Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

ADMINISTRACIÓN DE SERVIDORES LINUX A TRAVÉS DE UN CLIENTE MÓVIL

Informe final del Proyecto para optar al Título profesional de

Ingeniero de Ejecución en Informática

Autores:

Juan Francisco García Toledo - Sergio Andrés Salas Sánchez

Profesor Guía:

Iván Mercado Bermúdez

Profesor Co-referente:

José Miguel Rubio León

Diciembre 2007

“A mis padres y a mi novia,
a mi abuela materna por su inconmensurable labor
de cuidado y atención”

Juan Francisco García Toledo

“A mi madre por su constante esfuerzo, apoyo y confianza,
a mi padre por su respaldo en estos años de estudio”

Sergio Andrés Salas Sánchez

Resumen

La administración de servidores en la informática corresponde a la gestión, control y supervisión del correcto funcionamiento de los mismos, labores que demandan tiempo y dedicación, más aún si los servicios ofertados son de uso masivo. En la actualidad existen herramientas basadas en web con el fin de ayudar a dichos trabajos para el caso que se necesite alguna intervención remota, no obstante persiste la incertidumbre puesto que solo se conoce el real estado cuando se ejecutan dichas herramientas.

Con los avances en la tecnología móvil, es posible la creación de un sistema de administración y supervisión remota de servidores, Linux específicamente, utilizando solo como medio algún teléfono celular.

La inclusión de internet, la creación de protocolos específicos para navegación, plataformas para el desarrollo de aplicaciones y la integración de nuevas y cada vez más avanzadas formas de conexión, han permitido convertir al teléfono celular en un punto de convergencia de innovación tecnológica.

Palabras-claves: *tecnología móvil, administración remota, servidor, Linux.*

Abstract

The administration of servers in computing lies with the management, control and monitoring of proper functioning of the same, tasks that demand time and dedication, even more so if the services offered are of massive use. Actually exist tool web with the aim to help in that works in the case was necessary some intervention, however persist uncertainly because just only know the real state when execute that tools.

With the advance in the mobile technology is possible the creation to administration system and some server supervise, specifically Linux, using only the cell phone.

The internet inclusion, the creation to specific protocol for research, platforms for development applications and the integrated of new and more advances forms to connection, can make convert the cell phone in a convergence point of technology innovation.

Key-words: *mobile technology, remote administration, server, Linux.*

Glosario de Término

Applets: Se trata de pequeñas aplicaciones escritas en el lenguaje de programación JAVA y que se difunden a través de la red de Internet para ejecutarse en el visualizador cliente del usuario.

Bluetooth: Estándar de comunicación que permite poner en conexión distintos terminales (teléfonos móviles, ordenadores) por medios inalámbricos.

Log: Archivo que registra movimientos y actividades de un determinado programa (Log file).

Middleware: Es el término usados para referirse a los componentes de software que actúan como intermediarios entre otros componentes de software, generalmente, en el marco de la interacción cliente/servidor.

Midlet: Un Midlet es el nombre adoptado por pequeñas aplicaciones JAVA, esta puede utilizarse localmente sin conexión o estableciendo un canal de comunicación vía GSM, GPRS o UMTS para la transferencia de información con servidores remotos. Además, presenta otras características como el almacenaje de contenidos adicionales en los dispositivos donde se instalen. Generalmente, son los juegos y aplicaciones que funcionan en un teléfono celular.

Open Source: Es el término con el que se conoce al software distribuido y desarrollado libremente.

Servicio: Es un programa que se puede ejecutar utilizando Internet. Ejemplos: correo electrónico, Ftp, etc.

Servidor: Un ordenador que está conectado a Internet y presta algún servicio.

Servlet: Los Servlets son objetos que corren dentro del contexto de un servidor de aplicaciones (Ej.: Tomcat) y extienden su funcionalidad.

Shell: Interprete de órdenes o comandos en ambientes Linux y Unix.

Tomcat: Servidor web con soporte de Servlets y JSPs.

Netbeans: Se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma Netbeans.

Lista de Abreviaturas o Siglas

FTP: File Transfer Protocol.

FTPS: Comúnmente referido como FTP/SSL.

GSM: Global System For Mobile Communications.

GNU: Acrónimo recursivo que significa GNU not UNIX.

GPL: General Public License

GPRS: General Packet Radio Service.

GUI: Graphics User Interface.

HTTP: HyperText Transfer Protocol.

HTTPS: Versión segura del protocolo HTTP.

IP: Internet Protocol.

POP3: Post Office Protocol.

SSH: Secure Shell.

SMS: Short Message Service.

SSL: Secure Sockets Layer.

SMTP: Simple Mail Transfer Protocol.

NNTP: Network News Transport Protocol.

MAC: Message Authentication Code.

MIDP: Mobile Information Device profile.

TLS: Transport Layer Security.

TCP: Transmission Control Protocol, Protocolo de Control de Transmisión.

UML: Unified Modeling Language.

Capítulo 1
Introducción

El origen de los servidores comienza desde los inicios de los computadores al tener la necesidad de facilitar servicios a una cierta cantidad de clientes, desde la década de los 60’s con los primeros Mainframes de IBM que ya contaban con alrededor de 15.000 clientes, hasta la fecha el crecimiento de dichas maquinas ha sido explosivo, según las últimas estadísticas hacia el año 2006, se contaba en el mundo con alrededor 101.435.253 servidores activos con un número de clientes total alrededor mil cien millones de usuarios. La masificación de internet ha posibilitado este sombroso crecimiento llegando a la siguiente conjetura, un usuario nuevo que ingresa a la red es un potencial cliente.

Los usuarios al acceder a la red dirigen su atención a resolver sus demandas que se traducen simplemente en peticiones, peticiones encausadas generalmente a servidores específicos, especializados en dar respuestas a estas solicitudes, es labor del administrador de servidores dar gratificación de estos clientes ya que es responsabilidad de ellos el correcto funcionamiento de los servicios brindados por estos.

Por el hecho de ofrecer servicios se está sujeto a estar atentos al comportamiento de los clientes, la idea principal es retenerlos y dejar de alguna manera la mayor satisfacción en ello puesto que por cada cliente leal este acarreará otros 3 mas, esto se logra entregando servicios de calidad que persigan por sobre todas la cosas la gratificación del cliente, por lo tanto el administrador juega un papel fundamental en lo que implica tener clientes satisfechos, como encargado de los servidores y los servicios ofertados por los mismos, debe tener una constante preocupación por la seguridad, tiempos de respuesta, integridad de la información y estados de los servicio. En fin un sin número de tareas que requieren una intervención presencial.

Si consideramos que una jornada laboral comprende 8 hrs. diarias y tomando en consideración que los Servidores están en funcionamiento permanente, esto quiere decir las

24 hrs. del día, existe alrededor de un 70% de hrs. en las cuales el Administrador está en perfecto desconocimiento de los sucesos que en su servidores están ocurriendo, no hemos considerado el ausentismo del administrador por motivos ajenos a su labor que aumentan aun más el percentil.

De acuerdo a la anterior, se requiere contar con una herramienta capaz de entregar información veraz y oportuna del estado de los servicios y servidores en el momento en que el administrador no se encuentre en las estación de trabajo, el desconocimiento de los sucesos que allí ocurran sumado a esto la imposibilidad de ejercer un control remoto repercute sin duda alguna en la calidad del producto entregado para este caso servicios informáticos y de paso con esto en la satisfacción de los clientes o usuarios que hagan uso de estos.

Capítulo 2

Objetivos

2.1 Objetivo General

Implementar un software para dispositivos móviles con la lógica necesaria para llevar a cabo un control y administración eficaz tanto a servidores Linux como los servicios prestados por estos.

2.2 Objetivos Específicos

• Investigar las capacidades que nos entrega la tecnología móvil (comunicación, programación y arquitectura) para la integración en la administración de servidores Linux.

• Estudiar las plataformas de desarrollo de aplicaciones para teléfonos celulares, centrándose en las ventajas y deficiencias que ofrecen en torno al desarrollo este sistema.

• Conocer la administración de servidores y/o servicios para obtener las directrices que conforman el actuar del administrador.

• Conocer la forma de operar de los servicios dentro del servidor y la relación que tiene con el entorno mismo.

• Investigar mecanismos de seguridad que resguarden el flujo de información entre los Servidores y los Dispositivos Móviles.

• Crear una herramienta que apoye y/o facilite las labores del administrador.

Capítulo 3
Estado del Arte

3.1 Origen del problema

La administración de servidores desde su génesis siempre ha sido una tarea ardua, y que presenta a su vez una serie de inconvenientes propias del funcionamiento de este. El trabajo del administrador es un desafío constante del cual depende el buen servicio prestado a los usuarios y/o clientes finales de los servicios del servidor.

Dependiendo de la plataforma en la que se esté trabajando los procedimientos, funcionalidades y propósitos son diferentes, haciendo que la administración de un servidor Linux, Windows Server u otro, sea totalmente diferente. Independiente de la plataforma, los servicios en su mayoría pretenden un objetivo en común, que es entregar un servicio de calidad a los usuarios. La prestación de estos servicios (correo, ftp, bases de datos, etc.) es una tarea que requiere la inversión de una gran cantidad de conocimiento y tiempo, pero requiere una serie de herramientas que faciliten el control y constante monitorización de estos, haciendo que esto se transforme en algo central dentro de la administración de servidores.

Los problemas que se generan durante la ejecución de estos servicios pueden variar desde un error en la configuración, error en la programación durante el desarrollo, ataques maliciosos, poca robustez, inoperancia frente a errores inesperados, poca capacidad en la respuesta a peticiones múltiples, etc. Los errores que se presentan varían en cada plataforma y software que este en utilización y a su vez. Las soluciones a estos problemas también cambian de acuerdo a las versiones y de la implementación que se ha llevado a cabo, esto presenta un grave inconveniente dado que requiere una alta capacidad del administrador en conocer las diferentes aristas que se pueden dar, además debido a la gran

importancia de los servicios prestados por el servidor, es necesario que las correcciones sean realizadas en un tiempo menor.

Por lo tanto el contexto del problema se rige en la información con la que cuenta el administrador del servidor en el momento en que se presente un problema, además de la rapidez con que se actué para la solución de este, es por esto necesario contar con una herramienta que permita tener un conocimiento y control constante de los servicios prestados, además de una administración remota, haciendo que las respuestas a las dificultades que se puedan presentar sean de manera rápida y eficiente.

3.2 Tipos de servidores

De acuerdo a lo anterior podemos definir diferentes tipos de servidores (software) tal como:

• **Plataformas de Servidor (Server Platforms):** Un término usado a menudo como sinónimo de sistema operativo, la plataforma es el hardware o software subyacentes para un sistema, es decir, el motor que dirige el servidor.

• **Servidores de Aplicaciones (Application Servers):** Designados a veces como un tipo de middleware (software que conecta dos aplicaciones), los servidores de aplicaciones ocupan una gran parte del territorio entre los servidores de bases de datos y el usuario, y a menudo los conectan.

• **Servidores de Audio/Video (Audio/Video Servers):** Los servidores de Audio/Video añaden capacidades multimedia a los sitios Web permitiéndoles mostrar contenido multimedia en forma de flujo continuo (streaming) desde el servidor.

• **Servidores de Chat (Chat Servers):** Los servidores de chat permiten intercambiar información a una gran cantidad de usuarios ofreciendo la posibilidad de llevar a cabo discusiones en tiempo real.

• **Servidores de Fax (Fax Servers):** Un servidor de fax es una solución ideal para organizaciones que tratan de reducir el uso del teléfono pero necesitan enviar documentos por fax.

• **Servidores FTP (FTP Servers):** Uno de los servicios más antiguos de Internet, File Transfer Protocol permite mover uno o más archivos.

• **Servidores Groupware (Groupware Servers):** Un servidor groupware es un software diseñado para permitir colaborar a los usuarios, sin importar la localización, vía Internet o vía Intranet corporativo y trabajar juntos en una atmósfera virtual.

• **Servidores IRC (IRC Servers):** Otra opción para usuarios que buscan la discusión en tiempo real, Internet Relay Chat consiste en varias redes de servidores separadas que permiten que los usuarios conecten el uno al otro vía una red IRC.

• **Servidores de Listas (List Servers):** Los servidores de listas ofrecen una manera mejor de manejar listas de correo electrónico, bien sean discusiones interactivas abiertas al público o listas unidireccionales de anuncios, boletines de noticias o publicidad.

• **Servidores de Correo (Mail Servers):** Casi tan ubicuos y cruciales como los servidores Web, los servidores de correo mueven y almacenan el correo electrónico a través de las redes corporativas (vía LANs y WANs) y a través de Internet.

• **Servidores de Noticias (News Servers):** Los servidores de noticias actúan como fuente de distribución y entrega para los millares de grupos de noticias públicos actualmente accesibles a través de la red de noticias USENET.

• **Servidores Proxy (Proxy Servers):** Los servidores Proxy se sitúan entre un programa del cliente (típicamente un navegador) y un servidor externo (típicamente otro servidor Web) para filtrar peticiones, mejorar el funcionamiento y compartir conexiones.

• **Servidores Telnet (Telnet Servers):** Un servidor telnet permite a los usuarios entrar en un ordenador huésped y realizar tareas como si estuviera trabajando directamente en ese ordenador.

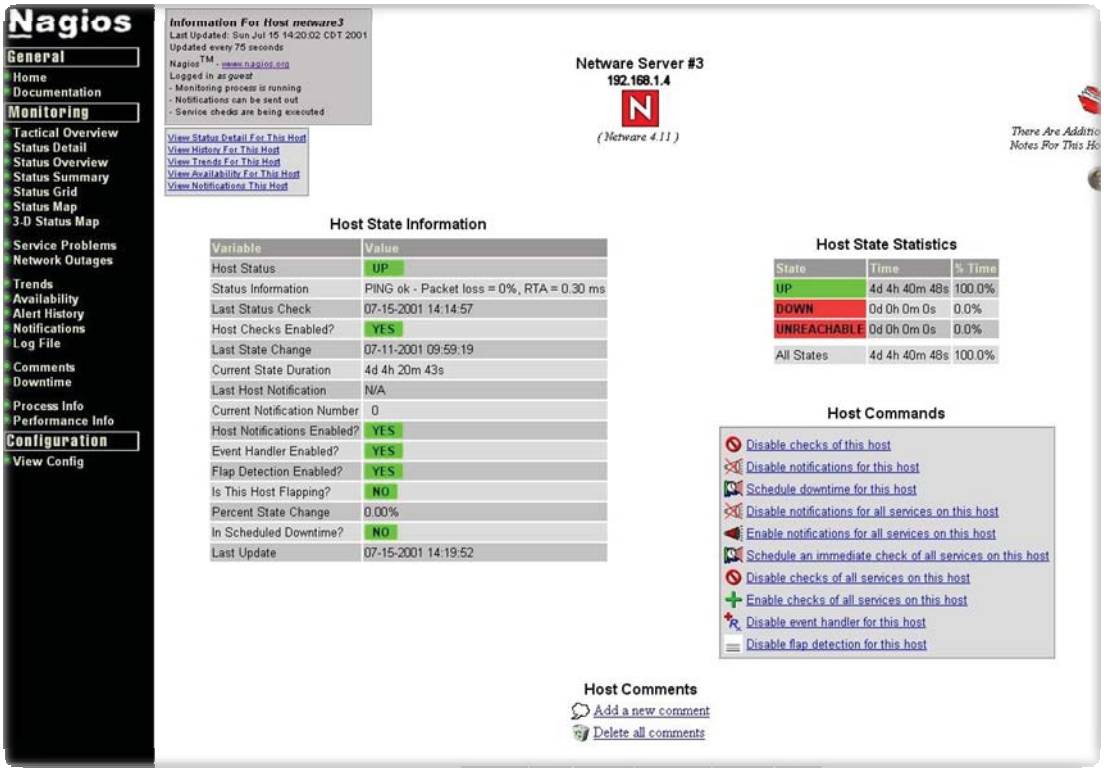
A lo largo de los años siempre ha existido cierto temor en utilizar los sistemas Linux como el medio en donde se proporcionen los servicios requeridos por los usuarios, pero hay que tener en cuenta el constante desarrollo y actualización lleva más de 20 años, haciéndolo uno de los sistemas más estables, potentes y confiables que existe en el mercado, además de ser de los más convenientes en los que respecta a dinero e inversión. A la vez la creación de empresas como Red Hat Enterprise Linux que proporciona su producto para servidores y aplicaciones de usuario otorga un soporte oficial, haciendo que Linux sus derivados (distribuciones) vayan aumentando a pasos agigantados su intromisión en el mercado, como un Sistema Operativo que actúa de manera eficiente en todos sus aspectos, como la seguridad, el aprovechamiento del hardware y precio.

3.3 Soluciones existentes en el mercado

En la actualidad se presentan una serie de alternativas de monitoreo que examinan la carga de los servidores y analizan los registros de los servicios, las cuales actúan de manera bastante eficiente como las que se presentan a continuación:

- **Nagios**

Ilustración 3.1 Nagios.



Esta herramienta además de monitorear el estado del servidor y de los servicios, también tiene una herramienta de notificación a través de mail y mensaje texto hacia teléfonos móviles, característica importante que informa el inconveniente no obstante existe un problema dado que si no se encuentra con una conexión a Internet simplemente no se puede hacer nada limitando la posible reparación del problema.

Algunas de las características de Nagios son:

› Monitoreo de servicios de red (SMTP, POP3, HTTP, NNTP, PING, etc.)

› Monitoreo de recursos de equipos (carga en el procesador, uso de disco duro y de la memoria, etc.)

› Supervisión de factores ambientales tales como temperatura

› Diseño simple de plugins que permite a los usuarios un desarrollo fácil de sus propias verificaciones de servicios.

› Revisión de servicios en paralelo.

› La habilidad de definir una jerarquía de los equipos de la red utilizando equipos "padre" (parent), permitiendo la detección y distinción de los equipos que están abajo y aquellos que son inalcanzables.

› Notificaciones a contactos cuando un servicio o equipo presenta problemas y necesita resolverse (vía e-mail, o método definido por el usuario).

› Habilidad de definir manejadores de eventos (event handlers) para ser ejecutados cuando un servicio o equipo presente eventos para que pueda ser proactiva la resolución del problema.

› Rotación de los archivos de Log automática.

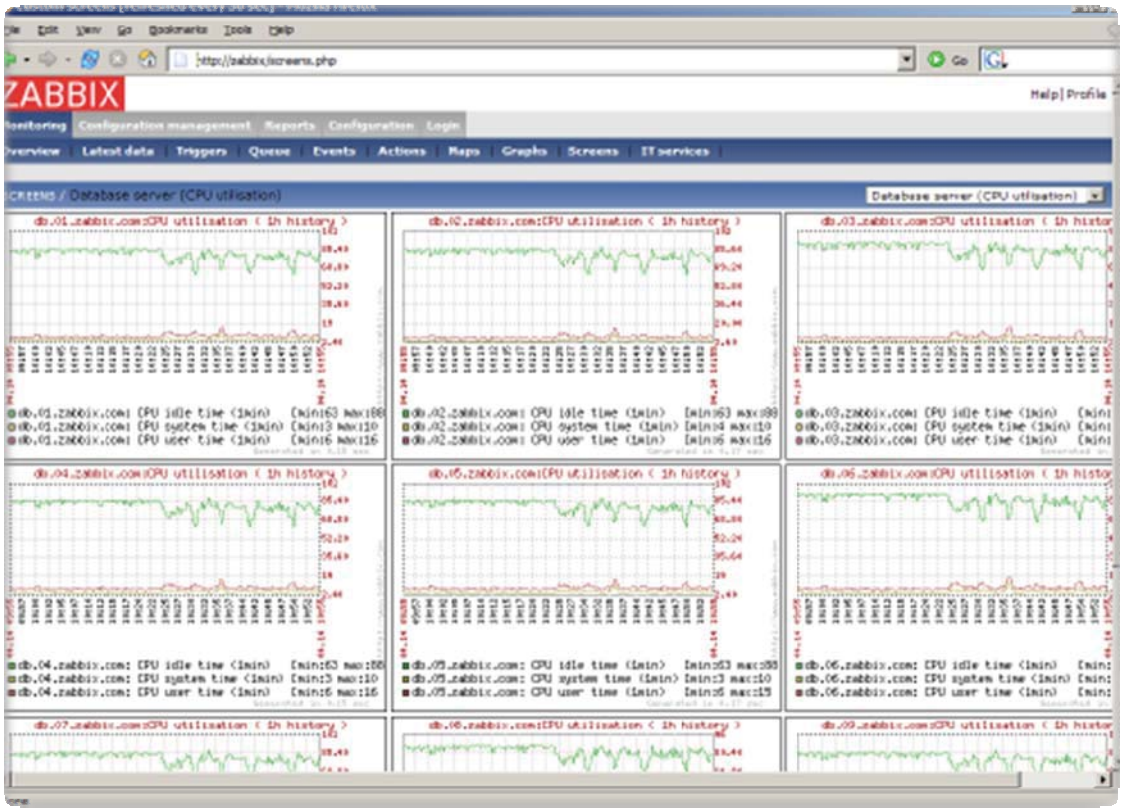
› Soporte para implementar equipos redundantes para monitoreo.

› Interface WEB opcional para ver el estado actual de la red, notificaciones, historial de problemas, archivo Log, etc.

- **Zabbix**

Otra herramienta que ayuda a la monitorización de servidores es Zabbix que permite supervisar tanto el uso, la red y los servidores. Consta de una interfaz de administrador vía Web y, siempre que haya algún problema, avisará con un correo electrónico de inmediato, pero el administrador al no encontrarse con una conexión a Internet no sabe lo que ocurre con su servidor.

Ilustración 3.2 Zabbix.



3.4 ¿Teléfonos celulares como solución?

Se puede observar que los software anteriormente presentados otorgan una serie de herramientas que permiten la administración eficiente de los servidores, la mayoría en forma remota, alertando de varias maneras (e-mail, Messenger, SMS, ventanas pop-up) del problema encontrado, esto representa una gran ventaja, pero al recibir la alerta es solo una parte de la solución del problema se hace necesario inaplazablemente resolverlo de manera dinámica e eficiente. Como se pudo observar la mayoría de las herramientas presentadas funcionan vía interfaz Web lo que implica tener una conexión constante con la Internet limitando así la posible solución del problema.

Todo esto conlleva a la creación e implementación de nuevas herramientas que ayuden y se complementen con las que existen en el mercado permitiendo a los administradores una administración más eficiente de los servidores, además de una constante monitorización de los servicios prestados por estos, auxiliando en los problemas más comunes que se

presentan al gestionar servidores, tales como: la detección de ataques maliciosos que mermen el buen funcionamiento del sistema en las prestaciones de los servicios más críticos y localización de errores de ejecución en los servicios.

Para lograr dicha administración, el método de resolución se enfoca en la constante entrega de información de parte del servidor al encargado de este y a la serie de personas designadas, que son capaces de dar una solución eficiente al problema presentado, esto se puede lograr monitoreando constantemente el servidor de acuerdo a su carga y estado de los servicios utilizados, obteniendo datos relevantes que permiten saber el real estado de la maquina, estos datos son analizados, resumidos y mandados al dispositivo móvil del o las personas encargadas del mantenimiento del servidor, para que a través de este se logre dar una solución, lo que representa una gran ventaja comparativa con las herramientas existentes en el mercado ya que como es conocido la Tecnología móvil otorga conectividad total durante las 24 horas del día, los 365 días del año y una cobertura casi absoluta, y así no depender de la Internet y todo lo que implica tener una conexión.

Capítulo 4
Metodología

4.1 Introducción

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantención, portabilidad, usabilidad, seguridad e integridad. La calidad del software es medible y varía de un sistema a otro o de un programa a otro.

Un software elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un software hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad; mientras que un producto de software para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas deriva dos de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantención y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

4.2 Importancia de la Metodología

Las metodologías de desarrollo de software son un conjunto de herramientas, procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

Actualmente es imprescindible considerar los riesgos, aunque habitualmente las empresas, no han sido concienciadas de los riesgos inherentes al procesamiento de la información mediante ordenadores, a lo que han contribuido, a veces, los propios responsables de informática, que no han sabido explicar con la suficiente claridad las consecuencias de una política de seguridad insuficiente o incluso inexistente. Por otro lado, debido a una cierta deformación profesional en la aplicación de los criterios de coste/beneficio, el directivo desconocedor de la informática no acostumbra a autorizar inversiones que no lleven implícito un beneficio demostrable, tangible y mensurable.

Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo.

Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí si toma sentido el basarnos en una metodología de desarrollo, y empezamos a buscar cual sería la más apropiada para nuestro caso. Lo cierto es que muchas veces no encontramos la más adecuada y terminamos por hacer o diseñar nuestra propia metodología, algo que por supuesto no está mal, siempre y cuando cumpla con el objetivo.

Muchas veces realizamos el diseño de nuestro software de manera rígida, con los requerimientos que el cliente nos solicitó, de tal manera que cuando el cliente en la etapa final (etapa de prueba), solicita un cambio se nos hace muy difícil realizarlo, pues si lo hacemos, altera muchas cosas que no habíamos previsto, y es justo éste, uno de los factores

que ocasiona un atraso en el proyecto y por tanto la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido. Obviamente para evitar estos incidentes debemos haber llegado a un acuerdo formal con el cliente, al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique al desarrollo del mismo.

Por experiencia, muchas veces los usuarios finales, se dan cuenta de las cosas que dejaron de mencionar, recién en la etapa final del proyecto, pese a que se les mostró un prototipo del software en la etapa inicial del proyecto.

Los proyectos en problemas son los que salen del presupuesto, tienen importantes retrasos, o simplemente no cumplen con las expectativas del cliente.

4.3 Evolución de los modelos de desarrollo

Durante mucho tiempo se ha utilizado el tradicional modelo en cascada, el cual ha demostrado que no refleja adecuadamente la complejidad que implica el proceso de desarrollo de software. Los problemas que presenta este modelo vienen de su propia estructura, al ser una secuencia de grandes etapas, las cuales requieren la documentación completa de la etapa previa, para poder continuar con la siguiente etapa. Para solucionar este problema se debe utilizar *métodos iterativos e incrementales*, que unidos a otras prácticas claves como la *orientación al manejo de riesgos y la planeación adaptable*, permiten de forma natural guiar adecuadamente el proceso de desarrollo de software

En los proyectos de tecnología se han incorporado progresivamente los cambios en los métodos de ingeniería, pero, desafortunadamente, no se ha hecho lo mismo en su administración.

Existe un conjunto de modelos de ciclo de vida en la gerencia de proyectos que han evolucionado, empezando por tradicional modelo en cascada, pasando por algunas propuestas de mejoramiento para el mismo, como son el caso de cascadas con fases solapadas o cascada con subproyectos. Estos modelos no permiten identificar tempranamente los riesgos, los cuales pueden aparecer en las etapas finales del desarrollo o

implantación, momento en que un cambio en el diseño del producto, en la arquitectura del sistema o en la infraestructura de software y hardware puede llevar al fracaso completo del proyecto.

Como respuesta a los problemas que presentan los modelos anteriores, los ciclos de vida evolucionaron y se han presentado propuestas, entre los cuales tenemos el modelo de entrega por etapas y el modelo de entrega evolutiva.

