red (network-based window system), independiente del hardware y del sistema operativo, y considerado ampliamente como el estándar en este contexto. El éxito de X viene dado en que:

- Es un entorno abierto, ya que el propietario de los derechos es MIT, que lo distribuye libremente. Los fuentes son gratuitos.
- Es un entorno gráfico de ventanas que nace en un momento en el que el mercado comenzó a demandar interfaces gráficas de usuario (GUI).
- Se ha beneficiado también del éxito de Unix como sistema operativo, aunque X no se diseñó específicamente para este.
- Es transportable, debido a que existen versiones para casi cualquier plataforma.
- X, a nivel de ejecución y debido a su naturaleza de protocolo de red, puede ejecutarse en una

máquina y sacar resultados en otra.

- Varios programas pueden utilizar los mismos recursos (ratón, teclado, pantalla).

Todas estas características convierte a X Window en una herramienta poderosa que justifica su utilización para manipular gráficos en SO[14]. Este manipulador de ventanas puede brindar datos importantes sobre lo que sucede en el ordenador, como las dimensiones de una ventana, conocer la aplicación a la que pertenece, saber el estado de visibilidad(totalmente visible, parcialmente visible o totalmente invisible), conocer la ventana padre y las hijas, permite realizar cálculos analíticos, partiendo de la obtención de la resolución de pantalla, etc. A diferencia de las diferentes herramientas de monitoreo estudiadas, con la utilización del sistema X Window se logra obtener una información más precisa sobre lo que el usuario está realizando en el máquina.

2.1. Xlib y XCB

El nivel más bajo de interface con X en lenguaje C es Xlib, que permite el acceso a todas las posibilidades del protocolo X. Este ha sido utilizado para el desarrollo de diversas aplicaciones. Se caracteriza por su potencial debido a que estas librerías permiten escribir aplicaciones efectivas sin herramientas de programación adicionales que permitan el uso de ciertas tareas de alto nivel.

A pesar del éxito de Xlib, surgió XCB con el objetivo de sustituir a Xlib. XCB (XC Binding) es una interfaz de programación de aplicaciones de bajo nivel para el servidor X Window. Este proyecto fue iniciado por Bart Massey. Los desarrolladores de la nueva alternativa reconocen a Xlib como un excelente trabajo, pero explican que hay aplicaciones para la que no es ideal.

A pesar de que en la actualidad ha sido utilizado XCB en aplicaciones como Awesome(gestor de ventanas), su extensibilidad aún no abarca el alcance de Xlib y la documentación en la web es mucho menor que la que posee Xlib. Sin embargo muchas funciones se asemejan a sus homólogas en Xlib, tanto en su nombre, como en los datos que requieren, facilitando la comprensión de la estructura de sus funciones. Por esta razón, se decide utilizar en la primera versión de la solución propuesta, Xlib.

[14] Sistema Operativo.

2.2. Arquitectura de la aplicación cliente

Para la implementación de un software, es importante tener en cuenta la arquitectura que se va a emplear. En el libro “Ingeniería de Software, un enfoque práctico”, de Roger S. Pressman, quien es una autoridad internacionalmente reconocida en la mejora del proceso del software y en las tecnologías de la Ingeniería del Software, cita tres razones claves por las que la arquitectura de software es importante:[13].

- Las representaciones de la arquitectura de software facilitan la comunicación entre todas las partes interesadas en el desarrollo de un sistema basado en computadora.
- La arquitectura destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería del software que sigue, y es tan importante en el éxito final del sistema como una entidad operacional.
- La arquitectura constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes.

Basado en estos elementos se realiza el análisis de la arquitectura de la aplicación. En el desempeño de este trabaja, inicialmente se descartó la necesidad de utilizar un modelo cliente/servidor, debido a que la información recopilada necesita estar centralizada. Para escoger este esquema, se hizo un estudio sobre las diferentes arquitecturas aplicables para la transportación de ficheros entre distintos ordenadores, y se determinó que esta sería la más indicada[15].

Para el desarrollo de la aplicación cliente, se define la utilización de diversos componentes, con el propósito de facilitar la integración de nuevas funcionalidades de forma independiente. En correspondencia a lo expresado se tomó como arquitectura a utilizar la “Arquitectura centrada en datos”.

En el centro de esta arquitectura se encuentra un almacén de datos (por ejemplo, un documento o una base de datos) al que otros componentes acceden con frecuencia para actualizar, añadir, borrar o bien modificar los datos del almacén. El software cliente accede a un almacén central. En algunos casos, el almacén de datos es pasivo. Esto significa que el software de cliente accede a los datos

[15] Ver capítulo1: Ventajas de la arquitectura cliente/servidor.

independientemente de cualquier cambio en los datos o de las acciones de otro software cliente. Una variación en este acceso trasforma el almacén en una pizarra que envía notificaciones al software de cliente cuando los datos de interés del cliente cambian.[14]

Las arquitecturas basadas en los datos promueven la capacidad de integración. Por consiguiente, los componentes existentes pueden cambiarse o los componentes del nuevo cliente pueden añadirse a la arquitectura sin involucrar a otros clientes (porque los componentes del cliente operan independientemente).[15] (Ver Anexo #4)

En la solución propuesta el almacén adopta una posición pasiva, por lo que no se utiliza un estilo de pizarra. La posición pasiva en la "Arquitectura centrada en datos" descrita por Roger S. Pressman también es nombrada como estilo de repositorio en bibliografías como el documento científico titulado "Estilos y Patrones en la Estrategia de Arquitectura de Microsoft"[16].

Entre las principales ventajas de la arquitectura usada están:

- Forma eficiente de compartir grandes cantidades de información.
- No hay necesidad de transmitir información entre los distintos componentes del sistema.
- Distintos componentes no necesitan qué datos producen o consumen otros.
- Las actividades de respaldo, seguridad, control de acceso y recuperación de errores están centralizadas.
- El formato de los datos es visible, de modo que es posible desarrollar nuevos componentes con ese formato.

2.3. Conclusiones

Con la realización de este capítulo, se realiza un análisis sobre la solución propuesta. Se describe de forma general como debe funcionar la aplicación y la importancia de la reutilización de código existente. Se explica por qué utilizar el sistema de ventanas X Window System, a diferencia de los sistemas de monitoreo estudiados. Para obtener un producto que sea flexible en la incorporación de nuevos

[16] Ver sección Bibliografía para localizar el acceso a la información.

componentes se realizó una investigación sobre las diferentes arquitecturas de software descritas por Roger S. Pressman, y se decidió utilizar una arquitectura centrada en datos.

3. Capítulo 3. Desarrollo ágil de la aplicación propuesta

En el transcurso de este capítulo se realiza la ingeniería de software a la herramienta de monitoreo, para ello se utiliza la metodología de desarrollo ágil SXP. Se explica toda la secuencia del proyecto en forma de historias de usuarios, prototipos de interfaces y algunos modelos auxiliares. Se analiza la arquitectura adecuada para desarrollar el sistema y se explica los diversos componentes que la integran.

3.1. Planificación del proyecto por roles

El desarrollo de software con metodologías ágiles exige de la creación de pequeños grupos de trabajo, donde los roles son pocos, pero están bien definidas sus actividades. El principal aspecto antes de comenzar el proceso de documentación es distribuir las tareas por cada uno de los roles existentes, lo que garantiza un trabajo organizado, de ahí la necesidad de tenerlos bien definidos. [16]

Planificación del proyecto por roles

Rol	Responsabilidad	Nombre
Gerente (Management)	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto.	Rodolfo González Pelegrino
Cliente (Customer)	El cliente participa en las tareas que involucran la lista de reserva del producto.	Abel Meneses Abad
Programadores (Programmers)	Es el encargado de producir el código y escribir las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.	Rodolfo González Pelegrino
Analista (Analyst)	Es el encargado de escribir las historias de usuario y las pruebas funcionales para validar su implementación.	Rodolfo González Pelegrino
Diseñadores	Encargados del diseño del sistema; así	Rodolfo González Pelegrino

(Designers)	como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.	
Encargado de Pruebas (Tester)	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Rodolfo González Pelegrino
Arquitecto (Architect)	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.	Rodolfo González Pelegrino

3.2. Modelo de Dominio

A partir de la concepción inicial de la solución propuesta se obtiene una visión general, por lo que se procede a modelar el negocio para comprender la interacción del sistema con los usuarios. En este caso, el negocio no esté bien definido entre el clientes y el ejecutor del proyecto, por lo que se realiza el Modelo de Dominio.

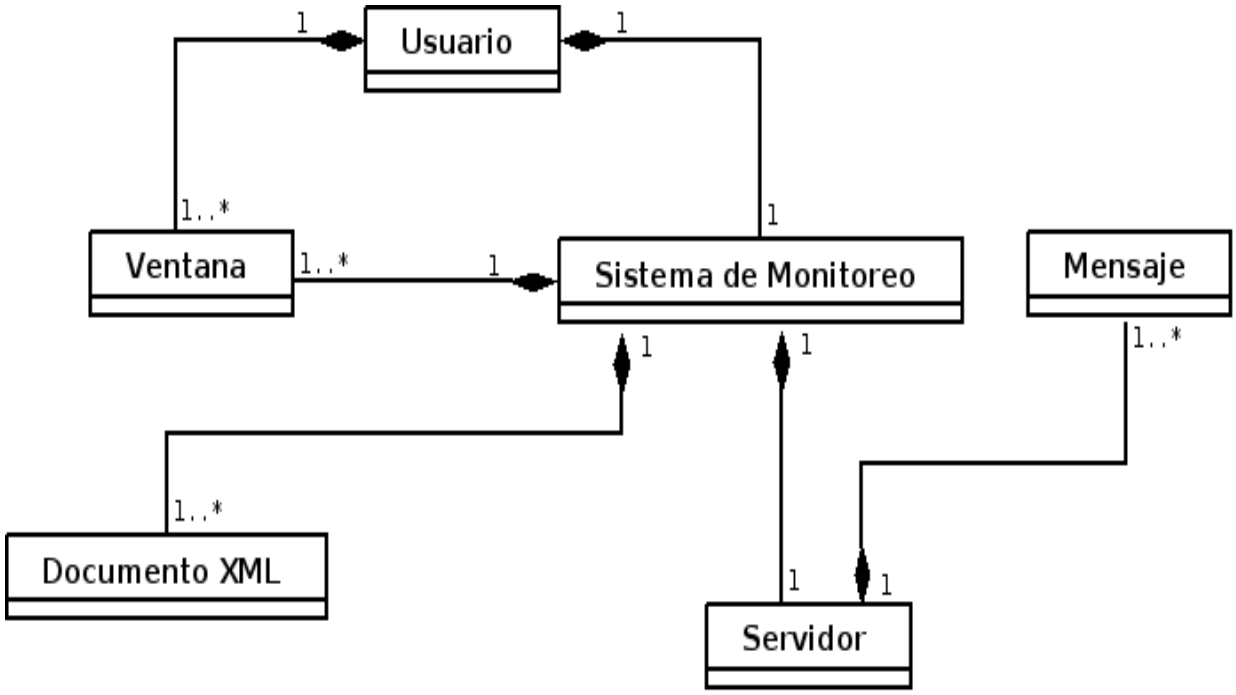


Fig. Modelo de dominio de la solución propuesta.

