Universidad de las Ciencias Informáticas

Facultad 3

Sistema para monitorear las aplicaciones ejecutadas en tiempo real sobre el sistema operativo NOVA.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Yasmany Avila Sarmiento

Tutor:

Ing. Maylevis Morejón Valdés

Co-Tutor:

Ing. Zénel Reyes Pérez

Junio de 2015

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yasmany Avila Sarmiento

_____

Firma del Autor

Maylevis Morejón Valdés

_____

Firma del Tutor

Zénel Reyes Pérez

_____

Firma del co-Tutor

DATOS DE CONTACTO

Nombre y apellidos del tutor: Maylevis Morejón Valdés
Institución: Universidad de las Ciencias Informáticas
Título: Ingeniero en Ciencias Informáticas
Correo electrónico: mmvaldes@uci.cu
Graduado de Ingeniero en Ciencias Informáticas de la UCI del año 2012, se desempeña actualmente como Especialista "B" en Ciencias Informáticas en el centro de desarrollo CEIGE.

Nombre y apellidos del co-tutor: Zénel Reyes Pérez
Institución: Universidad de las Ciencias Informáticas
Título: Ingeniero en Ciencias Informáticas
Correo electrónico: zenel@uci.cu
Graduado de Ingeniero Informático en 2006, se desempeña actualmente como Jefe del Departamento de Tecnología de la Facultad 3.

AGRADECIMIENTOS

Quiero agradecer a la Revolución cubana por darme la oportunidad de estudiar y superarme de forma gratuita y colocar en mi camino a todos esos educadores que desde la enseñanza preescolar me han ido instruyendo y educando, aportando cada uno un grano de arena para poder llegar a ser hoy el joven que soy, dispuesto a responder a la sociedad dando lo mejor de mí.

A mi hermosa familia por estar ahí siempre que lo eh necesitado, sobre todo a esas dos mujeres que más que mujeres son ángeles que me vieron nacer y me han visto crecer, mi abuela Ofelia y mi madre Xiomara, las palabras no son suficientes para expresar el amor que siento por las dos. A mi padre Luis que fue el que puso la semillita que me dio la vida y que a pesar de no estar todo el tiempo conmigo me ha apoyado en las decisiones que he decido tomar. A mi abuelo Cecilio quien ha sabido suplir el rol de padre en muchas ocasiones y a pesar de no hablar mucho me ha dado consejos que guían aún mis acciones y mi carácter. A mi tía Teresa y mis primas Yordelis y Yodelkis que desde pequeño han sido amigas incondicionales y me brindaron su ayuda, amor y cariño siempre que lo necesité. A mi otro padre José de quien no tengo ni una sola queja en más de 20 años que he convivido con él y que supo darme su apoyo incondicional siempre que lo necesité. A mi hermana a la que quiero y por la cual sería de darlo todo si es necesario con tal de que sea feliz, así como a mi hermanito Luisito que tantas alegrías me ha hecho sentir con sus travesuras y chistes. A mis segundos abuelos Orfelina y Amancio que me acogieron desde pequeño como si fuera otro más de sus nietos. A mi madrasta Mileidy por el cariño que me ha demostrado y por permanecer junto a mi padre y darme a ese hermanito loco que tengo. A mi tío Armando, mi primo Yuniel, mi tía abuela Eugenia, mi otra prima Sonia, a mi familia del Cerro a los que quiero muchísimo, mis primos Oscar y Leonardo, a Oclides, Carlos Alejandro, Serguei y Sergueisito, a todos los que de una forma u otra han estado ahí para mí en las buenas y las malas.

Quiero agradecer a mis tutores Maylevis y Zenel quienes me acogieron y me han guiado en esta ardua tarea y meta que representa un trabajo de diploma, el resumen de tantos años dedicados al estudio y la superación. Sobre todo reconocer el trabajo realizado por mi tutora Maylevis quien supo hacerse cargo de mi cuando fue necesario y con un nivel de profesionalidad único, por darme tantos momentos que nunca olvidaré y por ser la mejor. A ambos por impulsarme y llamarme la atención cuando fue necesario.

Quiero agradecer a todos mis compañeros de carrera, a los que ya se graduaron, Anidey, Aymeé, el Charlie, el Yoa (Yoannys), Albertaco, Alain, el Rafa, el Fifo, y a los que han compartido estos dos último años conmigo, Yanislay, Robertaco, Lise, Manuel, Rolando (El Tropa), el otro Rolando(el del nombre largo), Liannet, Gendry, Joel, Orlando, Ubesnel, a todos y los que se me quedan, muchas gracias por los momentos compartidos.

De igual forma agradecer a esa tropa que me ha acompañado en este último año, los compañeros del comité primario, Polanco, Josué, Thais, Jose, Carlos, Ariel, Pedro, Hernán, a todos muchas gracias por las alegrías compartidas en tan poco tiempo

A los profesores que durante los 5 años que pasé por esta universidad me pusieron a prueba y me instruyeron logrando inculcar en mi los mejores ejemplos y conocimientos posibles.

DEDICATORIA

Dedico esta investigación y su resultado a mis abuelos Ofelia y Cecilio quienes han sido mis segundos padres y a quienes admiro, quiero y respeto con todas las fuerzas de mi corazón y alma. A mis padres Xiomara y Luis sin quienes no podría haber llegado a este punto y a los cuales quiero y querré por siempre. A mi tía Teresa y mi prima Yordelis por todo su apoyo y ternura. A mi tutora Maylevis y a mi co-tutor Zénel. Sepan que siempre buscaré la forma de devolverles todo lo que han hecho por mí, aunque sé que no me alcanzará la vida entera para ello.

RESUMEN

La computadora constituye uno de los medios tecnológicos que más importancia ha cobrado en la vida diaria de la sociedad actual. Resulta prácticamente imposible entender la vida social sin el uso de estos equipos y sus redes. Cada vez se hace más necesario llevar un control de las actividades que se realizan en ellas, debido a incidencias provocadas por la interacción humana que afectan su integridad y resultan perjudiciales para sus propietarios.

Existe un gran número de sistemas destinados a monitorear el hardware y software de las computadoras, sin embargo no todos resultan aplicables a cualquier entorno. Los sistemas de monitoreo tienen un objetivo específico y la mayoría de ellos están atados a las condiciones y características de la situación en la cual fueron creados.

El presente trabajo muestra el proceso de desarrollo de un sistema informático que permite monitorear el uso del hardware y el software de una computadora o conjunto de ellas conectadas a través de una red, en tiempo real, (SMART). Además se presentan las herramientas y tecnologías empleadas en la solución y la situación que llevó a su creación.

Palabras claves: control, medio tecnológico, monitoreo, monitoreo en tiempo real.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

Los sistemas de control de inventario de hardware y software son herramientas que permiten a directivos y/o administradores mantener el control sobre los componentes físicos y programas de las computadoras de sus organizaciones. Dentro de sus características fundamentales se encuentra la existencia, en su mayoría, de clientes que instalados en las estaciones de trabajo de una red de computadoras obtienen la información necesaria para su posterior envío al servidor de aplicaciones, donde es procesada.

Una vez procesada la información el servidor provee una serie de reportes que permiten a los administradores monitorear cualquier incidencia ocurrida en las computadoras. Según la Real Academia Española (RAE), una incidencia es un acontecimiento que sobreviene en el curso de un asunto o negocio y tiene con él alguna conexión. Entre las incidencias relacionadas con las computadoras tenemos la sustracción o cambio de sus componentes, la instalación o desinstalación de un software, el inicio o fin de una aplicación o la interrupción de la comunicación de esta con el servidor. Sin embargo, una de las principales deficiencias de estos sistemas radica en la inmediatez con que se notifica la existencia de una incidencia, variando a lo largo del tiempo e incorporando nuevas funcionalidades a los sistemas según han ido evolucionando.

La Facultad 3 de la Universidad de las Ciencias Informáticas (UCI) tiene una infraestructura tecnológica que comprende un total de mil ciento cincuenta y siete computadoras distribuidas en oficinas, aulas y laboratorios, los cuales han sido objeto en disímiles ocasiones de sustracciones de hardware detectadas, en su mayoría, tiempo después de ocurridas. Si una incidencia vinculada a los componentes físicos de los ordenadores que se poseen es detectada con un alto desfasaje de tiempo, entre el momento de ocurrido el hecho y el momento en que se detecta, existe una baja probabilidad de que se encuentren a los responsables y se puedan evitar daños a la economía de la entidad.

Motivados por garantizar la seguridad de las computadoras y establecer un mecanismo de control, el departamento de tecnología de la facultad realizó un estudio de los diferentes sistemas de control de hardware y software existentes. Tras el análisis del estudio realizado se concluyó que no se podía hacer uso de ninguno de ellos pues no brindaban la información suficiente para mantener un control estricto sobre el hardware y el software de las computadoras de la facultad. Por tal motivo, en el

2014, se desarrolló e implantó una aplicación web (CEUS[1]) que recogía datos sobre las incidencias relacionadas a las computadoras que se monitoreaban y brindaba información útil para mantener el control sobre su hardware y software.

CEUS constituyó una potente herramienta, sin embargo, luego de un año de explotación, el departamento observó que existían otras irregularidades, las cuales ni CEUS, ni un nuevo sistema dispuesto por la universidad (GRHS)[2] solucionaban, por lo que se procedió a realizar una valoración de la permanencia o modificación de estas aplicaciones con el fin de garantizar su aporte a la gestión del departamento.

Dentro de las nuevas irregularidades se encuentran:

- La ocurrencia de algún incidente vinculado al hardware o software de las computadoras demora mucho tiempo en notificarse por lo que es necesario reducir los tiempos de respuesta. Un ejemplo lo constituye una sustracción de algún componente de una computadora, que no tiene sentido notificarse 24 horas después de ocurrido.

- Los laboratorios docentes no tienen restricciones de uso para los usuarios por lo que resulta difícil determinar quiénes han hecho uso de las PC, o ubicarlos cerca o en el lugar de algún incidente.

- Las aplicaciones instaladas en los laboratorios docentes tienen como objetivo apoyar el desarrollo académico de los estudiantes, sin embargo existe una tendencia a utilizar las computadoras en actividades de ocio que atentan contra la superación académica y causan daños al hardware de las computadoras. Por tal motivo el departamento necesita obtener información acerca del uso de las computadoras en los laboratorios docentes con el objetivo de conocer para qué son usadas realmente. ..

Ante tal situación se identifica como **problema**:

¿Cómo monitorear en tiempo real las incidencias relacionadas con las computadoras, de los laboratorios docentes de la Facultad 3, de manera que se solucionen las irregularidades encontradas?

[1] Sistema para el control de uso de los medios tecnológicos de la Facultad 3

[2] Gestión de Recursos de Hardware y Software

El **objeto de estudio** de la investigación se enmarca en el monitoreo en tiempo real.

Para dar solución al problema planteado anteriormente se define como **objetivo general:** Desarrollar una herramienta que permita monitorear en tiempo real las incidencias relacionadas con las computadoras, de los laboratorios docentes de la Facultad 3 con el fin de solucionar las irregularidades encontradas.

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

1. Fundamentar la investigación mediante la elaboración del marco teórico para sustentar los conceptos y la propuesta de desarrollo de la solución.
2. Obtener, describir y validar los requisitos que deben estar presentes en la solución que se propone desarrollar.
3. Diseñar e implementar un sistema de monitoreo de hardware y software en tiempo real que funcione sobre el sistema operativo NOVA [3] para solucionar las irregularidades detectadas.
4. Validar la solución propuesta mediante pruebas funcionales.

El **campo de acción** se centra en el monitoreo en tiempo real de incidencias relacionadas a las computadoras con sistema operativo NOVA.

La **idea a defender** es la siguiente: El desarrollo de un sistema informático para monitorear en tiempo real las incidencias relacionadas con las computadoras de los laboratorios docentes de la Facultad 3 solucionará las irregularidades detectadas.

Para dar cumplimiento a los objetivos específicos trazados se desarrollarán las siguientes **tareas de investigación:**

1. Elaboración de la fundamentación teórica de la investigación y la revisión bibliográfica acerca de temas relacionados con el monitoreo de las incidencias que tienen lugar en los ordenadores.
2. Análisis de la posibilidad de integrar herramientas de terceros para dar solución al problema que se pretende resolver.
3. Identificación de los requisitos funcionales y no funcionales de la herramienta.
4. Confección del Modelo Conceptual.

[3] Sistema operativo de software libre creado en la Universidad de las Ciencias Informáticas

5. Elaboración de las Historias de Usuarios.
6. Elaboración de las tarjetas CRC (Clase-Responsabilidad-Colaboración).
7. Identificación de los protocolos de comunicación y posibles agentes a utilizar para identificar las aplicaciones instaladas.
8. Confección del Modelo Entidad-Relación.
9. Descripción del modelo de datos.
10. Diseño e implementación de las interfaces de usuario.
11. Implementación de los requisitos descritos.
12. Diseño de casos de prueba a aplicar.
13. Aplicación de pruebas al sistema.
14. Documentación de los resultados de las pruebas realizadas.

Durante el desarrollo de la investigación se utilizaron los siguientes métodos científicos.

Métodos Teóricos:

- **Analítico-Sintético:** Para analizar teorías y documentos relacionados con los procesos que se estudian y extraer los datos necesarios.
- **Inductivo-Deductivo:** Para a partir del conocimiento que se pueda adquirir sobre la obtención de información de las incidencias relacionadas a las computadoras con sistemas operativos NOVA,DEBIAN o UBUNTU llegar a un conocimiento sobre cómo se deben hacer en todos los sistemas operativos de la familia LINUX.

Métodos Empíricos:

- **Entrevista:** Para adquirir información valiosa sobre el proceso de obtención y análisis de información sobre las incidencias relacionadas a las computadoras con sistemas operativos de la familia LINUX y poder identificar los requerimientos funcionales y no funcionales que debe cumplir el sistema que se propone desarrollar.
- **Observación:** Para la percepción selectiva de las restricciones y propiedades del sistema y para la determinación de la problemática que da origen a la investigación.
- **Modelación:** Para elaborar los artefactos que se generan durante todo el proceso de desarrollo de la solución propuesta.

Como resultado esperado se aspira a que el sistema desarrollado facilite la monitorización y control de las incidencias relacionadas con las computadoras de los laboratorios docentes de la Facultad 3 de la Universidad de las Ciencias Informáticas.

Para un mejor entendimiento de este documento, a continuación se hace una descripción de cada capítulo:

Capítulo1. Fundamentación Teórica: describe los principales conceptos de la investigación, las soluciones existentes en el ámbito nacional e internacional relacionadas con el campo de acción en el que se trabaja, así como las principales tecnologías, herramientas y lenguajes utilizados en el desarrollo de la propuesta solución.

Capítulo 2. Propuesta de solución: resume las bases del desarrollo del sistema propuesto, las funcionalidades que debe cumplir y su especificación, la arquitectura, los modelos de diseño y de datos, los patrones utilizados y los estándares de codificación empleados.

Capítulo 3. Implementación y validación del sistema propuesto: describe las métricas empleadas para validar el diseño realizado, representa y describe el diagrama de despliegue que guía la implementación de la solución propuesta, resume los estándares de codificación que se utilizan con el objetivo de hacer el código entendible ante los ojos de cualquier otro desarrollador. Por último analiza las pruebas de aceptación o pruebas funcionales aplicadas a la solución, permitiendo de esa forma comprobar que la solución desarrollada se encuentra lista y cumple con las exigencias del cliente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

En este capítulo se profundiza en los principales conceptos relacionados con el proceso de monitorización y control de aplicaciones en tiempo real. Se hace un estudio sobre sistemas de monitoreo que existen hoy en día en el ámbito internacional y nacional. Más adelante se exponen las principales características del marco de trabajo adoptado, así como los lenguajes y herramientas empleadas.

MARCO CONCEPTUAL

A continuación se explica una serie de términos asociados al objeto de estudio y campo de acción de la investigación necesarios para entender el problema que se desea solucionar:

Monitoreo:

Monitoreo es un término no incluido en el diccionario de la Real Academia Española (RAE). Su origen se encuentra en el monitor, un aparato que toma imágenes de instalaciones filmadoras o sensores y que permite visualizar algo en una pantalla. El monitor, por lo tanto, ayuda a controlar o supervisar una situación [1].

Esto permite inferir que monitoreo es la acción y efecto de monitorear, el verbo se utiliza para nombrar la supervisión o el control realizado a través de un monitor. Por extensión, el monitoreo es cualquier acción de este tipo, más allá de la utilización de un monitor [1].

Monitorización:

Según el diccionario de la lengua española la monitorización en la medicina es el control de las constantes vitales de un paciente a través de monitores. Si hablamos de redes la monitorización es la parte de la gestión de red que se ocupa de la observación y análisis del estado y el comportamiento de los recursos gestionados [2]. Por tanto, podemos decir que monitorización es el proceso mediante el cual nos aseguramos de que algo que deseamos realizar o que ocurra como esperamos suceda como tal.

Monitorización de sistemas:

Por monitorización de sistemas entendemos todo tipo de análisis periódico de un sistema de cuya información se desprenda su estado actual. Si analizamos más profundamente la definición podemos denotar que cuando nos referimos a análisis periódico queremos decir un escaneo del sistema que se repite cada cierto intervalo de tiempo. Este escaneo puede realizarse de forma automática, mediante software especializado, o de forma manual, es decir, un técnico se conecta al sistema a analizar y hace mediciones de diferentes parámetros para evaluar su estado [3].

Monitoreo en Tiempo Real:

Como se ha podido apreciar cuando se habla de monitorear está haciendo referencia a la acción de controlar o supervisar una situación determinada. Si se habla de monitoreo en tiempo real se está haciendo referencia a que los datos provenientes del monitoreo llegan al encargado de procesarlos con una diferencia de tiempo desestimable respecto al momento en que fueron producidos.