Cada uno de ellos adicionó mayores niveles de complejidad a la administración, pero aseguran poseer un marco de trabajo más sólido y ajustado para el desarrollo de proyectos con niveles moderados de riesgo. Se requería un modelo de ciclo de vida de proyectos que trabajara adecuadamente con altos niveles de riegos, así que se desarrolló el modelo en espiral. Este modelo tiene como objetivos la identificación de los riesgos para determinar la viabilidad del proyecto y definir planes de manejo para garantizar desde las fases iníciales la eliminación o disminución de los riesgos y la entrega desde las fases iníciales de productos probados, lo que permite un proceso continuo de pruebas y retroalimentación, entre las características que se encuentran están:

4.3.1 Desarrollo iterativo

El desarrollo iterativo es un método de construcción de productos cuyo ciclo de vida está compuesto por un conjunto de iteraciones, las cuales tienen como objetivo principal entregar versiones del software. Cada iteración es considerado como un subproyecto que genera productos de software, permitiendo al usuario tener puntos de verificación y control más rápidos y, por consiguiente, un proceso continuo de pruebas y de integración desde las primeras iteraciones. Las iteraciones están compuestas por el conjunto de disciplinas o actividades, las cuales son: la especificación de requerimientos, el análisis y diseño, las pruebas, la administración de la configuración y el proceso de gerencia de proyectos

4.3.2 Orientación al manejo del riesgo

Cada proyecto tiene asociado intrínsecamente un conjunto de riesgos que requieren un plan de manejo claramente establecido, documentado y con una implementación eficaz.

Con esto se pretende evitar posibles retrasos en los tiempos de entrega, problemas de calidad en el producto o en el peor de los casos, que puedan afectar la terminación del proyecto. Estos procesos pueden llegar a ser muy complejos, esto depende de la magnitud e importancia del proyecto. En las etapas iníciales se implementan las funcionalidades con mayor exposición al riesgo y las de mayor complejidad, lo cual incrementa la posibilidad de éxito del proyecto.

4.3.3 Desarrollo evolutivo

Cuando se trabaja con una especificación de requerimientos monolítica, se cae en el error de creer que se comprende completamente el concepto del producto sin haberlo validado con el cliente con algo más que documentos y modelos abstractos. Este proceso inicia con un concepto poco claro del producto a construir, y sólo se tiene claridad en la medida que se vaya desarrollando y verificando el producto con el cliente. Este tipo de proyectos se asemejan más al patrón que siguen los proyectos de investigación y desarrollo de nuevos productos. Estas prácticas claves en el desarrollo de software son implementadas en algunos métodos tales como Evo (Evolutionary Project Management, probablemente el primer método Iterativo e Incremental de Desarrollo de Software, publicado en 1976. y creado por Tom Gilb), el modelo en espiral y el proceso unificado (UP), siendo los dos últimos los que mayor trascendencia han tenido y serán explicados a continuación.

4.4 Proceso Unificado (UP)

El Proceso Unificado es un proceso de desarrollo de software configurable que se adapta a través de los proyectos variados en tamaños y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos en el desarrollo de software.

El Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto, para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura. El proceso describe qué entregables producir, cómo

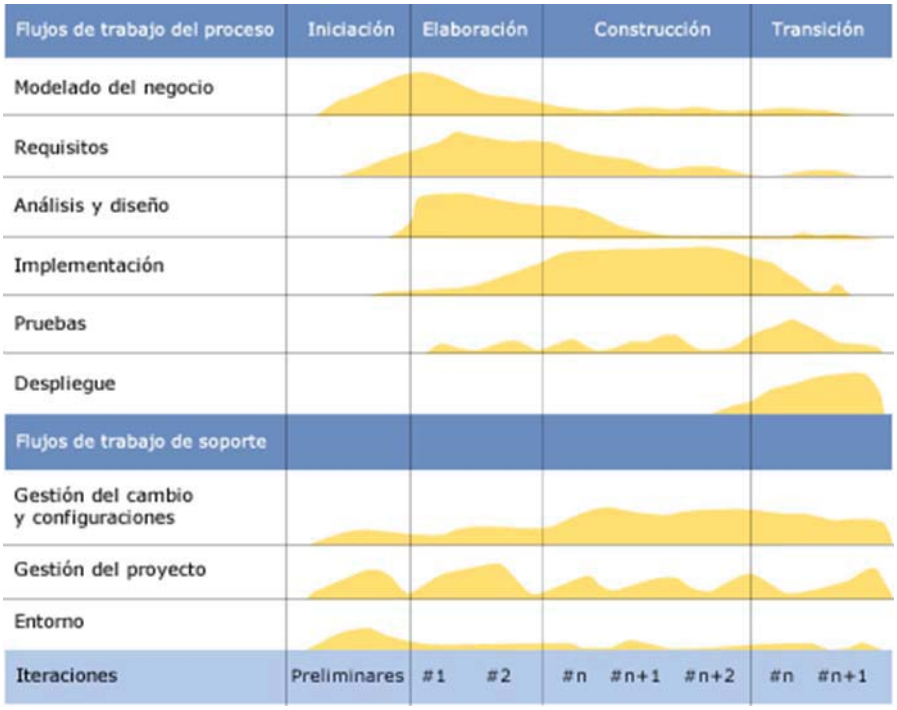
desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

La concepción de un sistema de información va mucho más allá de levantar los requerimientos, elaborar un conjunto de modelos y comenzar a programar. Esta concepción limitada ha permitido que durante años no podamos hacer uso adecuado de los conceptos y las herramientas con los que contamos. En este punto podemos considerar que la definición de la arquitectura del software se convierte en el eje orientador que permite controlar el desarrollo iterativo e incremental del sistema, a través de su ciclo de vida. Esta arquitectura se define en las primeras fases del proyecto, básicamente en la de elaboración, y se refina a través de todo el proyecto.

El ciclo de vida UP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

UP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

Ilustración 4.1 Relación fases y disciplinas.



Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una línea base de la arquitectura.

Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se

realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Por cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía

4.4.1 UP, Modelo de desarrollo para este sistema

Como se detallo en el punto anterior UP provee métodos para la identificación de riesgos en etapas tempranas de desarrollo en conjunto con la implementación del modelo en espiral desplegando el ciclo de vida en fases e iteraciones lo que permite un constante refinamiento en el desarrollo, a continuación se detalla las ventajas que provee UP sobre el desarrollo de este sistema.

4.4.1.1 Respecto a iteraciones e incrementalismo

Los desarrolladores basan la selección de las tareas a abordar en cada iteración partiendo de dos factores: la iteración maneja un grupo de casos de uso que extienden la utilidad del producto, además las iteraciones tratan siempre con los riesgos más altos en el estado actual del proyecto. Pero el desarrollo iterativo no es dividir, sino que las sucesivas iteraciones se construyen a partir de los artefactos en el estado en que fueron dejados en las iteraciones precedentes.

Cada iteración, considerada como un mini proyecto, a partir de los casos de uso aborda el análisis, diseño, implementación y test. Por supuesto, un incremento no es necesariamente aditivo, ya que, especialmente en las primeras fases del ciclo de vida, los desarrolladores pueden estar reemplazando un diseño superficial por un diseño más detallado. En las fases posteriores los incrementos suelen ser aditivos.

En cada iteración los desarrolladores analizan las casos de uso relevantes, crean un diseño utilizando como guía a la arquitectura elegida, e implementan el diseño en componentes

verificando que dichos componentes aportan la funcionalidad especificada en los caso de uso. Si una iteración cumple sus objetivos el proceso continuo con la siguiente iteración, por contra, si una iteración no cumple sus objetivos los desarrolladores deben revisar las decisiones previas e intentar una nueva aproximación.

Para alcanzar el mayor rendimiento un proyecto debería proceder casi siempre a lo largo de su curso directo con solo pequeñas desviaciones del curso inicialmente planeado. Por supuesto, desde el momento que problemas imprevistos a nadan iteraciones o alteren la secuencia planeada, el proceso requerirá un mayor esfuerzo y tiempo del calculado a priori. Minimizar los imprevistos es uno de los objetivos de la reducción de riesgos.

4.4.1.2 Respecto del Enfoque en los Riesgos

La gestión de los riesgos es contemplada desde el inicio del proyecto hasta el final del mismo a través de diferentes artefactos, fundamentalmente se manejan dos artefactos, el Plan de Gestión de Riesgos y el Registro de Riesgos, donde se describen los posibles riesgos de recursos, técnicos, o del negocio implicados en el proyecto, y formula un plan para abordar los mismos con medidas de mitigación y correctivas para afrontar cada uno de ellos.

El enfoque en los riesgos es una constante para cada proyecto informático más aun si este involucra requerimientos constantes o ámbitos tecnológicos innovadores, como es el caso de este sistema en donde la tecnología juega un papel fundamental en el desarrollo y posterior implementación pues es donde se sustenta la arquitectura de la misma.

El hecho de centrarse en identificar los riesgos críticos en una etapa temprana del ciclo de vida, la realización de iteraciones que involucran cada vez un análisis y refinamiento de los riesgos, en especial en la fase de Inicio y Elaboración, ahondan aun mas en los motivos de la elección de este modelo de desarrollo ya que asegura un buen desarrollo, con riesgos identificados y acotados sumando a ello planes de contingencia si el caso lo amerita.

4.4.1.3 Respecto al Concentración en la Arquitectura

La arquitectura de software permite a los ingenieros mayor control e introspección en el sistema en el proceso de desarrollo desde sus inicios, y promueve la temprana identificación y prevención de problemas. Como resultado, la arquitectura puede guiar el proyecto al éxito en vez de librarlo al fracaso por falta de entendimiento.

La arquitectura captura el conjunto de decisiones principales de diseño que se hacen sobre un sistema. Las decisiones de diseño son elecciones hechas pensando en cómo el sistema se desarrollará y cómo funcionará, y estas elecciones pueden incluir estructura, organización, funcionalidad, comportamiento, o más propiedades no funcionales como la usabilidad y estética [de la interfaz de usuario]. La importancia de estas decisiones de diseño varía para cada sistema de software y está en función de los interesados en el sistema, sus preocupaciones, y sus necesidades específicas.

Una cuestión clave es que, a pesar de que la arquitectura es fundamentalmente una actividad centrada en el diseño, afecta a todo el ciclo de vida. Las decisiones de la arquitectura se concentran en lo que es esencial en un sistema la influencia de los requerimientos y las actividades importantes con el diseño colaborativo, el desarrollo e implementación del sistema, la planeación de su evolución y su adaptación.

Las buenas prácticas de la arquitectura de software aplicadas durante el ciclo de vida, tienen el potencial de incrementar la facilidad para entender un sistema de software y de desarrollar procesos para crearlo, asegurando que cualidades particulares y relevantes sean alcanzadas, y que se reduzca el costo. Como una herramienta para mejorar la práctica de diseño, la arquitectura ofrece una forma efectiva de modelar sistemas de software mientras ayuda en el razonamiento y análisis del comportamiento antes de que el desarrollo sea terminado. Como las disciplinas y artefactos relativos al diseño, la arquitectura también tiene el potencial de garantizar beneficios importantes al proceso de desarrollo de todo el software incluyendo la evolución en tiempo de ejecución y adaptación.

Adicionalmente al acuerdo de la representación de la arquitectura, UP provee un proceso para diseñar la arquitectura a través de un conjunto de actividades y define un artefacto

fundamental llamado Arquitectura del Software para describir las vistas asociadas con los proyectos.

4.4.1.4 Guiado por los casos de uso

Como ya sabemos un caso de uso es una parte de la funcionalidad del sistema. El modelo de caos de uso, englobando todos los casos de uso del sistema, describe la funcionalidad completa del sistema. Este modelo reemplaza la especificación funcional tradicional del sistema: ``qué debe hacer el sistema" por ``qué debe hacer el sistema para cada usuario". Pero los casos de uso no son sólo una herramienta para especificar los requisitos del sistema, sino que también guían su diseño, implementación y testeado, o sea guían el proceso software.

Basado en el modelo de casos de uso, los desarrolladores crean modelos de diseño e implementación que realizan los casos de uso, los desarrolladores revisan cada uno de los modelos creados para comprobar si se corresponden con el modelo de casos de uso especificado. Igualmente los responsables del testeado del sistema comprueban la implementación para asegurarse de que los componentes del modelo de implementación implementan correctamente los casos de uso. En este sentido los casos de uso no solo inician el proceso software sino que permanecen ligados a él durante todo el ciclo de vida.

Capítulo 5

Desarrollo

5.1 Introducción

Como se expuso en el capítulo de metodología nuestro modelo de desarrollo de software ha sido Proceso Unificado.

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra una la distribución de tiempos y el número de iteraciones que tomó cada fase.

Tabla 5.1 Relación fases con número de iteraciones.

Fase	Nro. Iteraciones
Fase de Inicio	1
Fase de Elaboración	1
Fase de Construcción	2
Fase de Transición	-

El desarrollo se verá agrupado en la disciplinas que UP establece, en donde por cada una de ellas se expone el trabajo realizado y los documentos generados, como ya es sabido las iteraciones obligan a realizar el flujo de trabajo de estas disciplinas una y otra vez, poniendo énfasis en la Fase donde se aplica dicha iteración, esto genera revisiones y refinamientos de los documentos creados por cada disciplina, concibiéndose distintas versiones de un mismo documento.

El trabajo se ha organizado de forma de mostrar solo las últimas versiones de los artefactos, se refiere a esto a los documentos, modelos o diagramas agrupados en la disciplina que correspondan no reflejando la Fase en el cual se generaron.

5.2 Relación Artefactos y Fases

Las siguientes tablas muestran la correspondencia entre los artefactos y las fases en donde comienzan o se refinan para cada Disciplina, cabe destacar que los artefactos que aquí se muestran son los desarrollados durante el transcurso del desarrollo y reflejados en esta memoria.

Las disciplinas que conforman esta metodología se fundamentan en las propuestas por UP, las cuales se dividirán en dos grupos. El primero comprende las disciplinas fundamentales asociadas con las áreas de ingeniería y el segundo grupo lo integran aquellas disciplinas llamadas de soporte o gestión:

5.2.1 Primer Grupo

5.2.1.1 Disciplina requerimientos

Tabla 5.2 Artefactos de la disciplina de requerimientos.

Artefactos	Fi	Fe	Fc	Ft
Modelo de Casos de Uso	c	r	r	
Glosario del Sistema	c	r	r	r
Visión del Sistema	c	r		

5.2.1.2 Disciplina análisis y diseño

Tabla 5.3 Artefactos de la disciplina análisis y diseño.

Artefactos	Fi	Fe	Fc	Ft
Arquitectura del Software	c	r	r	
Mapa de Navegación		c	r	
Prototipo de la Interfaz de Usuario		c		
Modelo de Diseño		c	r	
Modelo de Implantación			c	r

5.2.1.3 Disciplina implementación

Tabla 5.4 Artefactos de la disciplina implementación.

Artefactos	Fi	Fe	Fc	Ft
Modelo de Implementación		c	r	r
Plan de Pruebas	C	r	r	R

5.2.1.4 Disciplina pruebas

Tabla 5.5 Artefactos de la disciplina pruebas.

Artefactos	Fi	Fe	Fc	Ft
Modelo de Implementación		c	r	r
Plan de Pruebas	C	r	r	r

5.2.1.5 Disciplina implantación

Tabla 5.6 Artefactos de la disciplina implantación.

Artefactos	Fi	Fe	Fc	Ft
Manual de Instalación			c	r
Manual de Usuario		c	r	r
Material de Adiestramiento			c	r

5.2.2 Segundo Grupo

5.2.2.1 Disciplina Gestión de Proyecto

Tabla 5.7 Artefactos de la disciplina gestión de proyecto.

Artefactos	Fi	Fe	Fc	Ft
Plan de Gestión de Riesgos	c	r	r	r
Términos de Referencia del Sistema	c			

5.2.2.2 Disciplina Gestión de Ambiente

Tabla 5.8 Artefactos de la disciplina gestión de ambiente.

Artefactos	Fi	Fe	Fc	Ft
Infraestructura de Desarrollo		c	r	

Fi: Fase de inicio	Fe: Fase de elaboración
Fc: Fase de construcción	Ft: Fase de transición
c: Comienzo de la construcción del artefacto	r: Refinamiento del artefacto

5.3 Resumen de las Iteraciones por Fases

5.3.1 Fase de Inicio

Durante la fase de inicio de este sistema, el equipo desarrollador se centro en la captura de los requerimientos otorgados por parte del cliente, en este caso el administrador de servidores de la escuela informativa de la PUCV, además de la constante consulta al profesor guía de este sistema. También se produjo la instancia donde los desarrolladores generaran una investigación meticulosa de las diferentes herramientas existentes en el mercado, que ayudarían el proceso de desarrollo, además de indagación de las diferentes posibilidades de solución del problema planteado por este sistema. A su vez se dio comienzo al modelado y se dio las directrices que se centraría el sistema a desarrollar, logrando obtener una mayor claridad del camino del sistema, observando sus ventajas y desventajas.

5.3.2 Fase de Elaboración

Durante la fase de elaboración se produjo un refinamiento de lo realizado durante la fase de inicio de este sistema, además de estar en el apogeo la disciplina de análisis y diseño, mejorando de esta manera el entendimiento y funcionamiento del sistema. Además se dio inicio a la creación de nuevos artefactos necesarios para el correcto funcionamiento del sistema, tal como el mapa de navegación, logrando fijar nuevas y mejoradas directrices del sistema. También se inicio la construcción del prototipo de la aplicación, actuando de manera real sobre el servidor, esto ayudo a comprobar las verdaderas capacidades del

sistema como de los desarrolladores, pudiendo así obtener un acercamiento real a la versión final del sistema y además de mejorar las confianzas del equipo desarrollador en lo que respecta las capacidades.

5.3.3 Fase de Construcción

5.3.3.1 Primera Iteración

Durante la primera iteración de esta Fase se presentaron problemas al momento de integrar los componentes que en la Arquitectura de Software se habían definido (*Ver Anexo 2, Arquitectura del Software*), en las primeras iteraciones de la Fase de elaboración de la arquitectura del software se definía el uso de una herramienta web que ayuda a la administración remota de Servidores para ser integrada al sistema con el propósito de que esta sirviese como componente intermedio entre el Midlet y el Servidor, gestionando las consultas y la interacción entre ambos componentes, las ventajas de este enfoque consistían en la reutilización de una serie de scripts que dicha herramienta poseía permitiendo visionar la construcción del sistema de forma más rápida y ágil.

No obstante, el resultado no fue lo esperado, no se logro integrar de buena manera por motivos de desconocimiento de la herramienta y la forma como integrar con el Midlet por tanto se debió cambiar la herramienta en busca de una solución que permitiese una rápida integración a modo de recuperar el tiempo invertido en una mala unificación.

5.3.3.2 Segunda Iteración

Para esta iteración los principales artefactos que se vieron afectado correspondieron al de Arquitectura de Software y el mismo sistema, como se expuso en el punto anterior no fue posible integrar los componentes que se habían elegido, específicamente el Midlet con la herramienta de administración web Nagios.

Se tomo la determinación de utilizar algún Middleware con la capacidad de interactuar con el Servidor como se refleja en la arquitectura de software (*Ver Anexo 2, Arquitectura del Software*) esta determinación se logró al momento de buscar una solución en conjunto con el profesor guía, en dicha reunión se instruyó a los alumnos del uso de comandos de

consola y el uso de script en Bash, que suplieran las funcionalidades que se esperaban obtener con el uso de Nagios. La funcionalidad que entregaría el Middleware seria además de gestionar la comunicación entre Midlet y Servidor la ejecución de comandos y los script ya mencionados.

Los resultados obtenidos fueron los esperados, la integración resultó exitosa, el middleware logro gestionar la comunicación.

5.3.4 Fase de Transición

Durante esta fase, se desarrollaron los distintos manuales que permiten al usuario final orientarse en los procesos de instalación y buen uso del sistema, cabe destacar que aunque la captura de requerimientos se hizo con un administrador de Servidores real, el sistema no fue pensado para un cliente especifico sino más bien un producto genérico que sirviese de utilidad a los administradores que cumpliesen con los requerimientos establecido para el uso del sistema.

Al no contar con un cliente definido, los artefactos generados bajo esta Fase se redujeron a solo la creación de los manuales ya mencionados obviando documentos que detallasen transición, implantación o integración, documentos relacionados con los procesos de entrega hacia clientes.

5.4 Disciplinas y Desarrollo de Flujos de Trabajo

5.4.1 Disciplina Requerimientos

5.4.1.1 Descripción

El objetivo principal de esta disciplina es establecer las funciones que se quiere que satisfaga el sistema a construir. En esta línea los requerimientos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requerimientos que se especifiquen.

Los objetivos específicos de la disciplina requerimientos son:

- Definir el ámbito del sistema.

- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

- Proveer a los desarrolladores un mejor entendimiento de los requerimientos del sistema.

- Proveer una base para estimar recursos y tiempo de desarrollo del sistema.

- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.

Los requerimientos pueden ser divididos en dos grupos: Los requerimientos funcionales, los cuales describen las funciones que el software va a ejecutar; por ejemplo, ajustarse a un formato de texto o modular una señal. Los requerimientos no funcionales, los cuales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus funciones específicas.

En esta disciplina, y como parte de los requerimientos de facilidad de uso, se diseña la interfaz gráfica del usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se validan con el usuario final.

5.4.1.2 Resumen Actividades, Sub-actividades y Tareas

a. Análisis del Problema

El problema radica en dar solución efectiva a la administración de Servidores Linux de forma remota para el caso en que el administrador no se encuentre en la estación de trabajo.

El impacto del problema reside en que al no saber el estado del o los Servidores y de los Servicios prestados por los mismos, crea ciertos grados de incertidumbre sobre todo si se habla de servicios cuyas prestaciones son críticas, el efecto resultante es desconocimiento sobre el real estado del Servidor y los Servicios, sumado a esto la inhabilitación para ejercer algún control de forma remota una vez enterado de algún malfuncionamiento o desperfecto.

Los afectados son todos aquellos sistemas, personas, sistemas o aplicaciones que hagan uso de los Servicios prestados por los Servidores administrados y gestionados por el administrador.

La satisfacción es el punto que se busca al momento de ofrecer servicios sobre todo si estos son de uso masivo o de gran impacto, aplicados en la informática estos servicios pueden ser de Email, Hosting o gestión Web o de entretenimiento masivo el impacto causado en los clientes es crítico cuando estos no están disponibles o su funcionamiento no es el correcto. Es labor del o los administradores o encargados a fines la disponibilidad y el correcto funcionar de los Servidores y sus Servicios, siendo la base para garantizar satisfacción y atraimiento de más clientes, sistemas o aplicaciones.

b. Determinación de Actores y Caso de Usos

La determinación de los Casos de Usos se logró gracias a una serie de entrevistas realizadas al Administrador de los Servidores de la Pontifica Universidad Católica de Valparaíso, Escuela de Ingeniería Informática específicamente y al profesor guía de esta memoria, junto a ellos se obtuvieron los requerimientos que en el Anexo de Requerimientos de software se especifican, no obstante, los alumnos quisieron dar un paso adelante y proponer la integración al sistema de una nueva funcionalidad , esta funcionalidad corresponde a la integración de una herramienta desarrollada para teléfonos celulares denominada MidpSSH (*Ver Anexo 2, Arquitectura del Software*), herramienta capaz de emular una consola de comando en Linux permitiendo control a bajo nivel de forma remota, el caso de uso que hace referencia a esta innovadora funcionalidad esta explicada en el mismo anexo.

5.4.1.3 Resumen de Artefactos

a. Especificación de Requerimientos del Software (ERS)

El objetivo de este artefacto es documentar todos los requerimientos del sistema, este describe las funciones del sistema, los requerimientos no funcionales, características del diseño, y otros elementos necesarios para proporcionar una descripción completa y comprensiva de los requerimientos para el software a desarrollar.

Los requerimientos pueden ser levantados con diferentes herramientas, también se pueden encontrar dispersos en varios artefactos. Es por ello, que la captura de todos los requerimientos para el ERS se complementa además por dos artefactos que describen los requerimientos que son: Modelo de Casos de Uso y Especificaciones Suplementarias.

Ver Anexo 5, Especificación de requerimientos.

b. Modelo de Casos de Uso

1. Actores

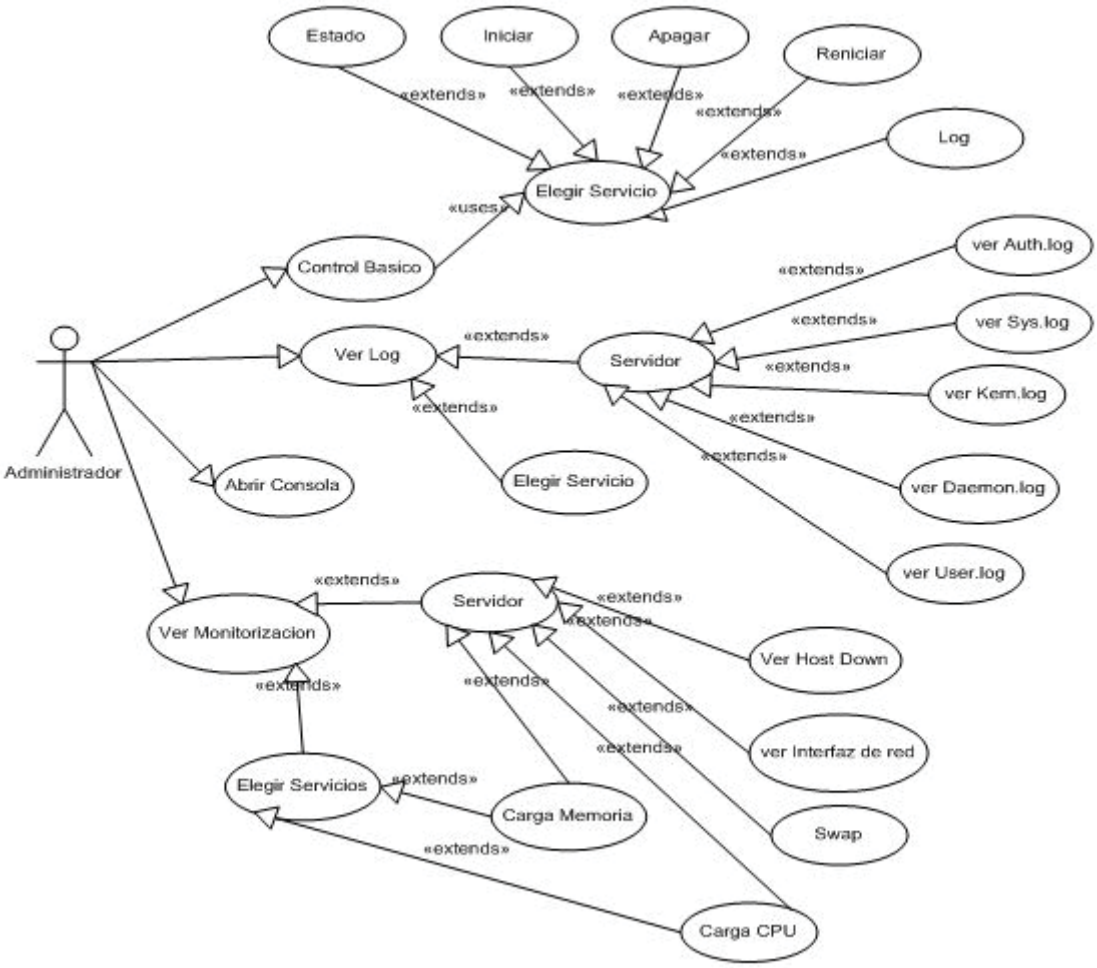
Para el caso de este sistema, se cuenta con un único actor que es llevado a cabo a su vez por una única persona, este es el Administrador, definiéndose como sigue.