3.3. Lista de Reserva del Producto

El levantamiento de requisitos es una actividad primordial para el desarrollo de cualquier sistema, en el que se definen las necesidades de los clientes para construir un software. Estos son reflejados en la plantilla Lista de Reserva del Producto (LRP), el cual es el primer artefacto generado en la captura de requisitos y en el que se incluyen las funcionalidades que el sistema debe cumplir para su implementación. Esta plantilla está conformada por una lista de actividades priorizadas que define el trabajo que se va a realizar.

Esta lista puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto. 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. [17]

Lista de Reserva del Producto de la solución propuesta

Asignado a	**Ítem ***	**Descripción**	**Estimación**	**Estimado por**
Prioridad		**Muy Alta**		
Rodolfo	1	Identificar display y pantalla de trabajo.	1	Programador
Rodolfo	2	Establecer comunicación con el servidor X Window System.	1	Programador
Rodolfo	3	Capturar eventos del servidor X.	1	Programador
Rodolfo	4	Calcular el tiempo de uso de una ventana.	1	Programador
Rodolfo	5	Obtener el identificador de una ventana.	1	Programador
Rodolfo	6	Obtener el nombre de la aplicación a la que pertenece una ventana.	1	Programador
Prioridad		**Alta**		
Rodolfo	1	Enviar y recibir datos entre procesos en el mismo ordenador.	1	Programador
Rodolfo	2	Identificar el uso de una aplicación.	1	Programador
Rodolfo	3	Diseñar la estructura del XML.	1	Diseñador
Rodolfo	4	Generar documentos XML.	1	Programador
Rodolfo	5	Definir etiquetas invariables	1	Analista
Rodolfo	6	Enviar reportes al servidor.	1	Programador
Rodolfo	7	Identificar usuario del sistema.	1	Programador

Rodolfo	8	Identificar la hora del sistema.	1	Programador
Rodolfo	9	Integrar todos los componentes creados.	1	Encargado de Pruebas
Rodolfo	10	Notificar tareas de producción a usuario	1	Programador
Prioridad		**Media**		
Rodolfo	1	Implementar script de inicio del sistema de monitoreo.	1	Programador
Rodolfo	2	Diseñar ventana.	1	Diseñador
Prioridad		**Baja**		
Rodolfo	1	Mostrar mensaje de advertencia.	1	Programador
RNF (Requisitos No Funcionales)				
Rodolfo	1	El sistema deberá funcionar sobre el sistema operativo GNU/Linux.	1	Gerente
Rodolfo	2	El sistema debe ser bien documentado, posibilitando que se consuma el menor tiempo durante su mantenimiento.	1	Gerente

3.4. Historias de usuarios

En la metodología SXP predominan factores importantes como la comunicación y el diseño simple, para ello se basan en las historias de usuarios donde se representa los requerimientos del sistema. Cada historia incluye una o varias tareas que responden a la solución de un requerimiento, se relacionan con la prioridad que tienen y los usuarios que se encargan de desarrollarlas. Además de contener prototipos[17] de interfaz de usuario.

[17] Los prototipos ayudan a identificar comunicar y probar un producto antes de crearlo.

Historia de Usuario U-AMVEX-01.

<table>
<tr><th colspan="2">Historia de Usuario</th></tr>
<tr><td>Número: U-AMVEX-01</td><td>Nombre Historia de Usuario: Establecer conexión con el servidor X.</td></tr>
<tr><td colspan="2">Modificación de Historia de Usuario Número: Ninguna.</td></tr>
<tr><td>Usuario: Rodolfo González Pelegrino.</td><td>Iteración Asignada: 1</td></tr>
<tr><td>Prioridad en Negocio: Alta
(Alta / Medio / Bajo)</td><td>Puntos Estimados: 3</td></tr>
<tr><td>Riesgo en Desarrollo: Alta
(Alta / Medio / Bajo)</td><td>Puntos Reales: 3</td></tr>
<tr><td colspan="2">Descripción: Se establece conexión con el servidor del sistema X Window.</td></tr>
<tr><td colspan="2">Observaciones: El sistema X Window trabaja en modo cliente servidor, donde el servidor es el encargado principal de construir cada ventana solicitada por un cliente, para realizar esta operación, el servidor necesita conocer el display que solicita la petición.</td></tr>
<tr><td colspan="2">Prototipo de interfaz:</td></tr>
</table>

Historia de Usuario U-AMVEX-02.

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número:U-AMVEX-02</td><td colspan="2">Nombre Historia de Usuario: Obtener datos de ventanas.</td></tr>
<tr><td colspan="3">Modificación de Historia de Usuario Número: Ninguna.</td></tr>
<tr><td colspan="2">Usuario: Rodolfo González Pelegrino.</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Alta
(Alta / Medio / Bajo)</td><td>Puntos Estimados: 3</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Medio
(Alta / Medio / Bajo)</td><td>Puntos Reales: 3</td></tr>
<tr><td colspan="3">Descripción: Se solicita al servidor del sistema X Window los datos de las ventanas que este construye a petición de un cliente (aplicación).</td></tr>
<tr><td colspan="3">Observaciones: Los datos a tener en cuenta están relacionados con la ventana contenedora del foco. La ventana puede contener varias hijas que también serán capturadas, y todas están relacionadas con el proceso principal que es el nombre de la aplicación, el cual también es incluido.</td></tr>
<tr><td colspan="3">Prototipo de interfaz: (Ver Anexo #5)</td></tr>
</table>

Historia de Usuario U-AMVEX-03.

Historia de Usuario	
Número:U-AMVEX-03	**Nombre Historia de Usuario:** Detectar aplicación no usada.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rodolfo González Pelegrino.	**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta / Medio / Bajo)	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio (Alta / Medio / Bajo)	**Puntos Reales:** 1
Descripción: Determina si la ventana enfocada está siendo usada o si el usuario ha abierto la misma y no la está utilizando.	
Observaciones:	
Prototipo de interfaz:	

Historia de Usuario U-AMVEX-04.

Historia de Usuario	
Número:U-AMVEX-04	**Nombre Historia de Usuario:** Generar estructura de datos representada en XML.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rodolfo González Pelegrino.	**Iteración Asignada:** 3
Prioridad en Negocio: Alta (Alta / Medio / Bajo)	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio (Alta / Medio / Bajo)	**Puntos Reales:** 2
Descripción: Implementar el componente generador de documentos xml.	
Observaciones: El nombre del documento lleva como formato: usuario-año-mes-día-minutos-segundos, para evitar en el mismo día existan documentos con nombres iguales.	
Prototipo de interfaz: (Ver Anexo #6)	

Historia de Usuario U-AMVEX-05.

<table>
<tr><th colspan="3" align="right">Historia de Usuario</th></tr>
<tr><td>Número: U-AMVEX-05</td><td colspan="2">Nombre Historia de Usuario: Enviar mensajes entre procesos.</td></tr>
<tr><td colspan="3">Modificación de Historia de Usuario Número: Ninguna.</td></tr>
<tr><td colspan="2">Usuario: Rodolfo González Pelegrino.</td><td>Iteración Asignada: 3</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Alta
(Alta / Medio / Bajo)</td><td>Puntos Estimados: 4</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Alta
(Alta / Medio / Bajo)</td><td>Puntos Reales: 4</td></tr>
<tr><td colspan="3">Descripción: Se integran los componentes de monitoreo y el que genera el documento XML. Luego el primer componente le envía la información recopilada al segundo.</td></tr>
<tr><td colspan="3">Observaciones: Los procesos se encuentran en el mismo ordenador.</td></tr>
<tr><td colspan="3">Prototipo de interfaz:</td></tr>
</table>

Historia de Usuario U-AMVEX-06.

Historia de Usuario	
Número: U-AMVEX-06	**Nombre Historia de Usuario:** Enviar documento XML.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rodolfo González Pelegrino.	**Iteración Asignada:** 3
Prioridad en Negocio: Alta (Alta / Medio / Bajo)	**Puntos Estimados:** 2
Riesgo en Desarrollo: Alta (Alta / Medio / Bajo)	**Puntos Reales:** 2
Descripción: Implementar el componente que envía el fichero XML hasta la computadora donde está el servidor.	
Observaciones: Se utiliza el protocolo TCP/IP.	
Prototipo de interfaz:	

Historia de Usuario U-AMVEX-07.

Historia de Usuario	
Número: U-AMVEX-07	**Nombre Historia de Usuario:** Recibir xml en el servidor.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rodolfo González Pelegrino.	**Iteración Asignada:** 3
Prioridad en Negocio: Alta (Alta / Medio / Bajo)	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alta (Alta / Medio / Bajo)	**Puntos Reales:** 1
Descripción: Implementar el programa que recibe el fichero XML de la computadora cliente.	
Observaciones: Se utiliza el protocolo TCP/IP.	
Prototipo de interfaz:	

Historia de Usuario U-AMVEX-08.

Historia de Usuario	
Número: U-AMVEX-08	**Nombre Historia de Usuario:** Implementar script de inicio.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rodolfo González Pelegrino.	**Iteración Asignada:** 4
Prioridad en Negocio: Media (Alta / Medio / Bajo)	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio (Alta / Medio / Bajo)	**Puntos Reales:** 1,5
Descripción: Se programa el script que ejecuta el sistema de monitoreo, este se encarga además de asegurar que el programa esté en ejecución hasta que el usuario abandone la sesión.	
Observaciones: Se utiliza Bash como lenguaje de programación.	
Prototipo de interfaz:	

Historia de Usuario U-AMVEX-09.

Historia de Usuario	
Número: U-AMVEX-09	**Nombre Historia de Usuario:** Notificar tareas productivas del DotProject.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rodolfo González Pelegrino.	**Iteración Asignada:** 4
Prioridad en Negocio: Alta (Alta / Medio / Bajo)	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio (Alta / Medio / Bajo)	**Puntos Reales:** 2,5
Descripción: Se programa el notificador de tareas productivas de la herramienta de gestión de proyecto DotProject. Para esto se accede a la base de datos del DotProject, se obtienen las tareas de los usuarios y se les muestra.	
Observaciones:	
Prototipo de interfaz:	

3.5. Plan de versiones

La plantilla de Plan de versiones, es un documento que se genera. En este el cliente define el valor que posee el negocio según las características deseadas. Las historias de mayor riesgo y mayor prioridad se incluyen en las primeras iteraciones. El lanzamiento y las iteraciones tienen fechas fijas para su consecución, así como un alcance variable. Esta plantilla proporciona ventajas, tales como:[18]

- Define cuales son las historias de usuario más significativas, y las ubican en las iteraciones según esta prioridad.
- Divide el proceso de desarrollo de software en iteraciones, planificando el trabajo a realizar en

cada una de ellas.

