Facultad 5

Título: Herramienta para el monitoreo y análisis de los procesos del sistema SCADA Guardián del ALBA.

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: José Manuel Batista Viltre

Tutor: Ing. José Antonio Aragón Cáceres

Ing. Yolier Galán Tassé

La Habana, Junio de 2012

"Año 54 de la Revolución"

Declaración de autoría.

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

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

_____

Firma del Autor
José Manuel Batista Viltre

_____ _____

Firma del Tutor
Ing. José A. Aragón Cáceres

Firma del Tutor
Ing. Yolier Galán Tassé

Datos de contacto.

Nombre y apellidos del tutor: José Antonio Aragón Cáceres.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: jaaragon@uci.cu

Especialista graduado en la Universidad de las Ciencias Informáticas (UCI), profesor instructor con 3 años de experiencia docente y 6 años de experiencia en la producción de software, específicamente en el desarrollo de sistemas SCADA. Arquitecto de software del SCADA-UX y líder de la línea de desarrollo de Comunicaciones del Departamento de Construcción de Componentes del CEDIN.

Nombre y apellidos del tutor: Yolier Galán Tassé.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: ytasse@uci.cu

Especialista graduado en la Universidad de las Ciencias Informáticas (UCI) con 4 años de experiencia en la producción de software, específicamente en el desarrollo de sistemas SCADA.

Agradecimientos

A mi familia por apoyarme y ayudarme en todo.

A mis tutores por ayudarme y confiar en mí.

A la decana Mayra por su preocupación y atención.

A todas las personas que se preocuparon por mi cuando estuve accidentado.

A mis amistades por aguantarme estos 5 años.

A las personas que me hicieron sugerencias.

A las personas que me ayudaron en la revisión y corrección de los errores.

A todas las personas que de una forma u otra ayudaron en el desarrollo de la tesis.

A todas estas personas, muchas gracias por todo, nombrarlas sería egoísta con las que pueda olvidar.

Dedicatoria

A mi mamá, mi abuela Edilia, mi hermana July, mi tío Manolo y mi sobrinita por estar siempre conmigo.

Resumen

El presente trabajo surge por la necesidad de crear una herramienta capaz de informatizar el proceso de supervisión del comportamiento en el tiempo del uso del procesador y el consumo real de la memoria de los procesos o servicios del sistema SCADA Guardián del ALBA (GALBA).

Para realizar la propuesta planteada se trazó como objetivo, desarrollar una herramienta que permita el monitoreo y análisis del consumo del procesador y la memoria de los procesos del GALBA. Se define *Extreme Programming* como metodología de desarrollo de software, para realizar el modelado del sistema se utiliza la herramienta CASE *Visual Paradigm*, apoyándose en el Lenguaje Unificado de Modelado (UML). Se hace uso de los lenguajes de programación *Python* y *JavaScript*, el entorno de desarrollo integrado *Eclipse*, el framework de desarrollo web *Django*, *PostgreSQL* como sistema gestor de base de datos y las bibliotecas gráficas *jQuery* y *Highcharts*.

El uso de la herramienta para el monitoreo y análisis de los procesos del GALBA permitirá detectar posibles fugas de memorias de dichos procesos en ejecución o un uso excesivo del procesador que pudiera ser dañino para el sistema. Además dotará al equipo de prueba de una herramienta sencilla, centralizada, con interfaz amigable que agilizará los procesos de prueba de software.

Palabras claves: monitoreo y análisis, monitoreo de proceso, SCADA, GALBA

Índice de contenido

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA Y TECNOLOGÍAS PARA EL DESARROLLO. ... 5