Administrador: Este actor representa la persona que hace uso del sistema, realizando las tareas necesarias para que el mantenimiento y funcionamiento del servidor sea optimo, por medio del conjunto de herramientas ofrecida por el software.

2. Diagrama general de casos de uso

A continuación se presentan el diagrama de casos uso y los casos de uso UML que se han identificado en el sistema, también se ha determinado la interacción de un actor.

Ilustración 5.1 Diagrama general de casos de uso.



3. Especificaciones de Casos de Uso

• **Control básico:** El administrador actúa sobre algún servicio prestado por el servidor, logrando así ejecutar alguna de las siguientes acciones: iniciar, detener o reiniciar.

• **Ver Log:** Los servicios prestados por el servidor y este mismo llevan registro de las actividades y movimientos dentro del uso de ellos, el administrador puede requerir el ver uno de estos para así obtener una mejor perspectiva del uso que se está dando a estos.

• **Abrir consola SSH:** El administrador desea realizar una operación directa dentro del servidor utilizando una consola de comando, en la cual se puede ingresar cualquier comando que sea necesario para interactuar directamente con el servidor.

• **Ver Monitorización:** Es de suma importancia conocer las cargas llevadas por la ejecución y/o utilización de los servicios o el mismo servidor para tomar acciones cuando se detecten cargas criticas.

c. Visión del Sistema

La visión del sistema es el artefacto hito con el cual la Fase de inicio finaliza para dar comienzo a la fase de Elaboración, la visión del sistema comprende la definición del problema, el sistema como solución y las limitantes y cualidades que en él se desprenden.

Ver Anexo 1, Visión del Sistema.

5.4.2 Disciplina de Análisis y Diseño

5.4.2.1 Descripción

El objetivo principal de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis fundamentalmente consiste en obtener una visión que se preocupa de ver que hace el sistema de software a desarrollar, por tal motivo este se interesa en los requerimientos funcionales. Por otro lado, el diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en como el sistema cumple sus objetivos.

Los objetivos específicos del análisis y diseño son:

- Transformar los requerimientos al diseño del futuro sistema.
- Desarrollar una arquitectura para el sistema.
- Adaptar el diseño para que sea consistente con el entorno de implementación.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los Casos de Uso con las interacciones de las clases de análisis. Durante la

fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes.

5.4.2.2 Resumen Actividades, Sub-actividades y Tareas

a. Definición Arquitectura y Comportamiento del Sistema

Sabido es la limitada capacidad operacional de los teléfonos celulares, el hardware contenido en estos ya sea la memoria o la velocidad del procesamiento son muy inferiores a la de un computador convencional, este factor se debió tener en cuenta al momento de proponer una Arquitectura y desarrollar la arquitectura de software (*Ver Anexo 2, Arquitectura del Software*).

Se optó por un estilo arquitectónico de Cliente-Servidor de dos capas, con cliente liviano. Se debía contar con un cliente cuya funcionalidad y aporte dentro del sistema estuviese restringido a solo funciones básicas, por otro lado se cuenta con una maquina, el servidor, que supera ampliamente en procesamiento al hardware de un teléfono celular además de esto se cuenta con un enlace de comunicación entre ambos, esto es la red GPRS, por tanto era factible y lógico usar un procesamiento distribuido, convirtiendo al cliente, el teléfono celular en una forma de “interprete” ya que este solo realizaría la función de GUI en el sistema.

5.4.2.3 Resumen de Artefactos

a. Arquitectura de Software

Es una especificación de las ideas principales del diseño. Este proporciona una descripción entendible de la arquitectura del sistema software y sirve como medio de comunicación entre el arquitecto de software y otros miembros de equipo del sistema con respecto a las decisiones arquitectónicamente significativas que se han tomado en el sistema. Contiene varias vistas que muestran aspectos distintos del sistema como son: Vista de Casos de Uso, Vista Lógica, Vista de Implementación, Vista del Proceso, Vista de Implantación y Vista de Datos.

Ver Anexo 2, Arquitectura del Software.

b. Estudio de Factibilidad

Los estudios de factibilidad consideran la factibilidad técnica, económica y operacional de cada alternativa, así como si el sistema es o no apropiado dados los factores políticos y otros del contexto institucional, se busca determinar si el sistema es realmente factible. Se agrupan en:

• **Factibilidad operacional:** Si se desarrolla e implanta, ¿será utilizado el sistema?, ¿Existirá cierta resistencia al cambio por parte de los usuarios que dé como resultado una disminución de los posibles beneficios de la aplicación?

• **Factibilidad Técnica:** El análisis de factibilidad técnica evalúa si el equipo y software están disponibles (o, en el caso del software, si puede desarrollarse) y si tienen las capacidades técnicas requeridas por cada alternativa del diseño que se esté considerando.

• **Factibilidad Económica:** Los estudios de factibilidad económica incluyen análisis de costos y beneficios asociados con cada alternativa del sistema. Con análisis de costos/beneficio, todos los costos y beneficios de adquirir y operar cada sistema alternativo se identifican y se hace una comparación de ellos.

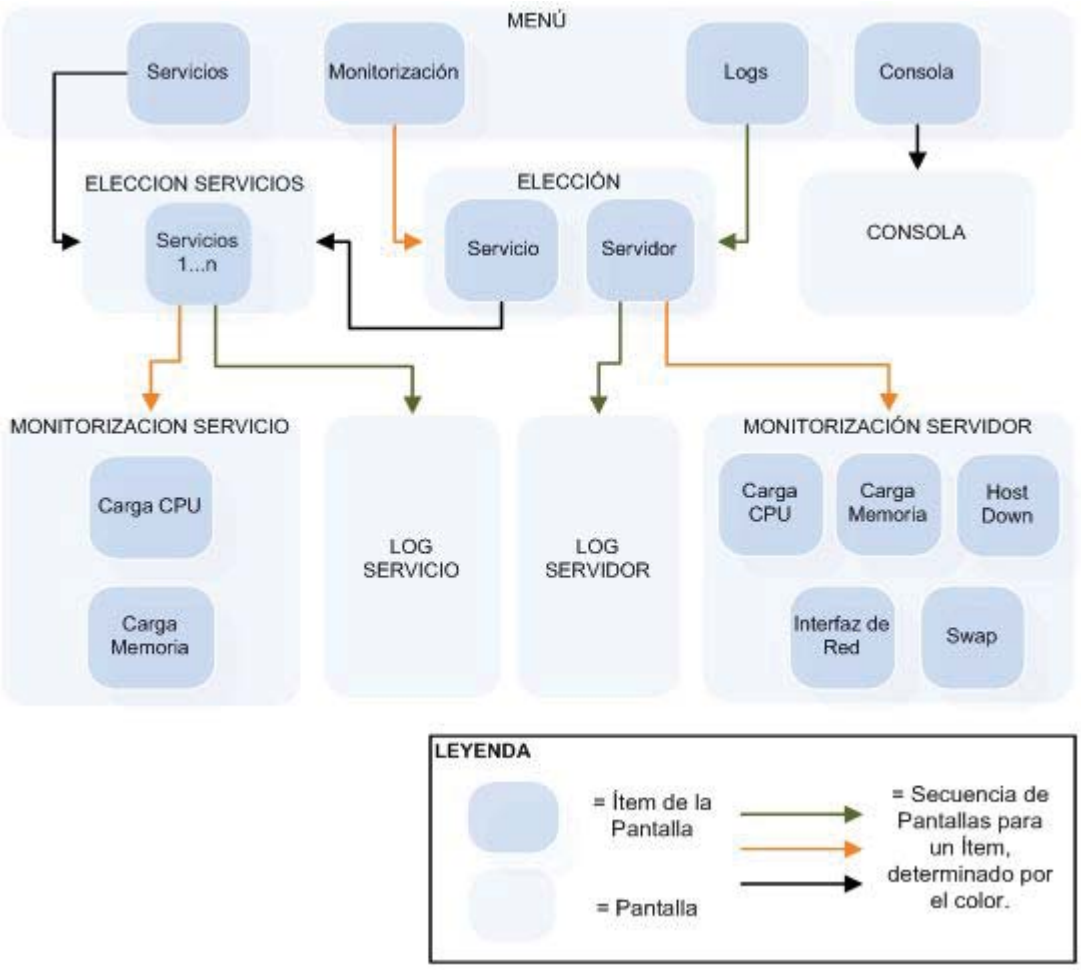
Ver Anexo 8, Estudio de Factibilidad.

c. Mapa de Navegación

Este artefacto expresa la estructura de los elementos de la interfaz de usuario del sistema, junto a los caminos de navegación principales.

Este permite al usuario una adecuada navegación en el sistema y sobre todo saber en qué punto del sistema se encuentra y hacia donde puede ir. Sin un Mapa de Navegación no se podría aprovechar al máximo un sistema. Cabe destacar que existirá solamente uno de estos artefactos en el sistema.

Ilustración 5.2 Mapa de navegación.



d. Interfaz tentativas de Usuario

Tabla 5.9 Interfaces tentativas de usuario.

e. Modelos de Diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución.

El Modelo de Diseño puede contener: los diagramas, las clases, paquetes, subsistemas, capsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.

Para representar los diagramas del Modelo de Diseño se pueden emplear diferentes diagramas de UML tales como:

- Diagramas de Clase.
- Diagramas de Colaboración.

- Diagramas de Estado.
- Diagramas de Paquetes.
- Diagramas de Secuencia.

1. Diagramas de Secuencia

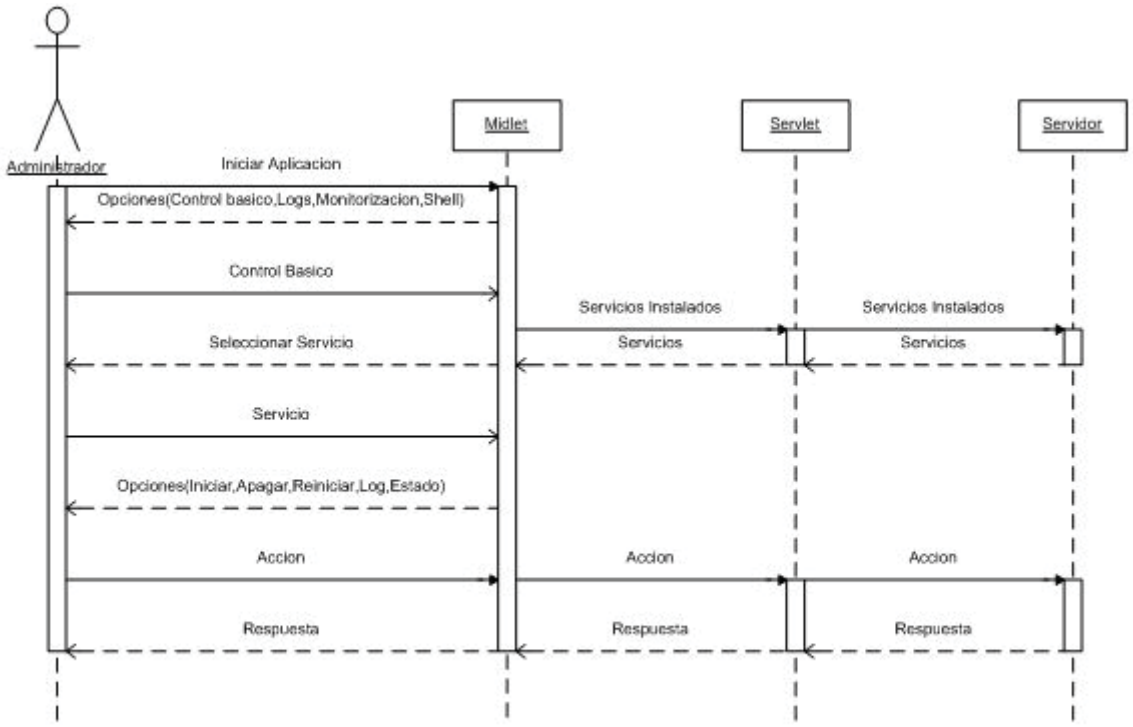
Los diagramas de secuencia exponen gráficamente los eventos que fluyen de los actores del sistema, además de ser una representación que muestra un determinado escenario de un caso de uso, de manera que se pueda conocer mejor el sistema en, estos se han construido durante la fase de análisis del ciclo de desarrollo de aplicación.

Los siguientes diagramas de secuencia muestran la interacción entre el actor identificado anteriormente (Administrador) y los componentes funcionales del sistema, haciendo más claro y sencillo el entendimiento del sistema en su conjunto, logrando una integración de los diferentes componentes que pertenecen al sistema.

› **Control básico**

El administrar elige la opción de control básico, luego se elige el servicio para posteriormente seleccionar alguna acción sobre este (iniciar, reiniciar, apagar, log, estado) para luego solicitar al Servidor que ejecute la acción sobre el servicio seleccionado.

Ilustración 5.3 Diagrama de secuencia control básico de los servicios.



› **Ver Log**

Al igual que las anteriores se presentan una serie de opciones al administrador y este elige ver Logs, luego el administrador debe elegir si ver el “Log File” de los servicios o el servidor. Para el primer caso el Midlet primero obtiene los servicios y los muestra para que posteriormente se elija por el cual consultar el Log, si se selecciona ver los Logs del servidor la aplicación muestra los registros disponibles requiriendo que el usuario elija uno.

Ilustración 5.4 Diagrama de secuencia Log de un servicio.

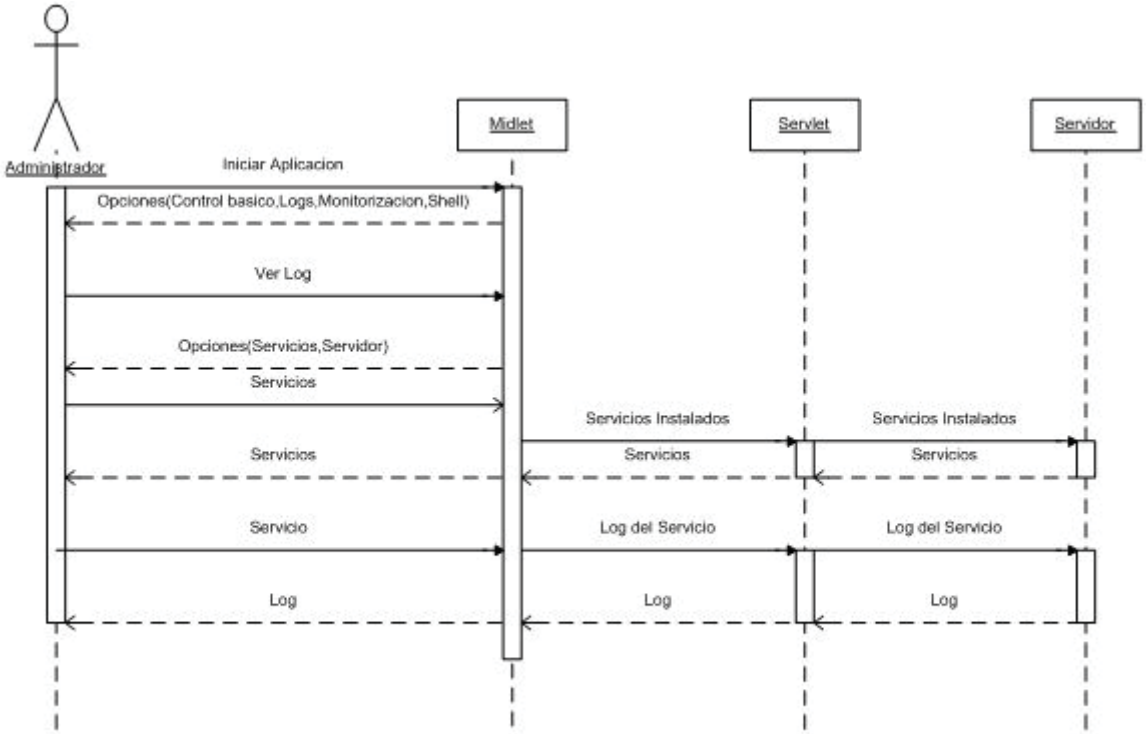
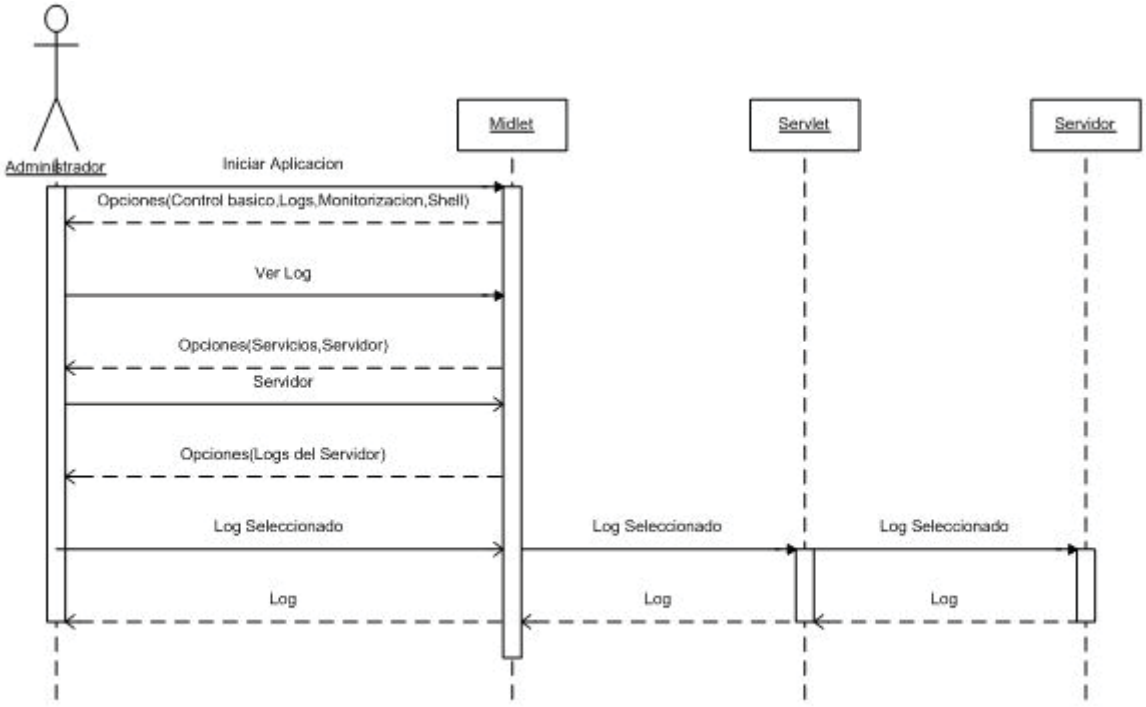


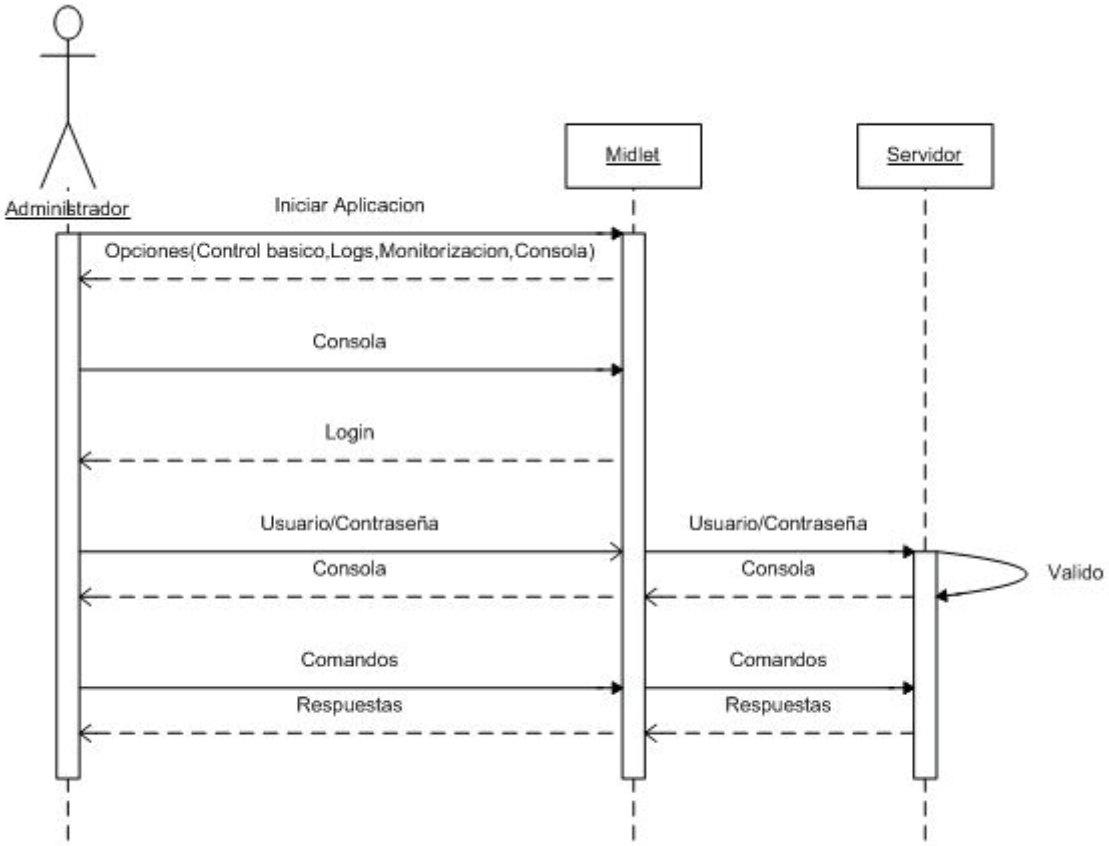
Ilustración 5.5 Diagrama de secuencia log del servidor.



› **Abrir consola (Shell Command)**

El administrador desea llamar una consola SSH para así interactuar directamente con el servidor, para que suceda esto se presentaran una serie de opciones que tiene el sistema (SSH, Log, monitorización, control básico), se elige la consola y el Midlet se comunicara con el servidor para obtener la conexión y así la respuesta requerida.

Ilustración 5.6 Diagrama de secuencia abrir consola (shell command).



› **Ver monitorización**

El administrador desea monitorear lo que está ocurriendo en su servidor o con los servicios prestados por este, para la segunda opción se obtiene previamente los servicios prestados por el servidor para luego seleccionar lo que desea monitorizar ya sea la carga del CPU o carga de la memoria.

Si se selecciona la monitorización del servidor, se presentan las opciones disponibles para llevar a cabo dicha tarea esto es; CPU, memoria, host Down, SWAP e Interfaz de Red.

Ilustración 5.7 Diagrama de secuencia monitorización servicios.

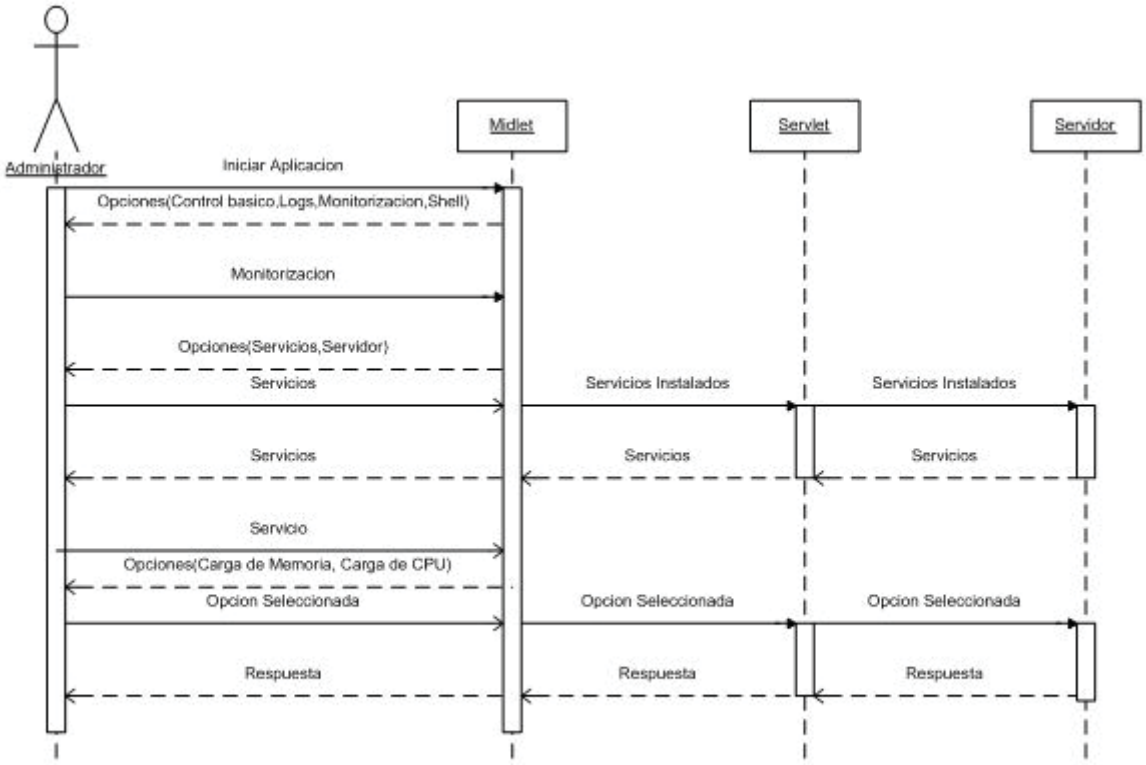
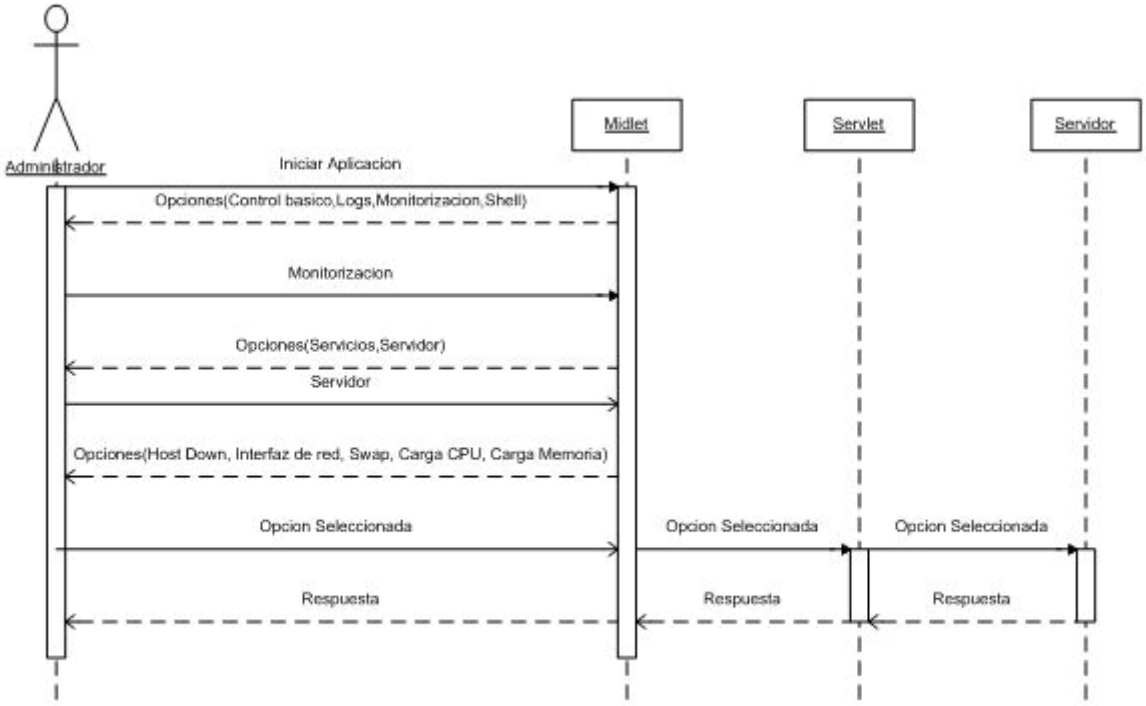


Ilustración 5.8 Diagrama de secuencia monitorización servidor.