Plan de releases

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 1	Se establece conexión con el servidor X Window System y se controla eventos del servidor X.	U-AMVEX-01 U-AMVEX-02	15/10/08-30/11/08
Iteración 1	Se implementa el componente generador de documentos XML y se corrigen errores en la captura de datos monitoreados.	U-AMVEX-03 U-AMVEX-04	01/12/08-15/01/09
Iteración 3	Se integra el componente de monitoreo al generador de documentos XML y se implementa el programa cliente que envía el fichero al servidor.	U-AMVEX-05 U-AMVEX-06 U-AMVEX-07	16/01/09-03/03/09
Iteración 4	Se implementan los script de inicio del sistema de monitoreo y el notificador de eventos de la herramienta de gestión de proyecto DotProject, además de integrar la aplicación al	U-AMVEX-08 U-AMVEX-09	04/03/09-20/04/09

	módulo GENEST.		

3.1. Tareas de Ingeniería

La metodología SXP durante la fase de desarrollo propone la realización de las tareas de ingeniería, las cuales están asociadas a las historias de usuarios. Esta plantilla organizan el trabajo en el desarrollo del producto, permite conocer datos como: el programador asignado a cada tarea y el tiempo que demora en realizarla.

Tareas de Ingeniería para la historia de usuario U-AMVEX-01.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-01.
Nombre Tarea: Estudio de xlib.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 3
Fecha Inicio: 15/10/08	**Fecha Fin:** 06/11/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se estudia la librería encargada de establecer comunicación con el servidor X.	
Observación: Es la librería de más bajo nivel.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-01.

Nombre Tarea: Obtener display de conexión.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 30/10/08	**Fecha Fin:** 06/11/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se obtiene el display a partir de: nombre del host en el que reside el servidor, el número del display y el número de la pantalla.	
Observación: Realizar la operación sobre un entorno gráfico.	

Tareas de Ingeniería para la historia de usuario U-AMVEX-02.

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-02.
Nombre Tarea: Máscaras de eventos.	
Tipo de Tarea : Estudio y Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 07/11/08	**Fecha Fin:** 14/11/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Estudio del conjunto de máscaras de eventos para determinar las máscaras que se van a emplear, así como los eventos de cada una de estas.	
Observación: La relación entre máscara de evento y tipo de evento no es siempre de uno a uno, puede ser de uno a muchos y de muchos a uno.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-02.
Nombre Tarea: Modo de espera.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 15/11/08	**Fecha Fin:** 22/11/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se determina el modo de bloquea de ejecución del código hasta que aparezca el evento relacionado con la máscara específica.	
Observación:	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** U-AMVEX-02
Nombre Tarea: Calcular tiempo.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 23/11/08	**Fecha Fin:** 30/11/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se calcula el tiempo que utiliza el usuario sobre la ventana activa.	
Observación: Si en la ventana del foco no se realiza ninguna acción durante un período específico de tiempo el valor del mismo se iguala a cero.	

Tareas de Ingeniería para la historia de usuario U-AMVEX-03.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-03.
Nombre Tarea: Manipular propiedades y eventos con el mouse y el teclado.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 01/12/08	**Fecha Fin:** 08/12/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se manipula cambio en las propiedades de una ventana, pulsación de teclas y redibujo sobre la pantalla.	
Observación: Si no existe cambio de propiedades en las ventanas del entorno de escritorio, existe alta probabilidad de que no se esté trabajando en el mismo.	

Tareas de Ingeniería para la historia de usuario U-AMVEX-04.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-04.
Nombre Tarea: Estudio de analizador sintáctico .	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 09/12/08	**Fecha Fin:** 17/12/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se estudia el analizador sintáctico libxml, para generar estructura de datos representado en XML.	

Observación: Se utiliza la versión 2.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-04.
Nombre Tarea: Diseñar la estructura del XML.	
Tipo de Tarea : Diseño	**Puntos Estimados:** 1
Fecha Inicio: 15/12/08	**Fecha Fin:** 22/12/08
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se diseña la estructura en que se genera el documento XML.	
Observación:	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** U-AMVEX-04.
Nombre Tarea: Implementar la estructura del XML.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 07/01/09	**Fecha Fin:** 14/01/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se implementa la estructura en que se genera el documento XML.	

Observación:

Tareas de Ingeniería para la historia de usuario U-AMVEX-05.

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-05.
Nombre Tarea: Estudio sobre recursos compartidos.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 16/01/09	**Fecha Fin:** 23/01/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se estudia la manera en que dos programas en ejecución pueden compartir información por medio de recursos compartidos.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-05.
Nombre Tarea: Implementación de colas de mensajes en el componente de monitoreo.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 24/01/09	**Fecha Fin:** 31/01/09
Programador Responsable: Rodolfo González Pelegrino.	

Descripción: Se implementa funcionalidades de colas de mensajes para establecer la comunicación entre el proceso de monitoreo y el componente de generador de XML.
Observación:

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** U-AMVEX-05.
Nombre Tarea: Implementación de colas de mensajes en el componente generador del XML.	
Tipo de Tarea : Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 01/02/09	**Fecha Fin:** 08/02/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se implementa funcionalidades de colas de mensajes para establecer la comunicación entre el proceso de monitoreo y el componente de generación de XML.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 4	**Número Historia de Usuario:** U-AMVEX-05.
Nombre Tarea: Recibir datos monitoreados.	
Tipo de Tarea : Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 10/02/09	**Fecha Fin:** 17/02/09

Programador Responsable: Rodolfo González Pelegrino
Descripción: El generador del XML recibe los datos que han sido monitoreados y los inserta en el documento XML según el nombre de la aplicación.
Observación: Para recibir los datos, la cola de mensajes debe tener la misma estructura de información que se envía. Luego la cola debe ser vaciada para evitar que esta se llene y los datos se pierdan.

Tareas de Ingeniería para la historia de usuario U-AMVEX-06.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-06.
Nombre Tarea: Estudio de programación con socket.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 18/02/09	**Fecha Fin:** 25/02/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se estudia la manera en que dos programas que estén en distintos ordenadores puedan establecer comunicación.	
Observación: Utilizar protocolo TCP/IP.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-06.
Nombre Tarea: Determinar la longitud del fichero.	
Tipo de Tarea : Desarrollo.	**Puntos Estimados:** 1

Fecha Inicio: 26/02/09	**Fecha Fin:** 02/03/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se calcula el tamaño del fichero que será enviado por la red.	
Observación: El fichero es un documento XML.	

		Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** U-AMVEX-06	
Nombre Tarea: Cargar y enviar fichero.		
Tipo de Tarea : Desarrollo		**Puntos Estimados:** 1
Fecha Inicio: 26/02/09		**Fecha Fin:** 02/03/09
Programador Responsable: Rodolfo González Pelegrino.		
Descripción: Se lee el fichero, se carga en un buffer y se implementa las funciones de envío.		
Observación: El fichero es un documento XML.		

		Tarea de Ingeniería
Número Tarea: 4	**Número Historia de Usuario:** U-AMVEX-06	
Nombre Tarea: Configurar archivos del sistema.		
Tipo de Tarea : Configuración		**Puntos Estimados:** 1
Fecha Inicio: 26/02/09		**Fecha Fin:** 02/03/09

Programador Responsable: Rodolfo González Pelegrino.
Descripción: Se configura los archivos del sistema operativo para que el cliente conozca el puerto por el que se comunica y localice al servidor.
Observación:

Tareas de Ingeniería para la historia de usuario U-AMVEX-07.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-07.
Nombre Tarea: Configurar archivos del sistema operativo.	
Tipo de Tarea : Configuración	**Puntos Estimados:** 1
Fecha Inicio: 26/02/09	**Fecha Fin:** 02/03/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se configura los archivos para determinar el puerto por el que atenderá el servidor peticiones de los clientes.	
Observación:	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-07.
Nombre Tarea: Recepción de datos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 26/02/09	**Fecha Fin:** 02/03/09

Programador Responsable: Rodolfo González Pelegrino.
Descripción: Se reciben los datos enviados por el cliente.
Observación: La estructura que se espera debe ser la que se envía.

Tareas de Ingeniería para la historia de usuario U-AMVEX-08.

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-08.
Nombre Tarea: Estudio de Bash.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 04/03/09	**Fecha Fin:** 11/03/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se estudia el intérprete de comando Bash para realizar el script de inicio.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U-K532-08.
Nombre Tarea: Configurar archivos del sistema operativo.	
Tipo de Tarea : Configuración	**Puntos Estimados:** 1
Fecha Inicio: 11/03/09	**Fecha Fin:** 18/03/09

Programador Responsable: Rodolfo González Pelegrino.
Descripción: Se configura el archivo que ejecuta el script al iniciar una sesión.
Observación:

Tareas de Ingeniería para la historia de usuario U-AMVEX-09.

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** U-AMVEX-09.
Nombre Tarea: Distribución de tablas de la base de datos del DotProject.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 19/03/09	**Fecha Fin:** 26/03/09
Programador Responsable: Rodolfo González Pelegrino.	
Descripción: Se estudia la distribución estructura de la base de datos del DotProject.	
Observación:	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** U-AMVEX-09.
Nombre Tarea: Gestión de tareas de producción.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 27/03/09	**Fecha Fin:** 8/04/09

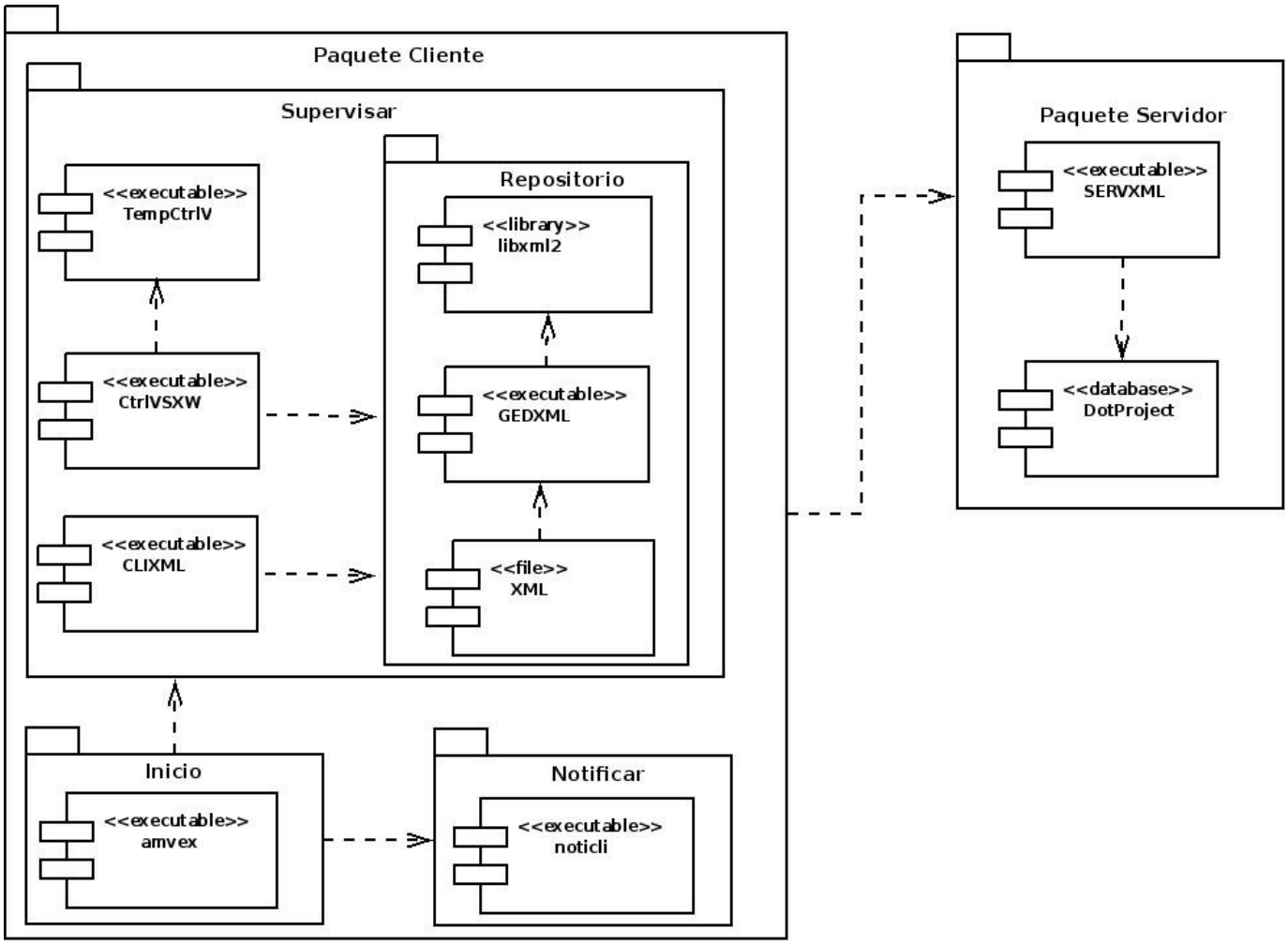
Programador Responsable: Rodolfo González Pelegrino.
Descripción: Se obtiene las tareas productivas de la base de datos del DotProject y se envía al usuario autenticado en el sistema operativo.
Observación:

3.2. Diseño con metáforas

La plantilla del Modelo de diseño, es el documento que se genera del diseño con las metáforas, donde se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume de forma evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto, se solventan con la existencia de una metáfora. [19]

Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Martin Fowler explica 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. [20]

El diseño con metáforas genera el modelo de diseño, que está compuesto por el diagrama de paquetes. Estos son útiles para agrupar elementos en estructuras lógicas y reflejan la organización de los paquetes. A continuación se muestra el diagrama del sistema que se propone.



En el cliente encontramos diferentes componentes, el primero en ejecutarse al iniciar el nivel de ejecución en el que trabaja el usuario es amvex (Ver Anexo #7), este es un ejecutable que detecta cuando el usuario entra a una sesión gráfica y lanza el paquete Notificar y Supervisar.

El paquete Notificar es el responsable de hacer una solicitud al servidor para obtener las tareas

productivas que tiene el usuario en la herramientas de gestión de proyecto DotProject, esta acción se produce cuando el usuario comienza a trabajar sobre un entorno gráfico.

El paquete Supervisar cuenta con varios componentes, CtrlVSXW es el encargado de gestionar la información sobre el uso de las ventanas y el tiempo que utiliza el usuario en cada una de estas. Para su funcionamiento usa la librería Xlib, la cual le proporciona la conexión con el servidor del sistema X Window (Ver Anexo #8). El paquete Repositorio se encarga de almacenar los datos supervisada utilizando la tecnología XML, esta información adopta una posición pasiva ya que solo se realizan acciones de lectura y escritura (Ver Anexo #9). CLIXML tiene como tarea enviar el reporte generado en el documento XML, se ejecuta una vez que el usuario termine una sesión de un entorno gráfico, y envía los datos al componente servidor ubicado en el Paquete Servidor (Ver Anexo #10). Para la implementación se utilizó un estándar de código que facilita la compresión del mismo (Ver Anexo #11).

Con esta estructura se describe una arquitectura centrada en datos, ya que al paquete Repositorio es el centro de acceso de diversos componentes y se le pueden añadir otros sin afectar la funcionalidad de los demás. Esta es una ventaja que proporciona el desarrollo de la aplicación sin necesidad de conocer la manera en que están implementados los componentes existentes, solo es imprescindible conocer la estructura de la información que requiere el Repositorio.

Para no violar la arquitectura a la hora de agregar nuevas funcionalidades a la aplicación, se debe evitar incluir en el código llamadas a otros componentes, así como el traslado de información entre estos, los distintos componentes no necesitan saber qué datos producen o consume otros. El control al Repositorio debe ser manejado por quienes lo acceden y el almacén no puede invocar a la ejecución de componentes.

También está presente el modelo cliente/servidor, a través del cual la información es depositada en el ordenador donde lo necesita el módulo GENEST.

4. Capítulo 4. Validación de la solución propuesta

En este capítulo se describen los casos de pruebas realizados a la aplicación en cada una de las iteraciones, se muestran los resultados obtenidos, se analizan y se presentan las funcionalidades alcanzadas hasta el período de desarrollo.

La plantilla de Plan de pruebas, es lo primero que se realiza cuando el producto va a ser probado. Este plantilla recoge los pasos y datos de cada uno de los involucrados en el proceso. Además de tener actualizadas las fechas en las que se realiza cada una de las funcionalidades a probar. Con esta plantilla se logra un plan de organización para la realización de las futuras pruebas a realizar, se recoge cada una de las funcionalidades a probar, y se realiza una evaluación de las pruebas realizadas.[21]

4.1. Casos de prueba de aceptación

Uno de los pilares de la metodología SXP es el uso de test para comprobar el funcionamiento de los códigos que se implementan. Estas pruebas se realizan entre iteraciones y son las que dan paso a la próxima iteración. Durante el desarrollo de la aplicación propuesta se realizó un conjunto de pruebas para verificar su correcto funcionamiento. Luego de elaborar el plan de pruebas, para organizar el desarrollo de las mismas, se define casos de prueba para cada historia de usuario, a continuación se hace una descripción de cada prueba correspondiente a la historia de usuario.

4.1.1. Caso de prueba para la Historia de usuario: U-AMVEX-01

En esta sección se realiza la prueba a la historia de usuario: Establecer conexión con el servidor X. Su objetivo principal es verificar que la conexión con el servidor X se realice con éxito.

	Caso de Prueba de Aceptación
Código Caso de Prueba: U-AMVEX-01-01	**Nombre Historia de Usuario:** Establecer conexión con el servidor X.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	

Descripción de la Prueba: Para establecer conexión con el servidor se obtiene el display de conexión y luego se comprueba solicitándole un servicio al servidor.
Condiciones de Ejecución: Se necesita estar trabajando sobre un entorno gráfico.
Entrada / Pasos de ejecución: Se tiene que conocer: el nombre del host en el que reside el servidor, el número del display y el número de la pantalla. En esta prueba se define estos valores con NULL, el cual se reemplaza por la cadena almacenada en la variable de entorno DISPLAY.
Resultado Esperado: Se realice la conexión con el servidor.
Evaluación de la Prueba: Satisfactoria

4.1.2. Casos de pruebas para la Historia de usuario: U-AMVEX-02

En esta sección se realizan un conjunto de pruebas a la historia de usuario: Obtener datos de ventanas. Se verifica la obtención de los datos de las ventanas que contienen el foco y se calcula el tiempo que utiliza el usuario en cada ventana.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-02-01	**Nombre Historia de Usuario:** Obtener datos de ventanas.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se comprueba que las máscaras de eventos determinan la ventana que el usuario está utilizando.	
Condiciones de Ejecución: El usuario tiene que abrir varias ventanas y cambiarlas de posición, puede cerrarla, minimizarla y abrir nuevas aplicaciones.	
Entrada / Pasos de ejecución: El usuario produce un evento correspondiente a la máscara.	
Resultado Esperado: Se obtiene el nombre de la ventana y la aplicación a la que corresponde.	

Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-02-02	**Nombre Historia de Usuario:** Obtener datos de ventanas.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se comprueba que las máscaras de eventos determinan la ventana que el usuario está utilizando.	
Condiciones de Ejecución: El usuario tiene que abrir varias ventanas y cambiarlas de posición, puede cerrarla, minimizarla y abrir nuevas aplicaciones.	
Entrada / Pasos de ejecución: El usuario produce un evento correspondiente a la máscara.	
Resultado Esperado: Se obtiene el nombre de la ventana y la aplicación a la que corresponde.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-02-03	**Nombre Historia de Usuario:** Obtener datos de ventanas.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se bloquea el modo de espera de un evento para cualquier ventana.	
Condiciones de Ejecución: El usuario tiene que abrir varias ventanas y cambiarlas de posición, puede cerrarla, minimizarla y abrir nuevas aplicaciones.	
Entrada / Pasos de ejecución: El usuario produce un evento sobre cualquier ventana.	

Resultado Esperado: Se obtiene el nombre de la ventana y la aplicación a la que corresponde.
Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-02-04	**Nombre Historia de Usuario:** Obtener datos de ventanas.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se calcula el tiempo que utiliza el usuario en cada ventana.	
Condiciones de Ejecución: El usuario abre una ventana, se demora el tiempo que necesita y luego enfoca otra ventana.	
Entrada / Pasos de ejecución: Se toma el tiempo cuando la ventana tiene el foco, luego se realiza la misma operación cuando la ventana pierde el foco, y se calcula la diferencia.	
Resultado Esperado: Se obtiene el tiempo que utiliza el usuario en cada ventana.	
Evaluación de la Prueba: Satisfactoria	

4.1.3. Casos de pruebas para la Historia de Usuario: U-AMVEX-03

En esta sección se realiza la prueba a la historia de usuario: Detectar aplicación no usada. El objetivo es verificar el correcto funcionamiento de detección de aplicaciones en foco sin ser utilizadas, para esto se mide la interacción del usuario con el ordenador a través de eventos de teclado y ratón para el dibujo de ventanas.

Caso de Prueba de Aceptación	
Código Caso de Prueba:	**Nombre Historia de Usuario:** Detectar aplicación no usada

U-AMVEX-03-01	
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se controla la utilización del teclado, el mouse y el dibujo en pantalla.	
Condiciones de Ejecución: El usuario abre una aplicación y utiliza el teclado, mouse y abre y cierra ventanas.	
Entrada / Pasos de ejecución: Se toma la diferencia del tiempo entre lo que sucede en cada operación, si el tiempo es mayor de 3 minutos se el valor del tiempo vuelve a tomar 0.	
Resultado Esperado: Si el usuario no ha causado ningún evento de los mencionados se muestra un mensaje para notificar su ausencia en el ordenador.	
Evaluación de la Prueba: Satisfactoria.	

4.1.4. Casos de pruebas para la Historia de Usuario: U-AMVEX-04

En esta sección se realizan un conjunto de pruebas a la historia de usuario: Generar estructura de datos representada en XML.. El propósito es lograr una estructura adecuada del documento xml, para esto es necesario definir las etiquetas que recogen la información y realizar una análisis sintáctico.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-04-01	**Nombre Historia de Usuario:** Generar estructura de datos representada en XML.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se comprueba la existencia de un documento xml.	
Condiciones de Ejecución: Existencia de un documento xml.	
Entrada / Pasos de ejecución: Se entra el nombre del xml, el texto que se va añadir y la etiqueta que recoge el contenido.	