1.1 Sistemas SCADA 5

1.1.1 Sistema SCADA Guardián del ALBA (GALBA) 6

1.2 Pruebas de software 7

1.2.1 Pruebas no funcionales 8

1.3 Monitoreo de procesos 10

1.3.1 Consumo del procesador. 11

1.3.2 Consumo de la memoria 11

1.4 Aplicaciones de monitoreo 12

1.4.1 Módulo psutil 12

1.4.2 Process snapshot (ps) 12

1.4.3 Top 13

1.4.4 Htop 13

1.4.5 Gnome System Monitor 14

1.5. Evaluación de aplicaciones de monitoreo para el desarrollo de la herramienta. 15

1.5.1. Criterios de evaluación 15

1.6. Módulo propuesto para el desarrollo de la herramienta. 16

1.7 Metodología, lenguajes y tecnologías para el desarrollo. 17

1.7.1 Metodología de desarrollo 17

1.7.2 Lenguaje de Modelado 19

1.7.3 Herramienta CASE 20

1.7.4 Framework de desarrollo 20

1.7.5 Bibliotecas gráficas 21
1.7.6 Lenguajes de programación. 22
1.7.7 Entorno de desarrollo 23
1.7.8 Sistema Gestor de Base de Datos (SGBD) 23
1.8 Conclusiones parciales. 24
CAPÍTULO 2 DESCRIPCIÓN DEL SISTEMA. PLANIFICACIÓN Y DISEÑO. 25
2.1 Flujo actual de eventos 25
2.2 Propuesta del sistema 25
2.2.1 Personal relacionado con el sistema 26
2.3 Historias de Usuario 26
2.4 Planificación y entrega 31
2.4.1 Plan de entrega 31
2.4.2 Plan de iteraciones 33
2.4.3 Plan de duración de las iteraciones 34
2.4.4 Historias de usuario divididas en tareas 34
2.5. Diseño del sistema 36
2.5.1 Tarjetas CRC (Contenido o Clase, Responsabilidad y Colaboración) 37
2.5.2 Arquitectura del sistema 38
2.5.3 Patrones de diseño 39
2.6. Conclusiones parciales 39
CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA 40
3.1 Estándares de codificación 40
3.1.1 Definiciones generales 40
3.1.2 Comentarios 41
3.2 Diagrama de despliegue 42

3.3 Diagrama Entidad-Relación 43
3.4 Prueba 44
3.4.1 Pruebas de aceptación 44
3.5 Conclusiones parciales 53
CONCLUSIONES GENERALES 54
RECOMENDACIONES 55
REFERENCIAS BIBLIOGRÁFICAS 56
ANEXOS 59
Anexo 1: Tareas abordadas en la primera iteración 59
Anexo 2: Tareas abordadas en la segunda iteración 62
Anexo 3: Tarjetas C.R.C del sistema 69
Anexo 4: Elementos a tener en cuenta en el desarrollo del modelo de decisión elaborado para la selección de la aplicación de monitoreo a utilizar en la implementación de la herramienta de monitoreo y análisis. 71

Índice de figuras

Figura 1: Salida del comando ps. 13

Figura 2: Salida de comando top 13

Figura 3: Salida comando htop 14

Figura 4: Imagen de la herramienta Gnome System Monitor. 15

Figura 5: Matriz de decisión de aplicaciones de monitoreo para la implementación de la herramienta de monitoreo y análisis. 17

Figura 6: Metodología Extreme Programing 18

Figura 7: Modelo Vista Controlador (MVC) 38

Figura 8: Diagrama de despliegue 42

Figura 9: Diagrama Entidad-Relación 43

Índice de tablas

Tabla 1 Personal relacionado con el sistema 26
Tabla 2 HU Autenticar usuario 27
Tabla 3 HU Gestionar usuario 27
Tabla 4 HU Gestionar computadora 28
Tabla 5 HU Gestionar proceso 28
Tabla 6 HU Monitorear proceso 28
Tabla 7 HU Visualizar comportamiento de la prueba. 29
Tabla 8 HU Generar reporte 29
Tabla 9: HU Requisitos de software 30
Tabla 10: HU Requisitos de Hardware. 30
Tabla 11: HU Restricciones en el diseño e implementación. 31
Tabla 12: HU Requisitos de apariencia o interfaz externa. 31
Tabla 13 Plan de entrega 32
Tabla 14 Estimación del esfuerzo por historia de usuarios 33
Tabla 15 Plan de duración de las iteraciones 34
Tabla 16 Tareas abordadas en la primera iteración 35
Tabla 17 Tarea genérica 35
Tabla 18 Tareas abordadas en la segunda iteración 36
Tabla 19 Modelo de Tarjeta CRC. 37
Tabla 20 Prueba 1 HU Autenticar usuario 45
Tabla 21 Prueba 1 HU Gestionar usuario 46
Tabla 22 Prueba 2 HU Gestionar usuario 46
Tabla 23 Prueba 3 HU Gestionar usuario 47
Tabla 24 Prueba 1 HU Gestionar computadora 47

Tabla 25 Prueba 2 HU Gestionar computadora 48
Tabla 26 Prueba 3 HU Gestionar computadora 48
Tabla 27 Prueba 1 HU Gestionar proceso 49
Tabla 28 Prueba 2 HU Gestionar proceso 49
Tabla 29 Prueba 3 HU Gestionar proceso 50
Tabla 30 Prueba 1 HU Monitorear proceso 50
Tabla 31 Prueba 1 HU Visualizar comportamiento de la prueba 51
Tabla 32 Prueba 2 HU Visualizar comportamiento de la prueba 52
Tabla 33 Prueba 3 HU Visualizar comportamiento de la prueba 53
Tabla 34 Prueba 1 HU Generar reporte 53
Tabla 35 Tarea 1: Sincronizar con la base de datos 59
Tabla 36 Tarea 2: Crear plantilla para formulario de autenticar. 59
Tabla 37 Tarea 3: Validar los campos del formulario. 60
Tabla 38 Tarea 4: Verificar los datos insertados. 60
Tabla 39 Tarea 5: Definir los privilegios del usuario 60
Tabla 40 Tarea 6: Definir tabla de usuarios. 61
Tabla 41 Tarea 7: Definir la tabla de computadoras 61
Tabla 42 Tarea 8: Definir tabla de procesos. 61
Tabla 43 Tarea 9: Definir tabla de pruebas. 62
Tabla 44 Tarea 10: Definir tabla de configuración de procesos. 62
Tabla 45 Tarea 11: Definir gestor de base de datos a utilizar. 63
Tabla 46 Tarea 12: Definir consultas sql. 63
Tabla 47 Tarea 13: Definir contexto de ejecución de prueba. 63
Tabla 48 Tarea 14: Crear plantilla de la vista principal. 64
Tabla 49 Tarea 15: Visualizar listado de computadoras. 64