Para tener una mejor perspectiva sobre el tema sería bueno analizar qué es un Sistema de Tiempo Real o STR por sus siglas en español. El Diccionario Oxford de Computación ofrece la siguiente definición de sistema en tiempo real:

Un sistema en tiempo real es cualquier sistema donde el tiempo en que se produce su salida no es significativo. Por lo general debido a que la entrada corresponde a algún instante del mundo físico y la salida tiene relación con ese mismo instante. El retraso transcurrido entre la entrada y la salida debe ser lo suficientemente pequeño para considerarse una respuesta puntual [4].

Cuando en este trabajo se habla de monitoreo de aplicaciones en tiempo real se hace referencia a que el sistema será capaz de producir una salida en un tiempo considerablemente pequeño. El tiempo de procesamiento de los datos será ínfimo por lo que el cliente tendrá acceso a la información casi al instante.

Los siguientes términos han sido extraídos del diccionario de la Real Academia Española:

Control: Comprobación, inspección, fiscalización o intervención sobre una cosa.

Local: Sitio cercado o cerrado y cubierto.

Computadora: Máquina electrónica, analógica o digital, dotada de una memoria de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas matemáticos y lógicos mediante la utilización automática de programas informáticos.

Incidencia: Un acontecimiento que sobreviene en el curso de un asunto o negocio y tiene con él alguna conexión.

Aplicación: Programa preparado para una utilización específica, como el pago de nóminas, formación de un banco de términos léxicos, etc.

ANÁLISIS DE SOLUCIONES EXISTENTES

En la actualidad sistemas informáticos son creados con el objetivo de monitorizar el comportamiento de aplicaciones informáticas, redes y otros sistemas, cambiando constantemente y adaptándose a todo tipo de plataforma. Diversas arquitecturas y funcionalidades son desarrolladas, todo en dependencia de lo que desea el cliente. Muchos de esos sistemas monitorizan una o varias aplicaciones específicamente, otros monitorizan todo lo que sucede en el sistema operativo. Algunas arquitecturas como cliente-servidor se han vuelto más populares debido al incremento de las redes y a que su estructura es sencilla. La arquitectura cliente-servidor se basa en un agente instalado en los clientes que recoge datos y los envía a un servidor que los procesa y convierte en información que ayuda a los directivos y/o administradores a tomar decisiones en su ámbito profesional.

En el ámbito internacional.

CONKY

Conky es un software libre que corre sobre las X[4] en Linux y es una BSD (*Berkeley Software Distribution (Distribución de Software de Berkeley*[5])). Desde su creación, Conky ha cambiado significativamente manteniendo al mismo tiempo la simplicidad y flexibilidad de configuración. Conky puede mostrar casi cualquier cosa, ya sea en el escritorio de la raíz o en su propia ventana. Conky no solo tiene muchos objetos integrados, también puede mostrar casi cualquier información mediante el uso de scripts y otros programas externos [5].

[4] X: interfaz gráfica multiplataforma para gestionar las ventanas en el sistema operativo.

[5] Berkeley: hace referencia a la Universidad Berkeley de California, Estados Unidos.

Características:

- Conky tiene más de 300 objetos construidos, incluyendo soporte para una gran cantidad de estadísticas del sistema operativo como el nombre del usuario, tiempo de actividad, uso del CPU[6], uso de la memoria y del disco, la estadística de los procesos con el comando "top", monitoreo de la red entre otras.

- Incorpora soporte IMAP[7] y POP3[8], incluye soporte para muchos reproductores de música como MPD[9], XMMS2[10], BMPx [11]y Audacious.[12]

- Puede ser extendido usando soporte para construir en el lenguaje de programación Lua o cualquier script o programa.

- Es construido en enlaces Imlib2 y Cairo para dibujo arbitrario con Lua [5].

PHPSYSINFO

PhpSysinfo es un script php que muestra información de monitoreo sobre sistemas basados en Linux. Muestra información en los sistemas Linux como la versión del kernel (núcleo), el tiempo de actividad, la red, disco y uso de memoria, información sobre el CPU y el modelo, IDE[13] conectado y dispositivos SCSI[14], tarjetas PCI[15], salidas de los Im-sensors[16] en arquitecturas que los soportan. También soporta múltiples lenguajes y temas [6].

[6] Unidad Central de Procesamiento.

[7] Protocolo de Acceso a Mensajes de Internet.

[8] Protocolo de Oficina Postal.

[9] Music Player Daemon (Demonio Reproductor de Música) .

[10] X-platform Music Multiplexing System 2 (Sistema de Multiplexión de Música para plataforma X).

[11] Reproductor Multimedia de Audio.

[12] Reproductor de Audio.

[13] Entorno Integrado de Desarrollo.

[14] SCSI: acrónimo en inglés de Small Computer System Interface (Interfaz de Sistema para Pequeñas Computadoras)

GNOME SYSTEM MONITOR

Gnome System Monitor es una utilidad de monitoreo del sistema GNOME, con una sencilla interfaz que permite hacer un seguimiento detallado de los procesos activos, gracias a su estructura en arborescencia. Gnome System Monitor muestra los procesos activos de tal forma que se puede localizar fácilmente los procesos dependientes o hijos, es decir los que derivan directamente de otros.

También cabe la posibilidad de ocultar determinados procesos, acceder a gráficas históricas del uso de la memoria, CPU y demás. Evidentemente, permite ejecutar los comandos "kill" o "renice" para cerrar o reiniciar un proceso, además de ofrecer todas las opciones habituales a la hora de trabajar con procesos del sistema.

MUNIN

El paquete munin permite monitorizar diversos parámetros de uno o varios sistemas, por ejemplo, el estado y carga de los servicios, temperaturas, espacio en disco, etc. Munin funciona según una arquitectura cliente-servidor, en que un sistema servidor recoge y organiza datos recibidos de varios sistemas clientes. Para monitorizar sólo un servidor, debe instalarse y configurarse los paquetes cliente y servidor en el mismo sistema. El paquete munin utiliza una interfaz web, por tanto, es necesario instalar un servidor http [7].

SISTEMA DE MONITOREO HYPERIC HQ

Hyperic administra y monitorea las aplicaciones e infraestructura, el control radica en verificar constantemente el óptimo rendimiento y notifica inmediatamente la degradación del rendimiento o la no disponibilidad del servicio. A través de una variedad de condiciones de alertas avanzadas y flexibles, acciones preparadas, esquemas de escalación de TI y equipos de operación web, brinda la sensación de confianza de que las aplicaciones e infraestructura se encuentran trabajando de manera óptima.

[15] Peripheral Component Interconnect (Interconexión de Componentes Periféricos)

[16] Programas que muestran datos como el estado del hardware de la computadora, la temperatura de la placa, microprocesador, gráficas, discos duros y las revoluciones de los ventiladores.

Hyperic HQ es una plataforma basada en Java para monitorear y administrar recursos de software. Los elementos clave de la arquitectura son: los servidores HQ, que proporcionan una administración y persistencia centralizada, y el Agente HQ, que facilita el monitoreo y control por plataforma [8].

Hyperic es una aplicación de monitoreo y gestión del rendimiento para infraestructuras virtuales, físicas y en la nube. Auto descubre los recursos de más de 75 tecnologías, incluyendo vSphere, y recoge la disponibilidad, el rendimiento, la utilización y métricas de rendimiento [9].

En el ámbito nacional.

A continuación se describen soluciones desarrolladas en nuestra universidad y que han sido utilizadas por el Departamento de Tecnología de la Facultad 3.

CEUS

CEUS recopila toda la información del tiempo que se estuvo utilizando un medio tecnológico y las aplicaciones ejecutadas en los mismos, haciendo uso de agentes remotos que se encargan del envío de esta información para su posterior procesamiento por parte del sistema. El procesamiento de los ficheros enviados por los agentes puede realizarse tanto de forma puntual como automática dependiendo de las necesidades del usuario. Permite generar reportes sobre las incidencias detectadas con un intervalo de tiempo de 24 horas desde ocurridos los eventos. Posee un sistema de control de acceso a la información bastante seguro.

GRHS

Realiza inventarios de piezas y programas instalados en una red de computadoras. GRHS permite que el usuario defina los cambios de componentes que constituyen incidencias. Envía notificaciones de incidencias mediante correos electrónicos, SMS y mensajería instantánea. Ejecuta acciones automáticas en las estaciones involucradas en las incidencias. Todas sus aplicaciones están diseñadas usando la ingeniería basada en componentes. Está escrita en tecnologías libres como Python, Django y PostgreSQL. Para la comunicación entre aplicaciones usa el protocolo HTTP en conjunto con REST. Está diseñado para las plataformas Microsoft Windows y GNU/Linux. Permite su aplicación en varios entornos con características específicas por su flexibilidad de configuración. La modularidad de sus aplicaciones permite adicionar nuevas funcionalidades y desactivar algunas de las existentes. Cuando ocurre una incidencia vinculada al hardware de una computadora de las que

monitorea no notifica inmediatamente a los interesados, sin embargo posee un sistema de control de acceso que permite gestionar quiénes pueden ver la información que se recoge.

La siguiente tabla muestra un resumen del estudio realizado, resaltando los indicadores que se necesita cumpla cada una de estas herramientas pues resultan clave para responder a las funcionalidades requeridas.

Tabla 1.1 Análisis del estudio realizado sobre los sistemas existentes.

	CONKY	**PHPSYSINFO**	**GNOME SYSTEM MONITOR**	**MUNIN**	**HYPERIC - HQ**	**CEUS**	**GRHS**
Muestra estadísticas sobre el hardware de la computadora donde corre.	x	x	X	x	x	x	x
Muestra estadísticas sobre las aplicaciones que son ejecutadas en las computadoras donde corren.	x	x	X	x	x	x	x
Posee una arquitectura cliente-servidor.				x	x	x	x
Tiene definido un sistema de control de acceso, seguro y eficiente, a la información.						x	x
Posee gráficos e interfaces sencillas y fáciles de entender.						x	x
Tiene definido un sistema de notificaciones y alertas en tiempo real ante incidentes vinculados a las computadoras que monitorea.							

Como se puede apreciar los sistemas CEUS y GRHS son los más completos ya que cumplen con la mayoría de las características deseadas, exceptuando la más importante, que es la correspondiente al monitoreo en tiempo real. Ante esta situación se decide desarrollar un nuevo sistema que posibilite la monitorización en tiempo real de las aplicaciones ejecutadas y las incidencias ocurridas en las computadoras de los laboratorios docentes de la facultad.

METODOLOGÍA DE DESARROLLO DE SOFTWARE

XP (Extreme Programming)

La Programación Extrema (PE) es una metodología ágil de desarrollo de software donde se vincula al cliente durante el proceso de desarrollo y que resulta muy útil si se desea construir un software mediante entregas frecuentes, funcionales, probadas y documentadas correctamente. Según grandes fuentes de información sobre Ingeniería de Software como el Instituto de Ingeniería de Software (Software Engineering Institute), los métodos ágiles son igual de robustos que las métodos tradicionales y pueden ser aplicados en soluciones seguras y confiables.

XP utiliza un enfoque orientado a objetos como su paradigma de desarrollo preferido. La PE abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades del marco de trabajo: planeación, diseño, codificación y pruebas [10]. En cada una de esas actividades se incluye al cliente permitiendo la agilidad del proceso y la retroalimentación directa garantizando cumplir las expectativas de quien usará el software.

La elección de la metodología de desarrollo de software XP para guiar el proceso de desarrollo del nuevo sistema tuvo su basamento en la necesidad de vincular al cliente durante el proceso de desarrollo e ir construyendo el software mediante entregas frecuentes, funcionales y probadas adecuadamente. Otros aspectos que influyeron en su selección son: el equipo de desarrollo solo cuenta con un desarrollador, XP propone la generación de una cantidad pequeña de artefactos, y es necesario una variante flexible y adaptable a los cambios que puedan surgir a medida que se avanza en el desarrollo.

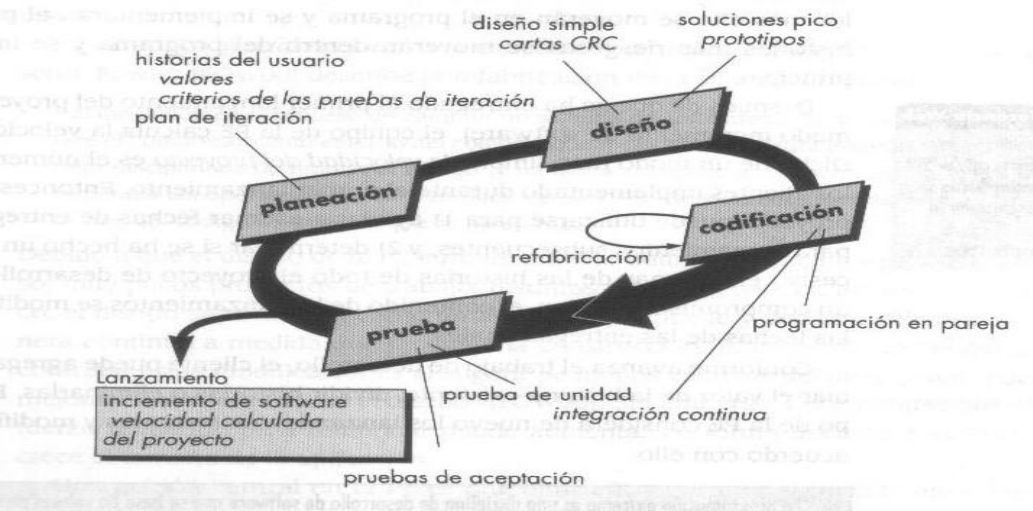


Figura 1.1 Las cuatro actividades de la Programación Extrema [10].

"La programación extrema es una disciplina de desarrollo de software que se basa en valores de simplicidad, comunicación, retroalimentación y audacia".

Ron Jeffries

MARCOS DE TRABAJO

Symfony2

Un marco de trabajo o framework en inglés se puede definir como un conjunto de componentes que facilita y agiliza el desarrollo de sistemas. El término se refiriere a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Esta una aplicación genérica incompleta y configurable a la que se le añade las últimas piezas para construir una aplicación concreta [11].

Al utilizar un marco de trabajo se acelera todo el proceso de desarrollo, se reutiliza código y como un marco promueve buenas prácticas de desarrollo de software los productos informáticos tienen un acabado eficiente.

Para el desarrollo de la aplicación web encargada de mostrar la información obtenida se decidió usar el marco de trabajo Symfony2 en su versión 2.5 debido a las grandes facilidades que brinda el mismo.

Este framework fue creado con el objetivo de desarrollar sitios y aplicaciones web con lo último en cuanto a los componentes que utiliza.

Symfony2 fue anunciado por primera vez a principios de 2009 (http://www.symfony.es/2009/03/06/asi-seran-las-novedades-de-symfony-20/) y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores.

Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto [12].

Las mejores aplicaciones PHP son construidas sobre componentes reusables como Drupal, phpBB y eZ Publish[17]. Existe una amplia comunidad de fans comprometidos a llevar PHP al nivel más alto posible. Es gracias a ello que el marco de trabajo se mantiene en constante actualización y cambio brindando siempre lo último en tecnologías de desarrollo de software orientado a la web.

Las herramientas y lenguajes que se utilizan al programar sobre Symfony son varias, ejemplo: HTML, PHP, CCS, JavaScript, Twig, YAML, SQL, Doctrine; y todas ellas, en el caso de Symfony 2.5, en su versión más reciente. La comunidad contribuye consustancialmente al desarrollo de Bundles que son muy útiles y que facilitan el trabajo y ahorran tiempo y recursos, logrando así que el proceso de desarrollo sea ágil pero robusto al mismo tiempo debido a las buenas prácticas de desarrollo de software que fomenta.

Bootstrap

Bootstrap brinda una base precodificada de HTML y CSS para armar el diseño de una página o una aplicación web, y al ofrecerse como un recurso de código abierto es fácil de personalizar y adaptar a múltiples propósitos. Permite crear interfaces que se adapten a los diferentes navegadores, tanto escritorio como tablets y móviles a distintas escalas y resoluciones. Se integra perfectamente con las principales librerías JavaScript, por ejemplo jQuery. Es un marco de trabajo ligero que se integra de forma limpia. Funciona con todos los navegadores, incluido Internet Explorer [13].

[17] Sistemas de Gestión de Contenidos.

LENGUAJES Y HERRAMIENTAS

Lenguaje de Modelado:

Un lenguaje de modelado provee un vocabulario y conjunto de reglas centradas en la representación conceptual y física de un sistema [14].

El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software [15].

Se decide usar UML como lenguaje de modelado porque mediante su empleo resulta fácil especificar, documentar, visualizar y comprender los artefactos que se generan en todo el proceso de desarrollo del software. El modelo conceptual realizado, las tarjetas CRC que describen las responsabilidades de las clases del dominio, el modelo entidad relación que describe el modelo de datos, así como el diagrama de despliegue de la solución fueron generados empleando UML.

Herramienta CASE (Computer Aided Software Engineering (Ingeniería de Software Asistida por Computadoras)):

Una herramienta CASE es aquella en la que se apoyan los desarrolladores para crear los diagramas que ayudan a la comprensión y análisis del trabajo tanto por parte del cliente como del equipo que creará la aplicación que se desea obtener [16].

Visual Paradigm (VP) es una herramienta CASE multiplataforma que contribuye al desarrollo de sistemas de software fiables, mediante un enfoque orientado a objetos. Su preferencia está determinada por las oportunidades que ofrece para la construcción de aplicaciones de calidad, con mayor rapidez y menor costo. Soporta el ciclo completo de desarrollo de software y permite su documentación en diferentes formatos, empleando UML como lenguaje de modelado [17].