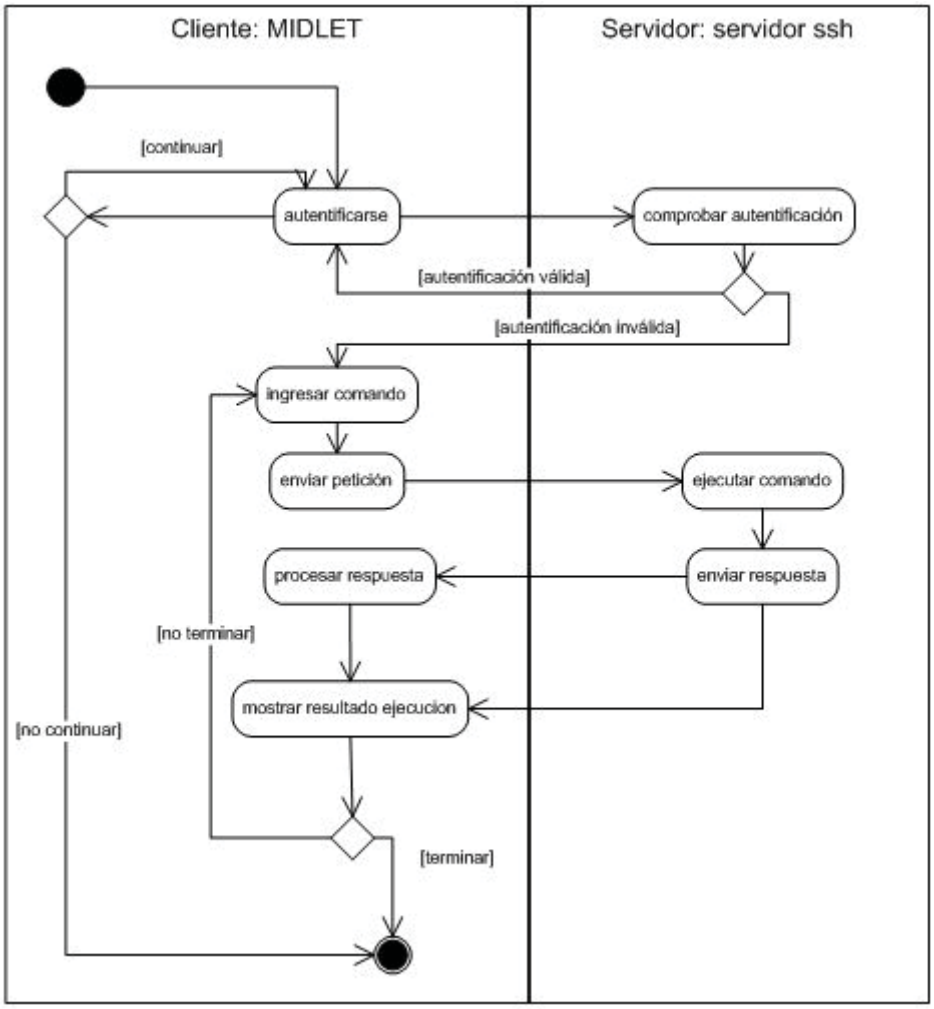


2. Diagramas de Actividades

Un diagrama de actividades es provechoso para entender el comportamiento de alto nivel de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes. Los parámetros de entrada y salida de una acción se pueden mostrar usando las relaciones de flujo que conectan la acción y un estado de flujo de objeto. Cuando una actividad termina se desencadena el paso a la siguiente actividad. A continuación de muestran los diagramas de actividades desarrollados según las necesidades del sistema presentadas:

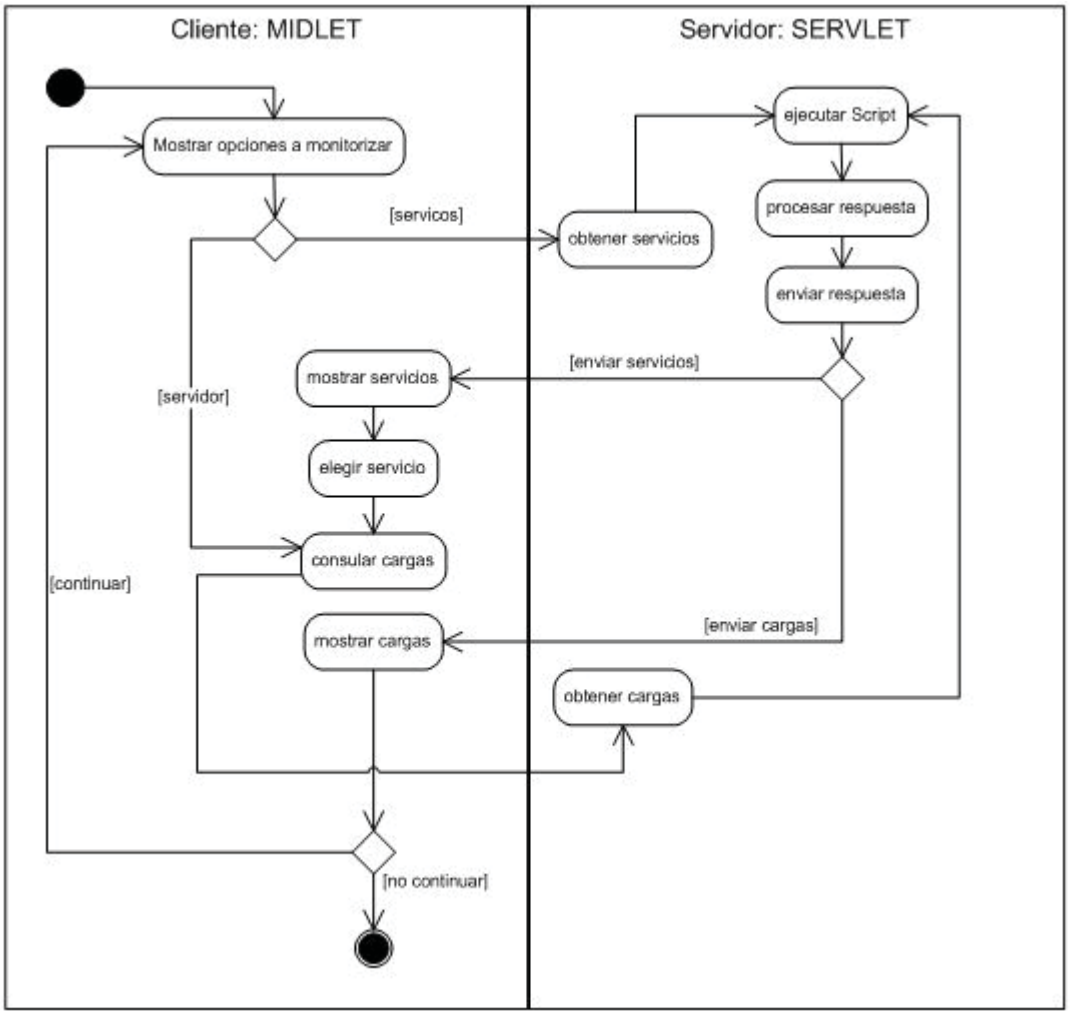
› Diagrama de actividades Consola (Shell Command)

Ilustración 5.9 Diagrama de actividades consola (Shell Command).



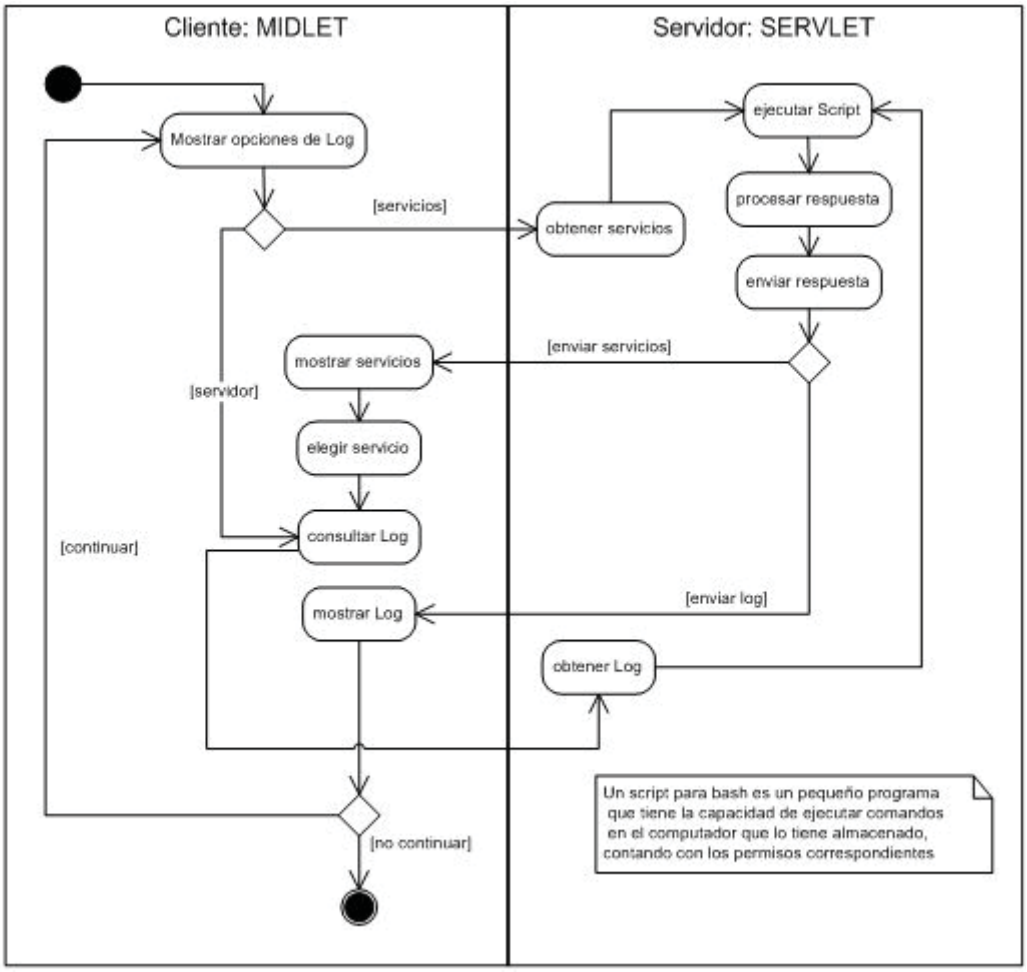
› **Diagrama de actividades Monitorización**

Ilustración 5.10 Diagrama de actividades monitorización.



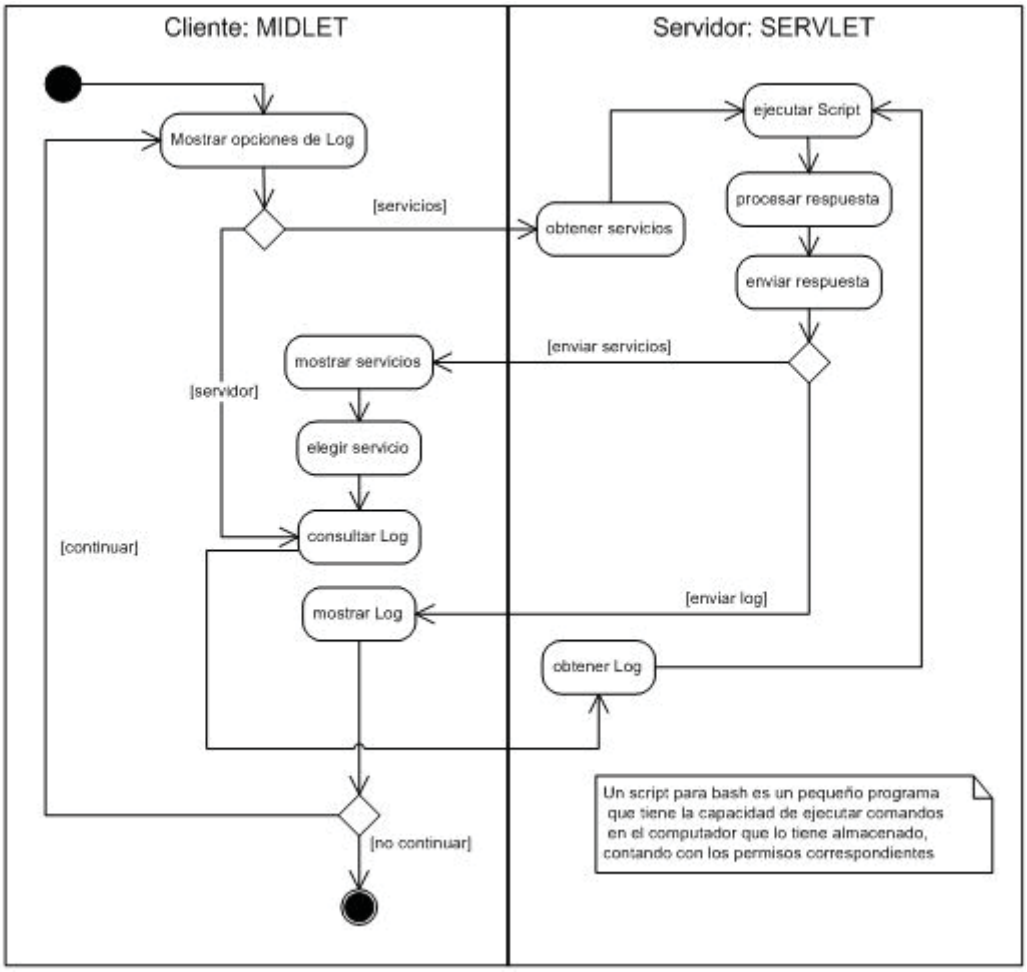
› **Diagrama de actividades Log**

Ilustración 5.11 Diagrama de actividades log.



› **Diagrama de actividades Control Básico**

Ilustración 5.12 Diagrama de actividades control básico.

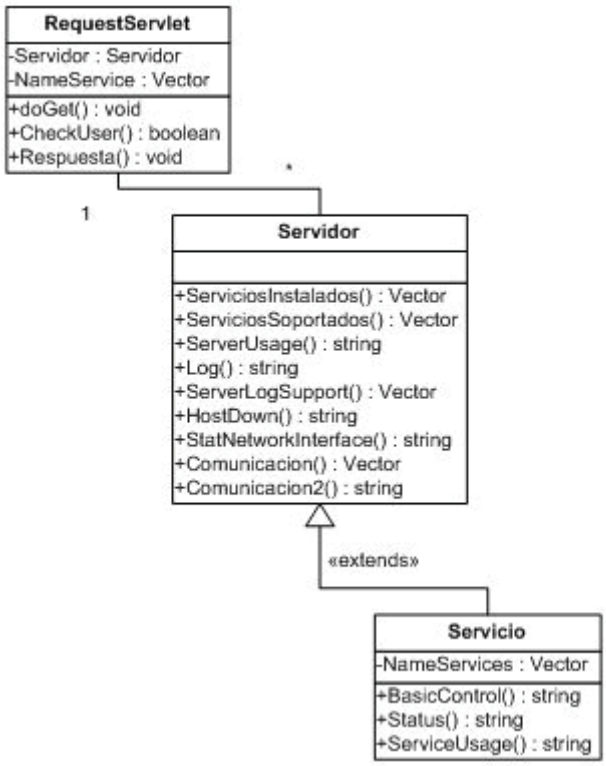


3. Diagramas de clases

- **Diagrama de clases servidor – Componente middleware**

El nodo servidor se compone principalmente de la interacción de 3 objetos: Una objeto principal llamado Request Servlet además de las clases Servidor y Servicio. Estas clases son necesarias y fundamentales para el correcto funcionamiento de las peticiones provenientes del Midlet hacia los dos módulos presentes en la aplicación, a su vez también es precisa la existencia de los script para Bash que se han creado, para efecto del procesamiento de los Logs, consultas de Monitorización tanto del servidor como de los servicios para así lograr una correcta consulta de lo requerido.

Ilustración 5.13 Diagrama de clases componente middleware.



- **Diagrama de clases dispositivo móvil – Componente Midlet**

Ilustración 5.14.a Diagrama de clases componente Midlet.

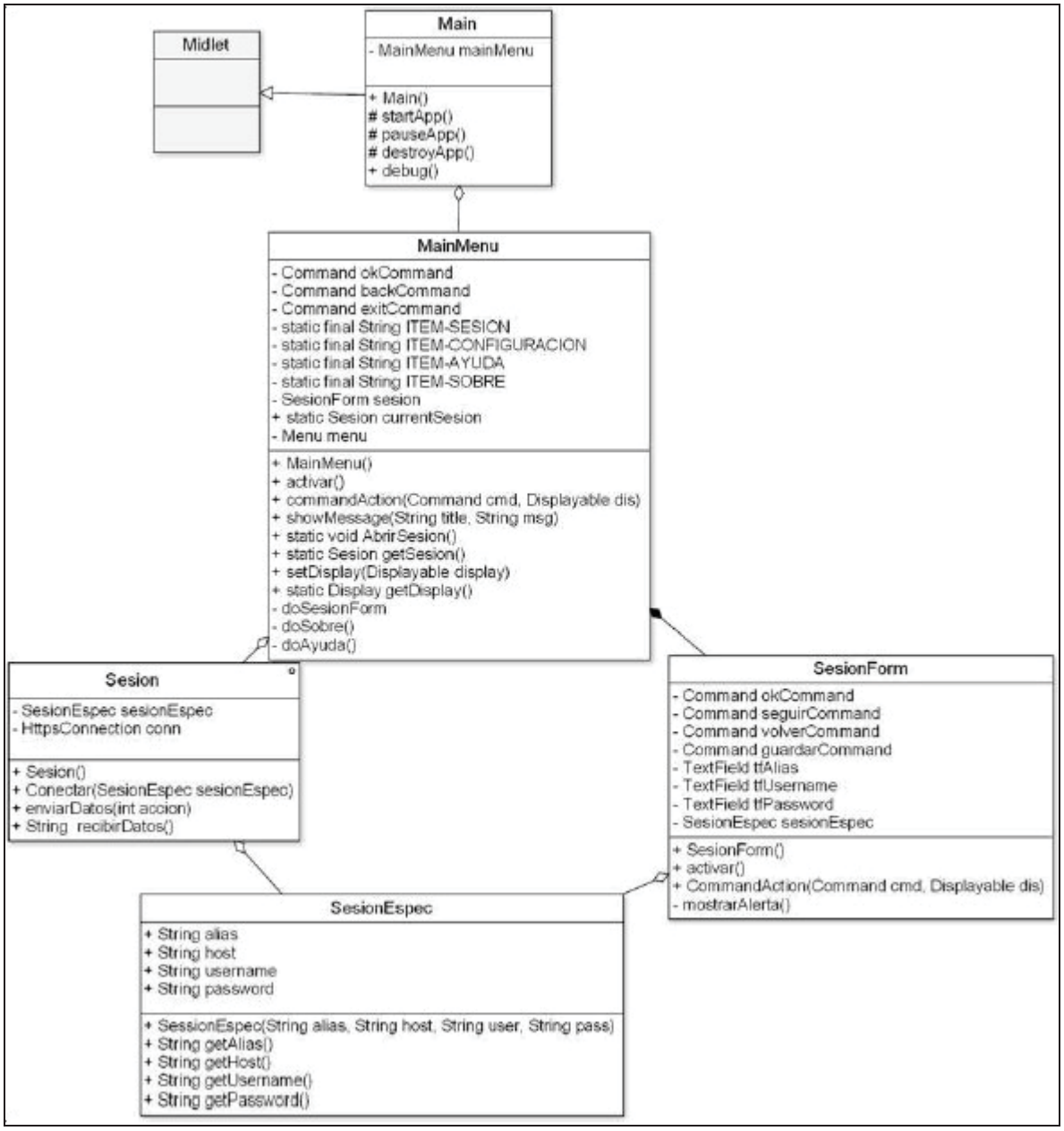
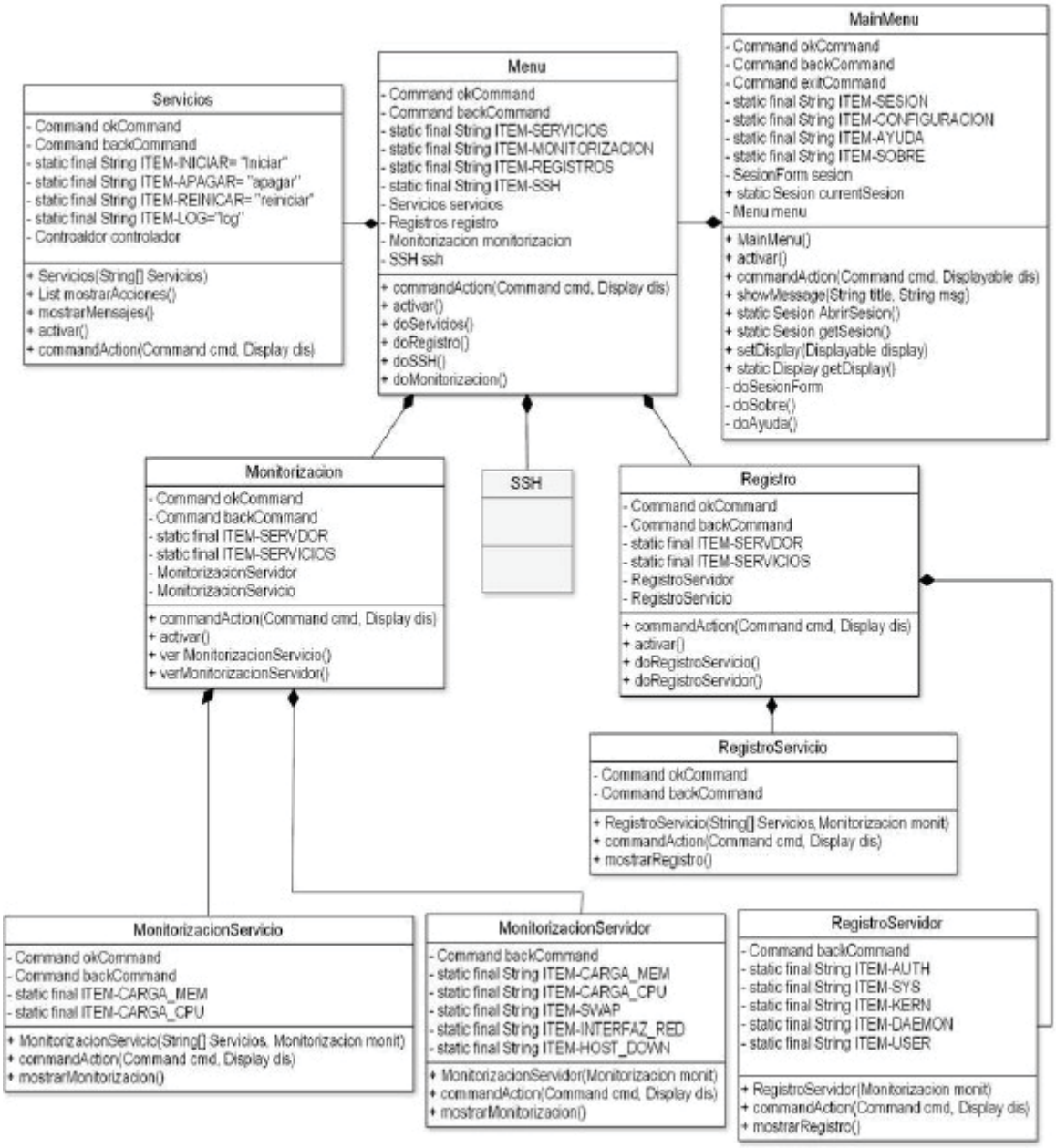


Ilustración 5.14.b Diagrama de clases componente Midlet.



5.4.3 Disciplina Implementación

5.4.3.1 Descripción

El objetivo principal de esta disciplina es convertir los elementos del diseño en elementos de implementación, dichos elementos son códigos fuentes, ejecutables, binarios, entre otros. Otra parte de esta disciplina son las pruebas de unidad, las cuales se limitan a los componentes de software implementados. De esta disciplina se obtiene un sistema

ejecutable estable, constituido de los resultados producidos por los programadores individuales.

- Los objetivos específicos de la implementación son:

- Planificar qué subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.

- Cada desarrollador decide en qué orden implementa los elementos del subsistema.

- Notificar los errores de diseño, si se encuentran.

- Probar los subsistemas individualmente.

- Integrar el sistema siguiendo el plan.

La estructura de todos los elementos implementados forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento sólo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

5.4.3.2 Resumen Actividades, Sub-actividades y Tareas

La implementación del sistema se llevo acabo de forma paralela y simultanea por los alumnos participantes de esta memoria, como ya es sabido el sistema comprende dos componentes esenciales, estos son el Midlet y el Middleware (*Ver Anexo 2, Arquitectura del Software*), estos dos componentes fueron desarrollados por los alumnos asignándose a cada una la labor en base a los conocimientos técnicos adquiridos en otros proyectos. Previo al desarrollo se estableció la forma de cómo estos componentes se comunicarían e interactuarían, una vez llegado a conceso, el trabajo se abordo de forma separada para cada componente estableciendo eso sí, momentos o hitos en los cuales se revisaba el avance de

cada componente, a modo de verificar que se seguían los acuerdos de integración, sirviendo a su vez de control, entendimiento y conocimiento del trabajo del otro.

5.4.4 Disciplina Pruebas

5.4.4.1 Descripción

El principal objetivo de esta disciplina es de evaluar la calidad del producto que se está desarrollando a través de las diferentes fases por las cuales este pasa, mediante la aplicación de pruebas concretas para validar que las suposiciones hechas en el diseño y los requerimientos se estén cumpliendo satisfactoriamente, esto quiere decir que se verifica que el producto funcione como se diseñó y que los requerimientos son satisfechos cabalmente. Esta disciplina brinda soporte para encontrar y documentar (y solucionar) defectos en la calidad del sistema a las otras disciplinas. Esta disciplina debe estar presente en todo el ciclo de vida del desarrollo del sistema para ir refinándolo y no al final del mismo.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos específicos son:

- Encontrar y documentar defectos en la calidad del software.
- Notificar la calidad percibida del software.
- Proveer un medio de validación para las suposiciones hechas en el diseño y especificaciones de requerimientos por medio de demostraciones concretas.
- Validar las funciones del producto de software según lo diseñado.
- Validar que los requerimientos fueron implementados apropiadamente.

El desarrollo de esta disciplina consistirá en planificar que es lo que hay que probar, diseñar cómo se va a llevar a cabo la prueba, implementar lo necesario para llevarlas a cabo, ejecutarlas en los niveles necesarios y obtener los resultados, de forma que la información obtenida sirva para ir refinando el producto a desarrollar.

El papel de las pruebas no es asegurar la calidad, pero sí evaluarla, y proporcionar una realimentación a tiempo, de forma que los aspectos de calidad puedan resolverse de manera efectiva en tiempo y costo.