Resultado Esperado: No se crea un nuevo documento xml.
Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-04-02	**Nombre Historia de Usuario:** Generar estructura de datos representada en XML.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se genera un documento xml.	
Condiciones de Ejecución: No existe un documento xml.	
Entrada / Pasos de ejecución: Se entra el nombre del xml, el texto que se va añadir y la etiqueta que recoge el contenido.	
Resultado Esperado: Se crea un nuevo documento xml.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-04-03	**Nombre Historia de Usuario:** Generar estructura de datos representada en XML..
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino.	
Descripción de la Prueba: Se agregan nuevos atributos a las etiquetas y nodos hijos.	
Condiciones de Ejecución: Puede existir o no un documento xml, en caso de no existir se crea.	
Entrada / Pasos de ejecución: Se entra el nombre del xml, el texto que se va a añadir, la etiqueta que	

recoge el contenido y un valor entero que representa el tiempo de uso de una ventana.
Resultado Esperado: El documento xml tendrá una etiqueta que recoge la cantidad de hijos y el tiempo total, y cada hijo recoge el tiempo y el nombre de la ventana.
Evaluación de la Prueba: Satisfactoria

4.1.5. Caso de prueba para la Historia de Usuario: U-AMVEX-05

En esta sección se realiza la prueba a la historia de usuario: Enviar mensaje entre procesos. Tiene como finalidad establecer comunicación entre procesos a través de colas de mensajes y enviar los datos que recopila el componente que monitorea las ventanas, al que genera el documento xml.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-05-01	**Nombre Historia de Usuario:** Enviar mensaje entre procesos.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se establece comunicación entre el proceso que monitorea y el que genera documentos xml, luego el primero le envía mensajes al segundo.	
Condiciones de Ejecución: El proceso que recibe los datos debe esperar la misma estructura de información que el proceso que envía.	
Entrada / Pasos de ejecución: Se ejecuta el proceso que genera documentos xml y este se queda esperando hasta que se ejecute el que envía los datos, luego comienza la transacción de información.	
Resultado Esperado: El componente generador de xml recibe los datos del proceso de monitoreo y se actualiza el documento xml.	
Evaluación de la Prueba: Satisfactoria	

4.1.6. Casos de pruebas para la Historia de Usuario: U-AMVEX-06

En esta sección se realiza la prueba a la historia de usuario: Enviar documento XML. Este caso de prueba

de aceptación consiste en verificar que el documento XML generado, es enviado al ordenador donde se encuentra el servidor y es colocado en su carpeta correspondiente.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-06-01	**Nombre Historia de Usuario:** Enviar documento XML.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino	
Descripción de la Prueba: Se envía el documento xml generado.	
Condiciones de Ejecución: Debe existir al menos un documento xml, así como un servidor esperando, además tanto el cliente como el servidor deben escuchar por el mismo puerto.	
Entrada / Pasos de ejecución: Se carga el contenido del documento, se calcula el tamaño del mismo y se envía al servidor.	
Resultado Esperado: El documento xml es transportado al ordenador donde se encuentra el servidor.	
Evaluación de la Prueba: Satisfactoria	

4.1.7. Casos de pruebas para la Historia de Usuario: U-AMVEX-07

En esta sección se realiza la prueba a la historia de usuario: Recibir xml en el servidor. En esta prueba se pretende confirmar que los reportes de las máquinas clientes llegan al servidor donde se encuentra el módulo de gestión de proyecto DotProject y cada uno de documentos xml es depositado en la carpeta del usuario correspondiente.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-07-01	**Nombre Historia de Usuario:** Recibir xml en el servidor.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino.	

Descripción de la Prueba: Se recibe el documento xml generado y se guarda en la carpeta del usuario al que pertenece.
Condiciones de Ejecución: Debe existir al menos un cliente que envíe al servidor y la carpeta de cada usuario debe estar creada.
Entrada / Pasos de ejecución: Se lanza el servidor y se envía un xml desde una máquina cliente.
Resultado Esperado: El documento xml enviado por el cliente es recibido.
Evaluación de la Prueba: Satisfactoria

4.1.8. Casos de pruebas para la Historia de Usuario: U-AMVEX-08

En esta sección se realiza la prueba a la historia de usuario: Implementar script de inicio. Este script es ejecutado cuando el usuario entra en el nivel de ejecución en el trabaja su entorno gráfico. Al usuario salir de la sesión este se encarga de ejecutar el componente que envía el documento xml hacia el servidor.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-08-01	**Nombre Historia de Usuario:** Implementar script de inicio.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino.	
Descripción de la Prueba: El script detecta cuando el usuario termina su sesión y ejecuta el componente que se encarga de enviar el xml generado.	
Condiciones de Ejecución: El usuario debe entrar a una sesión, realizar cualquier trabajo y salir.	
Entrada / Pasos de ejecución: El usuario entra en la sesión realiza cualquier trabajo y cierra la sesión, al ocurrir esto el script se encarga de ejecutar el componente que envía el xml hasta el servidor.	
Resultado Esperado: El documento xml generado por el usuario, es enviado al terminar la sesión de trabajo.	

Evaluación de la Prueba: Satisfactoria

4.1.9. Casos de pruebas para la Historia de Usuario: U-AMVEX-09

En esta sección se realiza la prueba a la historia de usuario: Notificar tareas productivas del DotProject. Esta prueba tiene como fin demostrar el correcto funcionamiento de la notificación de tareas del DotProject.

Caso de Prueba de Aceptación	
Código Caso de Prueba: U-AMVEX-09-01	**Nombre Historia de Usuario:** Notificar tareas productivas del DotProject.
Nombre de la persona que realiza la prueba: Rodolfo González Pelegrino.	
Descripción de la Prueba: El usuario entra en una sesión con entorno gráfico y recibe las tareas que tiene en el DotProject.	
Condiciones de Ejecución: El usuario debe entrar a una sesión con entorno gráfico.	
Entrada / Pasos de ejecución: El usuario al entrar en la sesión, se envía un mensaje al servidor para que este busque las tareas de ese usuario y las traslade hasta el cliente.	
Resultado Esperado: Al entrar en una sesión gráfica se visualiza las tareas que tiene en el DotProject.	
Evaluación de la Prueba: Satisfactoria	

4.2. Resultados obtenidos.

Con el desarrollo de esta herramienta de monitoreo y notificación de tareas productivas del sistema de gestión de proyecto DotProject, los estudiantes y profesores de Unicornios tienen un entorno orientado a mejorar la producción. Estos usuarios no necesitan perder tiempo para conocer sus tareas y pueden conocer en qué aplicaciones usan más tiempo. El jefe de proyecto tiene en sus manos la información

exacta de las actividades que se desarrolla en su equipo (Ver Anexo #12), lo que facilita conocer en qué tareas cada integrante es más eficaz, esto ayuda en gran medida realizar planificaciones con buenos resultados.

La aplicación lleva por nombre AMVEX (Aplicación para el Monitoreo de Ventanas X), y está integrada al módulo GENEST reportando documentos XML. A pesar de ser una herramienta desarrollada para GENEST, fue construida bajo una arquitectura que posibilita que se integre a cualquier herramienta de gestión de proyecto y que se le añadan nuevos componentes sin necesidad de conocer la lógica de los que ya están implementados.

Entre las principales funcionalidades que presenta el producto software en su primera versión se encuentran:

- Obtener información sobre el nombre de la ventana que está utilizando el foco.
- Obtener la aplicación correspondiente a cada ventana que tiene el foco.
- Calcular el tiempo de uso de cada ventana y aplicación.
- Detectar aplicaciones abiertas sin ser utilizadas.
- Detectar el uso del teclado, el dibujo de ventanas y el cambio de propiedades.
- Detectar la entrada y salida de un usuario en sesión.
- Generar documentos XML.
- Comunicar procesos en el mimo ordenador.
- Enviar ficheros desde la máquina cliente hacia el servidor.
- Notificar tareas productivas de la herramienta de gestión de proyecto DotProject a los usuarios.

Conclusiones

Para la realización del presente trabajo se plantearon un conjunto de tareas que posibilitaron el desarrollo y cumplimiento de los objetivos propuestos, con esta investigación se arribó a los siguientes resultados:

- Se analizó el funcionamiento de diferentes sistemas de monitoreo de procesos y sus características.
- Se implementó una aplicación cliente/servidor para la supervisión de procesos y notificación de tareas productivas de la herramienta de gestión de proyecto DotProject.
- Se obtuvo una herramienta que facilita integrarse al módulo GENEST.
- Se validó el software desarrollado, obteniendo resultados positivos.

Teniendo en cuenta todo lo expuesto en este trabajo, se concluye que los objetivos trazados se cumplieron de manera satisfactoria, con la documentación necesaria y la certeza de que el mismo puede utilizarse para futuras investigaciones.

Recomendaciones

- Se recomienda que sean agregadas nuevas funcionalidades al trabajo realizado, como manipular todas las ventanas visibles (Ver Anexo #11).
- Que sea utilizado en los laboratorios productivos con fines de mejorar la gestión de proyecto en las máquinas destinadas para la producción.
- Que se desarrolle una versión utilizando el protocolo XCB, debido a que este tiene como propósito sustituir a Xlib.
- Que se integre a otras herramientas de gestión de proyecto, aprovechando que su arquitectura lo permite.

Referencias Bibliográficas

[1]González Durán, Sergio. MANUAL BÁSICO DE ADMINISTRACIÓN DE PROCESOS [cited 4 April 2009]. Available from world wide web: <http://www.linuxtotal.com.mx/index.php?cont=info_admon_012>

[2]Pérez Costoya, Fernando. Curso del sistema de ventanas X [cited 10 November 2008]. Available from world wide web: <http://laurel.datsi.fi.upm.es/~fperez/cursoX/indice.html>

[3]Edwards, Grant. An Introduction to X11 User Interfaces [cited 20 November 2008]. Available from world wide web: <http://www.visi.com/~grante/Xtut/>

[4]Anjuta Team. Anjuta. [cited 4 December 2008] Available from world wide web: <http://anjuta.org/>

[5]Code::Blocks Team. Code::Blocks [cited 6 December 2008] Available from world wide web: <http://www.codeblocks.org>

[6]Code::Blocks Team. Code::Blocks [cited 6 December 2008] Available from world wide web: <http://www.codeblocks.org>

[7]Code::Blocks Team. Code::Blocks [cited 6 December 2008] Available from world wide web: <http://www.codeblocks.org>

[8]Meyer, Eric. Lenguaje C. [cited 20 October 2008]. Available from world wide web: http://www.ib.cnea.gov.ar/~icom/CursoC/historia.shtml

[9]chuidiang. Programación de sockets en C de Unix/Linux. [cited 10 February 2009]. Available from world wide web: http://www.chuidiang.com/clinux/sockets/sockets_simp.php

[10]chuidiang. Programación de sockets en C de Unix/Linux. [cited 10 February 2009]. Available from world wide web: http://www.chuidiang.com/clinux/sockets/sockets_simp.php

[11]Melian Montalvo, Marlene. XML el nuevo lenguaje universal. [cited 5 March 2009]. Available from world wide:
http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH0104/f016d031.dir/