Durante el proceso de desarrollo se empleó VP para el modelado de los artefactos propuestos por la metodología XP, haciendo uso de UML. Además su empleo facilitó la generación de la base de datos a partir del modelo de datos con la agregación de una librería llamada postgresql-9.1-901.jdbc4. Los prototipos de interfaz de usuario empleados en la validación de los requisitos funcionales que debe cumplir el sistema fueron generados haciendo uso de esta herramienta.

PHP 5

PHP (acrónimo de "PHP: Hypertext Preprocessor (Procesador de Hypertexto)") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Al nivel más básico, PHP puede hacer cualquier cosa que se pueda hacer con un script CGI[18], como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies. [16]

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente: [18]

- Adabas D Ingres Oracle (OCI7 and OCI8)
- dBase InterBase PostgreSQL
- Empress FrontBase Solid
- FilePro mSQL Sybase
- IBM DB2 MySQL Velocis
- Informix ODBC Unix dbm

PHP también soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos [18]. Su empleo resultó vital en el desarrollo de la aplicación web encargada de mostrar la información procesada por el sistema.

Doctrine 2:

Doctrine 2 es un asignador objeto relacional (ORM) para PHP 5.3.0+ que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (DBAL por DataBase Abstraction Layer). La principal tarea de los asignadores objeto relacionales es la traducción transparente entre objetos (PHP) y las filas relacionales de la base de datos [19].

[18] "Interfaz Común de Compuerta", es una norma para constituir una interfaz entre aplicaciones y servidores.(39)

Una de las características clave de Doctrine es la opción de escribir las consultas de base de datos en un dialecto SQL propio orientado a objetos llamado Lenguaje de Consulta Doctrine (DQL por Doctrine Query Language), inspirado en Hibernates HQL. Además DQL difiere ligeramente de SQL en que abstrae considerablemente la asignación entre las filas de la base de datos y objetos, permitiendo a los desarrolladores escribir poderosas consultas de una manera sencilla y flexible [19].

Su empleo permitió gestionar los datos guardados en la base de datos desde la aplicación web, haciendo uso del lenguaje php5. En el sistema son las clases controladoras las que hacen uso de Doctrine para poder realizar la mayoría de las acciones que les son solicitadas.

JavaScript:

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios [18].

Su empleo posibilitó la validación de los datos introducidos en los formularios utilizados para recoger información. Además el sistema muestra alertas de forma dinámica sobre las incidencias detectadas gracias al uso de código javascript.

HTML5:

HTML es un lenguaje de composición de documentos y especificación de líneas de hypertexto que define la sintaxis y coloca instrucciones especiales que son mostradas por el navegador, aunque sí le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. HTML también le indica cómo hacer un documento interactivo a través de ligas especiales de hypertexto, las cuales conectan diferentes documentos –ya sea en su computadora o en otras- así corno otros recursos de Internet, corno FTP y Gopher [20].

Los desarrolladores de navegadores dependen del estándar de HTML para programar el software que da formato y despliega documentos de HTML comunes. Los creadores de páginas utilizan el estándar para asegurarse de que sus documentos de HTML son correctos y efectivos [20].

HTML5 resulta ser la versión que utiliza Symfony2 y la más reciente de ellas [20]. Su empleo resultó vital en el sistema, no solo para generar las vistas con las que interactúa el usuario, sino también para poder hacer uso del protocolo websockets (este protocolo es abordado más adelante en el documento).

YAML:

Los archivos de configuración de Symfony2 se pueden escribir en PHP, en XML o en YAML. Desde el punto de vista del rendimiento no hay ninguna diferencia entre los tres, ya que todos ellos se transforman a PHP antes de ejecutar la aplicación.

El formato PHP es la forma más complicada de escribir la configuración de la aplicación, pero es la que más flexibilidad proporciona. El formato XML es el más largo de escribir, pero tiene la ventaja de que la configuración se puede validar antes de ejecutar la aplicación gracias a los esquemas definidos con XSD [12].

YAML es probablemente el formato más equilibrado, ya que es mucho más conciso que XML y es bastante flexible (permite ejecutar código PHP mientras se procesa el archivo YAML). Su gran desventaja es que no se puede validar automáticamente, por lo que la mayoría de los errores sólo puedes descubrirlos al ejecutar la aplicación. Los archivos de configuración de Symfony2 sólo utilizan una parte muy pequeña de todo el estándar YAML, por lo que es muy sencillo aprender su sintaxis [12]. En el sistema se evidencia el uso de este lenguaje en las vistas con las que interactúa el usuario.

Python

Python es un lenguaje de programación muy potente, simple y fácil de usar. Su sintaxis es simple, muy fácil de comprender, clara y sobre todo sencilla. Cuando se programa en Python no es necesario declarar el tipo de dato de una variable determinada, esto es llamado: Tipado dinámico [21].

Debido a la gran cantidad de librerías disponibles con que Python cuenta en la actualidad y la potencia del lenguaje se hace sencillo y rápido desarrollar aplicaciones con gran eficacia y eficiencia.

Ejemplos de éxito podemos verlos en Google, Yahoo, la NASA, y todas las distribuciones de Linux en las que un gran porciento de las aplicaciones son desarrolladas con Python [21].

Sobre todo Python es multiplataforma, aspecto que resulta muy útil si se desea que el software se pueda ejecutar indistintamente de la plataforma sobre cualquier plataforma. Algunas plataformas como UNIX, Solaris, Linux, DOS, Windows, OS/2 y Mac OS poseen librerías desarrolladas especialmente para ellas, si no se hace uso de una librería que sea específicamente para una plataforma determinada, el script o programa puede ser ejecutado en cualquier sistema operativo [21].

También es válido destacar que Python está orientado a objetos o sea que los principales conceptos que se tratan dentro del negocio que se informatiza están representados como clases y objetos. Al ser orientado a objetos se desarrollan interacciones entre estos lo que describe el funcionamiento del sistema que se crea. Los componentes del sistema encargados de obtener los datos en las computadoras clientes y procesarlos en el servidor fueron desarrollados haciendo uso de Python.

Entornos de Desarrollo Integrado (IDE): PhpStorm 8.0 y Pycharm 3.4

PhpStorm cuenta con un editor de código php que infiere el código y entiende profundamente su estructura; soporta todas las características del lenguaje php para proyectos modernos, proporciona un autocompletado limpio y buena refactorización. Mientras se escribe cientos de inspecciones se encargan de verificar el código analizando todo el proyecto [22].

Con PhpStorm se pueden realizar muchas tareas rutinarias desde el IDE, gracias a la integración de sistemas de control de versiones, el apoyo a la implementación remota, bases de datos / SQL, herramientas de línea de comando, Vagrant, Compositor, REST Client, y muchas otras herramientas [22].

Todas las tecnologías de front-end de vanguardia son incluidas, como HTML5, CSS, Sass, Menos, Stylus, CoffeeScript, Letra de imprenta, Emmet y JavaScript, con refactorizaciones, depuración y pruebas unitarias. LiveEdit permite ver los cambios al instante en el navegador [22].

Depuración Cero-configuración hace que sea muy fácil de depurar y debuggear las aplicaciones PHP, ya sea con Xdebug o el depurador Zend. El soporte de PHPUnit y Behat permite desarrollar y ejecutar pruebas desde el IDE [22].

Este IDE es multiplataforma y mediante la vinculación de un plugin se puede hacer uso del marco de trabajo Symfony2. Por las características expuestas se decide hacer uso de este IDE para el desarrollo del sistema.

Pycharm es un potente IDE destinado a la programación mediante Python facilitando el trabajo a la hora de producir y compilar código. Además de las características propias de Python, Pycharm soporta JavaScript, CoffeeScript, TypeScript, HTML/CSS, Cython, lenguaje de plantillas, AngularJS y Node.js. Las facilidades que brinda lo convierten en la principal opción a la hora de elegir una herramienta que permita programar utilizando el potente lenguaje de programación Python. Su elección se basó en las facilidades que brinda, sobre todo el autocompletado que brinda facilitando el proceso de implementación con Python.

Servidor Web: Apache2

Apache es un servidor web de código libre para la transferencia de hipertextos (Hypertext Transfer Protocol, HTTP por sus siglas en inglés) para plataformas Unix, Windows, y Macintosh. Su implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada [23].
Principales características:

- Es un servidor de web conforme al protocolo HTTP/1.1
- Soporta tanto host basados en IP como host virtuales.
- Apache soporta autentificación básica basada en la Web.
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.
- Personalización de las respuestas ante los posibles errores que se puedan dar en el servidor. [12]

Por las características expuestas se decide su empleo como servidor web.

Gestor de Bases de Datos: PostgreSQL

PostgreSQL es un sistema gestor de bases de datos objeto-relacional (ORDBMS por sus siglas en Inglés) basado en Postgres, versión 4.2. Desarrollado en la Universidad de California en Berkeley por el Departamento de Ciencias de la Computación. Postgres es pionero en muchos conceptos que sólo estuvieron disponible en algunos sistemas de bases de datos comerciales mucho más tarde.

PostgreSQL es un descendiente de código abierto del código original de Berkeley. Es compatible con una gran parte del estándar SQL y ofrece muchas características modernas como:

- Consultas complejas
- Claves externas
- Disparadores
- Vistas
- Integridad transaccional
- Control de concurrencia multiversión

También, PostgreSQL puede ser ampliado por el usuario de muchas maneras, por ejemplo mediante la adición de nuevo

- Tipos de datos
- Funciones
- Operadores
- Funciones agregadas
- Métodos de índice
- Lenguajes procedurales

Y debido a la licencia liberal, PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier fin, ya sea privado, comercial o académico [23].

Su elección se basó principalmente en las características expuestas pues permite gestionar la información guardada en la base de datos de una forma segura y eficiente. PostgreSQL es uno de los SGBD más empleados por los desarrolladores web alrededor del mundo y posee una amplia comunidad de desarrolladores que poco a poco han realizado aportes a su desarrollo y evolución desde su creación.

MENSAJERÍA

Para la comunicación y envío de la información se decidió hacer uso de Websockets y el protocolo AMQP.

El protocolo avanzado de encolamiento de mensajes (**AMQP**) es un estándar abierto para el intercambio de mensajes entre aplicaciones empresariales u organizaciones. Conecta sistemas, alimenta los procesos de negocio con la información que necesitan y transmite las instrucciones que se necesitan para lograr lo que se desea.

AMQP conecta a través de:

- Organizaciones - aplicaciones en diferentes organizaciones.
- Tecnologías - aplicaciones en diferentes plataformas.
- Tiempo - sistemas no tienen que estar disponibles de forma simultánea.
- Espacio - opera de manera confiable a distancia, o por redes de pobre infraestructura.

AMQP fue diseñado para brindar seguridad, confiabilidad, interoperabilidad y estándares [24]. Su empleo permitió la comunicación entre los componentes encargados de obtener la información en las computadoras clientes y con ello detectar las incidencias que ocurren en estas en tiempo real.

Websockets:

La especificación WebSocket fue desarrollada como parte de la iniciativa HTML5 e introdujo la interfaz WebSocket JavaScript que define una conexión en ambos sentidos única y completa sobre la cual se pueden enviar mensajes entre el cliente y el servidor. Este estándar simplifica mucho la complejidad en la comunicación web bidireccional y gestión de la conexión [25]. Websocket representa el siguiente paso en la comunicación web en comparación con otras tecnologías ya existentes hasta el momento.

La utilización de los websockets en este trabajo permite el acceso en tiempo real a la información sobre los eventos que se monitorean.

CONCLUSIONES PARCIALES DEL CAPÍTULO

La descripción de términos asociados al objeto de estudio y campo de acción de la investigación permitió lograr un mejor entendimiento del problema que se desea solucionar. Debido a las características propias del equipo de desarrollo y la necesidad de obtener un producto en un corto período de tiempo se decidió dirigir el proceso a través de las actividades propuestas por la metodología de desarrollo de software XP. El equipo de desarrollo optó por el empleo de los marcos de trabajo Symfony2 y Bootstrap con el objetivo de agilizar la implementación de la solución, ahorrar recursos y realizarla en el menor tiempo posible. Además se definieron un conjunto de herramientas y tecnologías que han de apoyar y servir de base al proceso de desarrollo del nuevo sistema que se desea obtener.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

INTRODUCCIÓN

En este capítulo se hace una propuesta general del sistema a desarrollar, luego de haber realizado un análisis de cómo debería ser el proceso de monitoreo en tiempo real de las aplicaciones ejecutadas y las incidencias ocurridas en las computadoras de los laboratorios docentes de la Facultad 3. Se define el modelo del dominio de la aplicación, se identifican los requisitos o funcionalidades que debe ser capaz de desarrollar el sistema de forma tal que se comporte según se espera y que el rendimiento general del producto sea de acuerdo a las necesidades reales de la situación que se quiere resolver.

Se realiza una descripción detallada de la solución propuesta a través de los artefactos que se generan durante las dos primeras actividades de la metodología de desarrollo de software XP (planeación y diseño). En la actividad de planeación se describen las “Historias de Usuario” (HU) que muestran los requisitos funcionales descritos por el cliente brindando una idea de cuál es la prioridad de cada uno de ellos; y constituye el primer artefacto generado de acuerdo a esta metodología. Se crea un “Plan de Iteraciones” con el objetivo de planificar el desarrollo del producto agrupando las historias de usuario según orden y prioridad con que serán implementadas.

Durante la planeación también se hace un “Plan de Entregas” con la propuesta de liberación de las versiones de la aplicación. En la actividad de diseño se describen las clases que forman el sistema, sus responsabilidades y cómo colaboran con las demás, mediante “Tarjetas Clase-Responsabilidad-Colaboración” (CRC), y se realiza el diseño de la base de datos.

De igual forma se describen los principales patrones que se emplearon en el desarrollo de la solución propuesta y cómo influyen estos en la calidad y eficiencia de las respuestas ante determinados eventos.

PROPUESTA DE SISTEMA

La solución propuesta ha de componerse de dos partes, una parte cliente y otra servidor. La primera es la encargada de obtener datos en las computadoras a monitorear y la segunda de procesar esos datos y convertirlos en información que el usuario sea capaz de entender y manejar.

El sistema debe detectar las incidencias relacionadas con el hardware y el software de las computadoras que se monitorean y mostrar la información referente en tiempo real. Para una mejor organización se podrá agrupar las computadoras por locales gestionados por usuarios con el rol de administrador. Cada incidencia relacionada con el software estará vinculada a una sesión determinada e incluirá la fecha en que se produjo.

Ha de contar con una serie de funcionalidades entre las que destacan listar y cerrar las aplicaciones que se encuentren en ejecución. Con la solución a desarrollar se genera información que resulta útil a la hora de tomar decisiones vinculadas a las computadoras que posee una entidad.

MODELO CONCEPTUAL

El modelo conceptual constituye una de las vías existentes de representar el conocimiento. Se puede describir como una red de objetos o conceptos donde los nodos de la misma los representan los conceptos y los enlaces entre ellos son las relaciones entre esos conceptos. Se trata de obtener el esquema conceptual de la base de datos a partir de los objetos y asociaciones identificadas durante la actividad de planificación. La representación debe ser no redundante y fácil de entender.

En la Figura 2.1 se puede apreciar la estructura del sistema, los principales conceptos implicados en el proceso de monitoreo y control de las aplicaciones y cómo estos se relacionan unos con otros. También se representa el sistema de acceso a la información mediante la gestión de usuarios y los roles que puedan ser asociados a estos.

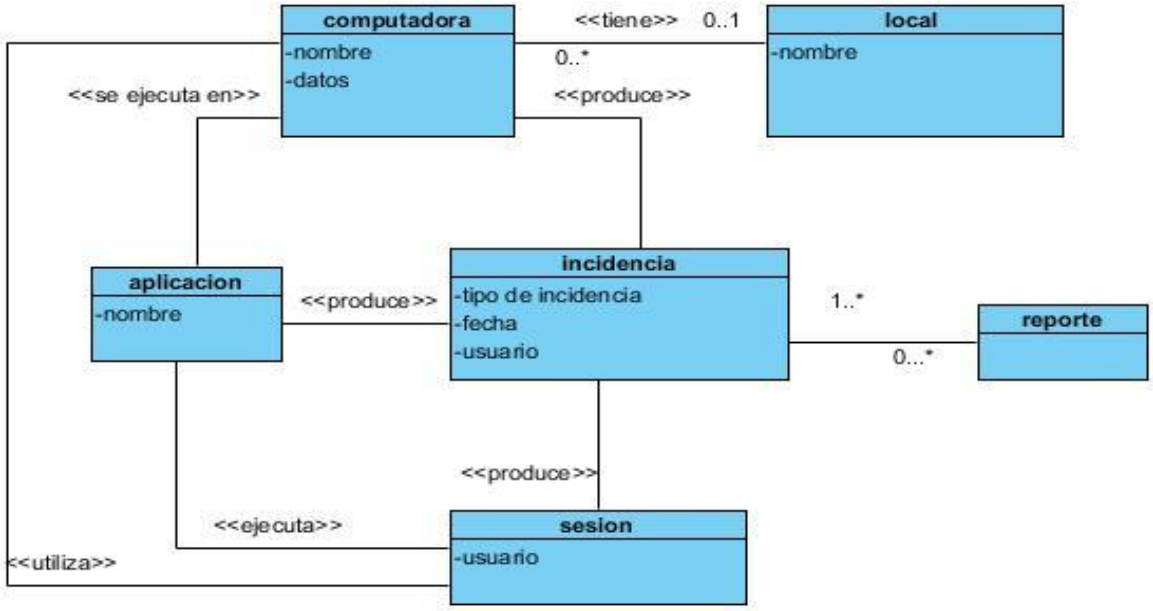


Figura 2.1 Modelo Conceptual.