Los principales aspectos a ser evaluados en un producto software son la Funcionalidad (hace lo que debe), la Fiabilidad (resistente a fallos), y el Rendimiento (lleva a cabo su trabajo de manera efectiva). Las pruebas pueden hacerse a diferentes niveles dependiendo del objetivo de los mismos, entre algunos tenemos: Pruebas de unidad (se prueban las unidades mínimas por separado, y normalmente se hace durante la implementación misma), de integración (varias unidades juntas), de sistema (sobre la aplicación o sistema completo) y de aceptación (realizado sobre el sistema global por los usuarios o terceros).

5.4.4.2 Resumen Actividades, Sub-actividades y Tareas

a. Definición Criterios, Enfoque y Detalles de las Pruebas

Como se definía en el punto anterior el objetivo principal que se busca el ejecutar pruebas es evaluar la calidad el producto construido a modo de verificar posibles defectos en el diseño, errores de programación (bugs) o defectos en la instalación como a su vez si los requerimientos previamente establecidos son cumplidos a cabalidad por el sistema contando con la aprobación del cliente y los desarrolladores.

Los criterios de evaluación son aplicados para todos los casos de prueba y variarán de la naturaleza de lo que se está probando. Los criterios fueron propuestos por los alumnos sumados a ello también criterios aplicados a cualquier tipo de proyectos informático.

5.4.4.3 Resumen de Artefactos

a. Plan de Pruebas

Es la colección formada por los casos de prueba y procedimientos de prueba. Este artefacto incluye el propósito de las pruebas, qué elemento se va a probar, las herramientas a utilizar y con qué recursos, así como el documento que va hacer entregado. Al tener el resultado de las pruebas se puede comparar lo obtenido con lo esperado.

En este artefacto también se reflejan las características de hardware y software que serán empleados para realizar el conjunto de las pruebas al sistema.

Ver Anexo 4, Plan de pruebas

5.4.5 Disciplina Implantación

5.4.5.1 Descripción

Esta disciplina tiene como objetivo distribuir e instalar con éxito el sistema elaborado por el equipo de desarrollo y asegurar la disponibilidad del producto para los usuarios finales.

Sus objetivos específicos son:

- Probar el producto en su entorno de ejecución final.
- Proveer asistencia y ayuda a los usuarios.
- Migrar el software existente.
- Instalar el software.
- Formar a los usuarios y al cuerpo encargado de distribuir el sistema.

Se lleva a cabo con mayor intensidad en la fase de transición, debido a que su propósito es asegurar una aprobación y adaptación sin que existan dificultades del software por parte del usuario. Esta disciplina debe iniciar en fases anteriores, para preparar el camino, sobre todo con actividades relacionadas a la planificación, pero también con la elaboración del manual de usuario y tutoriales. Debido al extenso nivel de aplicaciones que se pueden obtener y las diversas características de los productos necesarios para esta disciplina, se pueden obtener grandes variaciones dependiendo del tipo de sistema a desarrollar. El objeto clave es una distribución del producto.

Dado las diversas especificaciones de implantación que se pueden dar para cada proyecto, se le otorga en esta metodología pocos detalles a esta fase.

Aunque el sistema esté bien diseñado y desarrollado correctamente su éxito dependerá de su implantación y ejecución por lo que es importante capacitar al usuario con respecto a su uso y mantenimiento.

5.4.5.2 Resumen Actividades, Sub-actividades y Tareas

a. Desarrollo Plan de Implantación e Instalador

El plan de implantación corresponde al conjunto de tareas necesarias para poner en actividad el sistema, como se trata de un sistema distribuido se debió abordar el plan dividiéndolo en 2 partes, el cliente móvil ejecutado en o los teléfonos celulares de los usuarios finales y por supuesto el software a ser instalado en el Servidor, para el manejo de la interacción entre ambos, cada uno demandando requerimientos de hardware y software diferentes.

Para instalación del cliente en el teléfono celular son necesarios dos archivos, archivo .jar conteniendo el bytecode del programa y un archivo .jad que describe los contenidos del archivo .jar. en la actualidad el archivo .jad no es necesario incluirlo aparte ya que es posible ser agregado dentro el mismo archivo .jar, la instalación del cliente se realizara ejecutando el archivo .jar, se debe aclarar que dicha instalación no requiera intervención del cliente ya que no existe interfaces de instalación, esta es gestionada por entero por el sistema operativo del teléfono celular donde se cargue, una vez finalizada la instalación el archivo .jar que correspondía al instalador cambia de estado y pasar a convertirse en el ejecutable.

En lo que respecta al instalador del sistema en el servidor, es necesario ejecutar el pequeño programa desarrollado en un Bash script, logrando automatización de la instalación en el sistema logrando que el usuario no tenga la mayor intervención en la configuración. Además de la integración Bash se incorporo un programa desarrollado en el lenguaje de programación Perl para la intervención de archivos propios del sistema operativo tal como el archivo de configuración sudo.

b. Desarrollo de manuales

Bajo esta disciplina entre los artefactos a desarrollar se encuentran los relacionados con los manuales; manual de instalación e manual de usuario, en donde se describe como realizar una eficiente instalación y uso del sistema respectivamente, ambos fueron desarrollados utilizando lenguaje técnico pero de manera tal que los que usaran dichos documentos pudiesen entender sin mayores complicaciones, no obstante, se asumió que los usuarios finales poseen conocimiento mínimos requeridos para realizar una eficaz instalación en lo que respecta el buen manejo de la línea de comandos necesario para la instalación de una parte de este sistema, correspondiente al Middleware y todas las dependencias eso por parte del Servidor, la otra corresponde a la implantación del cliente móvil en el teléfono celular, no obstante, dicha facción no requiere más detalle puesto que la carga en el dispositivo y posterior instalación es guiada de forma ágil por los sistemas operativos de dichos teléfonos, no presentando mayores contratiempos.

5.4.5.3 Resumen de Artefactos

a. Manual de Usuario

Este artefacto provee una ayuda a las personas que manipularán directamente el producto, acerca del uso que le debe dar al sistema. Dicho artefacto debe ser discutido y aprobado por el cliente.

Elaborar el manual de usuario durante las primeras iteraciones del proyecto permitirá al equipo de probadores conocer el sistema antes de que comiencen las pruebas, adicionalmente provee los mecanismos básicos para elaborar los planes de pruebas y los casos de pruebas, y permite la elaboración de sistemas automatizados para las pruebas.

Según el tipo de sistema se define el comienzo del desarrollo del Manual de Usuario. Sistemas con interfaces complejas o con mucha interacción requerirán versiones tempranas del manual de usuario así como de prototipos de interfaces. Sistemas con poca interacción probablemente no requieran que la documentación del usuario se elabore muy temprano.

Ver Anexo 6, Manual de Usuario

b. Manual de Instalación

El manual de instalación es un artefacto que refleja los lineamientos que hay que seguir para instalar el sistema. Contiene información sobre la infraestructura de instalación e instrucciones para la instalación y actualización del software

Ver Anexo 7, Manual de Instalación

5.4.6 Disciplina Gestión de Proyectos

5.4.6.1 Descripción

El objetivo de la gestión del proyecto es conseguir alcanzar las metas propuestas con el desarrollo del sistema, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requerimientos de los clientes y usuarios.

Los objetivos específicos de este flujo de trabajo son:

- Proveer un marco de trabajo para la gestión de proyectos.
- Proveer guías prácticas para realizar planeación, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

Para conseguir estos objetivos el flujo de trabajo de esta metodología se centra en tres aspectos:

- Planificar un proyecto iterativo y cada iteración en particular.
- Administrar el riesgo.
- Monitorear el progreso del proyecto a través de métricas.

Monitorear un proyecto es importante para mantenerlo bajo control. Esto permite medir la magnitud en la que el proyecto se ajusta a los planes, la calidad requerida y los requerimientos. También es necesario para planificar de forma precisa y ver cuál es el comportamiento del proyecto frente a cambios.

La administración del riesgo consiste en ocuparse de las incógnitas de un proyecto, las cuestiones que puede afectar el desarrollo del proyecto y llevarlo al fracaso. En concreto hay que identificar los riesgos, típicamente en la fase de inicio, y hacerles frente, mitigar, transferirlos o asumirlos. En este último caso habrá que tratar de mitigar el riesgo y definir un plan de contingencia por si el riesgo se convierte en un problema real. En definitiva la administración del riesgo consistirá en gestionar una lista de riesgos.

5.4.6.2 Resumen Actividades, Sub-actividades y Tareas

a. Identificación, evaluación y análisis de Riesgos

Los métodos para la identificación, análisis y evaluación de riesgos son una herramienta muy valiosa para abordar con decisión su detección, causa y consecuencias que puedan acarrear, con la finalidad de eliminar o atenuar los propios riesgos así como limitar sus consecuencias, en el caso de no poder eliminarlos.

La identificación del riesgo es un intento sistemático para especificar las amenazas al plan del proyecto (estimaciones, planificación temporal, carga de recursos, etc.)

Existen dos tipos diferenciados de riesgos: genéricos y específicos del producto. Los riesgos genéricos son una amenaza potencial para todos los proyectos de software. Los específicos de producto sólo los pueden identificar los que tienen una clara visión de la tecnología y el entorno específico del proyecto en cuestión. Los alumnos participantes de esta memoria, durante el transcurso de sus años de estudio de la carrera en cuestión desarrollaron proyectos para diversos ramos de la misma, en algunos de abordaron tópicos similares a los que este sistema aborda; Tecnología Móviles, comunicación inalámbrica, arquitecturas cliente/servidor, sistemas distribuidos, etc. Creando en los mismos experiencia facilitando la identificación y posterior evaluación.

Básicamente, existen dos tipos de métodos para la realización de análisis de riesgos, si atendemos a los aspectos de cuantificación o evaluación:

Métodos cualitativos: se caracterizan por no recurrir a cálculos numéricos. Pueden ser métodos comparativos y métodos generalizados.

Métodos semicualitativos: los hay que introducen una valoración cuantitativa respecto a las frecuencias de ocurrencia de un determinado suceso y se denominan métodos para la determinación de frecuencias, o bien se caracterizan por recurrir a una clasificación de las áreas de una instalación en base a una serie de índices que cuantifican daños: índices de riesgo.

El método usado para la evaluación de riesgos corresponde al uso de Métodos Cualitativos, bajo este método la técnica usada corresponde al uso de métodos comparativos como generalizados.

b. Conducción y Evaluación del Proceso de Desarrollo

La evaluación del proceso de desarrollo fue realizada por los dos alumnos participante de esta memoria, el trabajo si bien en ocasiones fue realizado en paralelo a modo de agilizar tiempos, no proseguía sin contar con la aprobación de ambos, sin embargo, en ocasiones y por desconocimiento en algún tema, no se llegó a una completo consenso, para estas situaciones se tomaba la determinación de recurrir a la ayuda de los profesores guía y co-referente, en busca de los lineamientos y estrategias para enfrentar temas sensibles a discusión y no mutuo acuerdo.

5.4.6.3 Resumen de Artefactos

a. Plan de Gestión de Riesgos

Artefacto en el cual se describe los posibles riesgos de recursos, técnicos, o del negocio implicados en el proyecto, y formula un plan para abordar los posibles riesgos, con medidas de mitigación y correctivas para afrontar cada uno de ellos. Sirve de punto principal para la programar las actividades que deben realizarse y con base en este documento se deben plantear las iteraciones a ser realizadas.

Un Plan de Gestión del Riesgo debe ser documentado a comienzos del proyecto, durante la fase de inicio. El plan es emprendido ante la fase de elaboración para asegurar que ninguno los riesgos identificados sean direccionados durante la misma fase de elaboración.

Apenas el plan haya sido documentado, el proceso de prevención de riesgos estará ocupado para monitorear y controlar la probabilidad y el impacto de los riesgos sobre el proyecto.

Ver Anexo 3, Plan de Gestión de Riesgo

5.4.7 Disciplina Gestión de Ambiente

5.4.7.1 Descripción

La finalidad de esta disciplina es dar soporte al proyecto con los procesos, métodos y herramientas correctas. Ofrece una descripción de las herramientas que se van a necesitar para el apropiado desarrollo del software, que contendrá las herramientas de desarrollo y del proceso, plantillas, documentos, convenciones a seguir, y cualquier otro elemento necesario para llevar adelante con éxito el desarrollo del proyecto.

En concreto los objetivos específicos de esta disciplina de trabajo incluyen:

- Seleccionar y adquirir herramientas.
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configurar el proceso.
- Mejorar el proceso de desarrollo.
- Proveer los servicios técnicos necesarios.

5.4.7.2 Resumen Actividades, Sub-actividades y Tareas

a. Preparación de Ambiente para Proyecto

La infraestructura de trabajo comprendió 2 lugares, el primer lugar correspondió a las dependencias de la Pontificia Universidad Católica de Valparaíso en las salas de proyecto específicamente, esta instalación cuenta con un ambiente propicio que facilita el trabajo sobre todo al momento de desarrollar, se cuenta con una buena infraestructura de hardware, un lugar espacioso, climatizado, grato y cómodo. El lugar es compartido por otros alumnos

que asisten a esas dependencias con los mismos fines, creando un ambiente de motivación y de compartición de conocimientos que hacen del desarrollo una tarea menos tediosa. El Segundo lugar corresponde a los hogares de los alumnos participantes en esta memoria.

b. Selección y Adquisición de Herramientas

EL sistema fue desarrollado casi por completo utilizando la plataforma Java tanto en la programación del Midlet como el Middleware (Servlet) utilizando las ediciones J2ME y J2EE respectivamente, para lograr un eficiente uso de esta plataforma se evaluaron una serie de herramientas de programación IDE's (Integrated Development Environment). Entro estos IDE's fueron probados: Eclipse, Jdeveloper y Netbeans todas estas herramientas comparten similares características que facilitan al programador su labor, no obstante, a su vez cada una posee propiedades que la identifican como única, es por eso que la elección recayó en Netbeans entre las características que diferenciaron con sus pares se encuentran;

• La integración de múltiples herramientas y protocolos proporciona razones para la migración

• Facilidad de uso durante todo el ciclo de desarrollo

• Maneja la complejidad de la arquitectura orientada al servicio (SOA)

• Soporte al modelado que mejora la productividad del desarrollador

• Promueve las mejores prácticas para la productividad del grupo

• Creado para acelerar el desarrollo

• Multiplataforma

• Contiene Frameworks basado en asistentes (diálogos paso a paso) además de contar con bastante documentación es su website.

• Es gratuito y de código abierto para desarrolladores de software.

Además de la utilización de Netbeans, fue necesario la utilización las herramientas propias que otorga el sistema operativo Linux, como editores de texto, herramientas de evaluación, compiladores, línea de comandos. Esto fue preciso dado a la integración de las consultas provenientes del cliente móvil afectan directamente el estado del servidor y se comunican directamente con el SO haciendo necesario mantener la lógica del ambiente Linux.

c. Configuración y Verificación de Herramientas

La verificación de las herramientas esta dado por la propia experiencia de los desarrolladores de este sistema, además de la conocida integración con los diferentes tópicos que afectaron el desarrollo de la misma. A su vez la verificación se dio por la misma experiencia del profesor guía de este sistema, otorgando así una seguridad de que la elección de las herramientas del desarrollo que interactuaron fue la correcta.

Respecto a la configuración de las herramientas, esta se dio con la ayuda de la documentación oficial que otorgan estas, además de la lectura constante de foros incluidos en los propios sitios web de las ya mencionadas. En diferentes etapas del desarrollo se dieron problemas de configuración que se fueron solucionando, de la forma antes mencionada, pero en si no atraso ni afecto negativamente al desarrollo del sistema.

5.4.7.3 Resumen de Artefactos

a. Infraestructura de Desarrollo

- **Hardware**

Para el desarrollo del sistema, los recursos Hardware utilizados son los siguientes:

› Notebook Packard Bell, Modelo EasyNote mx456+, Intel Centrino Duo 1.73mhz, 1 GB (DDR SDRAM), 160 GB Hard Disk, SO: Windows Vista Home Edition.

› PC Intel Pentium 4, 2277 MHz, 512 MB (PC3200 DDR SDRAM), 160 GB Hard Disk, SO: Debian Gnu/Linux version 4.0 ETCH

› Teléfono Celular Sony Ericsson, Modelo K310a.

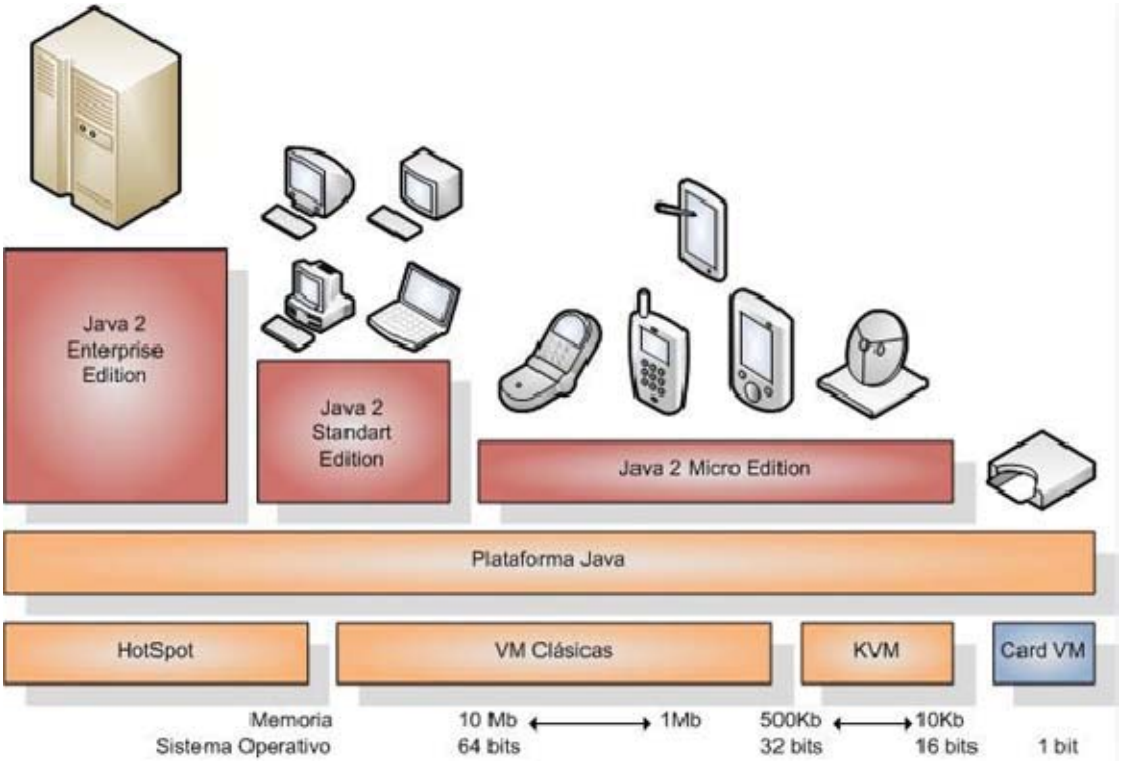
- **Software**

El desarrollo del sistema fue completado utilizando Java como plataforma de desarrollo, Java cumplía con una serie de requisitos necesarios para completar de forma exitosa el mismo. La capacidad de ser multiplataforma, Orientado a objeto además de poseer una gran cantidad de documentación en Internet y poseer ediciones especializadas para los diferentes intereses, para el caso particular de este sistema, la utilización de las ediciones Micro y Enterprise, lograron dar soporte a los requerimientos de Software para el desarrollo entorno a la cliente móvil como el servidor respectivamente.

Java 2 Micro Edition, está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes. Esta edición tiene unos componentes básicos que la diferencian de las otras versiones, como el uso de una máquina virtual denominada KVM (Kilo Virtual Machine, debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) en vez del uso de la JVM clásica, inclusión de un pequeño y rápido recolector de basura.

Java 2, Enterprise Edition, está enfocado para desarrollar y ejecutar software de aplicaciones con arquitectura de N niveles distribuidos, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc. y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, Servlets, Portales (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web.

Ilustración 5.15 Java 2.


Java 2
Enterprise
Edition
Java 2
Standart
Edition
Java 2 Micro Edition
Plataforma Java
HotSpot
VM Clásicas
KVM
Card VM
Memoria
Sistema Operativo
10 Mb
64 bits
1Mb
500Kb
32 bits
10Kb
16 bits
1 bit

Capítulo 6
Trabajos a Futuro

6.1 Alcance

Como todo software o sistema de información necesita ser revisado posterior implantación en busca de errores o para realizar mejoras que los clientes estimen convenientes, dichas mejoras o actualizaciones permiten lograr en mejor desempeño en el trabajo habitual del sistema logrando aun más gratificación en el o los clientes que de él hacen uso.

6.2 Mejoras

6.2.1 Respecto al cliente

6.2.1.1 Interacción y Diseño

El cliente móvil en su interfaz de usuario, fue desarrollado utilizando la API de alto nivel, esta API provee los elementos de interfaz comúnmente utilizados para el desarrollo de aplicaciones, los elementos como Forms o List permiten un rápido entendimiento en el cliente al momento de realizar alguna interacción sobre estos, no obstante el uso de la API de alto nivel tiene un limitante, la ubicación de los elementos dentro de la pantalla como así los colores, los eventos, y el mismo diseño son llevados a cabo por la misma API, no permitiendo al desarrollador crear aplicaciones más ricas en diseño, sin embargo J2ME cuenta con una API de bajo nivel desarrollada para la creación de juegos y aplicaciones interactivas, en realidad es un conjunto de clases que se basan en la clase abstracta Canvas.

Entre las mejoras sobre ese tópico, sería bueno reducir el número de pantallas cambiándolo por nuevos elementos, es decir, contar con pantallas en donde se despliegue

más información y más elementos de interacción, sumarle a ello, el despliegue de información en colores dependiendo de la naturaleza de la misma.

6.2.1.2 Integración y Comunicación

En la actualidad el correcto funcionamiento del cliente móvil esta dado en gran medida por la intensidad de la señal de recepción del teléfono celular por las antenas emisoras, si la calidad de la señal es deficiente, el envío de peticiones o solicitudes tarda mucho más que si se contara una calidad optima, el resultado de ejecutar alguna interacción en peores condiciones provoca la pérdida de paquetes resultando en la inutilización del cliente móvil.

No es posible llevar a cabo una mejora de la intensidad de la señal cuando se requiera, claro está, son limitantes fuera del alcance de este sistema, pero si el resguardo cuando se enfrente a la misma o se esté e presencia de ella, esto es implementando algún sistema de chuequeo de entrega de paquetes incluyendo códigos de errores, códigos de aceptación y verificación, para que ambas partes, teléfono celular y Servidor estén esterados de que la comunicación es efectiva y segura.

Siguiendo en este punto, la herramienta de administración de Servidores Web Nagios, proveía de la utilidad de aviso ante algún evento imprevisto que ocurriese en el Servidor, dicha utilidad se traducía en el envío de email a cuentas de correo electrónico previamente configuradas. Interesante seria poder integrar algún sistema de alerta ante algún imprevisto que afectase el normal funcionamiento del Servidor, tal vez, como Nagios con un sistema de email o aun mejor, algún procedimiento que entregase mediante SMS en el mismo teléfono celular del Administrador.

6.2.1.3 Desempeño y portabilidad

Como ya es sabido, la aplicación fue enteramente construida con la plataforma J2ME. Las aplicaciones desarrolladas bajo ese lenguaje suponen un soporte por todos los dispositivos móviles que tengan MIDP/CLDC integrado, no obstante en la realidad no es así. La aplicación fue probada en 2 modelos de teléfonos celulares estos fueron; Sony Ericsson y Nokia con similar desempeño, sin embargo, al momento de cargar en un modelo de LG,

simplemente no funciono, por ende, se debe tener presente al momento de desarrollar el o los modelos en el cual la aplicación se va a implantar, ya que conociendo el modelo es posible obtener la características técnicas y operacionales o de rendimiento con respecto a la ejecución de aplicaciones en J2ME, de esta manera de mejorar considerablemente la portabilidad.

La aplicación fue pensado para ser usadas en teléfonos celulares, sin embargo, y como J2ME recalca, las aplicaciones escritas en este lenguaje pude ser soportado mientras tengan Java y MIDP/CLDC requerido, en este grupo de dispositivos podemos incluir aparte de los teléfonos celulares los populares Blackberrys. Los Blackberrys superan ampliamente a los pequeños celulares, pueden lograr establecer conexión a internet mediante WIFI, la maquina virtual de java interna en muchísimas más poderosa ya que cuenta un procesador y memoria ampliamente superiores de los dispositivos móviles, además de un agradable diseño y GUI que permiten desplegar aplicaciones similares si se tratase de un PC. Interesante seria lograr implantar la aplicación en estos dispositivos, ya que se obtendría un mejor desempeño sobre todo al momento de establecer conexión con el Servidor, no se utilizaría GPRS ó EDGE medios con tasas de transferencia menores a redes WIFI. Para efectuar dicha implantación se necesitará convertir la aplicación con extensión .jar en una nueva con extensión .cod para ser soportado por los Blackberrys. Existe bastante documentación al respecto de cómo hacerlo.

RIM, compañía fabricante y distribuidora de Blackberrys, en su página provee a los desarrolladores de su API para la creación de aplicaciones para estos dispositivos. JDE el lenguaje de programación de los mismos, muy similar en sintaxis y semántica a J2ME. Si se logra coger la aplicación y desarrollarla para Blackberrys se abarcaría un nuevo segmento de dispositivos inalámbricos que no se habían pensado, con todos los beneficios que el mismo dispositivo entrega.

Otro punto sobre mejoras, en lo que respecta al desempeño tiene relación con el uso de WebServices como Middleware en vez de Servlets. Cuando se estudió el uso de Servlet se conocía el potencial que ofrecía el uso de WebServices, pero no se optó por esta opción, el motivo era principalmente que para dar soporte a esta tecnología se debía contar con teléfonos celulares que dieran soporte a la API de WebServices (JSR 172), siendo estos

dispositivos limitados, contraponiéndose a los requerimientos no funcionales para este sistema, en lo que respecta a soporte y operatividad (*Ver Anexo 5, Especificación de Requerimientos de software*), ya que la elección de esta opción hubiese reducido considerablemente la gama de modelos que lograsen haber integrar este sistema.

6.2.2 Respecto al Middleware y Software en el Servidor

6.2.2.1 Servicios

La integración de nuevos servicios ofrecidos comúnmente por los servidores representa un requerimiento básico para la constante actualización del sistema, pudiendo otorgar el control de cada vez más servicios. Además es necesario integrar los nuevos desarrollos que se van presentando para satisfacer las necesidades de las personas.

Para lograr la anterior, es que el sistema ha sido desarrollado para que en el futuro solo sea necesaria la integración de “plugins” no afectando mayormente el normal funcionamiento del sistema en su núcleo. Es necesario tener claro que para la integración de nuevos servicios corresponde analizar los diferentes tópicos que contiene el software a integrar para luego monitorear.

También es necesario el constante soporte a las nuevas versiones de servicios, actualmente soportados, debido a que cambios en las nuevas versiones pueden afectar el normal funcionamiento del sistema no logrando cumplir con su objetivo.

6.2.2.2 Servidores

Esta mejora se refiere a que siempre existen más de un servidor y que estos realizan tareas independientes unos de otros, como servidor de correo, web, etc., por lo cual la instalación de este sistema se tendría que realizar en cada servidor, pero si se lograra la integración del sistema y dar soporte a una cantidad n de maquinas, solo produciéndose la instalación de este sistema en un solo servidor, pero pudiendo administrar todos los servidores solo a través del teléfono celular, el sistema obtendría de forma centralizada la información de cada servidor y así proveerlo al administrador del sistema.