[12]Gladys Marsi, P. R., 2008, *MA-GMPR-UR2 Metodología ágil para proyectos de software libre,* Universidad de las Ciencias Informáticas, Facultad 10 [cited 2 February 2009] Available from world wide: http://bibliodoc.uci.cu/TD/TD_0693_07.pdf

[13]Pressman, Roger. *Ingeniería del Software. Un enfoque práctico.* Quinta Edición MacGraw-Hill [cited 10 March 2009]. Available from world wide web: <http://bibliodoc.uci.cu/pdf/reg02689.pdf>

[14]Idem[13]

[15]Idem[13]

[16]Gladys Marsi, P. R., 2008, *MA-GMPR-UR2 Metodología ágil para proyectos de software libre,* Universidad de las Ciencias Informáticas, Facultad 10 [cited 2 February 2009] Available from world wide: http://bibliodoc.uci.cu/TD/TD_0693_07.pdf

[17]Idem[16]

[18]Idem[16]

[19]Idem[16]

[20]Fowler, Martin. "Is Design Dead?".Fowler, M. "Is Design Dead?". [cited 2 December 2008] Available from world wide: www.martinfowler.com/articles/designDead.html

[21]Gladys Marsi, P. R., 2008, MA-GMPR-UR2 Metodología ágil para proyectos de software libre, Universidad de las Ciencias Informáticas, Facultad 10 [cited 2 February 2009] Available from world wide: http://bibliodoc.uci.cu/TD/TD_0693_07.pdf

Bibliografía

Eclipse. Fundación Eclipse. [cited 2 March 2009]. Available from world wide web: <http://www.eclipse.org/>

Free Software Foundation, Inc. *El sistema Operativo GNU* [cited 1 October 2008]. Available from world wide web: http://www.gnu.org/philosophy/philosophy.es.html#LicensingFreeSoftware

"El sistema X Window". [cited 3 November 2008]. Available from world wide web: http://ditec.um.es/laso/docs/tut-tcpip/3376c49.html

Gettys,James; Packard ,Keith."*The(Re)Architecture of the X Window System"*.2004 [cited 5 November 2008]. Available from world wide web: http://keithp.com/~keithp/talks/xarch_ols2004/]

Durán Toro, Amador. "*Programación en X Window*". [cited 20 November 2008]. Available from world wide web:http://www.lsi.us.es/cursos/xlib.html

Pérez Costoya, Fernando. "*Curso del sistema de ventanas X*". [cited 3 December 2008]. Available from world wide web: http://laurel.datsi.fi.upm.es/~fperez/cursoX/indice.html

Reynoso, Carlos ; Kiccillof, Nicolás . "Estilos y Patrones en la Estrategia de Arquitectura de Microsoft". [cited 10 December 2008].Available from world wide web: http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/estiloypatron.pdf

Marrique, Daniel. "*X Window System Architecture Overview HOWTO"*.2001. [cited 15 December 2008]. Available from world wide web: http://www.faqs.org/docs/Linux-HOWTO/XWindow-Overview-HOWTO.html

The X.Org Foundation. "*Documentation for the X Window System Version 11 Release 6.8.1 (X11R6.8.1)*". [cited 20 December 2008]. Available from world wide web: http://www.x.org/archive/X11R6.8.1/doc/

Freedesktop. "*XCB*". [cited 10 October 2008]. Available from world wide web: http://xcb.freedesktop.org/

The X.Org Foundation. "*Documentation for the X Window System Version 11 Release 6.9 and 7.0 (X11R6.9/X11R7.0)"*. [cited 10 January 2009]. Available from world wide web: http://www.x.org/archive/X11R7.0/doc/html/index.html

Nagios. [cited 6 January 2009]. Available from world wide web: http://www.nagios.org/

Conky. [cited 6 January 2009]. Available from world wide web: http://conky.sourceforge.net/

GkrellM. [cited 6 January 2009]. Available from world wide web: http://members.dslextreme.com/users/billw/gkrellm/gkrellm.html

GNOME Documentation Library. System Monitor Manual V2.1. [cited 12 January 2009]. Available from world wide web: http://library.gnome.org/users/gnome-system-monitor/

Manual de ksysguard. [cited 12 January 2009]. Available from world wide web: http://docs.kde.org/stable/es/kdebase-workspace/ksysguard/index.html

Spong - Systems and Network Montoring. [cited 12 January 2009]. Available from world wide web: http://spong.sourceforge.net/

PandoraFMS. [cited 12 January 2009]. Available from world wide: http://pandorafms.org/

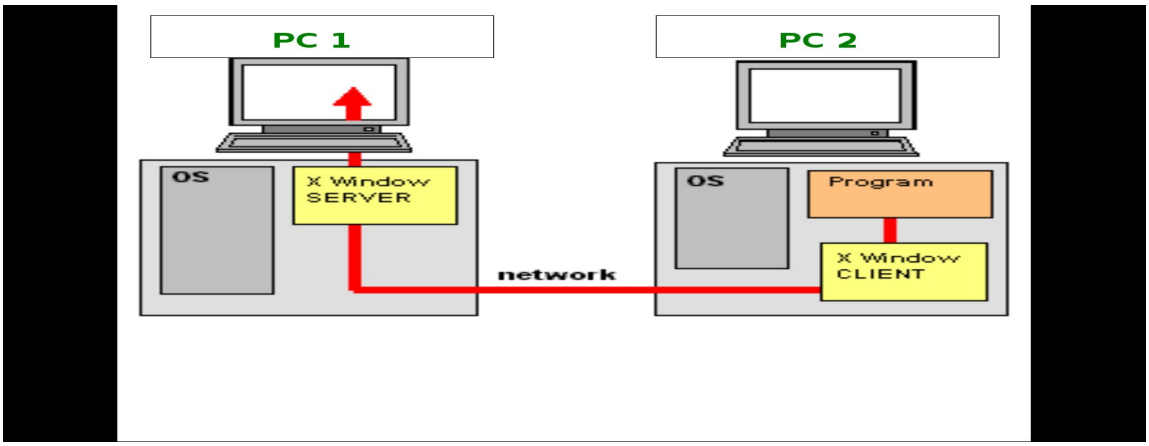
Monit. [cited 12 January 2009]. Available from world wide: http://mmonit.com/monit/

Jorba Esteve, Josep, Suppi Boldrito, Remo. Administración Avanzada de GNU/Linux. Available from world wide: http://10.33.2.200/Documentacion/Generales/Administracion/Adminsitracion/Admin_GNULinux.pdf

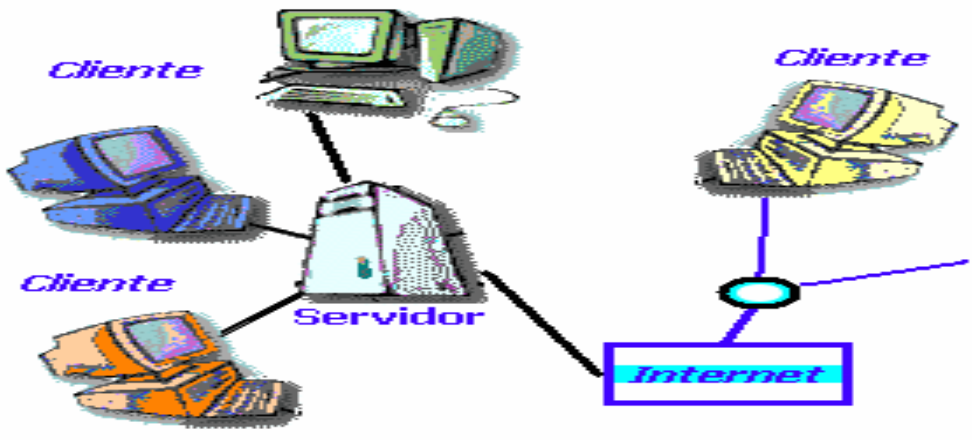
The GNOME Foundation. GNOME. [cited 20 March 2009]. Available from world wide web: http://www.gnome.org/

Anexos

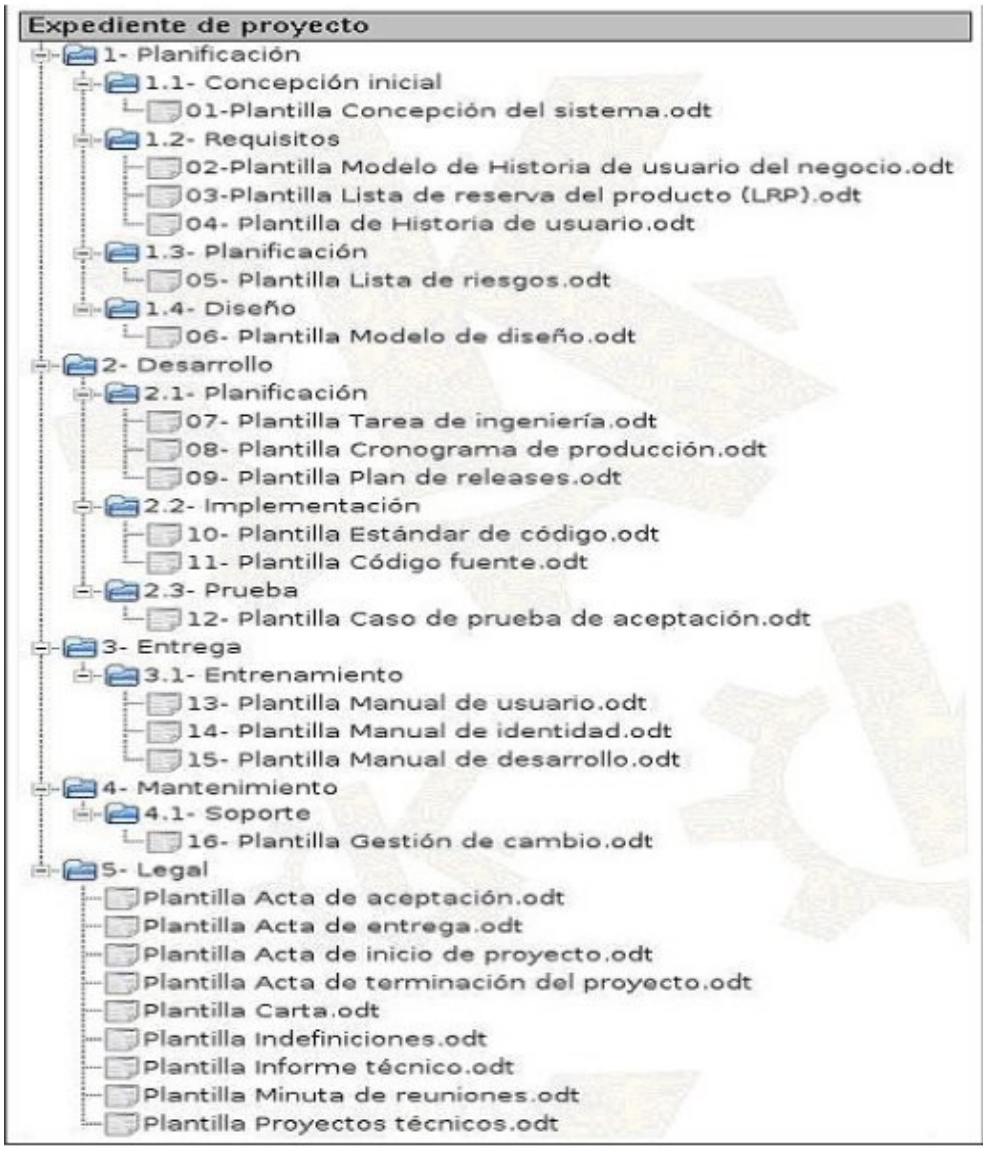
Anexo 1. Arquitectura del sistema X Window System.