INGENIERÍA DE REQUISITOS

La ingeniería de requisitos constituye el elemento base de todo proceso de desarrollo de software pues fija el punto de partida para la construcción del producto final. Si se identifican bien los requisitos que debe cumplir el sistema para dar solución al problema en cuestión se garantiza el desarrollo de un producto que cumpla con las expectativas del cliente.

Es un proceso centrado en el cliente y sus necesidades, con el objetivo de establecer los servicios que el sistema deberá proveer y las restricciones bajo las cuales deberá operar y ser desarrollado [2].

La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambiguedades, validar la especificación, y administrar los requisitos conforme éstos se transforman en un sistema operacional. El proceso de la ingeniería de requisitos se lleva a cabo a través de siete distintas funciones: inicio, obtención, elaboración, negociación, especificación, validación y gestión [26].

OBTENCIÓN DE LOS REQUISITOS

Para la obtención de los requisitos que debe cumplir el sistema se realizaron entrevistas a los miembros del Departamento Docente de Tecnología, en un inicio entrevistas abiertas para adquirir un conocimiento general del dominio del problema y seguidamente una mezcla de entrevistas cerradas y abiertas con el objetivo de profundizar en esos conocimientos. En las entrevistas cerradas, los stakeholders[19] responden a un conjunto predefinido de preguntas y en las entrevistas abiertas, el equipo de la ingeniería de requerimientos examina una serie de cuestiones con los stakeholders del sistema y, por lo tanto, desarrolla una mejor comprensión de sus necesidades [27].

Las entrevistas que se realizaron arrojaron información valiosa sobre las expectativas y necesidades de utilización del sistema, destacando la importancia de interfaces sencillas y fáciles de entender por los usuarios. Además se pudo ver la necesidad de proteger la información y regular el acceso a través de la autenticación de usuarios y la gestión de las funcionalidades a las que estos pueden acceder mediante la asignación de roles.

Los especialistas entrevistados resaltaron la necesidad de ver las aplicaciones que se abren en las computadoras en tiempo real y cerrar aquellas que no se consideren adecuadas para la producción y superación de los estudiantes. De igual forma se identificaron funcionalidades relacionadas con las características físicas del hardware que poseen las computadoras que se monitoreen y los requisitos no funcionales necesarios para el buen funcionamiento del sistema.

REQUISITOS FUNCIONALES

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, la forma en que debe reaccionar ante ciertas entradas y cómo se debe comportar en situaciones particulares [28].

[19] Son aquellos que se ven afectados directamente por las decisiones de la entidad y los que han de interactuar directamente con el sistema.

Los requisitos funcionales identificados son los siguientes:

Tabla 2.1 Requisitos funcionales.

Número	Requisito Funcional
1	Adicionar Local
2	Listar Locales
3	Modificar Local
4	Eliminar Locales
5	Exportar el listado de locales registrados en el sistema en formato PDF
6	Adicionar Computadora
7	Listar Computadoras registradas en el sistema
8	Modificar Descripción de Computadora
9	Eliminar Computadoras
10	Exportar listado de computadoras registradas en el sistema en formato PDF
11	Asociar computadoras a un local
12	Adicionar aplicación
13	Modificar aplicación
14	Listar aplicaciones instaladas por computadora
15	Listar aplicaciones instaladas en las computadoras asociadas a un local.
16	Exportar listado de aplicaciones instaladas en una computadora en formato PDF
17	Exportar listado de aplicaciones instaladas en las computadoras asociadas a un local en formato PDF
18	Listar los procesos asociados a una aplicación
19	Cerrar aplicación en ejecución
20	Mostrar proceso principal asociado a una aplicación
21	Exportar listado de procesos asociados a aplicación en formato PDF.
22	Adicionar aplicación a monitorear
23	Modificar descripción de aplicación a monitorear
24	Eliminar aplicación a monitorear
25	Listar aplicaciones a monitorear
26	Filtrar aplicaciones a monitorear
27	Exportar listado de aplicaciones a monitorear en formato PDF
28	Listar aplicaciones en ejecución
29	Filtrar aplicaciones en ejecución

30	Registrar cambio de hardware
31	Listar cambios de hardware
32	Filtrar cambios de hardware
33	Exportar listado de cambios de hardware en formato PDF
34	Registrar cambio de software
35	Listar cambios de software
36	Filtrar cambios de software
37	Exportar listado de cambios de software en formato PDF
38	Registrar inicio de aplicación
39	Listar inicios de aplicación
40	Filtrar inicios de aplicación
41	Exportar listado de inicios de aplicación en formato PDF
42	Registrar cierre de aplicación
43	Listar cierres de aplicación
44	Filtrar cierres de aplicación
45	Exportar listado de cierres de aplicación en formato PDF
46	Generar alertas sobre cambios de hardware
47	Generar alertas sobre cambios de software
48	Adicionar usuario
49	Modificar usuario
50	Listar usuarios
51	Filtrar usuarios por roles
52	Eliminar usuario
53	Generar reporte “Aplicaciones ejecutadas”
54	Generar reporte “Uso de aplicación por rango de fecha”

REQUISITOS NO FUNCIONALES

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Se aplican al sistema en su totalidad y no se refieren directamente a funciones específicas, sino a sus propiedades emergentes; la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento son algunas de ellas. Pueden estar dirigidos incluso al proceso de desarrollo, especificando herramientas o estándares de calidad a emplear en función de las necesidades del usuario [28].

A continuación se describen los requisitos no funcionales del sistema:

Tabla 2.2 Requisitos no funcionales.

Número	Requisito No Funcional
Usabilidad	
RNF1	Las interfaces del sistema deben ser entendibles, sencillas y amigables de manera que el usuario se sienta cómodo y pueda acceder a las funcionalidades de una forma rápida y cómoda.
Diseño e implementación	
RNF2	Se utilizará el sistema gestor de bases de datos PostgreSQl en su versión 9.0 o superior.
RNF3	Como principales lenguajes de programación se utilizarán Python en su versión 2.7 o superior, HTML, PHP5, JavaScript y SQL.
RNF4	Se hará uso de los marcos de trabajo Symfony2 y Bootstrap en sus versiones estables más recientes.
RNF5	Se utilizarán como entornos de desarrollo integrado los siguientes: PhpStorm en su versión 8.0 y Pycharm en su versión 3.4.
RNF6	Se utilizará como arquitectura del sistema Modelo-Vista-Controlador con un estilo basado en capas pues así se definió durante la actividad de planeación.
RNF7	Se utilizará NOVA como sistema operativo para el desarrollo del sistema, en su versión más reciente, pues así lo desea el cliente.
Hardware	
RNF8	**Recursos tecnológicos de las computadoras cliente.** Para el correcto funcionamiento de la parte cliente del sistema se establece que las computadoras donde se instale deben poseer las siguientes características: ➢ Tener al menos 512MB de memoria RAM. ➢ Poseer un disco duro con una capacidad superior a los 20GB. ➢ Contar con un procesador de 1.6 GHz o superior. ➢ Tener una tarjeta de red o capacidad de conectividad.
RNF9	**Recursos tecnológicos de la computadora servidor.** Es necesario que el ordenador donde se instale la parte servidor posea las siguientes propiedades: ➢ Memoria RAM: 1GB o superior.

	➢ Disco Duro: 40GB o superior. ➢ Microprocesador: con capacidad de procesamiento de 2.0 GHz o superior ➢ Poseer tarjeta de red o capacidades de conexión.
Software	
RNF10	Para el correcto funcionamiento de la parte cliente es necesario tener instalado un sistema operativo basado en GNU/Linux con los siguientes paquetes configurados correctamente: ➢ Python2.7 ➢ Python-amqplib ➢ Python-pika ➢ Python-psutil ➢ Python-psycopg2 ➢ Python-tornado ➢ Python-platform
RNF11	Para una correcta visualización de la información desde un ordenador cualquiera conectado a la red que se monitorea es necesario acceder mediante un navegador Firefox en una versión que soporte HTML5.
RNF12	La parte servidor debe ser instalada en una computadora con sistema operativo basado en GNU/Linux, tener instalado como sistema gestor de base de datos a PostgreSQL en su versión 9.0 o superior, como servidor web a Apache2, contar con PHP5 y Python2.7, python-amqplib, python-pika, python-psutil, python-psycopg2, python-tornado y python-platform.
Interconexión	
RNF13	La comunicación entre parte cliente y parte servidor debe ser mediante el empleo del protocolo de comunicación AMQP.
RNF14	Para la actualización de la información en tiempo real se debe emplear la tecnología de websockets.
Seguridad	
RNF15	La información que viaja a través de la red debe ser protegida haciendo uso de mecanismos de seguridad que lo permitan, evitando así el manejo de la información que

	llega al servidor.
RNF16	El acceso a la información debe ser a través de autenticación y mediante un sistema de control de acceso basado en roles.

ESPECIFICACIÓN DE REQUISITOS FUNCIONALES

La especificación de requisitos establece la base para el acuerdo entre usuarios y desarrolladores de software, quedando definido el comportamiento deseado del producto [29].

Para describir las funcionalidades del sistema se utilizan las Historias de Usuario (HU), estas contienen una descripción del comportamiento del sistema ante determinados eventos. Las HU permiten la comunicación entre el cliente y el equipo de desarrollo, también les permiten a los desarrolladores identificar las funcionalidades que deben implementar primero, teniendo en cuenta el nivel de prioridad, el riesgo en desarrollo y las iteraciones a las cuales se asignen.

A continuación se muestran tres HU correspondiente a los siguientes requisitos funcionales identificados:

- Listar aplicaciones en ejecución
- Cerrar aplicación en ejecución

Tabla 2.3 Historia de Usuario: Listar aplicaciones en ejecución.

HISTORIA DE USUARIO	
Número: HU-28	**Usuario:** Administrador
Nombre de Historia: Listar aplicaciones en ejecución	
Prioridad del Cliente: Alta	**Riesgo en desarrollo:** Alto
Estimación Técnica: 1	**Iteración asignada:** 3
Programador asignado: Yasmany Avila Sarmiento	
Descripción: Se muestran todas las aplicaciones bajo monitoreo que se encuentran en ejecución.	
Observaciones: Se brinda además la posibilidad de cerrar una de ellas, esta acción solo la pueden realizar los usuarios que tengan asignado el rol de administrador.	

Tabla 2.4 Historia de Usuario: Cerrar aplicación en ejecución.

HISTORIA DE USUARIO	
Número: HU-29	**Usuario:** Administrador
Nombre de Historia: Cerrar aplicación en ejecución	
Prioridad del Cliente: Alta	**Riesgo en desarrollo:** Alto
Estimación Técnica: 1	**Iteración asignada:** 3
Programador asignado: Yasmany Avila Sarmiento	
Descripción: Permite cerrar una aplicación que se encuentre en ejecución desde la aplicación web. Antes de cerrar la aplicación se le muestra un mensaje indicando la acción a realizar al usuario de la aplicación afectada.	
Observaciones: Esta acción solo puede ser realizada por los usuarios que tengan el rol de administrador asignado.	

VALIDACIÓN DE LOS REQUISITOS

La validación de los requisitos funcionales constituye un paso de avance pues permite concertar con el cliente las acciones realizadas y los diferentes puntos de vista de modo tal que se cumplan sus expectativas.

Para la validación de los requisitos se puso en práctica las siguientes técnicas:

- Revisión de los RF y sus descripciones

  El mecanismo primario para la validación de requisitos es la revisión técnica formal. El equipo de revisión que valida los requisitos forma un grupo de expertos incluyendo a los clientes, que examina la especificación y busca errores en el contenido o la interpretación de los requisitos, inconsistencias, omisiones o errores [26].

  El equipo de revisión encargado de validar los requisitos estuvo compuesto por el jefe del Departamento de Tecnología de la Facultad 3, la ingeniera en ciencias informáticas Maylevis Morejón Valdés y el desarrollador Yasmany Avila Sarmiento. Como resultado de la validación de los requisitos se aprobaron los requisitos de la solución propuesta, generándose como artefacto un acta de aceptación de requisitos, ver anexos #1,2 y 3.

- Construcción de prototipos :

  Los prototipos constituyen otra vía de validación de los requisitos, mediante ellos el cliente percibe la interpretación de los desarrolladores y si está de acuerdo da el visto bueno. A continuación, las figuras 2.2 y 2.3 muestran los prototipos correspondientes a “Listar aplicaciones en ejecución” y “Listar cambios de hardware” respectivamente.

Figura 2.2 Prototipo de interfaz de usuario: Listar aplicaciones en ejecución

Figura 2.3 Prototipo de interfaz de usuario: Listar cambios de hardware.

Como se puede apreciar en la figura 2.3, cuando el sistema detecta un cambio de hardware mientras el usuario está consultando el listado de cambios realizados, se muestra un mensaje o alerta informando a este sobre la nueva incidencia. Al dar click en el botón que se encuentra al lado de la alerta se actualiza el listado de los cambios de hardware registrados.

PLAN DE ITERACIONES

Con el objetivo de definir el orden de implementación de las historias de usuario y de realizar entregas continuas, al cliente, del producto de software se realiza un Plan de Iteraciones y un Plan de Entrega.

El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

A continuación se muestra el Plan de Iteraciones que recoge el resultado de un proceso de análisis de acuerdo al nivel de prioridad asignado a cada historia de usuario.

Tabla 2.5 Plan de iteraciones.

Iteraciones	Orden de las historias usuario a implementar	Prioridad	Duración de las iteraciones (en semanas)
1	Adicionar Local	Baja	3
	Listar Locales	Baja	
	Modificar Local	Baja	
	Eliminar Locales	Baja	
	Exportar el listado de locales registrados en el sistema en formato PDF	Baja	
	Adicionar Computadora	Alta	
	Listar Computadoras registradas en el sistema	Alta	
	Modificar Descripción de Computadora	Media	
	Eliminar Computadoras	Alta	
	Exportar listado de computadoras registradas en el sistema en formato PDF	Baja	
	Asociar computadoras a un local	Baja	
	Adicionar aplicación	Alta	
2	Modificar aplicación	Media	3
	Listar aplicaciones instaladas por computadora	Alta	
	Listar aplicaciones instaladas en las computadoras asociadas a un local.	Baja	
	Exportar listado de aplicaciones instaladas en una computadora en formato PDF	Baja	
	Exportar listado de aplicaciones instaladas en las computadoras asociadas a un local en formato PDF	Baja	

	Listar los procesos asociados a una aplicación	Alta	
3	Mostrar proceso principal asociado a una aplicación	Alta	3
	Exportar listado de procesos asociados a aplicación en formato PDF.	Baja	
	Adicionar aplicación a monitorear	Alta	
	Modificar descripción de aplicación a monitorear	Media	
	Eliminar aplicación a monitorear	Alta	
	Listar aplicaciones a monitorear	Alta	
	Filtrar aplicaciones a monitorear	Media	
	Exportar listado de aplicaciones a monitorear en formato PDF	Baja	
	Listar aplicaciones en ejecución	Alta	
	Cerrar aplicación en ejecución	Alta	
	Filtrar aplicaciones en ejecución	Media	
4	Registrar cambio de hardware	Alta	3
	Listar cambios de hardware	Alta	
	Filtrar cambios de hardware	Alta	
	Exportar listado de cambios de hardware en formato PDF	Baja	
	Registrar cambio de software	Alta	
	Listar cambios de software	Alta	
	Filtrar cambios de software	Alta	
	Exportar listado de cambios de software en formato PDF	Baja	
	Registrar inicio de aplicación	Alta	
	Listar inicios de aplicación	Alta	
	Filtrar inicios de aplicación	Alta	
	Exportar listado de inicios de aplicación en formato PDF	Baja	
	Registrar cierre de aplicación	Alta	

	Listar cierres de aplicación	Alta	
	Filtrar cierres de aplicación	Alta	
	Exportar listado de cierres de aplicación en formato PDF	Baja	
5	Generar alertas sobre cambios de hardware	Alta	3
	Generar alertas sobre cambios de software	Alta	
	Adicionar usuario	Alta	
	Modificar usuario	Alta	
	Listar usuarios	Alta	
	Filtrar usuarios por roles	Alta	
	Eliminar usuario	Alta	
	Generar reporte “Aplicaciones ejecutadas”	Alta	
	Generar reporte “Uso de aplicación por rango de fecha”	Alta	

PLAN DE ENTREGA

Tabla 2.6 Plan de entrega.