6.2.2.3 Distribuciones Linux

El soporte a nuevas distribuciones Linux es algo primordial dentro de las mejoras de este sistema debido a que este fue desarrollado en totalidad con la distribución Debian Gnu/Linux, por lo tanto esta como las distribuciones basadas en Debian pueden soportar este sistema. Es por esto que es esencial dar soporte a distribuciones tales como Red Hat debido a que su inclusión en el mercado de servidores ha sido de manera muy fuerte, convirtiéndose incluso en una de las más usadas en el mundo, incluso empresas como Hp solo permiten la instalación de Red Hat en sus maquinas. La integración va por el lado de la estandarización de los comandos Linux y de los posibles cambios que se puedan presentar respecto a Debian Gnu/Linux.

Capítulo 7
Conclusión

La integración de las tecnologías móviles con servidores Linux en este sistema sin duda ha sido un acierto, debido a que se logró desarrollar una herramienta que pretende ser un avance en las comunicaciones, gestión y administración de estas maquinas, concediendo que las tareas de administración remota puedan llevarse a otro nivel adquiriendo nuevas directrices y elementos que no se encontraban en el mercado, combinando los nuevos avances tecnológicos que permiten que la labor de los profesionales e especialistas en el ámbito de la computación e informática sea hecho de manera más profesional, didáctica e interactiva que la ya existente, transformando la vida diaria de un administrador de sistemas en todo un acontecimiento tecnológico, haciendo que su trabajo aumente en disponibilidad y eficacia.

El avance de la Internet y de las comunicaciones mundiales requieren que los servidores sean cada vez mas estables y confiables, es por esto que se hace necesario que la mantención y por ende la administración se ha tratada como una actividad de suma importancia, en este sentido el aporte de este sistema, radica en que las respuestas a problemas se resuelvan de forma más ágil tanto así que los planes de contingencia y decisiones importantes son tomadas en base a información veraz logrando obtener un trabajo por parte del administrador mas profesional y eficaz a la hora de enfrentar problemas en los servidores.

Sin duda este sistema ha presentado un avance en la optimización y rápida respuesta ante eventos inesperados que se obtienen en la administración de estas maquinas, aumentando tanto la conectividad como la rápida resolución de los problemas que se presenten en la entrega del servicio, debido a las características que otorgan las tecnologías móviles actuales. Esto adquiere más relevancia debido al mayor uso de herramientas pertenecientes en el mundo del software libre, como los sistemas operativos Linux, que ostentan en la

actualidad un mayor auge en el mercado por parte de las empresas del rubro, debido a las diferentes ventajas con respecto a otros sistemas operativos, como el menor costo y la mayor seguridad de estos, otorgando otro valor agregado a esta herramienta y haciéndola un gran avance en la administración de servidores.

Como ha quedado demostrado en este sistema, los avances tecnológicos están permitiendo la ubicuidad de las misma, es labor de los profesionales e especialistas el ámbito de la computación e informática difundir, abrir y sentar el camino para una nueva generación de sociedad, generación la cual estará inmersa en un mundo rodeado aspectos tecnológicos que elevará la calidad de vida, esperando eso sí, que el resultado de este salto esté disponible para todos, sin discriminación de ningún tipo, pues los avances tecnológicos se construyen con gente y para la gente.

Anexo 1

Visión del Sistema

1 Introducción

El objetivo principal de este Sistema es ayudar a los diferentes administradores de Servidores en el proceso rutinario de control, supervisión y administración general del sistema con la salvedad de que la manera de llevar acabo dichas labores será realizada de forma remota. Este sistema facilitará el proceso de control de las Servicios prestados por el Servidor, proporcionado paras ello una aplicación desarrollada para teléfonos celulares con las características únicas que permitirán satisfacer las necesidades de administración remota de servidores, eliminando incertidumbre que sucedía al no estar consciente sobre los sucesos que en la estación de trabajo ocurrían.

2 Aspectos del sistema

2.1 El Problema

El problema radica en dar solución efectiva a la administración de Servidores Linux de forma remota para el caso en que el administrador no se encuentre en la estación de trabajo.

2.2 Del impacto

El impacto del problema radica en que al no saber el estado del o los Servidores y de los Servicios prestados por los mismos, crea ciertos grados de incertidumbre sobre todo si se habla de servicios cuyas prestaciones son críticas, el efecto resultante es desconocimiento sobre el real estado del Servidor y los Servicios, sumado a esto la inhabilitación para ejercer algún control de forma remota una vez enterado de algún malfuncionamiento o desperfecto.

2.3 Los afectados

Los afectados son todos aquellos sistemas, personas o aplicaciones que hagan uso de los Servicios prestados por los Servidores administrados y gestionados por el administrador.

2.4 La solución

Se propone la creación sistema capaz de facilitar al administrador la gestión remota de sus Servidores mediante una aplicación desarrollada para Teléfonos Celulares. Dicha aplicación permitirá controlar los Servicios ofertados por el Servidor como este mismo, además de concebir la posibilidad de verificar estados operativos y Logs.

3 El sistema

3.1 Características del Sistema

Una de las características que hacen único al sistema tiene relación con el hecho de poder controlar el Servidor de forma remota aun bajo costo operacional, esto gracias a la utilización del Teléfono Celular como recurso para gestionar Servicios y Servidores supliendo otros computadores para dichos fines.

Conscientes del problema sobre una eficaz administración remota, se han desarrollado herramientas de tipo web que posibilitan ejercer control, no obstante para lograr dicho objetivo se requiere la utilización de computadores e Internet como mínimo, recursos innecesarios bajo el enfoque de esta solución, puesto que lo últimos avances en tecnología celular sumado a esto la inclusión de internet con conexiones de alta velocidad suplirán computador y conexión, respectivamente.

3.2 Beneficios del sistema

- Control del Servidor y Servicios, desde cualquier lugar y hora.
- Bajo costo operacional.
- Independencia de plataforma (Modelo de Teléfono) donde se cargue el cliente.
- Información coherente del estado del servidor.
- Intervención remota en el servidor frente a situaciones inesperadas.

3.3 Instalación

Para la instalación se deben crear 2 paquetes instaladores para cada componente que el sistema abarca, esto es un instalador para el Servidor que contenga las herramientas y dependencias que permitan obtener la información del mismo y los Servicios prestados por él, creando además el enlace de comunicación con el Cliente Móvil. El segundo instalador corresponderá al que será ejecutado en los Teléfonos Celulares el cual permitirá instalar el Cliente Móvil en dichos aparatos.

3.4 Costos y precios

Los costos asociados al desarrollo del sistema no existen dado que las herramientas necesarias no incluyen un costo por su adquisición o uso, esto se debe a que las licencias de estas en su mayoría son GPL permitiendo la libre modificación y aplicación, esto incluye toda la plataforma del servidor Linux utilizado. Para el directo desarrollo es utilizado un IDE llamado Netbeans proporcionado por Sun también de forma gratuita, haciendo que todo el desarrollo este exento de costos. El único costo monetario en que se puede incurrir es en el acceso del celular a las consultas mediante la red GPRS.

4 Limitaciones del sistema

Una de las limitantes de software, específicamente con el lado de la aplicación que se ejecuta en los teléfonos celulares tiene que relación con la misma naturaleza en el cual esta se desenvuelve; la telefonía móvil y comunicación inalámbrica.

El hecho de navegar, enviar mensajes de texto, establecer comunicación hablada mediante teléfonos celulares está condicionado por dos factores; costo y cobertura de señal, como el sistema está inmerso en esta misma tecnología comparte las mismas limitantes.

El uso del sistema no obtendrá frutos esperados si se quiere utilizar en momentos en los cuales la calidad e intensidad de la señal sea nula o muy deficiente, respectivamente o no se disponga de saldo suficiente en la cuenta telefónica del propietario del teléfono celular con la compañía que brinda el acceso.

5 Otros Requerimientos del Sistema

5.1 Desempeño

Tabla A1.1 Requerimientos de desempeño del sistema.

Requerimiento	Descripción
Tiempo de Respuestas	Las respuestas provenientes desde el servidor deben presentar un tiempo razonable para la presentación al usuario del sistema.
Seguridad	Debido a que la información tratada por este sistema es de suma importancia, es necesario mantener estándares de seguridad, que no permitan la fácil obtención de información por partes de usuarios maliciosos.
Estabilidad	Aunque no es posible tener un sistema cien por ciento estable, para este sistema es necesario que los errores y/o fallos sean mínimos o que estén bien acotados.
Portabilidad	Debido a la gran variedad de modelos de celulares existentes en el mercado, se requiere que la aplicación que en estos se ejecutará, sea soportada por el mayor número de modelos posible sin variar su desempeño.

5.2 Hardware

Tabla A1.2 Requerimientos de hardware.

Requerimiento	Cantidad	Descripción
Teléfono Celular	>= 1	Teléfono celular memoria libre y conexión a internet.
Servidor	>=1	Debido a las especiales funciones que cumplen estas maquinas, se entiende que el nivel de hardware es superior a cualquier maquina normal, logrando que el sistema funcione correctamente.

5.3 Software

Tabla A1.3 Requerimientos de software del sistema.

Requerimiento	Versión	Fabricante	Descripción
Debian Gnu/Linux	4.0 Etch	Comunidad Debian Gnu/Linux	Este requerimiento se entiende como mínimo, es decir, aceptando nuevas versiones de este SO.
Apache Tomcat	5.5	Apache Software Foundation.	También es posible utilizar otros contenedores de middleware, tal como Glassfish.
Java	1.5	Sun Microsystems, Inc.	Para el servidor es necesaria la versión 1.5 de J2EE. Para el teléfono celular se requiere la versión J2ME con perfil Midp 2.0.

Anexo 2

Arquitectura del Software

1 Introducción

El siguiente documento tiene como finalidad describir la arquitectura de software detallando en si mismo su propósito y como se llegó a establecer dicho diseño.

Mediante la descripción de sus componentes se logra dimensionar la plataforma en donde se sostiene este sistema, como estos logran comunicarse e interactuar, haciendo hincapié de forma constante en la seguridad.

Dichos componentes establecen requerimientos hacia las unidades que los contienen, las unidades, el teléfono celular y por supuesto el servidor, deben cumplir con requisitos que obligan a ser considerados como críticos para un correcto funcionamiento del sistema.

2 Documentos Relacionados

- Anexo 1, Visión del sistema
- Anexo 5, Especificación de Requerimientos

3 Resumen Arquitectónico

3.1 Hechos más Importantes

La plataforma de desarrollo para aplicaciones para teléfonos celulares J2ME desde sus comienzos ha puesto hincapié en permitir al desarrollador la creación de aplicaciones capaces de establecer comunicación con su entorno, estos es mediante Bluetooth, Infrarrojos ó HTTP. Este último con bastante documentación pues la base donde se sostiene las actualizaciones para toda aplicación desarrollada en J2ME.

3.2 Estilo Arquitectónico

El estilo arquitectónico corresponderá al tipo Cliente-Servidor de 2 capas. Con cliente ligero.

3.2.1 Capa presentación o Capa de Usuario

La capa de presentación de la mayoría de aplicaciones resulta con mucha frecuencia esencial para su éxito. Después de todo, representa la interfaz entre el usuario y el resto de la aplicación. Es, por así decirlo, el aspecto fundamental. Si el usuario no puede interactuar con la aplicación de forma que le permita realizar su trabajo con eficacia, el éxito global de la misma se verá seriamente perjudicado.

Para la Capa de presentación se pretende implementar una aplicación cliente desarrollada para teléfonos celulares con la lógica suficiente para interactuar con el Servidor, dicha interacción permitirá realizar una completa gestión y administración remota. Mediante una serie de intuitivas interfaces de usuario que guiarán al administrador para el rutinario ejercicio de control con la salvedad que se realizara todo desde un teléfono celular.

Este cliente será del tipo ligero. El teléfono celular realizará la función de solo GUI en el sistema, solo se hará cargo de la presentación, las validaciones. El real procesamiento será ejecutado por el Servidor. Las limitadas características operacionales de los teléfonos celulares avalan aun más este estilo de arquitectura.

3.2.2 Capa Negocio

Bajo la capa de negocio, se encontrará el componente que facilitará la comunicación entre el Servidor y el Cliente (teléfono celular), genéricamente conocido como Middleware mantendrá el sincronismo, privacidad e integridad y ejecutará las validaciones correspondientes en la información transferida, además será este quien se comunicará con el Servidor mediante comandos Linux y la ejecución de una serie de scripts que actuarán directamente con el sistema operativo para la obtención de estados, registros e información relevante del Servidor y los Servicios que en él se ejecutan.

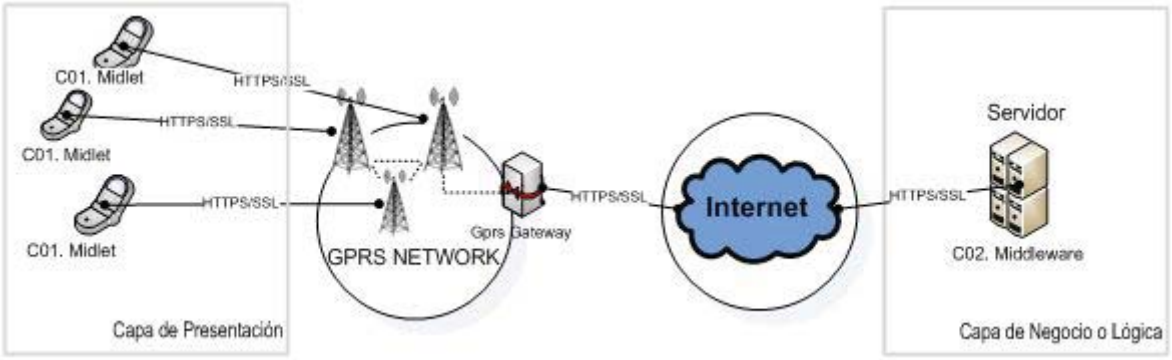
La capa de negocio en realidad corresponde al conjunto representado por el Middleware y el Servidor, no obstante, el Middleware es ejecutado dentro del Servidor, por lo cual se visualiza como un solo componente bajo esta capa, representado por el Middleware.

3.3 Objetivos de la Arquitectura

El objetivo final que persigue la arquitectura es que por el hecho de estar divididas en Capas o componentes, la construcción y posterior mantención del sistema, visto como un todo, se realizará de forma en que el esfuerzo de trabajo se enfoca solo en la o las capas de interés, sin interferir con las otras o al menos provocar el menor impacto en ellas, permitiendo un trabajo modular asincrónico por parte del equipo desarrollador con las beneficios de obtener una mejor manutención, expansibilidad, funcionalidad, reutilización y por supuesto el objetivo principal que se persigue, calidad.

4 Componentes Significativos de la Arquitectura del Sistema

Ilustración A2.1 Diagrama de arquitectura.



4.1 Capa de Presentación

Tabla A2.1 Capa de presentación, componente Midlet.

Componente 01. Midlet	
Descripción	Corresponde a la aplicación o Cliente móvil desarrollada para teléfonos celulares. Esta desarrollada enteramente en J2ME.
Requerimientos	Los teléfonos celulares deben contar en donde la aplicación se cargue debe contar con: Conexión a Internet. Memoria libre disponible. Soporte Java.

	MIDP 2.0

4.2 Capa de Negocio o Lógica

Tabla A2.2 Capa del negocio o lógica, componente Middleware.

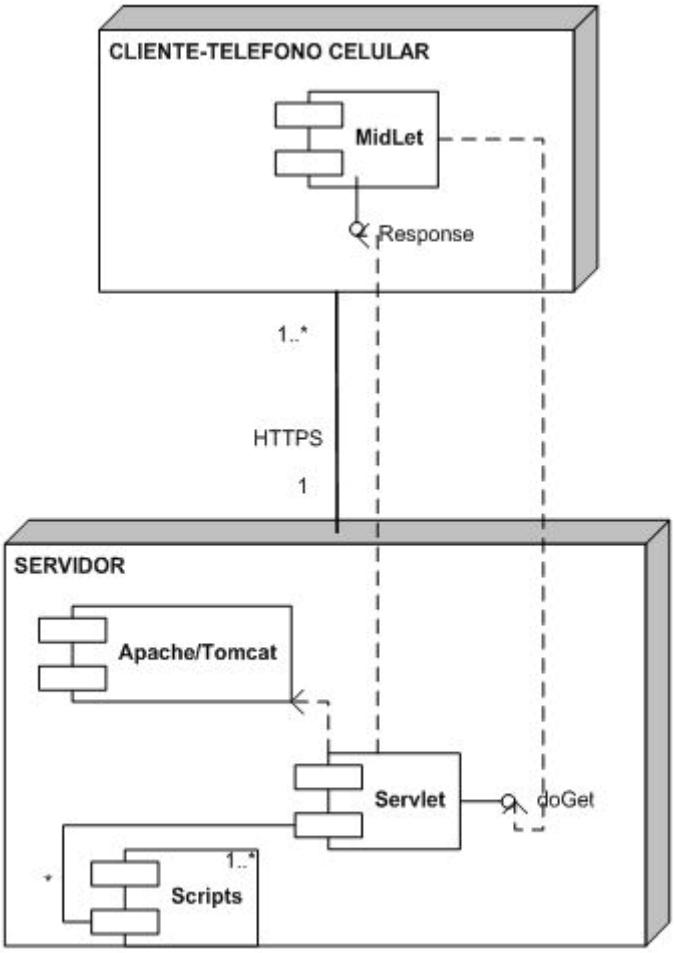
Componente 02. Middleware	
Descripción	Software necesario para “interconectar” el Midlet con el Servidor. Manejará el sincronismo en la comunicación de ambos componentes, gestionara las consultas y la información transferida.
Requerimientos	Los requerimientos a que se enfrenta el middleware son los siguientes: Contenedor de Servlet. Recomendado Apache Tomcat Soporte Java Soporte SSL Linux

5 Vista de Implementación

El siguiente diagrama de despliegue muestra la disposición física de los distintos nodos que componen la arquitectura y el reparto de los componentes sobre dichos nodos.

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. A continuación se presenta el diagrama de despliegue del sistema:

Ilustración A2.2 Diagrama de despliegues.



6 Integración los Componentes y su Comunicación

6.1 Entre el Midlet y Middleware

La comunicación entre estos dos componentes es mediante peticiones HTTPS y consiste básicamente en establecer conexiones HTTP sobre SSL (Secure Socket Layer).

Las peticiones se realizan mediante el uso de GET.

Las peticiones independientes de la naturaleza, serán protegidas, ya que HTTPS provee de sistemas de encriptación, transmisión segura e integridad.

6.1.1 SSL y HTTPS

SSL aporta las siguientes características:

• **Confidencialidad:** Mediante el uso de la Encriptación se garantiza que los datos enviados y recibidos no podrán ser interpretados por ninguna otra persona que no sea ni el emisor ni el receptor.

• **Integridad:** Se garantiza que los datos recibidos son exactamente iguales a los datos enviados, pero no se impide que al receptor la posibilidad de modificar estos datos una vez recibidos.

• **Autentificación:** El vendedor se autentifica utilizando un Certificado Digital emitido por una empresa llamada Autoridad Certificadora, este documento es totalmente infalsificable y garantiza que el Vendedor es quien dice ser.

La idea que persigue SSL es encriptar la comunicación entre servidor y cliente mediante el uso de llaves y algoritmos de encriptación.

¿De qué están compuestos?

Estos protocolos se componen de dos capas: el Record Protocol y el Handshake Protocol.

• **El Record Protocol:** es la capa inmediatamente superior a TCP y proporciona una comunicación segura. Principalmente esta capa toma los mensajes y los codifica con algoritmos de encriptación de llave simétrica como DES, RC4 aplicándole una MAC (Message Authentication Code) para verificar la integridad, logrando así encapsular la seguridad para niveles superiores.

• **El Handshake Protocol:** es la capa superior a la anterior y es usada para gestionar la conexión inicial.

¿Cómo funcionan?

En resumidas cuentas, después que se solicita una comunicación segura, servidor y el cliente se deben poner de acuerdo en cómo se comunicaran (SSL Handshake) para luego comenzar la comunicación encriptada. Luego de terminada la transacción, SSL termina.

• **Solicitud de SSL:** Típicamente este proceso ocurre en el momento que un cliente accede a un servidor seguro, identificado con "https://...". Pero como se mencionó, no necesariamente es usado para HTTP. La comunicación se establecerá por un puerto distinto al utilizado por el servicio normalmente. Luego de esta petición, se procede al SSL Handshake.

• **SSL Handshake:** En este momento, servidor y cliente se ponen de acuerdo en varios parámetros de la comunicación. Se puede dividir el proceso en distintos pasos:

› **Client Hello:** El cliente se presenta. Le pide al servidor que se presente (certifique quien es) y le comunica que algoritmos de encriptación soporta y le envía un número aleatorio para el caso que el servidor no pueda certificar su validez y que aun así se pueda realizar la comunicación segura.

› **Server Hello:** El servidor se presenta. Le responde al cliente con su identificador digital encriptado, su llave pública, el algoritmo que se usará, y otro número aleatorio. El algoritmo usado será el más poderoso que soporte tanto el servidor como el cliente.

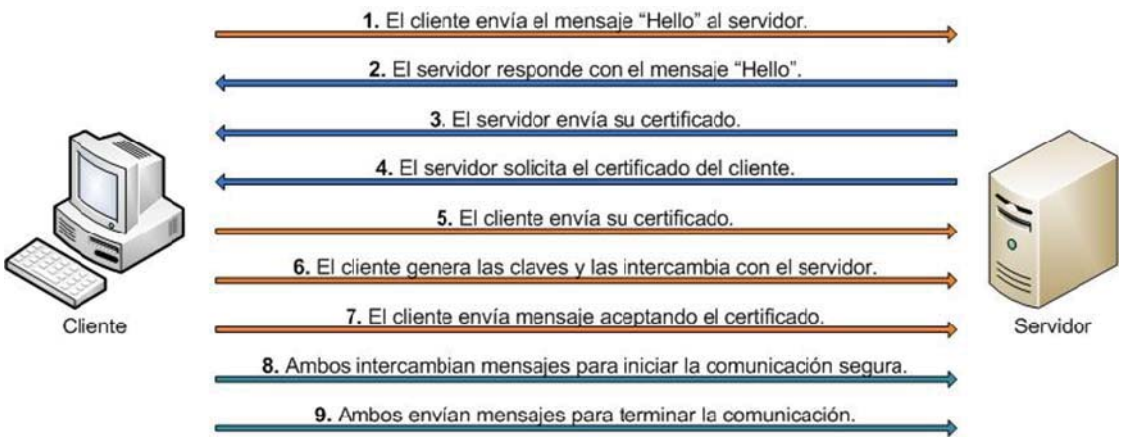
› **Aceptación del cliente:** El cliente recibe el identificador digital del servidor, lo desencripta usando la llave pública también recibida y verifica que dicha identificación proviene de una empresa certificadora .Luego se procede a realizar verificaciones del certificado (identificador) por medio de fechas, URL del servidor, etc. Finalmente el cliente genera una llave aleatoria usando la llave pública del servidor y el algoritmo seleccionado y se la envía al servidor.

› **Verificación:** Ahora tanto el cliente y el servidor conocen la llave aleatoria (El cliente la generó y el servidor la recibió y desencriptó con su llave privada). Para asegurar que nada ha cambiado, ambas partes se envían las llaves. Si coinciden, el Handshake concluye y comienza la transacción.

• **Intercambio de Datos:** Desde este momento los mensajes son encriptados con la llave conocida por el servidor y el cliente y luego son enviados para que en el otro extremo sean desencriptados y leídos.

• **Terminación de SSL:** Cuando el cliente abandona el servidor, se le informa que terminara la sesión segura para luego terminar con SSL.

Ilustración A2.3 SSL Y HTTPS.



Durante el proceso de comunicación segura SSL existen dos estados fundamentales, el estado de sesión y el estado de conexión. A cada sesión se le asigna un número identificador arbitrario, elegido por el servidor, un método de compresión de datos, una serie de algoritmos de encriptación y funciones hash, una clave secreta maestra de 48 bytes y un Flag de nuevas conexiones, que indica si desde la sesión actual se pueden establecer nuevas conexiones. Cada conexión incluye un número secreto para el cliente y otro para el servidor, usados para calcular los MAC de sus mensajes, una clave secreta de encriptación particular para el cliente y otra para el servidor, unos vectores iniciales en el caso de cifrado de datos en bloque y unos números de secuencia asociados a cada mensaje.

Desde el punto de vista de su implementación en los modelos de referencia OSI y TCP/IP, SSL se introduce como una especie de nivel o capa adicional, situada entre la capa de Aplicación y la capa de Transporte, sustituyendo los sockets del sistema operativo, lo que hace que sea independiente de la aplicación que lo utilice, y se implementa generalmente en el puerto 443. (NOTA: Los puertos son las interfaces que hay entre las aplicaciones y la pila de protocolos TCP/IP del sistema operativo).

SSL proporciona servicios de seguridad a la pila de protocolos, encriptando los datos salientes de la capa de Aplicación antes de que estos sean segmentados en la capa de

Transporte y encapsulados y enviados por las capas inferiores. Es más, también puede aplicar algoritmos de compresión a los datos a enviar y fragmentar los bloques de tamaño mayor a 214 bytes, volviéndolos a reensamblarlos en el receptor.

6.1.2 Certificados Digitales

Un certificado SSL es una credencial digital que permite a los visitantes de un sitio Web verificar la autenticidad del sitio y comunicarse con él de manera segura mediante el protocolo SSL. Este lo concede una entidad certificadora: la Certification Authority. Esta entidad concede dicho certificado después de haber controlado la correcta configuración del proceso de encriptación (SSL) y haber comprobado los datos de la empresa solicitante. El certificado de servidor seguro se concede a una entidad cuyas referencias han sido comprobadas, para asegurar que efectivamente quien recibe los datos encriptados es quien debe de recibirlos.

Cuando nos conectamos a un servidor seguro, los navegadores avisan de esta circunstancia mediante un candado de color amarillo en alguna parte de la ventana, en el caso de Firefox la barra de direcciones aparecerá también en color amarillo. Para más seguridad podemos comprobar que la dirección empieza por https://www..., con una ‘s’ después de la ‘p’. Si pinchamos sobre el candado aparecerá la información contenida en el certificado digital que lo autoriza como servidor seguro.

6.2 Entre el Middleware y el Servidor

La comunicación entre el middleware y el servidor es de suma importancia dado que esta es la que interpreta y ejecuta las órdenes provenientes desde el administrador a través del cliente móvil. La comunicación entre el middleware y el servidor se realiza a través de la interpretación de las órdenes recibidas, procediendo luego a la ejecución directa de estas en el servidor a través de mecanismos de ejecución inherentes en el servidor, estas se entienden como comandos Linux. Estos comandos se pueden interpretar como pequeños programas que se ejecutan cada vez que es necesaria una información o ejecución directa de una orden proveniente del administrador, estos programas se hacen llamar Bash Scripts.

6.2.1 Bash Script

Linux, así como la mayoría de los Unix, utilizan Shell scripts para realizar una infinidad de tareas. Un Shell script es un programa que se escribe con una sintaxis particular, en un archivo de texto plano, para que sea interpretado por un Shell, en este caso /bin/bash.

Un Shell script es básicamente un programa que llama a otros programas, con la posibilidad de hacer algún tipo de procesamiento propio (como control de flujo, operaciones matemáticas simples, etc.).