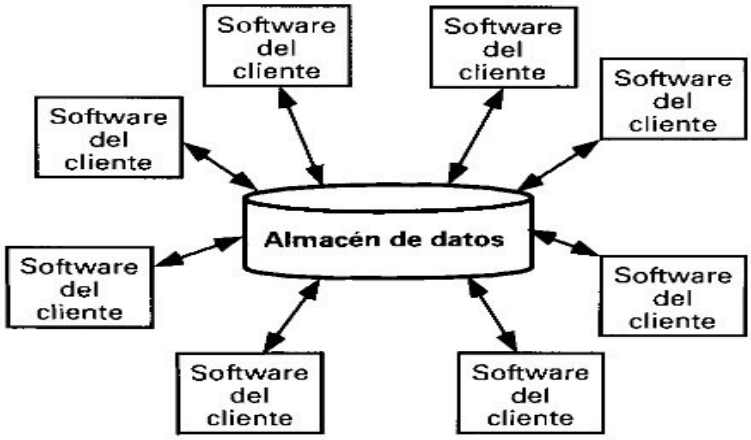
Anexo 2. Modelo Cliente/Servidor.



Anexo 3. Expediente de proyecto de la metodología SXP.



Anexo 4. Arquitectura centrada en datos.



Anexo 5. Prototipo de la Histotia de Usuario "Obtener datos de ventanas".

```
Usuario: pepito

Aplicacion: Anjuta
Ventana: Anjuta - /home/pepito/Desktop/script.sh

Aplicacion: Nautilus
Ventana: PROGRAMACION - Navegador de archivos

Aplicacion: OpenOffice.org
Ventana: Tesis_pepito_v2 - OpenOffice.org
Ventana: PDF Options
Ventana: Save
rpelegrino@debian:~/Desktop$
```

Shell

rpelegrino@debian: ~/Desktop - KDE Terminal Emulator

Anexo 6. Prototipo de la Histotia de Usuario “Generar documento XML”.

```
−<report>
    <user> rpelegrino </user>
    <date> 2009-04-24 </date>
    <ip> 10.33.2.117 </ip>
  −<process t="9">
    +<aplication name="nautilus" time="18" t="1"></aplication>
    +<aplication name="mozilla" time="321" t="6"></aplication>
    +<aplication name="gedit" time="155" t="2"></aplication>
    +<aplication name="gajim" time="30" t="1"></aplication>
    +<aplication name="konqueror" time="543" t="6"></aplication>
    +<aplication name="gimp" time="316" t="4"></aplication>
    +<aplication name="anjuta" time="343" t="3"></aplication>
    +<aplication name="stardict" time="32" t="6"></aplication>
    +<aplication name="Terminal" time="316" t="1"></aplication>
    </process>
  </report>
```

Done

Anexo 7. Fragmentos de código de amvex .

```
# Cantidad de usuarios que tiene cuenta en el sistema el sistema
 ls /home/ | wc -l
 let cuentas=$(ls /home/ | wc -l)
# Enviar xml
 if [ $logeado = falso ]; then
    echo el usuario $user no esta logueado
    #verificar se existen xml en SALVAXML para enviarlo
```

```
    let cantxml=$(ls /usr/bin/SUPERVNE/SALVAXML/ | wc -l)
    if [ $cantxml -gt 1 ]; then
       cd /usr/bin/SUPERVNE/SALVAXML
       ./CLIXML
    fi
  ...
 fi

# Limpiar procesos que queden vivo al salir de la sesión gráfica
# detener el GEDXML del usuario
 xml=$(ps aux | awk '{print $11}' | grep  GEDXML$user)
 while [ -n "$xml" ]; do
   idpr=$(ps aux | grep GEDXML$user  | awk '{print $2}'| head -1 | tail -1)
   kill $idpr
   xml=$(ps aux | awk '{print $11}' | egrep  GEDXML$user)
 done

# detener el  CtrlVSXW del usuario
 proceso=$(ps aux | grep CtrlVSXW$user  | awk '{print $11}')
 while [  -n "$proceso" ]; do
   idpr=$( ps aux | grep  CtrlVSXW$user  | awk '{print $2}'| head -1 | tail -1)
   kill $idpr
   proceso=$(ps aux | grep CtrlVSXW$user  | awk '{print $11}')
 done

 # detener el  TempCtrlV del usuario
 temp=$(ps aux | awk '{print $11}' | grep  TempCtrlV$user)
```

```
 while [  -n "$temp" ]; do
   idpr=$( ps aux | grep  TempCtrlV$user | awk '{print $2}'| head -1 | tail -1)
   kill $idpr
   temp=$(ps aux | grep TempCtrlV$user  | awk '{print $11}')
 done

# Lanzando procesos para supervisar y notificar tareas del DotProject
 # Verificar si están corriendo los procesos: CtrlVSXW, GEDXML y TempCtrlV.
 corriendo=$(ps aux | awk '{print $11}' | grep CtrlVSXW$usuario_logueado)
 xml=$(ps aux | awk '{print $11}' | grep GEDXML$usuario_logueado )  #> /usr/bin/trash 2> /usr/bin/trash
 temp=$(ps aux | awk '{print $11}' | grep TempCtrlV$usuario_logueado ) #> /usr/bin/trash 2> /usr/bin/trash
  # preguntar si está corriendo CtrlVSXW, de lo contrario lanzarlo.
if [ -z "$corriendo" ]; then
    #entra en caso de no estar corriendo el proceso
   cd /usr/bin/SUPERVNE/$usuario_logueado
  ./CtrlVSXW$usuario_logueado &
  echo LANZADO CtrlVSXW$usuario_logueado
else
  echo Esta corriendo CtrlVSXW$usuario_logueado
 fi

 # preguntar si está corriendo GEDXML, de lo contrario lanzarlo.
 if [ -z "$xml" ]; then
  #entra en caso de no estar corriendo el proceso
  cd /usr/bin/SUPERVNE/$usuario_logueado
   # se lanza el DATASERV para gestionar la hora en el servidor
  ./DATASERV &
```

```
  sleep 2
  cd /usr/bin/SUPERVNE/$usuario_logueado
   #se lanza noticli para notificar las tareas productivas del DotProject
  ./noticli &
  cd /usr/bin/SUPERVNE/$usuario_logueado
   #se lanza GEDXML para generar una estructura de datos
  ./GEDXML$usuario_logueado &
 else
   echo Esta corriendo GEDXML$usuario_logueado
 fi

# preguntar si está corriendo TempCtrlV, de lo contrario lanzarlo.
 if [ -z "$temp" ]; then
   cd /usr/bin/SUPERVNE/$usuario_logueado
   ./TempCtrlV$usuario_logueado &
   echo LANZADO TempCtrlV$usuario_logueado
 else
    echo Esta corriendo TempCtrlV$usuario_logueado
 fi
```

Anexo 8. Fragmentos de código de CtrlVSXW.

Librerías para trabajar con xlib

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xresource.h>
#include <X11/keysym.h>
```

```
#include <unistd.h>
#include <X11/Xatom.h>

Obterner nombre de la aplicación
 #variable para guardar el nombre de la aplicación
 char * aplicacion;
 #variable de tipo wm_class que facilita la datos de una ventana
 XClassHint wm_class;
 #entero es lavariable que guarda si se realizó la conexión con el servidor X, la función que se llama
 #obtiene el  nombre de la ventana que se solicite, en este caso el parámetro ventana es un atributo de la
 #clase que guarda la ventana que tiene el foco, la variable text_prop es una estructura que posee
valores
 #de la ventana
int entero = XGetWMName(display, ventana, &text_prop);
 #se declara una variable de tipo ventana
 Window ventana_auxiliar = ventana -1;
   if (entero == 1)
    {
      #función que se llama para obtener una estructura donde se guarda el nombre de la aplicación
      #de la ventana que tiene el foco
       XGetClassHint(display,ventana, &wm_class);
       #se obtiene el nombre de la aplicación
       aplicacion = wm_class.res_name;

     }
     else
     {
```

```
    #función que se llama para obtener una estructura donde se guarda el nombre de la aplicación
    #de la ventana que tiene el foco
     XGetClassHint(display,ventana_auxiliar, &wm_class);
     #se obtiene el nombre de la aplicación
     aplicacion = wm_class.res_name;
   }
```

Anexo 9. Fragmentos de código de GEDXML.

Estructura de datos para recibir mensajes.

```
struct Mi_Tipo_Mensaje
{
  #identificador del mensaje
  long Id_Mensaje;
  #nombre de la aplicación
  char Mensaje1[100];
  #nombre de la ventana
  char Mensaje2[100];
  #tiempo que usó la ventana
  int tiempo;
}cola;
```

Recibir información a través de colas de mensajes

```
 key_t Clave1;
 int Id_Cola_Mensajes;
 /* Igual que en cualquier recurso compartido (memoria compartida, semaforos  o colas) se obtien una
clave a partir de un fichero existente cualquiera  y de un entero cualquiera. Todos los procesos que
```

```
quieran compartir este semáforo, deben usar el mismo fichero y el mismo entero.
*/
 Clave1 = ftok ("puente", 33);
 if (Clave1 == (key_t)-1)
 {
   printf("Error al obtener clave para cola mensajes\n") ;
   return;
 }
//Se crea la cola de mensajes y se obtiene un identificador para ella
// El IPC_CREAT indica que cree la cola de mensajes si no lo está ya.
// el 0600 son permisos de lectura y escritura para el usuario que lance
// los procesos. Es importante el 0 delante para que se interprete en
// octal.