Módulos	**Historias de Usuario que incluye el módulo**
Administración de Entidades	Adicionar Local
	Listar Locales
	Modificar Local
	Eliminar Locales
	Exportar el listado de locales registrados en el sistema en formato PDF
	Adicionar Computadora
	Listar Computadoras registradas en el sistema
	Modificar Descripción de Computadora
	Eliminar Computadoras
	Exportar listado de computadoras registradas en el sistema

	en formato PDF
	Asociar computadoras a un local
	Adicionar aplicación
	Modificar aplicación
	Listar aplicaciones instaladas por computadora
	Listar aplicaciones instaladas en las computadoras asociadas a un local.
	Exportar listado de aplicaciones instaladas en una computadora en formato PDF
	Exportar listado de aplicaciones instaladas en las computadoras asociadas a un local en formato PDF
	Listar los procesos asociados a una aplicación
	Mostrar proceso principal asociado a una aplicación
	Exportar listado de procesos asociados a aplicación en formato PDF.
	Adicionar aplicación a monitorear
	Modificar descripción de aplicación a monitorear
	Eliminar aplicación a monitorear
	Listar aplicaciones a monitorear
	Filtrar aplicaciones a monitorear
	Exportar listado de aplicaciones a monitorear en formato PDF
Monitoreo	Listar aplicaciones en ejecución
	Cerrar aplicación en ejecución
	Filtrar aplicaciones en ejecución
	Registrar cambio de hardware
	Listar cambios de hardware
	Filtrar cambios de hardware
	Exportar listado de cambios de hardware en formato PDF
	Registrar cambio de software
	Listar cambios de software
	Filtrar cambios de software

	Exportar listado de cambios de software en formato PDF
	Registrar inicio de aplicación
	Listar inicios de aplicación
	Filtrar inicios de aplicación
	Exportar listado de inicios de aplicación en formato PDF
	Registrar cierre de aplicación
	Listar cierres de aplicación
	Filtrar cierres de aplicación
	Exportar listado de cierres de aplicación en formato PDF
	Generar alertas sobre cambios de hardware
	Generar alertas sobre cambios de software
Acceso	Adicionar usuario
	Modificar usuario
	Listar usuarios
	Filtrar usuarios por roles
	Eliminar usuario
Reportes	Generar reporte “Aplicaciones ejecutadas”
	Generar reporte “Uso de aplicación por rango de fecha”

Tabla 2.7 Plan de entrega por módulos.

	Iteración 1	**Iteración 2**	**Iteración 3**	**Iteración 4**	**Iteración 5**
Módulo	**4ta semana de febrero**	**3ra semana de marzo**	**2da semana de abril**	**1ra semana de mayo**	**3ra semana de mayo**
Administración de Entidades	V1.0	V1.1 Final	Finalizado	Finalizado	Finalizado
Monitoreo		V1.2	V1.3 Final	Finalizado	Finalizado
Acceso			V1.4	V1.5 Final	Finalizado
Reportes				V1.6	V1.7 Finalizado

MODELO DE DISEÑO

La metodología de desarrollo de software XP propone el uso de las tarjetas CRC (Clase Responsabilidad Colaboración) durante la actividad de diseño, en lugar de realizar diagramas de clases del diseño para la descripción del sistema. Cada tarjeta CRC representa una clase del modelo de dominio y en ella se representan las responsabilidades de la clase a la cual hace referencia y una referencia a otras con las cuales colabora.

A continuación se muestra la tarjeta CRC correspondiente a la clase controladora relacionada con las aplicaciones:

Tarjeta CRC AplicacionController.php	
Super Classes: Controller.php	
Sub Classes:	
Descripción: Esta clase se encarga de gestionar todas las acciones solicitadas por el usuario que estén vinculadas a las aplicaciones que registra el sistema. No interactúa directamente con la base de datos, para guardar los datos que maneja utiliza clases proporcionadas por le ORM Doctrine y mediante estas almacena y consulta la información de la base de datos.	
Attributes:	
Nombre	Descripción
Responsabilidades:	
Nombre	Collaborator
idexAction()	AdministracionEntidadesBundle\Entity\Aplicacion.php
newAction()	AdministracionEntidadesBundle\Form\AplicacionType.php
	AdministracionEntidadesBundle\Entity\Aplicacion.php
showAction()	Symfony\Component\HttpFoundation\Request.php
	AdministracionEntidadesBundle\Entity\Aplicacion.php
	AdministracionEntidadesBundle\Form\AplicacionType.php
editAction()	Symfony\Component\HttpFoundation\Request.php
	AdministracionEntidadesBundle\Entity\Aplicacion.php
updateAction()	Symfony\Component\HttpFoundation\Request.php
	AdministracionEntidadesBundle\Entity\Aplicacion.php
deleteAction()	Symfony\Component\HttpFoundation\Request.php
	AdministracionEntidadesBundle\Entity\Aplicacion.php

Figura 2.4 Tarjeta CRC de la clase controladora AplicacionController.php

PATRONES DE DISEÑO EMPLEADOS

Un patrón de diseño define un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos. Este describe una estructura común y recurrente de componentes interrelacionados, que resuelve un problema general de diseño dentro de un contexto particular [30].

PATRONES GRASP

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones [31]. Durante el diseño de la solución se emplearon los patrones GRASP experto, creador, controlador, bajo acoplamiento y alta cohesión.

Experto: Este patrón se puso en práctica con el objetivo de asignar la responsabilidad de creación de un objeto o la implementación de un método a la clase que conoce toda la información necesaria para crearlo, y de esta forma obtener un diseño con mayor cohesión y así la información se mantenga encapsulada (disminución del acoplamiento) [31]. Su uso se puede apreciar en las clases del modelo (Ejemplos: Local.php, Computadora.php, Aplicación.php, Usuario.php, Rol.php, Evento.php), las cuales tienen toda la información necesaria para crear los objetos a los cuales representan.

Creador: Este patrón se emplea para asignar a una clase la responsabilidad de crear una instancia de otra. En este trabajo se puede apreciar su uso en el caso de las clases controladoras que se utilizan en la aplicación web como parte de la estructura que define el marco de trabajo Symfony2. Un ejemplo lo constituye la clase ComputadoraController.php que crea instancias de la clase Computadora.php y modifica sus atributos en el momento de la creación.

Controlador: El patrón controlador**:** propone asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase [31]. Su uso se ve reflejado en la clase EventoController.php que maneja los mensajes que son enviados hacia el servidor desde el cliente.

Bajo acoplamiento: garantiza la asignación de responsabilidades de manera que el acoplamiento entre las clases permanezca bajo [31]. El acoplamiento mide la fuerza con que una clase está conectada a otra. En el sistema se puede apreciar como las clases se encuentran conectadas con otras de una forma débil pues las responsabilidades de cada una no dependen de la interacción con más de dos clases.

Alta cohesión: Este patrón se utiliza con el propósito de asignar las responsabilidades a las clases de manera que permanezcan lo más cohesionadas posible, garantizando que ninguna realice un trabajo excesivo gracias a la colaboración con otras clases [31]. A pesar de que la mayoría de las responsabilidades de cada clase no dependen de la interacción con más de dos clases ajenas, se mantiene la alta cohesión pues las mismas no realizan trabajo excesivo.

PATRÓN ARQUITECTÓNICO

Los patrones de arquitectura expresan un esquema de organización estructural para sistemas de software, especificando las responsabilidades de una serie de subsistemas predefinidos. Constituyen una plantilla para una arquitectura de aplicaciones y especifican las propiedades generales de la estructura del sistema, que trascienden en la arquitectura de los subsistemas. La selección de un patrón de arquitectura debe ser una decisión cuidadosa y fundamental al desarrollar un sistema de software [32].

Para el diseño arquitectónico del sistema se hizo uso del patrón arquitectónico Modelo Vista Controlador (MVC). MVC organiza el código fuente en tres componentes fundamentales: el modelo de los datos del dominio, la vista con la que interactúa el usuario y el controlador que gestiona y se encarga de procesar las peticiones del usuario enviadas por la vista y darles una respuesta haciendo uso del modelo.

El modelo es la capa donde se trabaja con los datos, contiene mecanismos para acceder a la información y para cambiar su estado. Debido a que se ha decidido emplear el marco de trabajo Symfony2, el modelo del sistema está vinculado al uso del ORM Doctrine permitiendo la abstracción de la base de datos y la persistencia de los datos.

Las vistas, como su nombre indica, contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá renderizar los estados de la aplicación en HTML. En las vistas nada se tienen los códigos HTML y PHP que permiten mostrar la salida. En la vista generalmente se trabaja con los datos, sin embargo, no se realiza un acceso directo a estos [33]. Ejemplo de vistas lo constituyen las plantillas .html.twig como aplicacionesEjecucion.html.twig perteneciente al módulo de Administración de Entidades y Usuario\index.html.twig relacionado al módulo de control de acceso.

Las controladoras contienen el código que se necesita para responder a las acciones que se solicitan en la aplicación. Una clase controladora funciona de intermediaria entre el modelo y las vistas. En el sistema existen muchas clases controladoras como AplicacionMonitorearController.php y ComputadoraController.php.

DISEÑO DE LA BASE DE DATOS

Luego de haber realizado el diseño del sistema es necesario modelar las entidades que forman parte de este y cómo se relacionan entre sí. El siguiente Modelo-Entidad-Relación (MER) permite apreciar las relaciones entre las entidades de la base de datos del sistema que se desarrolla.

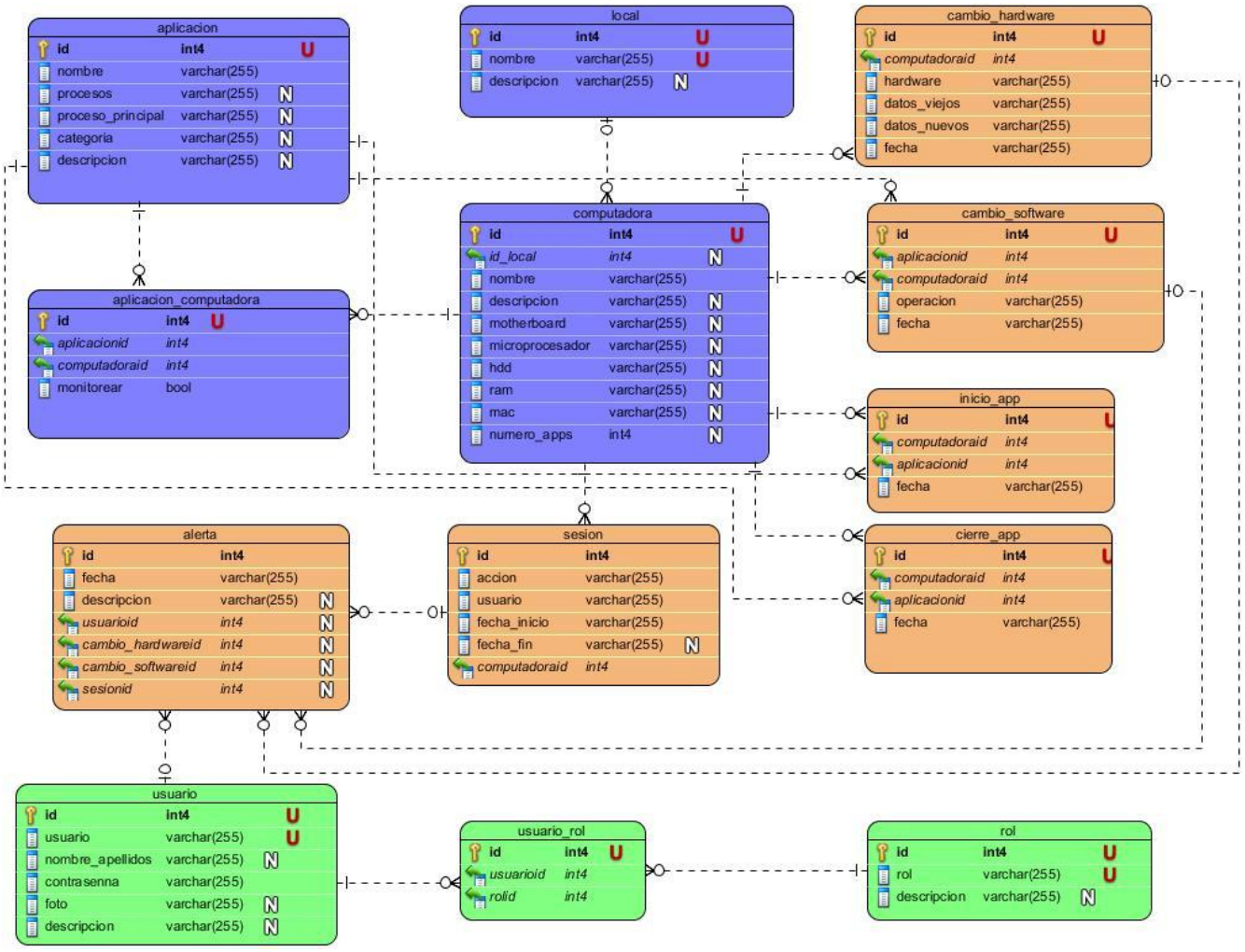


Figura 2.5 MER del sistema SMART

En el MER presentado en la figura anterior se diferencian las tablas y se agrupan por colores, estos colores representan partes distintas de la solución propuesta. Las tablas con color azul son aquellas en donde se almacena los datos necesarios para realizar las funcionalidades del sistema, datos que son introducidos y empleados en el módulo de administración de entidades. El módulo de administración de entidades recoge y analiza información sobre los locales, las computadoras y las aplicaciones que se monitorean.

Las tablas con color naranja corresponden al módulo de monitoreo y almacenan los datos referentes a las incidencias detectadas durante el monitoreo. La tabla “alerta” almacena los datos correspondientes a las alertas realizadas a cada usuario con el rol de administrador, a los usuarios que no tengan este rol solo se le mostrará las alertas en forma de mensaje mientras visualiza la información . Las alertas están relacionadas con los cambios de hardware, cambios de software, apagado de una computadora e inicio de una aplicación con extensión .exe. Por otro lado las tablas con color verde representan las entidades relacionadas con el módulo de acceso. El módulo de acceso gestiona los usuarios que pueden acceder al sistema y los roles que le son asignados a cada uno.

CONCLUSIONES PARCIALES DEL CAPÍTULO

El desarrollo de las dos primeras actividades propuestas por la metodología XP permitió obtener los requisitos funcionales que debe cumplir el sistema, describirlos y validarlos mediante entrevistas y encuestas, historias de usuario y prototipos de interfaz de usuario respectivamente. La generación de artefactos como las tarjetas CRC permitió describir las clases del sistema para un mejor entendimiento de las acciones que realizan. El plan de iteraciones y el plan de entrega describieron los distintos momentos en que se realizaron las entregas de código funcional al cliente. El uso de patrones de diseño permitió obtener un diseño coherente y de acuerdo a las necesidades existentes. Además, la descripción del modelo de datos fue posible a través del modelo entidad relación y en él se especifican las relaciones existentes entre las clases persistentes del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA PROPUESTO

INTRODUCCIÓN

En el presente capítulo se describen las principales métricas empleadas en la validación del diseño realizado y se pueden apreciar los resultados de su aplicación. Se representa el diagrama de despliegue de la solución propuesta y se explica los nodos que lo componen. Se describen los estándares de codificación definidos y cómo se ven reflejados en la implementación. Además se exponen las pruebas realizadas para la validación de la solución y los resultados obtenidos.

MÉTRICAS EMPLEADAS PARA LA VALIDACIÓN DEL DISEÑO

Las métricas, dentro del contexto de la ingeniería del software son un grupo de medidas efectuadas sobre programas, documentación, procesos de desarrollo o al mantenimiento, que permite una previa comparación con valores que proporcionan una indicación cuantitativa de extensión, cantidad, dimensión, capacidad y tamaño de algunos atributos de un proceso o producto. Las métricas permiten descubrir y corregir problemas potenciales, antes de convertirse en defectos catastróficos. Se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo [34].

En su libro sobre métricas orientadas a objetos, Lonrenz y Kidd dividen las métricas basadas en clases en cuatro amplias categorías, cada una con un diseño al nivel de componentes: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño aplicadas a una clase de diseño orientado a objetos se concentran en el conteo de atributos y operaciones de una clase individual, así como en valores promedio para el sistema orientado a objetos como un todo. Las métricas basadas en la herencia se concentran en la manera en que las operaciones se reutilizan en la jerarquía de clases. Las métricas para los valores internos buscan cohesión y aspectos orientados al código, y las métricas de valores externos examinan el acoplamiento y la reutilización [35].

A continuación se explican las métricas empleadas durante la evaluación del diseño realizado y los resultados obtenidos.

Tamaño de Clase (TC):

Según Pressman el tamaño general de una clase se determina con las siguientes medidas:

- El número total de operaciones (tanto heredadas como propias) que están encapsuladas dentro de la clase.
- El número de atributos (heredados o privados) que están encapsulados por la clase.

Los atributos que se miden a través de esta métrica son:

Responsabilidad: Consiste en las responsabilidades que son otorgadas a una clase en el ámbito de la propuesta solución.

Complejidad de implementación: Consiste en el grado de dificultad que posee la implementación de un diseño de clases determinado.

Reutilización: Consiste en el grado de reutilización del código de una clase o estructura de clase dentro del diseño de un producto.

A continuación se muestra una tabla con los criterios que se tuvieron en cuenta para clasificar los atributos que mide la métrica en cuestión:

Tabla 3.1 Criterios de clasificación de los atributos a medir con la métrica de validación del diseño Tamaño de Clase (TC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Complejidad de Implementación	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio

Los siguientes gráficos de pastel muestran los resultados de aplicar la métrica TC en la validación del diseño realizado:

Figura 3.1 Criterio Responsabilidad al aplicar la métrica TC.

Figura 3.2 Criterio Complejidad – Implementación al aplicar la métrica TC.

Figura 3.3 Criterio Reutilización al aplicar la métrica TC.

Como se puede apreciar, en las Figuras 3.1, 3.2 y 3.3, al aplicar la métrica de validación del diseño, Tamaño de Clase (TC), se demuestra la calidad del diseño realizado pues aproximadamente el 54% de las clases posee un nivel de responsabilidad y un nivel de complejidad de implementación bajo, además, más del 50% de las clases poseen un nivel alto de reutilización.

Relaciones entre Clases (RC):

El número de relaciones que tiene una clase está dado por la cantidad de relaciones de uso que tenga con el resto de las clases del diseño.

Los atributos que se miden en esta métrica son los siguientes:

- **Acoplamiento:** entre mayor sea el nivel de relación de una clase, mayor será el nivel de acoplamiento de la misma.
- **Complejidad de mantenimiento:** entre mayor sea el nivel de relación de una clase, mayor será el nivel de complejidad a la hora de realizar mantenimiento haciendo uso de ella.
- **Reutilización:** un aumento del nivel de relación de una clase influye en una disminución del nivel de reutilización de esta.
- **Cantidad de pruebas:** Un aumento del nivel de relación de una clase implica un mayor número de pruebas de unidad necesarias para probar su funcionamiento.