Un script para Bash es un archivo tipo texto que es básicamente un programa que llama a otros programas, con la posibilidad de hacer algún tipo de procesamiento propio (como control de flujo, operaciones matemáticas simples, etc.). Para lograr que el intérprete de comandos intérprete las líneas de un archivo puede:

• Ejecutar /bin/bash seguido del nombre del archivo (o redireccionar la entrada estándar para que provenga del archivo).

• Emplear el comando source seguido del nombre del archivo.

• Emplear el carácter '.' seguido de un espacio y el nombre del archivo.

• Agregar en la primera línea del archivo la cadena #!/bin/bash, dar permiso de ejecución al archivo y teclear el nombre del archivo desde el intérprete de comandos como si fuera un nuevo comando.

En el desarrollo de este sistema se ha utilizado la cuarta forma, es decir, #!bin/bash. Para la implementación se utilizan pequeños script para Bash que contiene las suficientes órdenes para obtener la información necesaria desde un log. Uno de los script implementado es el siguiente:

```
#!/bin/bash
if [ $1 = user ];then
        tail /var/log/user.log | awk '{printf $1" "$2"\n"$3"\n"
        for(i=5 ;i<=NF;i++){
        printf($i" ")}
                printf("\n")
        }'
fi
```

Este script obtiene las ultimas 10 líneas del archivo user.log, esto dependiendo si el primer argumento de cuando es ejecutado es igual a user, gracias a AWK los campos 1, 2 y 3 de cada línea son impresos así obteniendo la fecha, hora y descripción respectivamente.

Para efectos de las peticiones del Midlet se han utilizado los siguientes Scripts que son invocados por el Servlet (Middleware):

- **Logs:** Dependiendo de los argumentos retorna los Logs del servidor y de los servicios

- **RedUsage:** Obedeciendo al argumento entregado por la invocación, este puede retornar los Host que se encuentran apagados o con problemas dentro de la red y la estadística de la interfaz de red del servidor.

- **ServerUsage:** Dependiendo del argumento puede entregar el estado de la Memoria, Swap y CPU del servidor.

- **ServiceInstall:** Retorna los servicios soportados por el Software y a su vez devuelve los servicios instalados en el servidor, todo esto dependiendo del argumento con el cual sea invocado.

- **Status:** Devuelve el estado del servicio especificado en el argumento, este puede ser Running y Not Running.

- **Usuarios:** Retorna los usuarios que están identificados y autorizados a utilizar la aplicación haciendo tareas y funciones sobre el servidor.

• **ServiceUsage:** Retorna la memoria y la CPU utilizada por un servicio instalado en el servidor este debe ser especificado en el argumento.

• **ServicePid:** Devuelve el PID (identificador del proceso) de cada servicio instalado.

La forma de ejecutar cada uno de los Scripts declarados es (esto como usuario root):

$sh [Script] [Argumento1] [Argumento2] ……[Argumento N]

Además de estos programas nos encontramos con comandos ejecutados de forma directa que permiten el control de directo de los servicios que se encuentran en el servidor, tal como el que logra el control básico del sistema. Para lograr esta acción se ejecuta el siguiente comando:

```
/etc/init.d/[servicio] [acción]

acción = start, restart, Stop
```

6.2.2 Logs

Los Logs son registros oficiales de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática un Log es usado para registrar datos o información sobre quién, que, cuando, donde y porque (who, what, when, where y why, W5) un evento ocurre para un dispositivo en particular o aplicación.

Por defecto, la mayoría del logging a ficheros tiene lugar en /var/log/, y generalmente los programas que manejan su propio logging (la mayoría de servidores manejan sus Logs internamente) lo hacen a /var/log/nombreprograma/.

De acuerdo al análisis del sistema se analizaron los Logs de cada servicio prestado por la maquina, y a su vez los Logs del servidor en sí. La aplicación permite mostrar los siguientes Logs:

• /var/log/auth.log: Es en donde se guardan Logs referidos principalmente con la "seguridad" del sistema

• /var/log/syslog: En general se logea todo lo relacionado con accesos (o intentos) a los servicios del sistema (telnet, rpc). Así también como algunos avisos referentes a los módulos.

• /var/log/daemon.log: Contiene un log de los Daemons (Servicios) que se cargan en el sistema así también como información referente a avisos en los módulos.

• /var/log/kern.log: Los mensajes producidos por klogd, que es el k maneja los mensajes del kernel, en su mayoría son almacenados aquí.

• /var/log/user.log: En su mayoría son mensajes enviados a usuarios en el sistema, como por ejemplo un aviso de "shutdown".

Los Logs anteriormente listados vienen definidos por defecto en /etc/syslog.conf.

También se ha procesado e implementado las vistas de los de algunos Logs de los servicios prestados por el servidor, que son los siguientes:

• /var/log/postgresql/postgresql-8.1-main.log: Contiene todo lo relacionado con el motor de base de datos Postgresql (no contiene consultas ejecutadas).

• /var/log/apache2/error.log: Contiene los errores y noticias que se han producido en el servidor Web apache.

• /var/log/exim4/mainlog: Contiene los errores y novedades de Exim.

• Para Mysql y SSH los registros, noticias, errores son obtenidos de los Logs del sistema de forma específica para cada uno.

7 Especificación de Componentes

7.1 Midlet

El Midlet integrará una aplicación externa para dar soporte a la funcionalidad que se especifica en el Anexo de Especificación de Requerimientos, esto corresponde al Caso de Uso, “Abrir Consola”.

Como se detalla en el Anexo de Especificación de Requerimientos, ese caso de uso permitirá al usuario final poder interactuar con el Servidor mediante una interfaz que emulará una consola de comandos dentro del Midlet, permitiendo así lograr tener un mayor control en la administración, ya que la rutina de este Caso de Uso se basa por completo en los conocimientos y expertiz en comandos Linux que tenga el administrador.

MidpSSH, es la aplicación que compondrá el Midlet, MidpSSH es un cliente de SSH y Telnet para teléfonos móviles compatibles con Java MIDP 1.0 / 2.0 (J2ME) licenciado bajo GPL, lo cual concede la libre modificación y uso.

7.2 Middleware

El Middleware a utilizar corresponderá al uso de Servlets. Un Servlet es un objeto que se ejecuta en un servidor o contenedor JEE, fue especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente es HTML. Otras opciones que permiten generar contenido dinámico son con los lenguajes ASP, PHP, JSP (un caso especial de Servlet) y Python.

Los Servlets forman parte de JEE (Java Enterprise Edition), que es una ampliación de JSE (Java Standard Edition).

Un Servlet implementa la interfaz javax.servlet.Servlet o hereda alguna de las clases más convenientes para un protocolo específico (ej: javax.servlet.HttpServlet). Al implementar esta interfaz el Servlet es capaz de interpretar los objetos de tipo HttpServletRequest y HttpServletResponse quienes contienen la información de la página que invocó al Servlet.

Entre el servidor de aplicaciones (o web content) y el Servlet existe un contrato que determina cómo han de interactuar. La especificación de éste se encuentra en los JSR (Java Specification Requests) del JCP (Java Community Process).

Anexo 3

Gestión del Riesgo

1 Introducción

El reconocer los riesgos que se pueden presentar durante el desarrollo e implementación del sistema o software, puede facilitar la pronta mitigación de estos, haciendo que no se transformen en algo catastrófico que impida el buen funcionamiento del sistema, es por esto que a continuación se especifican los principales riesgos de este sistema.

2 Documentos Relacionados

- Anexo 1, Visión del sistema.
- Anexo 5, Especificación de Requerimientos del Sistema.

3 Identificación y Control de Riesgos

3.1 Clasificación de los Riesgos

- **Recursos:** Se refiere a todas las herramientas y tecnologías necesarias para el desarrollo e implementación del sistema.
- **Tiempo planificado:** Describe a los tiempos de desarrollo e implantación que se han presentado en el inicio del sistema.
- **Personas:** Se refiere a las personas directamente involucradas en las distintas iteraciones del sistema durante el desarrollo, es decir, los encargados y desarrolladores.
- **Requerimientos:** Son los requerimientos expresados por el cliente al inicio y durante el desarrollo del sistema.

3.2 Identificación de los Riesgos

Tabla A3.1 Riesgos del sistema.

Riesgo	Clasificación	Especificación
Cambio de Tecnología	Requerimientos	La tecnología con la cual es construido el sistema ha sido sustituida por nueva tecnología.

Cambios en los Requerimientos	Requerimientos	Dependiendo de la evolución del sistema irán apareciendo otros requerimientos en forma anticipada.
Desarrolladores con problemas de motivación.	Personas	Los desarrolladores presentan baja motivación, no participan activamente del sistema.
Desempeño de las herramientas CASE	Recursos	Las herramientas CASE que apoyan el sistema no tienen el desempeño anticipado.
Falta de experiencia	Personas	La Falta de experiencia de los desarrolladores podría conducir a errores.
Mala elección de Herramientas	Recursos y Personas	Las herramientas seleccionadas para el desarrollo del sistema presentan un desempeño deficiente.
No disponibilidad de hardware	Recursos	El Hardware que es esencial para el sistema, no será adquirido o entregado dentro de lo planificado.
Poca documentación	Recursos	La documentación de las herramientas y lenguajes de programación a utilizar es deficiente.
Retrasos en la especificación	Tiempo Planificado	Las especificaciones principales del sistema no estarán listas antes de la fecha de entrega.
Subestimación de tamaño	Tiempo planificado	El tamaño del sistema ha sido subestimado.

Tamaño del Midlet demasiado grande	Personas	Mala programación.

3.3 Ponderación de los Riesgos

3.3.1 Escala de Ponderación de los Riesgos

Tabla A3.2 Escala de ponderación de riesgos.

Probabilidad del Riesgo	Porcentajes
Muy bajo	< 10%
Bajo	10% - 25%
Moderada	25% - 50%
Alta	50% - 75%
Muy alto	> 75%

3.3.2 Probabilidad de Ocurrencia e Impacto

Tabla A3.3 Ocurrencia e impacto de los riesgos.

Riesgo	Probabilidad	Efectos
Cambio de Tecnología	Alto	Tolerable
Cambios en los Requerimientos	Alta	Serio
Desarrolladores con problemas de motivación.	Bajo	Serio
Desempeño de las herramientas CASE	Moderada	Insignificante

Falta Capacitación	Bajo	Catastrófico
Mala elección de Herramientas	Moderada	Catastrófico
No disponibilidad de hardware	Bajo	Serio
Poca documentación	Moderada	Serio
Retrasos en la especificación	Moderada	Serio
Subestimación de tamaño	Alta	Serio
Tamaño del Midlet demasiado grande	Alta	Serio

3.4 Planificación del Plan de Riesgos

Tabla A3.4 Indicadores de la presencia de los riesgos.

Riesgo	Indicadores Potenciales
Cambio de Tecnología	Nueva tecnología sale al mercado reemplazando la ya existente.
Cambio en los Requerimientos	Demasiadas peticiones de cambios y surgimientos de nuevas necesidades por parte del cliente.
Desarrolladores con problemas de motivación	Desarrolladores no cumplen a cabalidad con las tareas designadas.
Desempeño de las herramientas CASE	Rechazo del equipo para utilizar la herramienta, quejas sobre la herramienta.
Falta Capacitación	Atraso en el desarrollo de la aplicación.
Mala elección de Herramientas	Mal funcionamiento de la herramienta elegida.

No Disponibilidad de Hardware	Entrega retrasada del hardware, problemas tecnológicos detectados.
Poca documentación	La investigación de las tecnologías y herramientas no ha tenido resultados.
Retrasos en la especificación	Fracaso en el cumplimiento de alguno de los tiempos acordados.
Subestimación de tamaño	Aparecen nuevas funcionalidades que dan mayor complejidad al sistema
Tamaño del Midlet demasiado grande	El tamaño del archivo ".jar" crece por cada nueva compilación del sistema.

Tabla A3.5 Estrategia frente a un riesgo.

Riesgo	Estrategia
Bajo desempeño de la herramienta CASE	Desechar la herramienta y elegir una nueva, o no utilizar alguna.
Cambio de tecnología	Averiguar las posibilidades que ofrecen los distintos equipos que se encuentran en el mercado para la implementación del sistema.
Cambios en los requerimientos	Rastrear la información para valorar el impacto de los requerimientos, maximizar la información oculta en ellos
Desarrolladores con problemas de motivación.	Incentivar trabajos en grupos y fijar objetivos a corto plazo.
Falta Capacitación	Aumentar la capacitación de los desarrolladores.

Mala elección de Herramientas	Buscar nuevas herramientas. Rehacer trabajo.
No disponibilidad de hardware	Reestructurar la planificación de manera de que el hardware que va estar no disponible, no sea esencial en algún periodo determinado.
Poca documentación	Consultar a personal especializado en los diferentes tópicos que existan problemas.
Retrasos en la especificación	Reorganizar las labores dentro del grupo de trabajo, y potenciar las virtudes del grupo laboral.
Subestimación de tamaño	Aumentar la asignación de h/h a partes del sistema que han sido subestimadas.
Tamaño del Midlet demasiado grande	Reescribir el código aplicando los conceptos de orientación a objeto.

Anexo 4

Plan de Pruebas

1 Introducción

El siguiente documento tiene como objetivo, verificar que los requerimientos de software ya sean los funcionales y no funcionales fueron abordados a cabalidad,

2 Documentos Relacionados

- Anexo 1, Visión del sistema.
- Anexo 2, Arquitectura de software.
- Anexo 5, Especificación de Requerimientos del Sistema.

3 Antecedentes y Propósito

3.1 Propósito de la Evaluación

"Calidad" se refiere a todas las cosas buenas que nos gustaría ver en el producto. La idea fundamental es hacer un producto de calidad y esto se logra manteniendo calidad en mente todo el tiempo y realizando las actividades para esto. Las pruebas son una actividad de aseguramiento de calidad. Es necesario un plan para seleccionar y coordinar todas las actividades para asegurar la calidad del producto durante el ciclo de vida del sistema, para ello a de especificarse para cada iteración a realizarse cual es el objetivo a conseguir con la aplicación de este plan:

- Encontrar tantos errores como sea posible.
- Supervisar si se cumple las especificaciones de diseño.
- Supervisar si se cumple los requisitos del análisis.
- Realizar pruebas de rendimiento y capacidad.
- Encontrar los problemas importantes y determinar los riesgos percibidos de la calidad.
- Otros.

3.2 Motivadores de la prueba

La prueba tiene como fin evaluar si se cumplen con:

- Requerimientos Funcionales
- Requerimientos no Funcionales

3.3 Objetos a ser Evaluados

Los objetos a ser evaluados corresponden a los siguientes componentes:

- Componente Midlet

- Componente Middleware

Dichos componentes no serán evaluados como unidades aisladas ya que el enfoque de esta prueba es verificar el real comportamiento del sistema ya finalizado, lo cual involucra la interacción de ambos componentes verificando entre ellos la comunicación como a su vez el comportamiento individual ante la interacción del componente contiguo y humano, tal como se espera que suceda cuando este sea lazado.

4 Enfoque de las Pruebas

4.1 Tipo de enfoque y prueba

El enfoque para esta entrega corresponderá a un enfoque de Caja Negra, los datos de prueba se escogerán atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Criterios mínimos que guiarán al escoger los datos de prueba:

- Valores Fáciles: El programa se depurará con datos de fácil comprobación.

- Valores típicos realistas: se ensayará un programa con datos seleccionados para que representen como se aplicará. Los Datos han de ser sencillos, de modo que los resultados sean verificables en forma manual.

- Valores extremos.

- Valores ilegales: Cuando en un programa entra basura, su salida habrá de ser un mensaje de error adecuado. Es preferible que el programa ofrezca indicación de errores en la entrada y que realice cálculos que sigan siendo factibles luego de desechar la entrada equivocada.

El método de la caja negra se centra en los requisitos fundamentales del software y permite obtener entradas que prueben todos los requisitos funcionales del programa.

Para los Casos de prueba, estas se pueden catalogar en:

- Pruebas de Función

- Pruebas de Interfaces de usuario
- Pruebas de Carga
- Pruebas de Desempeño
- Pruebas de Fallas y Recuperación
- Pruebas de Configuración
- Pruebas de Seguridad y Control de Acceso

5 Herramientas para las Pruebas

5.1 Software

No se necesitaron herramienta de software externas adicionales para llevar a cabo las pruebas.

5.2 Hardware

- PC Intel Pentium 4, 2277 MHz, 512 MB (PC3200 DDR SDRAM), 160 GB Hard Disk, SO: Debian Gnu/Linux version 4.0 ETCH
- Teléfono Celular Sony Ericsson, Modelo K310a.

6 Casos de Prueba

6.1 Para el Caso de uso Control Básico

Tabla A4.1 Primer caso de prueba para el caso de uso control básico.

Caso de Uso: Control básico	**Tipo de Prueba:** Prueba de función
Requerimiento: -	**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de

	Pontificia Universidad Católica de Valparaíso.
Escenario: El control de un Servicio en particular estando este no activo.	**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.
Versión del Caso de Prueba: 1.5	**Nombre del Probador:** Juan Francisco García y Sergio Salas.

Condición(es) para que se ejecute el Caso de Prueba
Haber seleccionado algún servicio de la lista de servicios desplegada por la aplicación. Para este Caso de prueba se trabajó con el servicio PostgreSql, inicialmente estando abajo.

Para la ejecución del Caso de Prueba			
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se presiona la acción de control “iniciar”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien iniciará el servicio seleccionado.	Se obtuvo lo requerido
Se presiona la acción de control “parar”	-	El servicio no está arriba por lo que esta acción en la aplicación no debiese estar disponible o estar presente	Se obtuvo lo requerido
Se presiona la acción de control “reiniciar”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien reiniciará el servicio seleccionado.	Se obtuvo lo requerido
Se presiona la acción de control	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet que obtendrá el Log de los últimos eventos	Se obtuvo lo requerido

“ver Log”		sucedidos para el servicio seleccionado.	
Decisión de aprobación de Caso de Prueba: Aprobó			

Tabla A4.2 Segundo caso de prueba para el caso de uso control básico.

Caso de Uso: Control básico		**Tipo de Prueba:** Prueba de función	
Requerimiento: -		**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de Pontificia Universidad Católica de Valparaíso.	
Escenario: El control de un Servicio en particular estando activo.		**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.	
Versión del Caso de Prueba: 1.5		**Nombre del Probador:** Juan Francisco García y Sergio Salas.	
Condición(es) para que se ejecute el Caso de Prueba:			
Haber seleccionado algún servicio de la lista de servicios desplegada por la aplicación. Para este Caso de prueba se trabajó con el servicio PostgreSql, inicialmente estando arriba.			
Para la ejecución del Caso de Prueba			
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se presiona la acción de control	-	El servicio no está arriba por lo que esta acción en la aplicación no debiese estar	Se Obtuvo

“iniciar”		disponible o estar presente	lo requerido
Se presiona la acción de control “parar”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien parará el servicio seleccionado.	Se Obtuvo lo requerido
Se presiona la acción de control “reiniciar”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien reiniciará el servicio seleccionado.	Se Obtuvo lo requerido
Se presiona la acción de control “ver Log”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet que obtendrá el Log de los últimos eventos sucedidos para el servicio seleccionado.	Se Obtuvo lo requerido
Decisión de aprobación de Caso de Prueba: Aprobó			

6.2 Para el Caso de Uso Ver Log

Tabla A4.3 Primer caso de prueba para el caso de uso ver log.

Caso de Uso: Ver Log.	**Tipo de Prueba:** Prueba de función.
Requerimiento: -	**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de Pontificia Universidad Católica de Valparaíso.
Escenario: La Visualización del log de un Servicio en particular, estando este activo.	**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.

Versión del Caso de Prueba: 1.0	**Nombre del Probador:** Juan Francisco García y Sergio Salas.
Condición(es) para que se ejecute el Caso de Prueba:	
Haber seleccionado algún servicio de la lista de servicios desplegada por la aplicación. Para este Caso de prueba se trabajó con el servicio PostgreSql.	
Para la ejecución del Caso de Prueba	

Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se presiona la acción “ver log”	-	Se debe desplegar en la aplicación los últimos 10 o menos eventos, relacionado con el Servicio seleccionado.	Se Obtuvo lo requerido
Decisión de aprobación de Caso de Prueba: Aprobó			

Tabla A4.4 Segundo caso de prueba para el caso de uso ver log.

Caso de Uso: Ver Log	**Tipo de Prueba:** Prueba de función
Requerimiento: -	**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de Pontificia Universidad Católica de Valparaíso.
Escenario: La Visualización del log del Servidor	**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.

Versión del Caso de Prueba: 1.0	**Nombre del Probador:** Juan Francisco García y Sergio Salas.
Condición(es) para que se ejecute el Caso de Prueba:	
Haber seleccionado la opción “ver log Servidor” desplegada por la aplicación.	

Para la ejecución del Caso de Prueba			
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se presiona la acción “Auth.log”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien devolverá las últimas 10 líneas de este log, haciendo referencia a los eventos producidos, de acuerdo a la seguridad, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Se presiona la acción “Sys.log”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien devolverá las últimas 10 líneas de este log, haciendo referencia a los eventos producidos, de acuerdo a los intentos de acceso a los servicios del sistema, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Se presiona la acción “Kern.log”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien devolverá las últimas 10 líneas de este log, haciendo referencia a los eventos	Se Obtuvo lo requerido

		producidos, de acuerdo a los mensajes provenientes del kernel, para ser desplegada por la aplicación en la pantalla del teléfono celular.	
Se presiona la acción “Daemon.log”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien devolverá las últimas 10 líneas de este log, haciendo referencia a los eventos producidos, de acuerdo a los servicios que se cargan en el sistema, y así ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se presiona la acción “User log”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien devolverá las últimas 10 líneas de este log, haciendo referencia a los eventos producidos, de acuerdo a los mensajes enviados a los usuarios del sistema, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Decisión de aprobación de Caso de Prueba: Aprobó			

6.3 Para el Caso de Uso Ver Monitorización

Tabla A4.5 Primer caso de prueba para el caso de uso ver monitorización.

Caso de Uso: Ver Monitorización	**Tipo de Prueba:** Prueba de función
Requerimiento: -	**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de Pontificia Universidad Católica de Valparaíso.
Escenario: La monitorización de un Servicio en particular, estando este activo	**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.
Versión del Caso de Prueba: 1.0	**Nombre del Probador:** Juan Francisco García y Sergio Salas.

Condición(es) para que se ejecute el Caso de Prueba:
Haber seleccionado la opción “Monitorización” y dentro de esta “Monitorización del Servicios” desplegada por la aplicación.

Para la ejecución del Caso de Prueba			
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se presiona la acción de control “ver carga de CPU”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien obtendrá la carga de CPU que está utilizando el servicio seleccionado, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Se presiona la acción de control	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet	Se Obtuvo

“ver carga memoria”		quien obtendrá el uso de la memoria que está siendo utilizada por el servicio seleccionado, para ser desplegada por la aplicación en la pantalla del teléfono celular.	lo requerido
Decisión de aprobación de Caso de Prueba: Aprobó			

Tabla A4.6 Segundo caso de prueba para el caso de uso ver monitorización.

Caso de Uso: Ver Monitorización	**Tipo de Prueba:** Prueba de función
Requerimiento: -	**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de Pontificia Universidad Católica de Valparaíso.
Escenario: La monitorización del Servidor	**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.
Versión del Caso de Prueba: 1.0	**Nombre del Probador:** Juan Francisco García y Sergio Salas.
Condición(es) para que se ejecute el Caso de Prueba:	
Haber seleccionado la opción “Monitorización” y dentro de esta “Monitorización del Servidor” desplegada por la aplicación.	
Para la ejecución del Caso de Prueba	

Acción o Condición	Valores	Resultado Esperado	Resultado Obtenido
Se presiona la acción de control “ver carga de CPU”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien, obtendrá la carga de CPU que se está produciendo en el servidor de acuerdo a todos los procesos que se están ejecutando, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Se presiona la acción de control “ver carga memoria”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien obtendrá la carga de la memoria que está siendo utilizada por el total de procesos que se están ejecutando, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Se presiona la acción de control “ver Interfaz de red”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien obtendrá la información de las interfaces de red del servidor, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Se presiona la acción de control “ver Host Down”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien obtendrá los host que se encuentran en la red y tienen estado “down”, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**

Se presiona la acción de control “ver Swap”	-	Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet quien obtendrá la carga de memoria Swap que se están produciendo en el servidor de acuerdo a todos los procesos que se están ejecutando, para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se Obtuvo lo requerido
Decisión de aprobación de Caso de Prueba: Aprobó			

6.4 Para el Caso de Prueba de Abrir Consola

Tabla A4.7 Caso de prueba para el caso de uso abrir consola.

Caso de Uso: Abrir Consola	**Tipo de Prueba:** Prueba de función
Requerimiento	**Ambiente de Prueba:** Prueba realizada en las dependencias de la Escuela de Ingeniería Informática de Pontificia Universidad Católica de Valparaíso.
Escenario	**Autor Caso de Prueba:** Juan Francisco García y Sergio Salas.
Versión del Caso de Prueba: 1.0	**Nombre del Probador:** Juan Francisco García y Sergio Salas.
Condición(es) para que se ejecute el Caso de Prueba:	
Seleccionar en el menú principal, la opción Abrir consola, además el usuario se debe logiar con su user y pass del servidor.	

Para la ejecución del Caso de Prueba			
Acción o Condición	**Valores**	**Resultado Esperado**	**Resultado Obtenido**
Se ingresa el comando Valido	hostname	El servidor debiese retornar su nombre para ser desplegada por la aplicación en la pantalla del teléfono celular.	Se obtuvo lo requerido
Se ingresa el comando Valido	top	El servidor debiese retornar la información de los procesos que se están ejecutando, y el uso de estos con respecto al CPU, y memoria. Para ser desplegada por la aplicación en la pantalla del teléfono celular con la característica de que la información que se muestra se refresca cada 5 segundos o tras pulsar la barra espaciadora.	Se obtuvo lo requerido
Se ingresa un comando invalido	123abcd	El servidor debiese retornar algún mensaje de alerta advirtiendo que no se conoce el comando o este es inválido.	Se obtuvo lo requerido
Se ingresa un comando invalido vacio		Se espera que al ejecutar esta acción en la aplicación, esta sea comunicada al Servlet que obtendrá el Log de los últimos eventos sucedidos para el servicio seleccionado.	Se obtuvo lo requerido
Decisión de aprobación de Caso de Prueba: Aprobó			

7 Criterios para el Lanzamiento

7.1 Criterios de Evaluación

El lanzamiento principal de la solución está vinculado a la gravedad y la prioridad de errores sin resolver de acuerdo con los criterios siguientes:

• No existen errores sin solucionar de Gravedad 1 o Gravedad 2.

• No existen errores sin solucionar de Prioridad 1 o Prioridad 2 de ningún nivel de gravedad.

• Todos los casos de prueba del entorno de laboratorio de prueba se han completado satisfactoriamente.

7.2 Clasificación de los errores

Tabla A4.8 Clasificación de los errores en las pruebas.

Calificación	Definición de gravedad	Definición de prioridad
1	El error provoca el loquea del sistema o perdida de datos	El error debe corregirse lo antes posible. El error bloquea el progreso en esta área.
2	El error causa problemas graves en la funcionalidad u otros aspectos importantes; el producto se bloquea en casos poco claros.	El error debe corregirse antes del lanzamiento del producto.