  Id_Cola_Mensajes = msgget (Clave1, 0600 | IPC_CREAT);
  if (Id_Cola_Mensajes == -1)
  {
    printf("Error al obtener identificador para cola mensajes\n");
   return;
  }
  // Se recibe un mensaje del otro proceso. Los parámetros son:
  //  Id de la cola de mensajes.
  // Dirección del sitio en el que queremos recibir el mensaje,
  //convirtiéndolo en puntero a (struct msgbuf *).
  //Tamaño máximo de nuestros campos de datos.
  // Identificador del tipo de mensaje que queremos recibir. En este caso
  //se quiere un mensaje de tipo 1, que es el que envia el proceso
```

```
 //flags. En este caso se quiere que el programa quede bloqueado hasta
 //que llegue un mensaje de tipo 1. Si se pone IPC_NOWAIT, se devolverá
 //un error en caso de que no haya mensaje de tipo 1 y el programa
 //continuará ejecutándose.
   printf("RECIBE:\n");
 msgrcv (Id_Cola_Mensajes, (struct msgbuf *)&rodo,
sizeof(rodo.Mensaje1)+sizeof(rodo.Mensaje2)+sizeof(rodo.minutos),1, 0);
```

Anexo 10. Fragmentos de código de CLIXML.

Conectar con un servidor remoto a traves de socket INET

```
struct sockaddr_in Direccion;
struct servent *Puerto;
struct hostent *Host;
int Descriptor;

//obtener puerto para la conexión ubicado en el fichero /etc/service
Puerto = getservbyname (Servicio, "tcp");
 if (Puerto == NULL)
{
   return -1;
}
 //obtener el host del servidor ubicado en el fichero /etc/hosts
 Host = gethostbyname (Host_Servidor);
 if (Host == NULL)
 {
   return -1;
```

```
 }
 //llenar estructura para realizar la conexión con el servidor
  Direccion.sin_family = AF_INET;
  Direccion.sin_addr.s_addr = ((struct in_addr *)(Host->h_addr))->s_addr;
  Direccion.sin_port = Puerto->s_port;
 //obtener descriptor para la conexión
  Descriptor = socket (AF_INET, SOCK_STREAM, 0);
  if (Descriptor == -1)
  {
    return -1;
 }
   //establecer conexión con el servidor
  if (connect ( Descriptor,(struct sockaddr *)&Direccion,sizeof (Direccion)) == -1)
  {
    return -1;
  }
```

Anexo11. Estádar de código utilizado.

Variables:

No manejar más de una instrucción por línea.

Declarar las variables en líneas separadas.

Ejemplo:

int a;

int b;

Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.

Deben incluirse espacios en ambos lados de los operadores binarios.

Ejemplo:

y = 50 + 15 - x;

Para hacer más clara la expresión, es aceptable agregarle paréntesis innecesarios. Dichos paréntesis se llaman paréntesis redundantes.

Identación:

El identado debe ser de cuatro espacios por bloques de código. No se recomienda utilizar el tabulador, ya que el mismo puede variar según su configuración.

Llaves:

Los inicios y cierres de llaves, deben ir en una nueva línea. Se recomienda su utilización aunque exista una sola instrucción.

Ejemplo:

```
int Calcular_Tiempo()
{
  int tiempo = minuto;
  return tiempo;
}
```

Cuerpo de declaraciones compuestas por if, while, for , etc:

En el caso de las instrucciones que permiten alternativas de implementación sin el uso de llaves como el caso del if, pueden tenerlas para lograr una mejor compresión del código, la instrucción debe ser colocada en una línea independiente.

Ejemplo:

```
int Calcular_Tiempo()
{
  int tiempo = minuto;

  if ( tiempo > 0 )
  {
```

```
      tiempo += segundo;
    }
    return tiempo;
  }
```

Comentarios:

Los comentarios deben ser completos y exactos. Se debe comentar a medida que se escribe el código, esto ayuda a tener mejor conocimiento de lo que se está programando. Siempre describir lo que está pasando, cómo se está haciendo y qué significan los parámetros, qué variables se usan y cuáles son modificadas. Además se tiene que alertar de cualquier restricción en el código.

Los comentario de línea, mejoran la legibilidad del código y pueden usarse para separar fragmentos de difícil compresión.

Ejemplo:

```
  int Calcular_Tiempo()
  {
    código 1;
    //-----------------------------------
    código difícil de comprender;
    //-----------------------------------
  }
```

Uso de espacios en blancos:

Los operadores binarios (lógicos, aritméticos, de comparación, etc) deberán estar separados por un espacio antes y después del mismo.

Ejemplo:

```
if ( ( a < 2 ) && ( b < 34 ) )
```

Clases:

El nombre de la clase debe estar relacionado con la funcionalidad que va a realizar. Debe ser concreto, compuesto por una sola palabra y comenzar con letra mayúscula, colocándose la llave que le sigue,

debajo del nombre de la clase.

Ejemplo:

```
class Supervisar
{


};
```

Otras características a considerar:

Ejecución correcta: funciona de acuerdo a las especificaciones.

Ejecución eficiente: uso mínimo de recursos de tiempo y memoria.

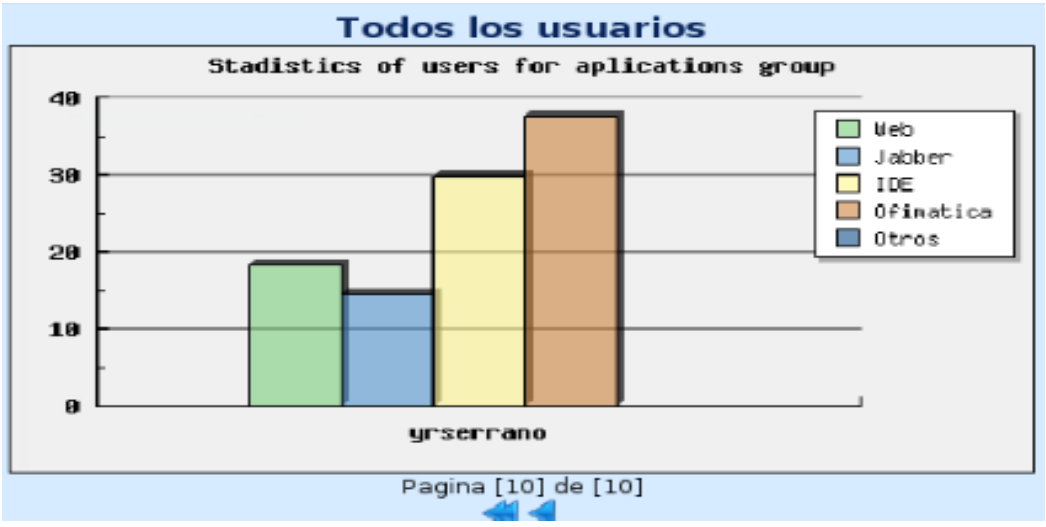
Fácil de leer y comprender.

Fácil de depurar.

Fácil de mantener.

Interfaz de usuario independiente de las funciones de cálculo.

Anexo12. Estadísticas sobre el uso de aplicaciones.




Estadisticas por aplicaciones
% de uso de los grupos de aplicaciones
65%
35%
gnumeric
openoffice

Glosario de términos

X WINDOW: El sistema X Window (abreviado X o X11 para la versión 11) es un método independiente para proporcionar capacidades gráficas a un sistema operativo.

PROTOCOLO: Método estándar que permite la comunicación entre procesos (que

potencialmente se ejecutan en diferentes equipos).

GPL: 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 a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

FREEWARE: Es un programa completamente gratis, normalmente los autores únicamente piden que no sea distribuido como propio o revendido. Fácilmente tu puedes encontrar un programa que cubra tus necesidades tan bien como un programa comprado.

RAM: Son las siglas de Random Access Memory, un tipo de memoria de ordenador a la que se puede acceder aleatoriamente; es decir, se puede acceder a cualquier byte de memoria sin acceder a los bytes precedentes. La memoria RAM es el tipo de memoria más común en ordenadores y otros dispositivos como impresoras.

CPU: Abreviatura de Central Processing Unit (unidad de proceso central). La CPU es el cerebro del ordenador. A veces es referido simplemente como el procesador o procesador central, la CPU es donde se producen la mayoría de los cálculos. En términos de potencia del ordenador, la CPU es el elemento más importante de un sistema informático.

SWAP: Es el espacio de intercambio de una zona del disco (un fichero o partición) que se usa para guardar las imágenes de los procesos que no han de mantenerse en memoria física.

GTK+: Es un conjunto de bibliotecas escritas en C para crear interfases gráficas de usuario. Tiene un

arquitectura orientada a objetos desarrollada en tres bibliotecas:

- GLib: Brinda varias herramientas para usar cuando se programa con GTK+.
- GDK: Abstrae la funciones de dibujo de ventanas de bajo nivel
- GTK: Brinda un conjunto de widgets para usar cuando creas tus GUI.

PLUGIN: Un **plug-in** es un módulo de hardware o software que añade una característica o un servicio específico a un sistema.

GNOME: es un entorno de escritorio para sistemas operativos de tipo Unix bajo la tecnología X Window.

KDE: Acrónimo que significa "K Desktop Environment", uno de los GUI más importantes para sistemas UNIX.

XFCE: Es un rápido y ligero entorno de escritorio para sistemas operativos basados en Unix.

PERL: (**P**ractical **E**xtraction and **R**eport **L**anguage), es un lenguaje de programación desarrollado por Larry Wall (lwall at netlabs.com) inspirado en otras herramientas de UNIX como son: sed, grep, awk, c-shell, para la administración de tareas propias de sistemas UNIX.

OPEN SOURCE: Código abierto (open source en inglés) es un término que empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre, usándolo como reemplazo al ambiguo nombre original, en inglés, del software libre (free software). El software de código abierto (OSS por sus siglas en inglés) es software para el que su código fuente está disponible públicamente.

TCP/IP: Son las siglas de Protocolo de Control de Transmisión/Protocolo de Internet (en inglés *Transmission Control Protocol/Internet Protocol*), un sistema de protocolos que hacen posibles servicios Telnet, FTP, correo electrónico, y otros entre ordenadores que no pertenecen a la misma red.

PING: Es un rastreador de paquetes de Internet (Packet Internet Groper) que es empleado para verificar si un host o servidor está funcionando.

SMTP: Simple Mail Transfer Protocol, o protocolo simple de transferencia de correo.

POP3: Siglas de Post Office Protocol 3 (protocolo de oficina de correos 3). Es el protocolo utilizado para recuperar los mensaje de correo almacenados en un servidor.

HTTP: (Siglas inglesas de *Hypertext Transfer Protocol*). Es el conjunto de reglas para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos multimedia) en la World Wide Web.

SNMP: *Network Management Protocol* (por sus siglas en inglés) es un protocolo de la capa de aplicación de la suite de protocolos TCP/IP, que facilita el intercambio de información administrativa entre dispositivos de red a fin de que los administradores puedan supervisar el desempeño de la red, buscar y resolver sus problemas, y planear su crecimiento.

ICMP: Por sus siglas de Internet Control Message Protocol*,* es el subprotocolo de control y notificación de errores del Protocolo Internet (IP).

SSL: Secure Socket Layer, es un protocolo que proporciona seguridad a la transmisión de datos en transacciones comerciales en Internet, como pueden ser compras a través de una tienda virtual.

SCRIPTS: Son pequeños programas formados a partir de una serie de instrucciones que serán usadas por otra aplicación.

HOST: Un host o anfitrión es un ordenador que funciona como el punto de inicio y final de las transferencias de datos. Más comúnmente descrito como el lugar donde reside un sitio web. Un host de Internet tiene una dirección de Internet única (dirección IP) y un nombre de dominio único o nombre de host.

FIFO: Método de estructuración de datos *que es* utilizado en colas que significa primero que entra, primero que sale.

KERNEL: Es el núcleo de un sistema operativo. Es el software que tiene la función de facilitar a los programas el acceso seguro al hardware y administrar los recursos del sistema, permitiendo el correcto funcionamiento del ordenador.

GENEST: Módulo desarrollado por el proyecto Unicornios de la facultad 10 de la Universidad de las

Ciencias Informáticas para el manejo de estadísticas del uso de aplicaciones.