La siguiente tabla muestra los criterios que se tuvieron en cuenta para clasificar los atributos, antes mencionados, durante la aplicación de la métrica de validación del diseño RC:

Tabla 3.2 Criterios de clasificación de los atributos a medir con la métrica de validación del diseño Relaciones entre Clases (RC).

Atributo	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	> 2
Complejidad de mantenimiento	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2 * Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio
Cantidad de Pruebas	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Los resultados de la aplicación de esta métrica se muestran en la Figura 3.4.

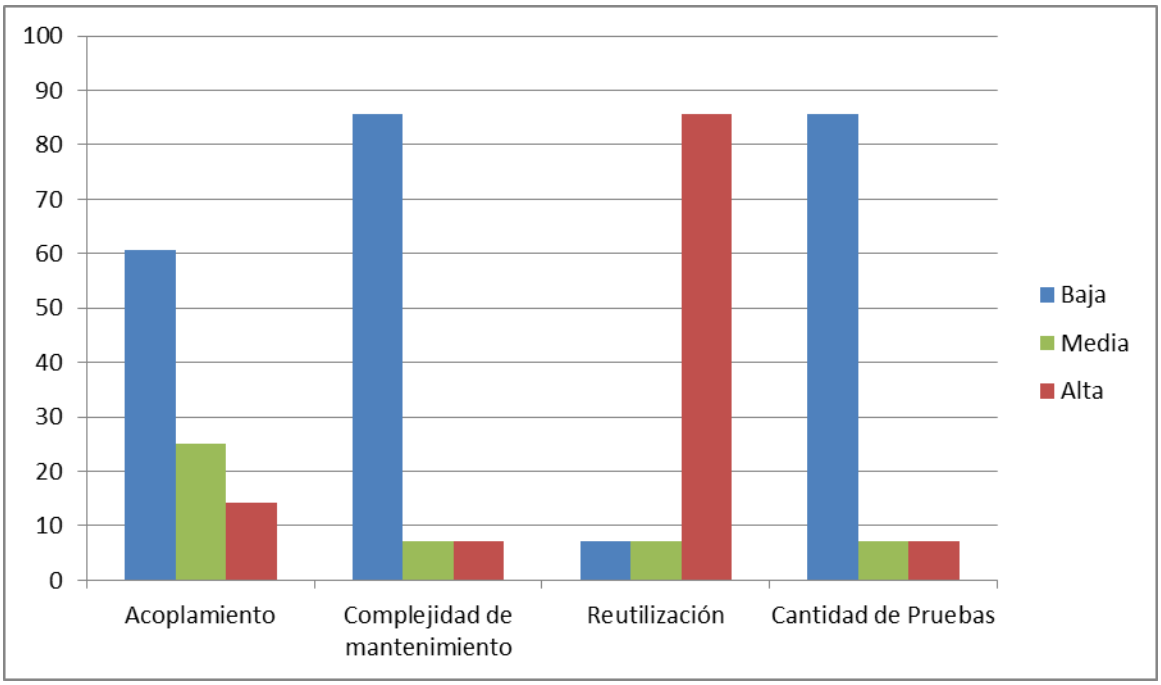


Figura 3.4 Resultados de la aplicación de la métrica de validación del diseño RC.

Haciendo un análisis de los resultados obtenidos se puede concluir que el diseño realizado tiene una calidad aceptable pues aproximadamente el 61% de las clases poseen un nivel de acoplamiento bajo,

el 85% de ellas requieren una complejidad de mantenimiento baja y un nivel de reutilización alto. Además, se necesitan pocas pruebas para más del 85% de las clases que forman parte de la solución.

La aplicación de las métricas Tamaño de Clase (TC) y Relaciones entre Clases (RC) demuestra que el diseño de la solución desarrollada posee una calidad aceptable. De esta forma queda validado el buen diseño de la solución desarrollada.

IMPLEMENTACIÓN DE LA SOLUCIÓN PROPUESTA

En la actividad de implementación se obtiene el código funcional de la solución propuesta que responde a las funcionalidades requeridas. El equipo de desarrollo implementa cada funcionalidad de acuerdo a las especificaciones recogidas en las historias de usuario haciendo uso de las herramientas y lenguajes definidos con anterioridad. De igual forma se deben emplear estándares de codificación para garantizar la reusabilidad y fácil entendimiento del código creado.

DIAGRAMA DE DESPLIEGUE

El Modelo de Despliegue o Diagrama de Despliegue es un tipo de diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. En muchos casos el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema.

El diagrama de despliegue que se presenta está compuesto por los nodos que se describen a continuación:

Computadora: Es el elemento que se monitorea y posee varios componentes que son instalados para enviar datos hacia el servidor web mediante el protocolo AMQP, en donde son procesados y convertidos en información entendible por el cliente.

Cliente: Accede a la aplicación a través de un ordenador, donde es ejecutada desde un navegador web Mozilla Firefox 29.0 o superior que soporte HTML5, el ordenador desde el cual se accede puede tener instalado cualquier sistema operativo. Se comunica con el servidor a través del protocolo AMQP y Websockets.

Servidor Web: Es el servidor de aplicaciones donde radica la parte del negocio encargada de procesar los datos enviados desde el monitor y debe contar con un servidor web Apache en una

versión igual o superior a la 2.0. De igual forma funciona sobre un sistema operativo basado en GNU/Linux y debe tener instalado el paquete Python2.7 y contar con la Máquina Virtual de Java en una versión 7.0 o superior. Se comunica con el cliente mediante el protocolo HTTPS y Websockets, con el servidor de base de datos a través de TCP y con el monitor lo hace a través del protocolo AMQP.

Servidor de Base de Datos: Utiliza PostgreSQL en su versión 9.0 como Sistema Gestor de Bases de Datos para almacenar los datos con los cuales trabaja el sistema.

A continuación se representa el modelo de despliegue elaborado para el Sistema de Monitoreo de Aplicaciones en Tiempo Real (SMART):

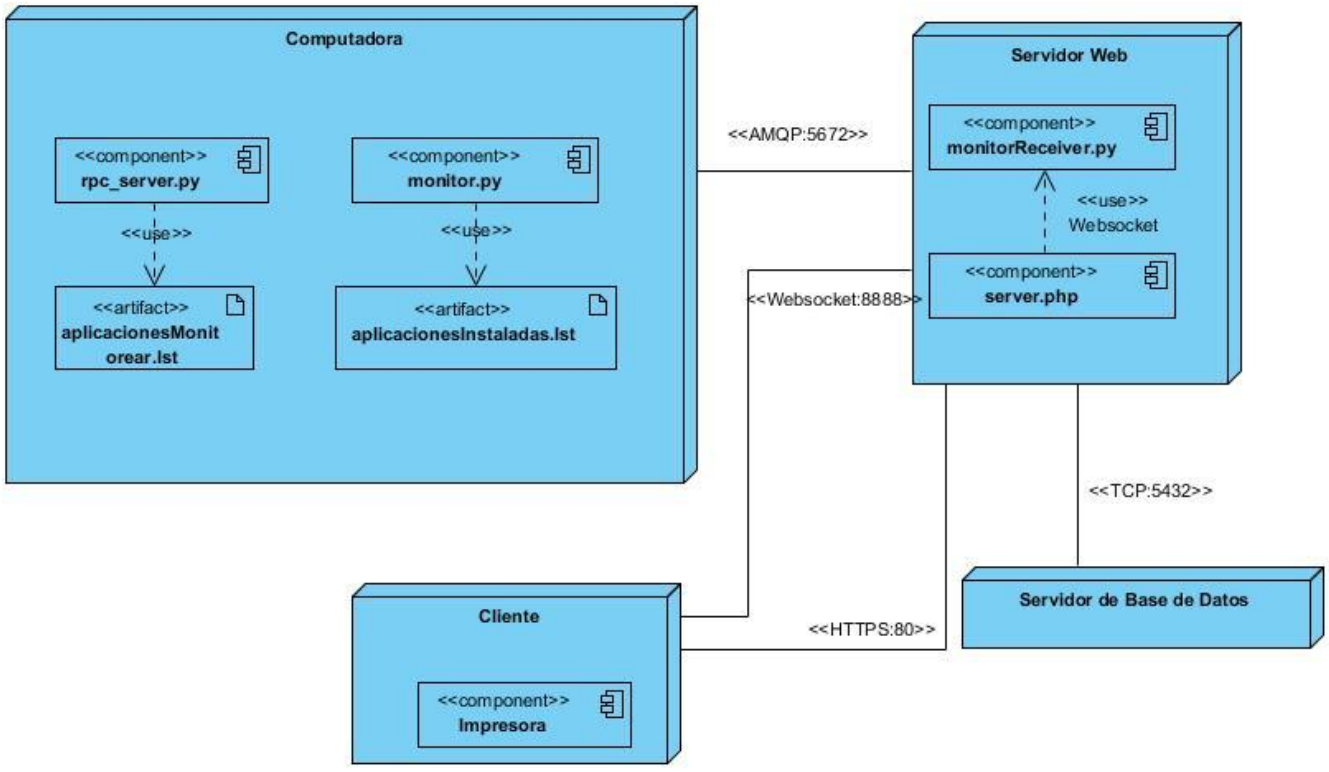


Figura 3.5 Modelo de despliegue.

El monitoreo en tiempo real se puede apreciar gracias al funcionamiento de un script, realizado haciendo uso del lenguaje de programación Python y ubicado en las computadoras clientes, que recoge datos continuamente y detecta cualquier incidencia para reportarla al instante. La comunicación con el servidor se realiza a través del protocolo avanzado de mensajería AMQP que

permite enviar mensajes desde una computadora a otra siempre y cuando la parte que envía el mensaje conozca el nombre y las credenciales necesarias para acceder a las pilas o colas de mensajes que se han creado en la parte que recibe.

En el servidor existe un script, igualmente realizado con Python, que lee de las pilas o colas que se definen para recibir los mensajes y los procesará para gestionar la información que brindan. Cuando el script mencionado anteriormente reconozca, por los mensajes procesados, que ha ocurrido una incidencia vinculada a las computadoras que se monitorean realizará las acciones pertinentes para registrarlas en una base de datos histórica y notificar a todos los usuarios, que se encuentren visualizando la información proveniente del monitoreo, sobre lo ocurrido.

La actualización de las vistas en los navegadores web conectados a la aplicación se realiza mediante el uso de websockets, empleados en las páginas html para recibir mensajes sobre los cambios en la información que se está mostrando; y en el script encargado de procesar los mensajes provenientes de los clientes, a través de AQMP, para enviar mensajes a un servidor, creado haciendo uso del lenguaje de programación PHP, que se encarga de reenviar esos mensajes a las páginas html relacionadas y que estén siendo solicitadas..

ESTÁNDARES DE CODIFICACIÓN

Con el objetivo de lograr homogeneidad en la implementación de manera que se facilitase el entendimiento del código se utilizó un conjunto de estándares que se describen a continuación.

Cabecera de archivo.

Todos los archivos deben iniciar con una cabecera que especifique su autor y en algunos casos, de ser necesario, la fecha de los últimos cambios realizados, además de otros datos de interés.

```
#/usr/bin/env python
#-*- coding: utf-8 -*-
#__author__ = 'yasmany'
```

Figura 3.6 Cabecera de archivo en Python.

```
/**
 * @author 'yasmany'.
 *
 */
```

Figura 3.7 Cabecera de archivo en PHP.

Estilo de codificación "Upper Camel Case".

Establece que los nombres de las clases iniciarán con letra mayúscula y de poseer más de una palabra estas comenzarán de igual forma con letra mayúscula y el resto en minúsculas. No se permiten letras mayúsculas sucesivas a menos que se trate de siglas definidas en el sistema.

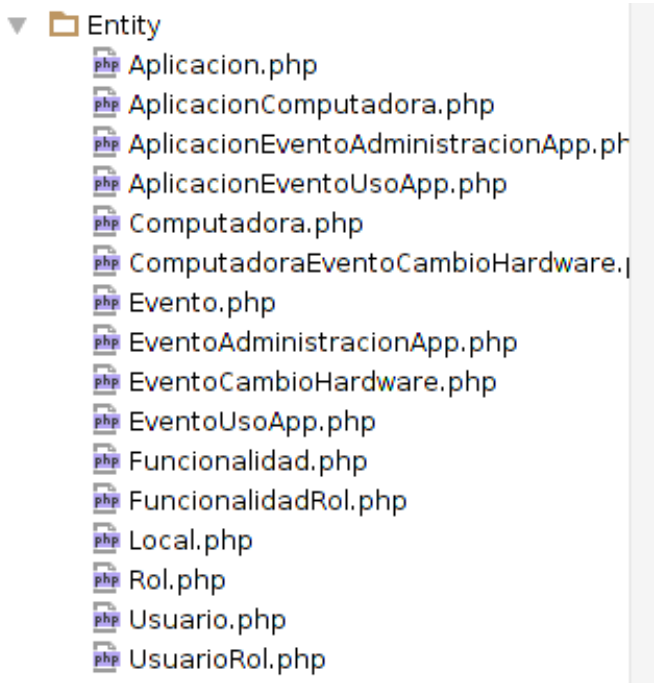


Figura 3.8 Aplicación del estilo de codificación "Upper Camel Case".

Definición de variables y constantes.

Los nombres de las variables deben ser descriptivos y no se usarán abreviaciones que puedan llevar a malentendidos.

```
$ids=$request->get('ids');
$entityManager = $this->getDoctrine()->getManager();
$entidades = $entityManager->getRepository('AdministracionEntidadesBundle:Aplicacion')->findById($ids);
$categorias= new ArrayCollection();
$categorias->add('Accesorio');
```

Figura 3.9 Definición de variables y constantes.

Estilo de codificación "Lower Camel Case".

Los nombres de las variables y constantes deberán iniciar con letra minúscula y cada nueva palabra comenzará con letra mayúscula.

```
$entidad = new Aplicacion();
$formularioAplicacion = $this->createCreateForm($entidad);
$formularioAplicacion->handleRequest($request);

if ($formularioAplicacion->isValid()) {
    $entityManager = $this->getDoctrine()->getManager();
    $entityManager->persist($entidad);
    $entityManager->flush();

    return $this->redirect($this->generateUrl('backend_aplicacion_show', array('id' => $entidad->getId())));
}
```

Figura 3.10 Aplicación del estilo de codificación “Lower Camel Case”.

Comentarios al inicio de cada función.

Al inicio de cada función se deberá poner un breve comentario sobre el objetivo de esa función y otros datos que faciliten la comprensión del código.

```
/**
 * Muestra una página (Aplicacion:edit.html.twig) con los datos de la aplicación
 * que se pretende modificar
 */
public function editAction(Request $request)
{
```

Figura 3.11 Comentarios al inicio de cada función.

PRUEBAS FUNCIONALES O DE ACEPTACIÓN

Cualquier estrategia de prueba, debe incorporar la planeación de pruebas, el diseño de caso de pruebas, la ejecución de pruebas y la recolección y evaluación de los datos resultantes [36].

Las pruebas funcionales, conocidas también como pruebas de aceptación o pruebas de caja negra, tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales [39]. Al realizar pruebas funcionales lo que se pretende es asumir la posición del usuario, usar el sistema como él lo usaría, sin embargo el analista de pruebas debe ir más allá que cualquier usuario, generalmente se requiere apoyo de los usuarios finales ya que ellos pueden aportar mucho en el desarrollo de casos de prueba complejos, enfocados

básicamente al negocio, posibles particularidades que no se hayan contemplado adecuadamente en el diseño funcional, el analista de pruebas debería dar fuerza a las pruebas funcionales y más aún a las de robustez, generalmente los usuarios realizan las pruebas con la idea que todo debería funcionar, a diferencia del analista de pruebas que tiene más bien una misión destructiva, su objetivo será encontrar alguna posible debilidad y si la llega a ubicar se esforzará por que deje de ser pequeña y posiblemente se convertirá en un gran error, cada error encontrado por el analista de pruebas es un éxito [37].

Las pruebas de caja negra, se concentran en los requisitos funcionales del software, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa [38].

Las pruebas realizadas se dividieron en pruebas alfas aplicadas al final de cada iteración del desarrollo y pruebas betas al concluir el desarrollo de todas las funcionalidades. La aplicación de las pruebas alfas se realizó en un entorno natural donde el desarrollador pudo seguir de cerca el trabajo de los usuarios típicos y registrar los errores y los problemas de uso. Por otro lado, las pruebas beta se realizaron una vez finalizado el sistema, se colocó a los usuarios finales en su ambiente de trabajo y a medida que fueron utilizando el sistema estos fueron registrando todos los problemas que detectaron.

A continuación se muestra el diseño del caso de prueba empleado en la validación de la historia de usuario **Listar aplicaciones en ejecución**, que corresponde a la tercera iteración de desarrollo. Los campos de las tablas que se muestran son los siguientes:

Historia de Usuario: Hace referencia al código y nombre de la historia de usuario que se prueba.

Descripción: Información descriptiva de la funcionalidad que se desea probar.

Condiciones de ejecución: Muestra las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba. Si estas condiciones no se cumplen no se garantiza que se alcancen los resultados esperados.

Escenarios: Muestra los diferentes escenarios por los que puede pasar la historia de usuario en dependencia de los datos introducidos y las acciones realizadas por el usuario.

Descripción: Muestra una breve descripción sobre el escenario al que hace referencia.