7.3 Resultados de la prueba

Se obtuvieron resultados positivos de todos los casos de prueba. No hubo errores sin resolver de Gravedad 1 y 2 ni de un nivel de Prioridad 2. Esto demuestra la consecución de los objetivos de las pruebas, con lo que la solución está lista para el lanzamiento.

Anexo 5

Especificación de Requerimientos

1 Introducción

Para lograr la especificación de los requerimientos del sistema fue necesario la entrevista con una persona que estuviera directamente ligada al tema, es por esto es que se opto por el encargado de los servidores en la Escuela de informática de la Pontificia Universidad Católica de Valparaíso, logrando captar una visión del sistema y obtener los requerimientos necesarios para que el sistema fuera un verdadero aporte al trabajo de los administradores de servidores Linux

2 Documentos Relacionados

- Anexo 1, Visión del sistema

3 Casos de Uso

3.1 Actores

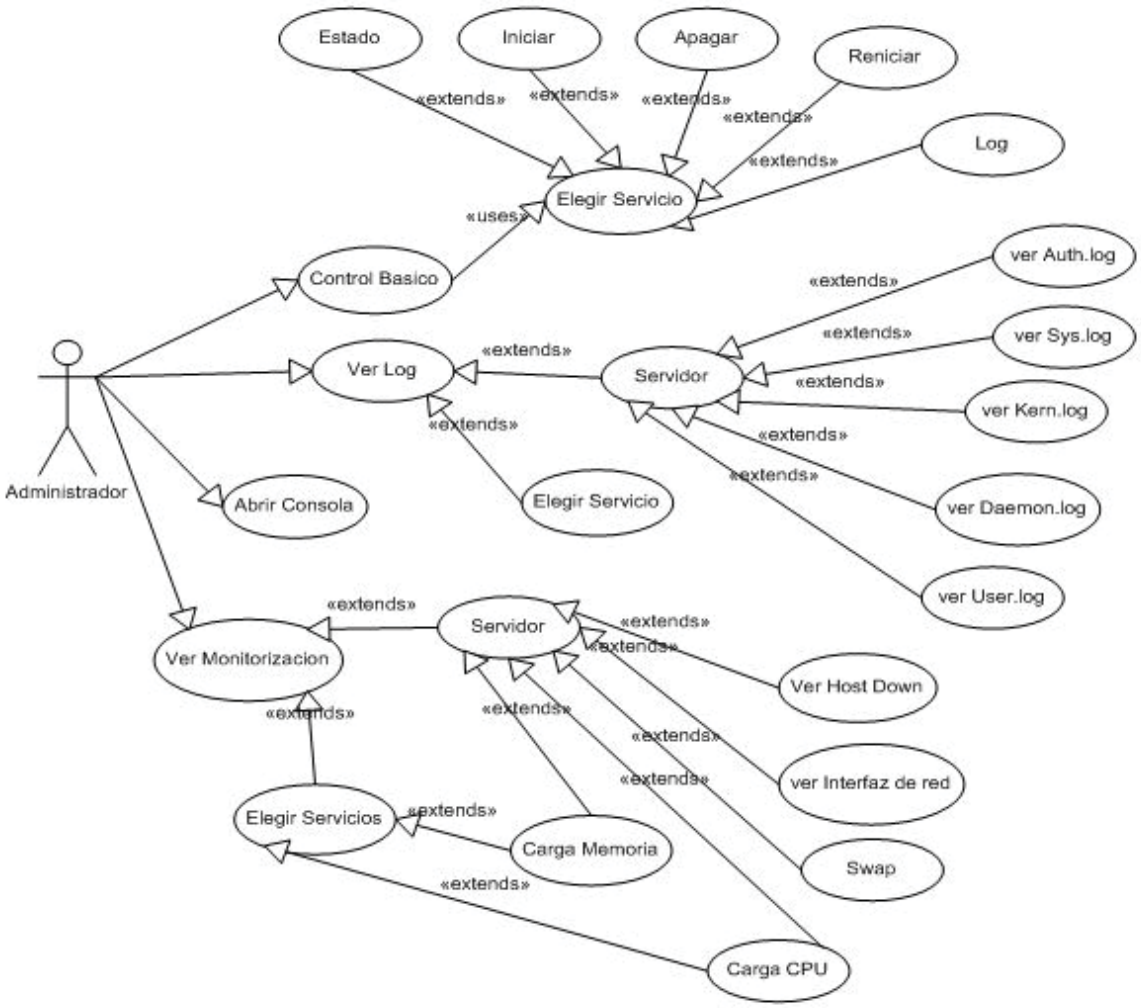
Para el caso de este sistema, se cuenta con un único actor que es llevado a cabo a su vez por una única persona, este es el Administrador, definiéndose como sigue.

Administrador: Este actor representa la persona que hace uso del sistema, realizando las tareas necesarias para que el mantenimiento y funcionamiento del servidor sea optimo, por medio del conjunto de herramientas ofrecida por el software.

3.2 Diagrama

A continuación se presentan el diagrama de casos uso y los casos de uso UML que se han identificado en el sistema, también se ha determinado la interacción de un actor.

Ilustración A5.1 Diagrama General de casos de uso.



3.3 Especificaciones de Casos de Uso

• **Control básico:** El administrador actúa sobre algún servicio prestado por el servidor, logrando así ejecutar alguna de las siguientes acciones: iniciar, detener o reiniciar.

• **Ver Log:** Los servicios prestados por el servidor y este mismo llevan registro de las actividades y movimientos dentro del uso de ellos, el administrador puede requerir el ver uno de estos para así obtener una mejor perspectiva del uso que se está dando a estos.

• **Abrir consola SSH:** El administrador desea realizar una operación directa dentro del servidor utilizando una consola de comando, en la cual se puede ingresar cualquier comando que sea necesario para interactuar directamente con el servidor.

• **Ver Monitorización:** Es de suma importancia conocer las cargas llevadas por la ejecución y/o utilización de los servicios o el mismo servidor para tomar acciones cuando se detecten cargas criticas.

4 Requerimientos

4.1 Requerimientos Funcionales

• Permitir el monitoreo de los servicios y del sistema, además de entregar mecanismo de alerta ante sucesos imprevistos.

• Permitir obtener los últimos registros (Logs) de los servicios y del servidor que los aloja.

• Permitir tener control de los servicios o servidor mediante interfaces o a través de consola de comandos.

4.2 Requerimientos No Funcionales

4.2.1 Usabilidad

Interfaz amigable, simple e intuitiva que permite una rápida adaptación de los usuarios al sistema.

4.2.2 Confiabilidad

Se debe tener presente que al comunicación entre el Midlet y el Servidor puede verse afectado por la calidad de recepción de la señal del Teléfono Celular o en el peor de los casos la pérdida total por causa de nula cobertura, por tanto, es necesario la creación de mecanismo que garanticen una comunicación fluida y fiable por medio de algún protocolo o norma de comunicación.

4.2.3 Seguridad

Entregar los mecanismos de seguridad pertinentes que resguarden la comunicación entre el servidor y el sistema residente en el teléfono móvil (Midlet).

4.2.4 Soporte y Operatividad

Hacer del sistema (Midlet) sea soportado por la mayoría de los teléfonos móviles existentes en el mercado sin afectar el desempeño de los mismos ni del sistema.

4.2.5 Restricción de Diseño

La disposición del contenido dentro de las pantallas de los teléfonos celulares no se debe ver afectada por el modelo en donde esta se visualice.

4.2.6 Requerimientos de Documentación en Línea y de Sistemas de Ayuda

Contar con manual de usuario disponible en formato digital (.pdf) como a su vez en papel.

Anexo 6

Manual de Usuario

1 Introducción

El siguiente documento presentara las diferentes opciones del sistema administración de servidores a través de un cliente móvil, permitiendo que el usuario empiece la utilización de este de forma sencilla, rápida y eficaz. El usuario final se entiende como una persona preparada y capacitada para la administración de servidores y los servicios prestados por estos, debido al carácter crítico que presenta el funcionamiento de un servidor.

Este manual de usuario esta contiene las capacidades del cliente móvil y las incidencias en el servidor que se presenten al realizar alguna acción sobre algún servicio.

El sistema desea otorgar al administrador de servidores una herramienta rápida y confiable, para así contar con información en línea y disponible en todo lugar, haciendo que la que la solución de los problemas que se puedan presentar sea de forma rápida y eficaz, logrando no afectar el normal funcionamiento del servidor y que tanto la performance como la presentación de servicios a clientes no sufra ningún inconveniente.

2 Documentos Relacionados

- Anexo 1, Visión del sistema.

- Anexo 2, Arquitectura del Sistema.
- Anexo 5, Especificación de Requerimientos del Sistema.
- Anexo 7, Manual de instalación.

3 Funcionalidades

Ilustración A6.1 Interfaz de usuario, advertencia celular.

Luego de producida la instalación del Midlet en algún celular o dispositivo móvil, esta quedara alojada en el sistema de ficheros de cada celular, es necesario recalcar que dependiendo del modelo del celular y del fabricante la ubicación puede variar, pero es normal que se encuentre en el apartado de “aplicaciones” o “juegos” del celular. Al ubicar la aplicación dentro del sistema operativo del dispositivo se procederá a la ejecución de esta, cabe recordar que para lograr la comunicación con el servidor es necesario contar con saldo para poder conectarse a la red. Luego se muestra una advertencia, diciendo que se iniciara una aplicación que utilizara internet, obviamente contestamos que sí.

Ilustración A6.2 Interfaz de usuario, menú principal.

El menú mostrado luego de la aceptación corresponde a las cuatro funcionalidades más generales del sistema.

3.1 Control Básico de Servicios

Ilustración A6.3 Interfaz de usuario, servicios instalados en el servidor.

Ilustración A6.4 Interfaz de usuario, control básico de un servicio.

Al seleccionar esta funcionalidad se presentaran los servicios instalados en el servidor a consultar y a la vez que sean soportados por el sistema. Luego de seleccionar el servicio sobre el cual se desea realizar algún tipo de acción como: inicio, apagado reinicio o ver log, se manda la orden directa al servidor y este retorna si la petición fue ejecutada con éxito. Además arriba de las opciones se muestra el estado actual del servicio, con lo cual se desactiva la opción no requerida, es decir, si el servicio está con el estado UP, se desactiva la opción Iniciar. Es importante recalcar que las operaciones de este tipo sobre algún servicio , presentan un grave riesgo cuando se están otorgando prestaciones de servicios a personas externas, por lo cual al utilizar este apartado del sistema, es necesario tener un conocimiento acabado sobre las posibles consecuencias de realizar alguna de estas acciones.

Ilustración A6.5 Interfaz de usuario, respuesta del servidor a la acción solicitada.

3.2 Monitorización

Ilustración A6.6 Interfaz de usuario, menú principal monitorización.

Presenta el estado actual de la maquina en general y de los servicios prestados por esta. Al llegar a esta pantalla del sistema se debe seleccionar si se desea monitorear el servidor en si o los servicios alojados en este.

• **Servidor:** Dentro de este apartado se pueden monitorear lo que está pasando en servidor en sí, y se puede obtener información como:

› **Carga CPU:** Presenta la carga que está sufriendo en el mismo instante de la consulta, esta es obtenida a través de comandos que interactúan directamente con el servidor.

› **Carga Memoria:** Presenta el uso de la memoria que se está realizando en el instante de la consulta.

› **Swap:** Presenta el uso de la swap que está siendo requerida por el SO y sus procesos.

› **Interfaces de red:** Muestra información detallada del estado de las tarjetas de red instaladas en el servidor.

› **Host Down:** Muestra los host que se encuentran en la red, y que se encuentran con algún problema de conectividad.

Ilustración A6.7 Interfaz de usuario, opciones de monitorización del servidor.

• **Servicios:** Al seleccionar esta funcionalidad, se presentan los servicios instalados y soportados por el sistema, luego el usuario selecciona el que desea y puede obtener información precisa sobre la el uso del procesador y memoria del servidor por parte del servicio especificado.

Esta información puede ser vital para la toma de una decisión sobre algún problema y ejecutar alguna operación de control básico. La información presentada por este apartado, es el promedio entregado por el servidor en el momento exacto de realizada la petición.

Ilustración A6.8 Interfaz de usuario, logs disponibles del servidor.

3.3 Logs

Los Logs son registros oficiales de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática un Log es usado para registrar datos o

información sobre quién, que, cuando, donde y porque (who, what, when, where y why, W5) un evento ocurre para un dispositivo en particular o aplicación.

Al presentarse esta pantalla muestra la opción de elegir saber la información del servidor como de los servicios prestados por este, para el servidor se presentan los siguientes Logs:

- /var/log/auth.log
- /var/log/syslog
- /var/log/daemon.log
- /var/log/kern.log
- /var/log/user.log

En el contexto de los servicios, solo se presentaran los Logs pertenecientes a los servicios soportados por este sistema, que son:

- Cupsys: Servidor de impresoras.
- Postgresql: Motor de BD.
- Mysql: Motor de BD.
- Exim4: Servidor de Correo o MTA.

Es necesario aclarar que se muestran los últimos 10 registros de los archivos Logs del sistema y de los servicios prestados. Esta información es registrada por el sistema operativo mostrando los posibles problemas y últimos eventos que se han presentado.

3.4 Consola (Shell)

Se presenta una consola de comandos del sistema, igual que las presentadas por el sistema operativo, para así ejecutar alguna orden que no se encuentre en las otras funcionalidades

del sistema, logrando de esta manera un completo control de la maquina a distancia, produciéndose una administración proactiva, sencilla y rápida.

Anexo 7

Manual de Instalación

1 Introducción

Este documento desea expresar la forma de obtener una instalación satisfactoria del sistema de control y monitorización de servidores Linux a través de un dispositivo móvil.

Esta instalación se puede separar en dos grandes tópicos en el Servidor y en Celular, pero antes de ejecutar la instalación del software desarrollado se deben cumplir ciertas especificaciones y además de la instalación de las dependencias requeridas para el buen funcionamiento del sistema

2 Documentos Relacionados

- Anexo 1, Visión del sistema.
- Anexo 2, Arquitectura del Sistema.
- Anexo 5, Especificación de Requerimientos del Sistema.
- Anexo 7, Manual de instalación.

3 Requerimientos Mínimos del Sistema

3.1 Servidor

Los requerimientos mínimos en lo que se respecta a Hardware son:

- Mínimo PIII 550, recomendado PIV o superior.
- RAM mínimo 128, recomendado 256 o superior.

Estos requerimientos es si es una recomendación en lo que respecta a la necesidades de los software externos que se integran con el sistema desarrollado, pero debido a las características del sistema y donde se realizara la implantación, es decir un servidor, se entiende que el hardware de la maquina superara con creces esta especificación debido al uso tan especial que se le da, por lo cual el sistema funcionara de manera optima.

3.2 Celular

Los requerimientos mínimos en lo que respecta al celular o dispositivo móvil son:

- Soporte para Java.
- Dispositivos móviles con soporte para el perfil MIDP 2.0, necesario para realizar un https/ssl.
- Acceso a internet para realizar las consultas.
- Memoria disponible.

4 Instalación

4.1 Instalación de otros Software necesarios

4.1.1 JVM

Para lograr la ejecución de sistema es necesario contar en el servidor con la maquina virtual de java, para el desarrollo fue utilizada la versión 1.5 de esta y la instalación se realizo de la siguiente forma:

```
apt-get install sun-java-5-jre
```

Si se desea utilizar otra versión de la maquina virtual es necesario bajar el ejecutable proporcionado por Sun y seguir las instrucción de instalación.

4.1.2 Apache Tomcat

Para el alojamiento de middleware que es el que logra la comunicación entre el dispositivo móvil y el servidor es necesario contar con un contenedor, para esto fue elegido el proporcionado por la Fundación apache que es Tomcat, esta elección se baso en la ya famosa estabilidad proporcionada por esta, además de que la adquisición de este es totalmente gratuita.

Debido a que el desarrollo se realizo sobre la distribución Debian GNU/Linux con la versión estable 4.0 “Etch” la instalación se realizo con la Herramienta APT proporcionada por esta ejecutando la siguiente orden:

```
apt-get install tomcat5.5 tomcat5.5-admin tomcat5.5-webapps
```

Como queda claro en el comando ejecutado la versión utilizada de Tomcat es la 5.5, si se deseara utilizar otra versión del producto de apache es necesario bajar el archivo comprimido (tar.gz) y seguir los pasos proporcionados por la documentación oficial de de apache Tomcat.

4.2 Instalación del sistema

Para realizar la instalación del sistema es necesario obtener el archive comprimido que contiene los ejecutables como el sistema, este archivo contiene un archivo escrito en bash que logra la instalación automática del sistema, y se deben ejecutar los siguientes pasos:

```
chmod +x Install

sh Install
```

Este proceso instalara los archivos necesarios tales como:

- Middleware
- Archivos que obtienen el estado del servidor

Además proporcionara la configuración y permisos necesarios para el correcto funcionamiento del sistema.

4.2.1 Aspectos de configuración

La configuración básica del sistema se logra ejecutando el archivo especificado en el punto anterior, lográndose automáticamente, pero es necesario nombrar que es lo que se cambia para que el administrador del servidor tenga presente que es lo que se ha hecho y cambiado en el sistema.

4.2.1.1 Permisos de los archivos

El sistema de permisos en Linux se basa en un esquema de usuarios/grupos que lo convierte en la base principal de la seguridad en Linux, a estos usuarios y grupos se les asignan distintos derechos sobre los archivos y directorios.

Esta es una de las características que ayudan a que Linux sea casi inmune a los Virus de computadora, los virus deben ser capaces de escribir sobre un archivo para poder infectarlo y ejecutarse de alguna manera para poder infectar mas archivos, con el sistema de permisos de Linux los virus no pueden copiarse a cualquier archivo, si el usuario carece de permisos el virus no podrá infectar más archivos y por lo tanto no podrá reproducirse.

Todos los archivos y directorios en Linux tienen permisos que verifican quien puede hacer o no alguna acción con él.

Los permisos propiamente dichos son tres:

• r: read (lectura): Cuando el permiso de lectura está activo sobre un directorio significa que se podrá listar los recursos almacenados en él, si está asignado a un archivo se podrá leer su contenido.

• w: write (escritura): Cuando el permiso de escritura está activo sobre un directorio significa que se podrá crear y borrar archivos en su interior, si esta activado para un archivo significa que se podrá modificar su contenido.

• x: execute (ejecución): Si el permiso de ejecución está activo sobre un directorio significa que el usuario podrá realizar otras funciones dentro de él mediante los otros permisos de

lectura y escritura, y si está activo sobre un archivo se podrá ejecutarlo desde la línea de comandos.

Los archivos necesarios para el funcionamiento del software se encuentran ubicados en:

```
/etc/proyectoucv/
```

Estos archivos son de propiedad del usuario root y del grupo root, y estos tiene permisos del tipo 700, es decir, que solo root puede escribir sobre el archivo, leer y ejecutar este.

```
-rwx------ 1 root root 121 2007-09-10 18:35 Logs
-rwx------ 1 root root 121 2007-09-10 16:16 RedUsage
-rwx------ 1 root root 147 2007-09-09 14:23 ServicePid
-rwx------ 1 root root 125 2007-09-09 14:25 ServiceUsage
-rwx------ 1 root root 185 2007-09-09 21:28 Status
-rwx------ 1 root root 127 2007-09-09 14:32 ServiceInstall
-rwx------ 1 root root 121 2007-09-10 16:16 ServerUsage
-rwx------ 1 root root 147 2007-09-09 14:30 Usuarios
```

4.2.1.2 Sudo

Sudo (SUperuser DO) es una herramienta que permite otorgar a un usuario o grupos de usuarios normales, permisos para ejecutar algunos comandos como root (o como otros usuarios) sin necesidad de conocer su password.

El fundamento de sudo reside en su fichero de configuración, el fichero /etc/sudoers. Este fichero tiene, en los casos más sencillos, dos partes: una parte de alias y otra parte de reglas. La parte de alias, lo que hace es "agrupar" de alguna manera listas de usuarios y listas de aplicaciones (incluso listas de máquinas de una red, pero esto es más específico y no lo explicaremos aquí). La parte de reglas define qué grupos de usuarios pueden usar qué grupos de programas con permisos distintos de los suyos y en qué condiciones pueden hacerlo.

En la parte de alias, Cmnd_Alias indica una lista de comandos (programas) que serán luego referidos mediante el nombre que le demos (asignar alias aquí tiene similitud con asignar variables de entorno en el Shell). User_Alias agrupa a una lista de usuarios bajo un mismo nombre.

En cuanto a la parte de reglas, primero se especifican los usuarios, el ALL que sigue hace referencia a en qué máquinas podrán hacer esto, y el NOPASSWD: indica que lo harán con permisos de root y sin necesidad de teclear su password. Después viene el alias con los comandos que podrán ejecutar en las condiciones que hemos dado. La configuración de sudo que se está usando actualmente es la siguiente:

```
# /etc/sudoers

# This file MUST be edited with the 'visudo' command as root.

# See the man page for details on how to write a sudoers file.

Defaults        env_reset

# Host alias specification

# User alias specification

User_Alias PROYECTO = usuario, tomcat55

# Cmnd alias specification

Cmnd_Alias PROCESARLOG = /etc/init.d/postgresql-8.1 restart , /etc/init.d/postgresql-8.1 start , /etc/init.d/postgresql-8.1 stop , /etc/init.d/exim4 start , /etc/init.d/exim4 stop , /etc/init.d/exim4 restart , /etc/init.d/cupsys start , /etc/init.d/cupsys restart , /etc/init.d/cupsys stop , /etc/init.d/mysql start , /etc/init.d/mysql restart , /etc/init.d/mysql stop , /bin/ps -A , /bin/grep , /etc/proyectoucv/Status , /etc/proyectoucv/ServiceInstall , /etc/proyectoucv/ServicePid , /etc/proyectoucv/ServiceUsage , /etc/proyectoucv/Logs , /etc/proyectoucv/Services , /etc/proyectoucv/ServerUsage , /etc/proyectoucv/RedUsage , /bin/sh , /etc/proyectoucv/Usuarios, /bin/sh, /bin/bash

# User privilege specification

PROYECTO ALL=NOPASSWD: PROCESARLOG

root    ALL=(ALL) ALL
```

Para mantener una mayor seguridad en la ejecución de los scripts, se creó un User alias llamado Proyecto y a su vez un Cmnd Alias que contiene los comandos que pueden ser

ejecutados por el alias especificado. Entonces para poder ejecutar alguno de los script creados es necesario hacer lo siguiente:

```
sudo /etc/proyectoucv/[script a ejecutar] [Argumento1] [Argumento2] ……[Argumento N]
```

4.3 Instalación del Midlet en dispositivo móvil

Para lograr la instalación de Midlet en el teléfono o dispositivo móvil es necesario descárgalo de una página wap habilitada para estos efectos. Debido a las nuevas configuraciones de los teléfonos celulares luego de descargar la aplicación esta se instalara de forma automática.

Otra de las formas de instalación es la utilización de los programas que acompañan a los teléfonos celulares, es decir, conectar el teléfono a un computador y utilizar estos programas para copiar y luego instalar la aplicación, automatizando este proceso, algunos ejemplos de estos son:

- Sony Erickson: Sony Erickson PC Suite
- Nokia: Nokia PC Suite

4.4 Desinstalar el Producto

Para lograr la desinstalación del sistema es necesario ejecutar el archivo Uninstall que se encuentra dentro de la carpeta de la siguiente forma:

```
sh Unistall
```

Anexo 8

Estudio de Factibilidad

1 Introducción

Un estudio de factibilidad técnica trata de determinar si el sistema es útil y viable para la empresa u organización, asegurando así el desarrollo y la implementación de la aplicación en proceso de construirse.

Factibilidad se refiere a la disponibilidad de los recursos necesarios para llevar a cabo los objetivos o metas señaladas, todo esto se apoya en 4 aspectos básicos: técnico, operativo, legal, y económico.

2 Factibilidad Técnica

Un análisis de factibilidad técnica evalúa si el equipo o los equipos y software están disponibles, o si realmente el software se puede desarrollar y si tienen las capacidades técnicas requeridas por cada alternativa del diseño que se esté considerando, a la vez también considera las interfaces entre los sistemas actuales y nuevos.

De acuerdo a las necesidades técnicas del sistema sería necesario lo siguiente:

- Contenedor de Servlet, Apache Tomcat 5.5 o superior.
- Servidor con Sistema operativo Linux (preferentemente Debian GNU/Linux versión 4.0 “Etch”).
- RAM mínimo 128, recomendado 256 o superior.
- Mínimo PIII 550, recomendado PIV o superior.
- Soporte para java.
- Dispositivos móviles con soporte para el perfil midp2.0, necesario para realizar un https/ssl.

3 Factibilidad Operacional

Radica en determinar la capacidad potencial de la organización para llevar a cabo el sistema en término de los planes, políticas y procedimientos vigentes, es decir, se trata de reconocer a qué se expone la organización al añadir un nuevo sistema, cuál será la reacción

que producirá en los demás procesos, en los recursos humanos y en general en toda la organización, considerando incluso los clientes.

De acuerdo a la naturaleza del sistema esto es difícil de determinar dado a que no se cuenta con un cliente formal y la vez que su utilización radica generalmente a una persona o un grupo de trabajo más bien cerrado. Pero de acuerdo a lo analizado la implementación del software no alterara en si el funcionamiento general de la organización o entidad que lo utilice, lo que si puede que afecte son los protocolos actuales en lo que respecta a la solución de problemas que se presentan en la administración de servidores, pero el impacto sería mínimo y seria más bien un aporte al mejoramientos de estos procesos.

4 Factibilidad Legal

De acuerdo con este punto del análisis en la implementación del sistema en una entidad, no representaría problemas dado la naturaleza del sistema y al uso total de herramientas libres (open source) que cuentan con licencias tales como GPL, que permiten y protegen la libre modificación, distribución y uso de los software que cuenta con este tipo de licencias, por lo tanto el uso de las herramientas que se hace utilización para el desarrollo e implementación de este software no violaría ningún marco legal vigente en el ámbito nacional en lo que corresponde a leyes informáticas. También es importante considerar las políticas internas de la organización, tratando de afectar de la menor forma posible el accionar o el desarrollo interno de la empresa y que al utilizar el sistema incurra en algún delito, en este caso en particular se deben verificar las licencias actuales del software actual.

5 Factibilidad Económica

En lo que respecta a la factibilidad económica es importante resaltar lo comentado en el punto anterior, dado a que están directamente relacionados, porque el uso de herramientas open source hace que los costos se disminuyan al mínimo en lo que respecta a la instalación e implementación del software y herramientas necesarias para el correcto funcionamiento de esta aplicación.

El costo incurrido en el desarrollo del sistema es casi nulo en lo que respecta al pago de licencias y herramientas. Por lo tanto se concluye que el sistema es económicamente factible de realizar.

Referencias

[Apache] www.apache.org

[Debian] www.debian.org/doc/

[Jacobson, 2000] Jacobson, Ivar y Booch, Grady y Rumbaugh, James Pearson Educación. Madrid. España, año 2000, Proceso Unificado de Desarrollo de Software.

[Java] www.java.com

[Larman, 2002] Larman, Graig. Uml y patrones. 2da edición. Madrid, España, año 2002.

[Nagios] www.nagios.org

[MidpSSH] www.xk72.com/midpssh

[Pressman, 2002] Pressman, Roger S. Libro: Ingeniería del software: un enfoque práctico. (5a. edicion), McGraw Hill. Madrid. España, año 2002

[Tomcat] www.tomcat.apache.org

[Zabbix] www.zabbix.org