Variable: Representa las variables con las que se trabaja, describe el estado de la variable y el dato que contiene en cada escenario, el estado puede ser V, I o N/A: V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Respuesta del sistema: Se describe las respuestas del sistema ante el flujo de eventos desarrollado.

Flujo central: Pasos a desarrollar para probar la funcionalidad que se indicó.

Tabla 3.3 Caso de Prueba de Aceptación: Listar aplicaciones en ejecución.

CASO DE PRUEBA DE ACEPTACIÓN				
Historia de Usuario:		HU-28 Listar aplicaciones en ejecución		
Descripción:		Se muestran todas las aplicaciones que se encuentran en el listado de aplicaciones a monitorear que se encuentren en ejecución.		
Condiciones de ejecución:		El usuario debe haber accedido al sistema y encontrase en la página de inicio.		
Escenarios	**Descripción**	**Variable usuario**	**Respuesta del sistema**	**Flujo Central**
El usuario no está autenticado.	El usuario accede al sistema pero no se ha autenticado.	N/A	Muestra el siguiente mensaje de alerta: “Para acceder a esta funcionalidad debe estar autenticado”.	El usuario accede al sistema, da click en el enlace “Monitoreo”, se despliega un menú y selecciona la opción “Aplicaciones en ejecución”.
El usuario está autenticado pero no tiene el	El usuario ha accedido al sistema, se	I	Muestra el siguiente mensaje de alerta: “Para	El usuario accede al sistema, se autentica, da click en el enlace

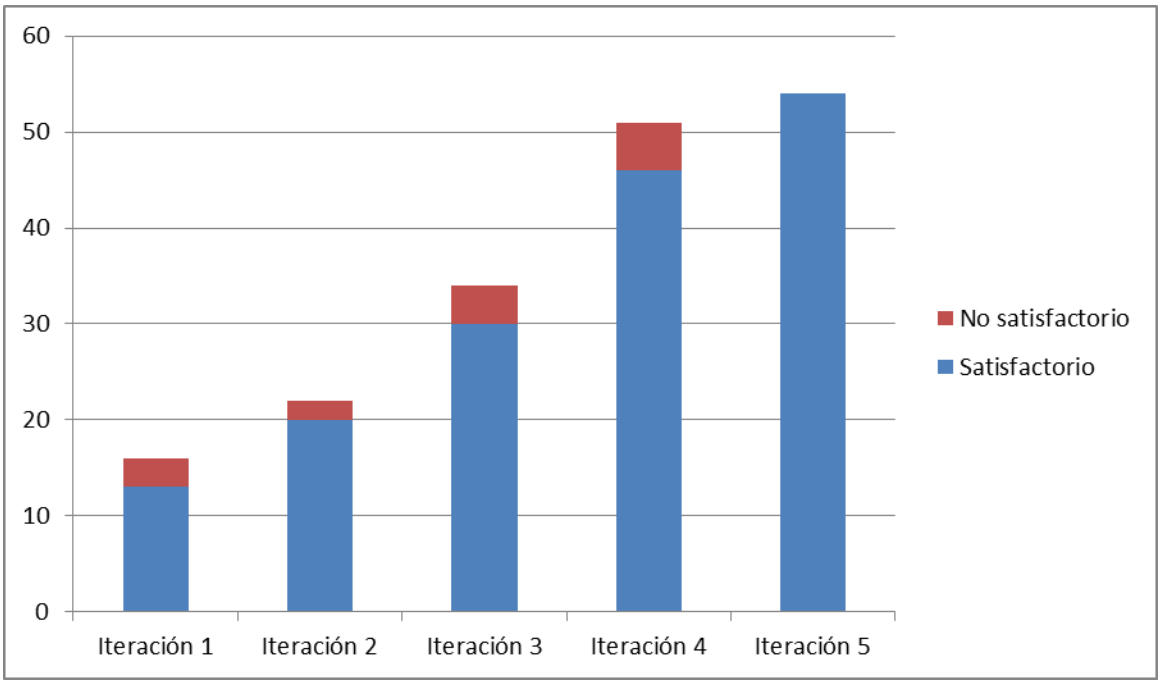
rol de Controlador o el Administrador.	encuentra autenticado pero no tiene el rol de Controlador o el de Administrador.	Invitado	acceder a esta funcionalidad debe tener permisos, usted no los tiene!! ".	"Monitoreo", se despliega un menú, selecciona la opción "Aplicaciones en ejecución".
El usuario está autenticado y tiene el rol de Controlador o el Administrador.	El usuario ha accedido al sistema, se encuentra autenticado y tiene el rol de Controlador o el de Administrador.	V Controlador/ Administrador	Muestra el listado de las aplicaciones que se están ejecutando en las computadoras que se monitorean.	El usuario accede al sistema, se autentica, da click en el enlace "Monitoreo", se despliega un menú, selecciona la opción "Aplicaciones en ejecución".

Resultados de las pruebas de aceptación:

Para la validación del correcto funcionamiento del sistema se realizaron varias pruebas funcionales durante el proceso de desarrollo, divididas en subgrupos de acuerdo a la iteración a la cual correspondían. Al final de la primera iteración de desarrollo se realizaron un total de 13 pruebas alfas, al final de la segunda iteración un total de 20 pruebas alfas y en las siguientes iteraciones se realizaron 30, 46 y 54 pruebas alfas respectivamente debido a la incorporación de las funcionalidades implementadas en cada una de ellas. En la primera iteración fueron detectadas tres no conformidades (NC), en la segunda iteración se corrigieron las NC encontradas en la primera y se detectaron dos NC que fueron corregidas en la tercera iteración. Durante la realización de las pruebas de la tercera iteración se generaron cuatro NC que fueron solucionadas en la cuarta iteración en donde al finalizar su desarrollo se encontraron cinco NC. Durante la quinta iteración fueron solucionadas las NC encontradas en la iteración anterior y se realizó el último grupo de pruebas alfas sin encontrarse NC.

Luego de finalizada la quinta iteración del proceso de desarrollo se procedió a realizar una prueba beta en el lugar de trabajo de los usuarios finales por un período de una semana y en un entorno no controlado por el equipo de desarrollo.

A continuación se muestra una gráfica donde se recogen los resultados obtenidos durante la realización de las pruebas alfas.



Durante la realización de la prueba beta el cliente pudo trabajar con el sistema e introducir datos y poner a prueba su funcionamiento. Finalmente el cliente expresó su satisfacción por la calidad de la solución brindada, haciéndolo constar formalmente en una carta de aceptación firmada el 5 de junio del 2015.

IMPACTO SOCIAL

La solución desarrollada consiste en un sistema de monitoreo de hardware y software en tiempo real orientado al sistema operativo NOVA pero que puede ejecutarse en ambientes con GNU/Linux. Su uso puede resultar útil para instituciones que prestan alta importancia a los medios tecnológicos que poseen. SMART brinda información relevante que puede ser empleada en la toma de decisiones vinculada a los activos fijos tangibles de una entidad y los recursos humanos con que cuenta.

Con el sistema se logró garantizar que cualquier incidencia detectada, ya sea un cambio de hardware, un cambio de software o el inicio o cierre de una aplicación, que ocurra sobre una computadora bajo monitoreo, sea de conocimiento de la persona que lo administra casi instantáneamente. Al asegurar la inmediatez de la información se evitan situaciones que puedan causar pérdidas económicas a quien usa el sistema. SMART constituye un sistema configurable que

se puede aplicar a distintos ambientes y que sirve de apoyo a la toma de decisiones en las entidades que decidan su empleo.

CONCLUSIONES PARCIALES DEL CAPÍTULO

La aplicación de las métricas Tamaño de Clase y Relaciones entre Clases permitió validar que, en efecto, el diseño está realizado de una forma simple y con una calidad aceptable. La estandarización del código creado posibilitó agregar aspectos a la implementación que facilitan su entendimiento y reusabilidad en un futuro.

El diagrama de despliegue presentado contribuyó a un mejor entendimiento de la solución propuesta dado que recoge los principales aspectos que la componen. Tras aplicar pruebas funcionales o de aceptación se pudo apreciar que la solución desarrollada se encuentra lista para ser utilizada por el cliente y cumple con sus expectativas. El nuevo sistema de monitoreo de hardware y software en tiempo real funciona correctamente y las irregularidades encontradas que dieron origen a la investigación han sido resueltas.

CONCLUSIONES

El estudio realizado de las soluciones existentes tanto en el ámbito internacional como en el nacional, condujo a la fundamentación de la investigación realizada, permitiendo definir los conceptos tratados y la propuesta de solución. La metodología de desarrollo de software definida, los marcos de trabajo utilizados y las herramientas y tecnologías empleadas guiaron el proceso de desarrollo de la solución y posibilitaron la obtención de un producto con una calidad aceptable y que cumple con las expectativas del cliente.

En total el sistema cuenta con 54 funcionalidades descritas y validadas con el cliente que fueron recogidas en un acta de aceptación de requisitos. Durante la validación de los requisitos se emplearon técnicas como la revisión de requisitos funcionales y su descripción y la generación de prototipos no funcionales en las cuáles la opinión del cliente siempre jugó un papel decisivo. El diseño del sistema fue validado mediante la aplicación de métricas y su posterior estudio generó resultados positivos permitiendo pasar a la próxima etapa: la implementación. La aplicación de pruebas funcionales o de aceptación permitió validar la solución desarrollada al ponerla en uso por parte del cliente y logrando su familiarización con las funcionalidades implementadas.

El diseño realizado y la implementación lograda permitieron obtener un sistema de monitoreo de hardware y software en tiempo real que funciona sobre el sistema operativo NOVA y resuelve las irregularidades detectadas en un inicio por el Departamento de Tecnología de la Facultad 3.

RECOMENDACIONES

Se recomienda la agregación al sistema de una funcionalidad que permita calcular el uso real que se realiza de una computadora en función de apoyar la toma de decisiones en los proyectos productivos.

REFERENCIAS BIBLIOGRÁFICAS

[1]. Definición .de. *Monitoreo.* [En línea] 2015. http://definicion.de/monitoreo/.

[2]. **BRUNNER, G.BÁEZ y S. B.** *Metodología DoRCU para la Ingeniería de Requerimientos.* s.l. : Instituto Superior Politécnico Jose Antonio Echeverría, 2001.

[3]. **Francisco Ortega Belmonte, Jordi Pons Aróztegui.** *Monitorización remota de sistemas. Memoria del proyecto de Ingeniería Técnica en Informática de Sistemas.* s.l. : Universidad Autónoma de Barcelona, 2009.

[4]. Sistemas distribuidos de tiempo real. [En línea] [Citado el: 06 de 04 de 2015.] http://dias-pensados.blogspot.com/2013/12/sistemas-distribuidos-de-tiempo-real.html.

[5]. **Conky.** Conky A light-weight system monitor. *Conky.* [En línea] 2015. [Citado el: 16 de 02 de 2015.] http://conky.sourceforge.net/.

[6]. Softonic. [En línea] [Citado el: 10 de 04 de 2015.] http://www.softonic.com/linux/php-sysinfo.

[7]. Munin. *Servidor Debian | Debian Server.* [En línea] 06 de 07 de 2013. [Citado el: 23 de 01 de 2015.] http://servidordebian.org/es/squeeze/monitoring/system/munin.

[8]. Ecured. [En línea] [Citado el: 09 de 04 de 2015.] http://www.ecured.cu/hyperic-hq.

[9]. Hyperic HQ. *SourceForge.* [En línea] [Citado el: 06 de 02 de 2015.] http://sourceforge.net/projects/hyperic-hq/#reviews.

[10]. **Pressman.** *Pressman. 6ta Edición, Capitulo 4 Desarrollo Ágil.* 2010.

[11]. **APONTE, ÁNGELA MARÍA VALBUENA.** *GUÍA COMPARATIVA DE FRAMEWORKS PARA LOS LENGUAJES HTML 5,CSS Y JAVASCRIPT PARA EL DESARROLLO DE APLICACIONES WEB.* s.l. : UNIVERSIDAD TECNOLÓGICA DE PEREIRA, 2014.

[12]. **Eguiluz, Javier.** *Desarrollo Web Ágil con Symfony2 páginas 18-19.* 29-12-2011.

[13]. **Mateu, Carles.** *Desarrollo de aplicaciones web.* Barcelona : Eureka Media, 2007. ISBN: 84-9788-118-4.

[14]. **BOOCH, G., J. RUMBAUGH y I. JACOBSON.** *El Lenguaje Unificado de Modelado/The Unified Software Development Process. edited by A. WESLEY.* s.l. : Wiley, 2001. ISBN 0 471 95889 7.

[15]. monografias.com. [En línea] [Citado el: 15 de 05 de 2015.] https://modulopoo.wordpress.com/unidad-ii/.

[16]. Sistemas de Información. [En línea] [Citado el: 15 de 05 de 2015.] http://uesasistemas.blogspot.com/2011/11/herramientas-case.html.

[17]. **Paradigm, V.** Visual Paradigm Suite User´s Guide. 2005.

[18]. **Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson,Rasmus Lerdorf, Zeev Suraski, Andrei Zmievski, Jouni Ahto.** *Manual de PHP página 45.* s.l. : Grupo de documentación de PHP, 2001.

[19]. **Team, Doctrine Project.** *Doctrine 2 ORM Documentation Release 2.1.* 2011.

[20]. **Kennedy, Chuck Musciano y Bill.** *HTML La Guía Completa página 13.* s.l. : ISBN 970-10, 1999.

[21]. **Duque, Raúl González.** *Python para todos.*

[22]. **Mateu, Carles.** *Desarrollo de aplicaciones web.* Barcelona : Eureka Media, 2007. ISBN: 84-9788-118-4.

[23]. **Ford, A.** *Apache 2 pocket reference.* s.l. : O'Reilly Media.

[24]. MQP Advanced Message Queuing Protocol. [En línea] 2015. [Citado el: 06 de 04 de 2015.] http://www.amqp.org/product/overview.

[25]. Websockets.org. [En línea] [Citado el: 15 de 05 de 2015.] https://www.websocket.org/.

[26]. **Pressman.** *Pressman Capítulo 07 Ingeniería de Software.*

[27]. Técnicas de Recopilación de Requisitos. [En línea] [Citado el: 07 de 04 de 2015.] http://clases3gingsof.wikifoundry.com/page/T%C3%A9cnicas+de+Recopilaci%C3%B3n+de+Requerimientos.

[28]. **SOMMERVILLE, IAN.** *Software Engineering.* s.l. : Pearson Education, 2007. ISBN 7-111-19770-4.

[29]. **IEEE.** *Guide to the Software Engineering Body of Knowledge (SWEBOK).* 2004.

[30]. **Marquina, Ernesto.** *Guía de Patrones, Prácticas y Arquitectura .NET.* 2008.

[31]. **Larman, Craig.** *UML y Patrones Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall.

[32]. **GAMMA, Erich.** *Design Patterns: Elements of Reusable ObjectOriented Software.* 2013.

[33]. **PARRA, E. MARQUINA y J. D.** *Guía de Patrones, Prácticas y Arquitectura .NET.* s.l. : In MICROSOFT, 2008.

[34]. DesarrolloWeb.com. [En línea] [Citado el: 08 de 04 de 2015.] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

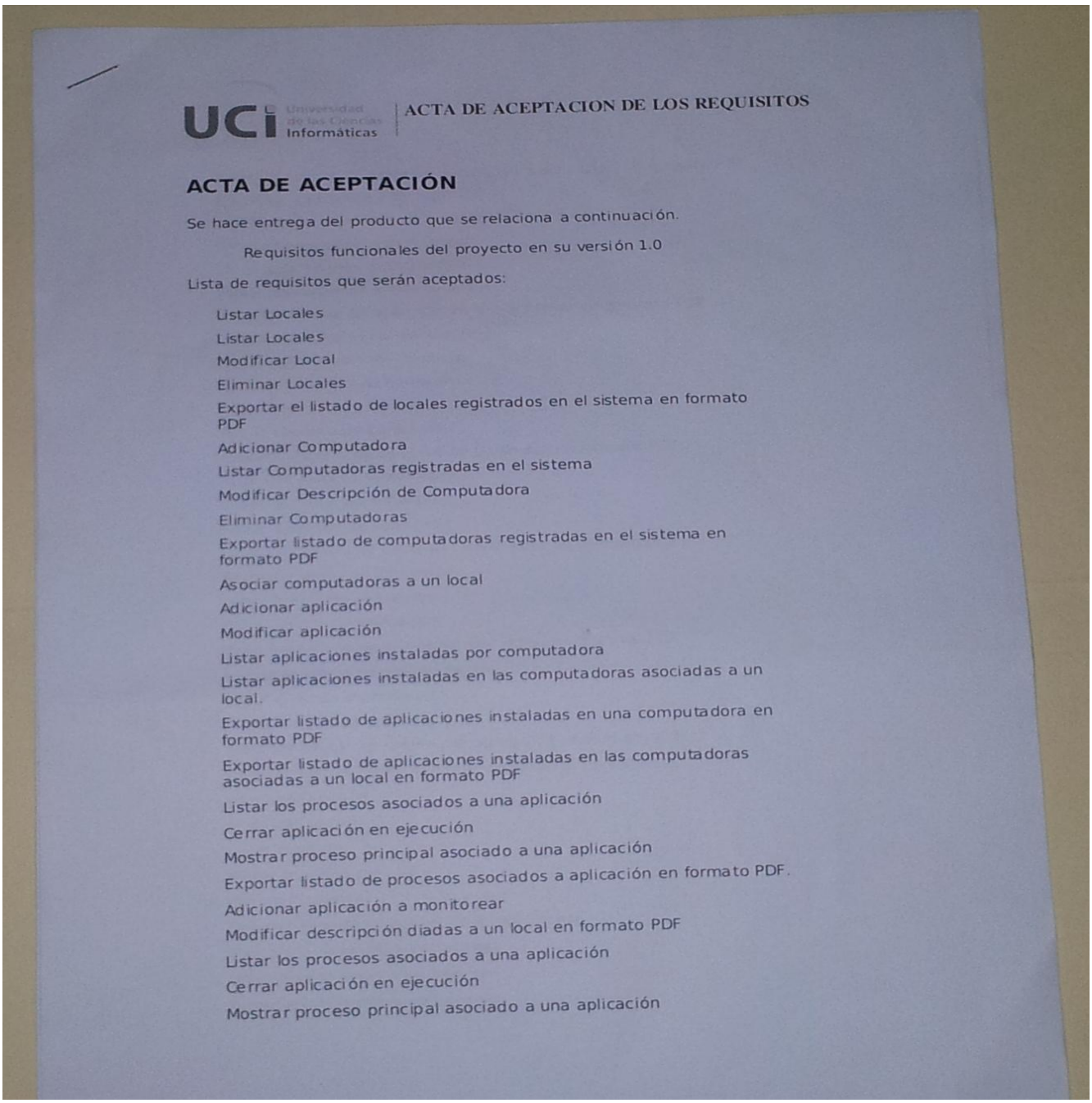
[35]. **Pressman, Roger S.** *Pressman 5ta Edición Capítulo 15.*

[36]. **Pressman.** *Pressman Capítulo 13 6ta Edición.*

[37]. Calidad y Software. *Pruebas Funcionales.* [En línea] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

[38]. **Pressman.** *Pressman Capítulo 14 6ta Edición.*

ANEXOS



UCI Universidad de las Ciencias Informáticas | ACTA DE ACEPTACION DE LOS REQUISITOS

ACTA DE ACEPTACIÓN

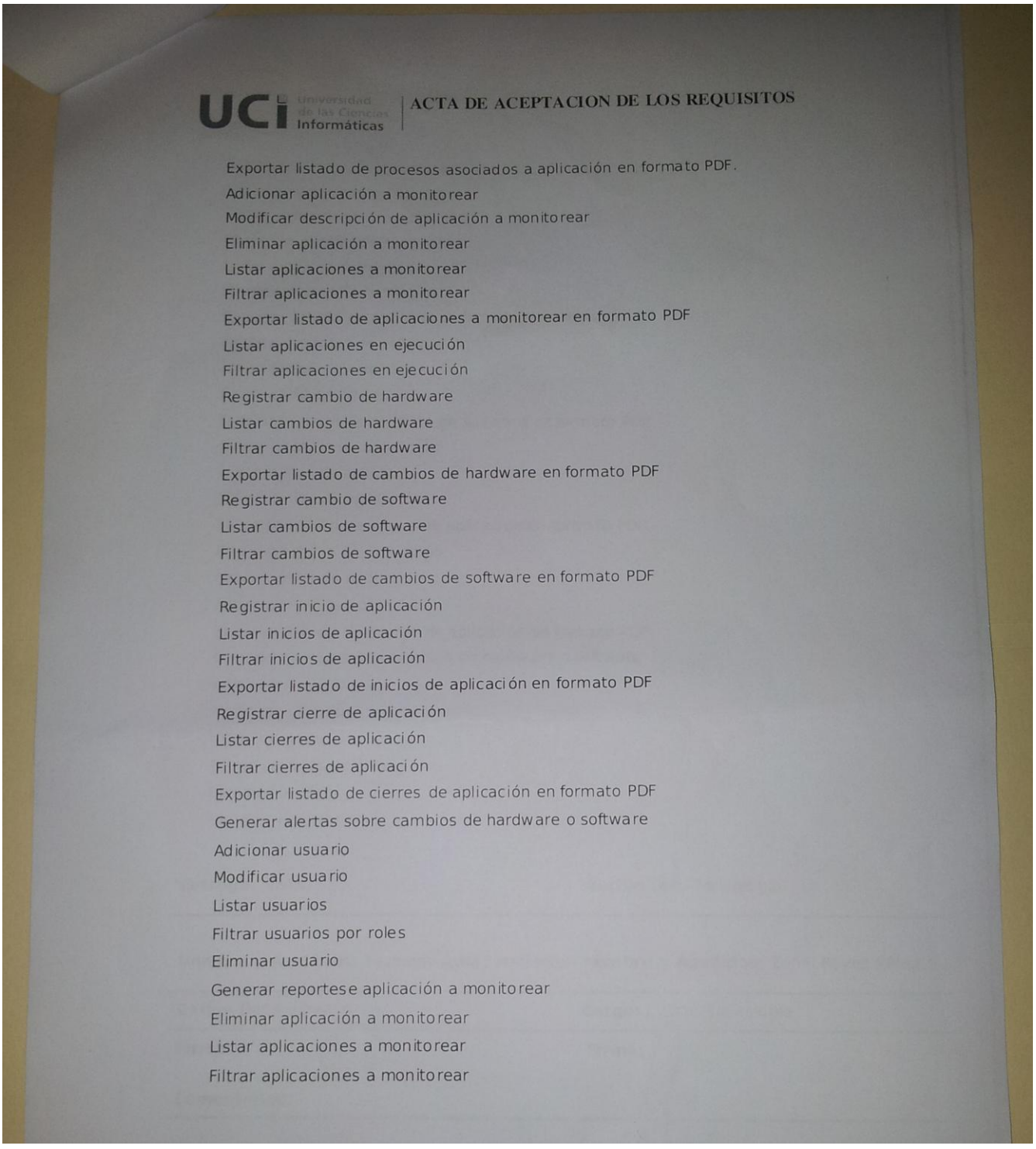
Se hace entrega del producto que se relaciona a continuación.

Requisitos funcionales del proyecto en su versión 1.0

Lista de requisitos que serán aceptados:

- Listar Locales
- Listar Locales
- Modificar Local
- Eliminar Locales
- Exportar el listado de locales registrados en el sistema en formato PDF
- Adicionar Computadora
- Listar Computadoras registradas en el sistema
- Modificar Descripción de Computadora
- Eliminar Computadoras
- Exportar listado de computadoras registradas en el sistema en formato PDF
- Asociar computadoras a un local
- Adicionar aplicación
- Modificar aplicación
- Listar aplicaciones instaladas por computadora
- Listar aplicaciones instaladas en las computadoras asociadas a un local.
- Exportar listado de aplicaciones instaladas en una computadora en formato PDF
- Exportar listado de aplicaciones instaladas en las computadoras asociadas a un local en formato PDF
- Listar los procesos asociados a una aplicación
- Cerrar aplicación en ejecución
- Mostrar proceso principal asociado a una aplicación
- Exportar listado de procesos asociados a aplicación en formato PDF.
- Adicionar aplicación a monitorear
- Modificar descripción diadas a un local en formato PDF
- Listar los procesos asociados a una aplicación
- Cerrar aplicación en ejecución
- Mostrar proceso principal asociado a una aplicación

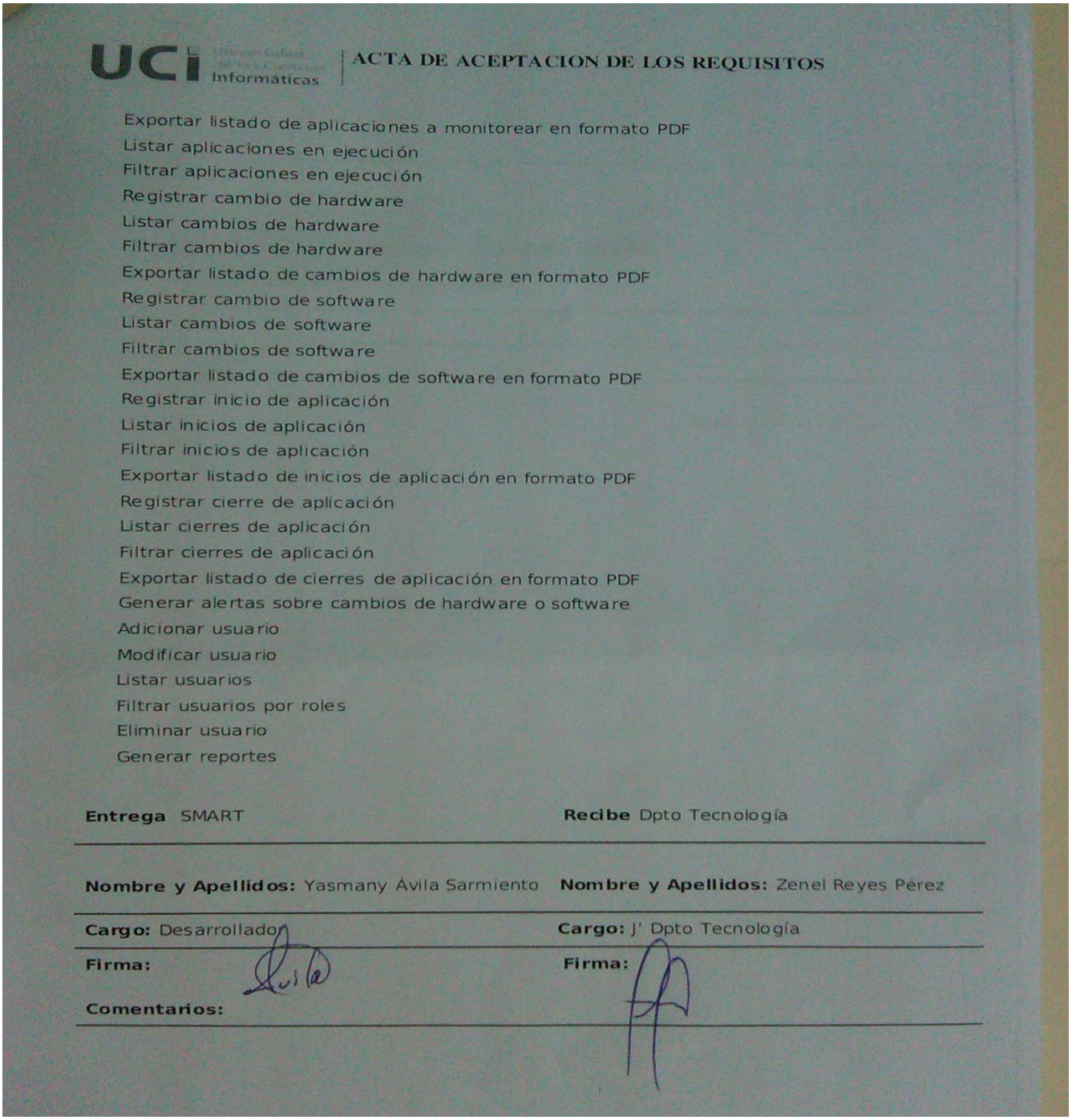
Anexo #1: Acta de aceptación de los requisitos. (Parte 1)



UCI Universidad de las Ciencias Informáticas | ACTA DE ACEPTACION DE LOS REQUISITOS

Exportar listado de procesos asociados a aplicación en formato PDF.
Adicionar aplicación a monitorear
Modificar descripción de aplicación a monitorear
Eliminar aplicación a monitorear
Listar aplicaciones a monitorear
Filtrar aplicaciones a monitorear
Exportar listado de aplicaciones a monitorear en formato PDF
Listar aplicaciones en ejecución
Filtrar aplicaciones en ejecución
Registrar cambio de hardware
Listar cambios de hardware
Filtrar cambios de hardware
Exportar listado de cambios de hardware en formato PDF
Registrar cambio de software
Listar cambios de software
Filtrar cambios de software
Exportar listado de cambios de software en formato PDF
Registrar inicio de aplicación
Listar inicios de aplicación
Filtrar inicios de aplicación
Exportar listado de inicios de aplicación en formato PDF
Registrar cierre de aplicación
Listar cierres de aplicación
Filtrar cierres de aplicación
Exportar listado de cierres de aplicación en formato PDF
Generar alertas sobre cambios de hardware o software
Adicionar usuario
Modificar usuario
Listar usuarios
Filtrar usuarios por roles
Eliminar usuario
Generar reportese aplicación a monitorear
Eliminar aplicación a monitorear
Listar aplicaciones a monitorear
Filtrar aplicaciones a monitorear

Anexo #2: Acta de aceptación de los requisitos. (Parte 2)



UCI Universidad de las Ciencias Informáticas | **ACTA DE ACEPTACION DE LOS REQUISITOS**

Exportar listado de aplicaciones a monitorear en formato PDF
Listar aplicaciones en ejecución
Filtrar aplicaciones en ejecución
Registrar cambio de hardware
Listar cambios de hardware
Filtrar cambios de hardware
Exportar listado de cambios de hardware en formato PDF
Registrar cambio de software
Listar cambios de software
Filtrar cambios de software
Exportar listado de cambios de software en formato PDF
Registrar inicio de aplicación
Listar inicios de aplicación
Filtrar inicios de aplicación
Exportar listado de inicios de aplicación en formato PDF
Registrar cierre de aplicación
Listar cierres de aplicación
Filtrar cierres de aplicación
Exportar listado de cierres de aplicación en formato PDF
Generar alertas sobre cambios de hardware o software
Adicionar usuario
Modificar usuario
Listar usuarios
Filtrar usuarios por roles
Eliminar usuario
Generar reportes

Entrega SMART — **Recibe** Dpto Tecnología

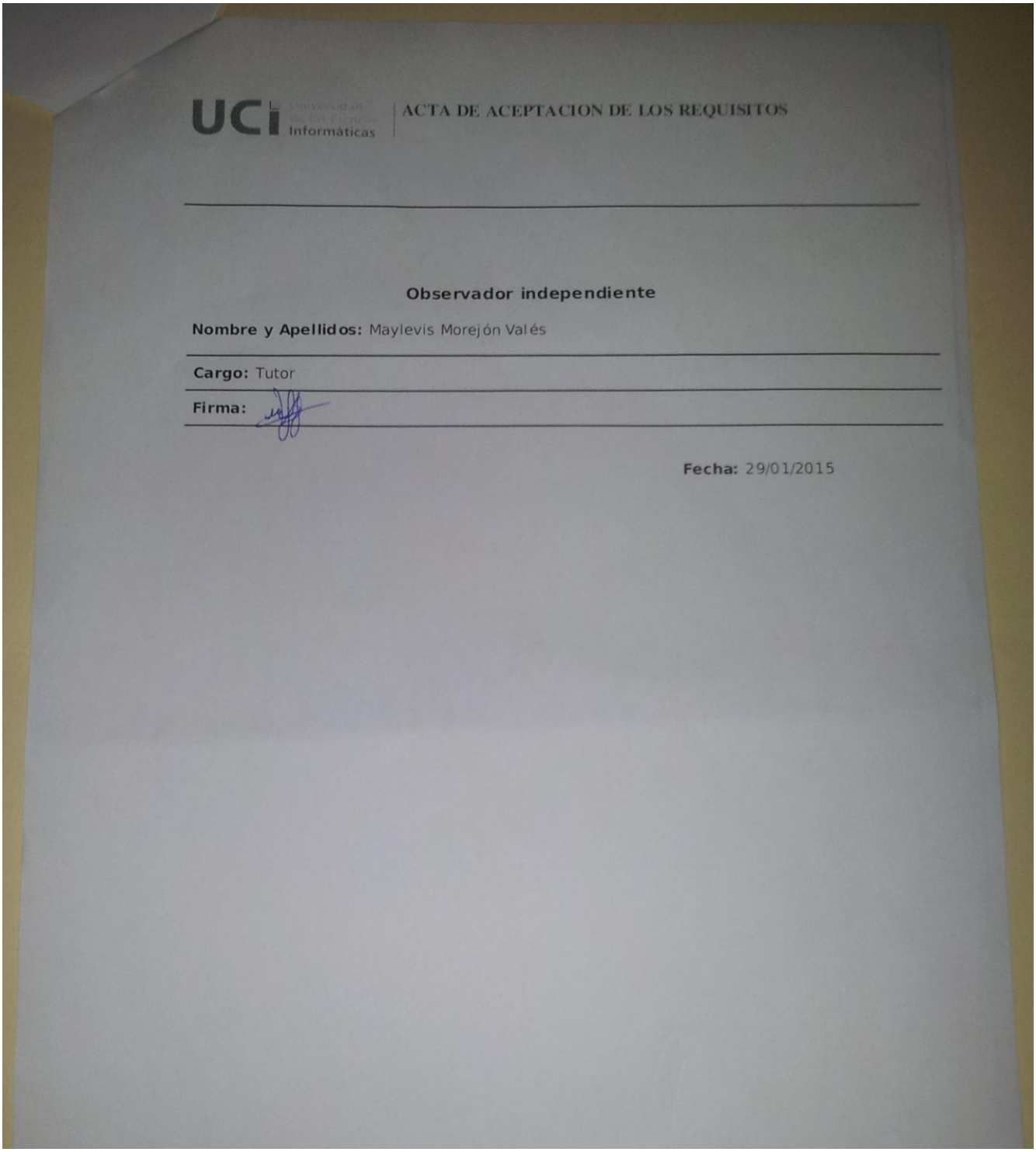
Nombre y Apellidos: Yasmany Ávila Sarmiento — **Nombre y Apellidos:** Zenel Reyes Pérez

Cargo: Desarrollador — **Cargo:** J' Dpto Tecnología

Firma: — **Firma:**

Comentarios:

Anexo #3: Acta de aceptación de los requisitos. (Parte 3)



UCI Universidad de las Ciencias Informáticas | ACTA DE ACEPTACION DE LOS REQUISITOS

Observador independiente

Nombre y Apellidos: Maylevis Morejón Valés

Cargo: Tutor

Firma:

Fecha: 29/01/2015

Anexo #4: Acta de aceptación de los requisitos. (Parte 4)