Tabla 50 Tarea 16: Visualizar listado de procesos. 64
Tabla 51 Tarea 17: Visualizar listado de pruebas. 65
Tabla 52 Tarea 18: Consultar valores en la base de datos. 65
Tabla 53 Tarea 19: Consultar últimos valores en la base de datos. 65
Tabla 54 Tarea 20: Crear gráfica para visualizar todos los valores. 66
Tabla 55 Tarea 21: Crear gráfica para visualizar un intervalo. 66
Tabla 56 Tarea 22: Exportar datos a mostrar. 67
Tabla 57 Tarea 23: Crear plantilla para el reporte. 67
Tabla 58 Tarea 24: Ajustar datos del reporte. 67
Tabla 59 Tarea 25: Guardar reporte en formato pdf. 68
Tabla 60 Tarjeta CRC clase DBObject 69
Tabla 61 Tarjeta CRC clase DBPostgresql 69
Tabla 62 Tarjeta CRC clase DBSQLite 69
Tabla 63 Tarjeta CRC clase ExecutePMonitorQuery 69
Tabla 64 Tarjeta CRC clase TestThread 69
Tabla 65 Tarjeta CRC clase TreeViewAdmin 70
Tabla 66 Tarjeta CRC clase ChartViewAdmin 70
Tabla 67 Tarjeta CRC clase ConvertSVG2PNG 70
Tabla 68 Tarjeta CRC clase Report 70
Tabla 69: Elementos de la matriz de decisión. 71
Tabla 70: Tabla que muestra cómo calificar una opción. 71
Tabla 71: Modelo de decisión 72

INTRODUCCIÓN

Los sistemas SCADA (*Supervisory Control And Data Acquisition*) son aplicaciones para la supervisión, adquisición y control de datos, diseñados especialmente para funcionar sobre ordenadores en el control de procesos industriales en una amplia gama de industrias. Estos sistemas ejecutan tareas críticas como la recolección periódica, procesamiento y monitoreo de información, control remoto de dispositivos de campo y visualización de alarmas.

El Centro de Informática Industrial (CEDIN), perteneciente a la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), está dedicado a desarrollar productos y servicios informáticos de automatización industrial y computación gráfica. Estos productos tienen un alto valor agregado que cumplen con las necesidades y expectativas de los clientes, potenciando la formación especializada y la investigación (1). En la actualidad tiene contrato con la industria petrolera de Venezuela, para la cual desarrolla y despliega el sistema SCADA Guardián del ALBA (GALBA).

Este sistema, por el volumen de datos que manipula, demanda gran cantidad de recursos en la computadora o servidor donde debe ser instalado. Como producto a ser entregado, el mismo debe pasar por un laboratorio de pruebas en el que se deben desarrollar pruebas funcionales y no funcionales. Las pruebas de rendimiento son un caso particular de pruebas no funcionales a realizar en un sistema, este proceso suele tornarse engorroso pues se tendría que estar chequeando de una manera periódica el comportamiento del computador, los procesos del sistema y así detectar una posible fuga de memoria o un uso excesivo de CPU entre otros parámetros.

En la actualidad no se cuenta con una herramienta adecuada que sea capaz de informatizar el proceso de supervisión del comportamiento en el tiempo del uso del procesador y el consumo real de la memoria.

Como resultado de lo analizado anteriormente surge el siguiente **problema científico** a resolver: ¿Cómo mejorar los procesos de prueba de software a partir del monitoreo del consumo de procesador y memoria?

A partir del problema científico se puede definir como **objeto de estudio,** los sistemas de monitoreo de software, teniendo como **campo de acción,** el monitoreo del consumo de recursos en el SCADA GALBA.

De acuerdo con el problema científico planteado, la **idea a defender** es la siguiente: Si se implementa la herramienta para el monitoreo y análisis de los procesos del sistema SCADA Guardián del ALBA, entonces se dotará al equipo de prueba de una herramienta que facilitará y agilizará los procesos de prueba de software.

Para dar solución al problema planteado anteriormente se propone como **objetivo general:** desarrollar una herramienta que permita el monitoreo y análisis del consumo del procesador y la memoria de los procesos del GALBA.

Para desarrollar satisfactoriamente la investigación se han trazado las siguientes **tareas investigativas:**

- Estudio del arte de las herramientas de monitoreo.
- Estudio del lenguaje Python, para el mejor entendimiento del framework y el desarrollo del módulo de recolección de información de los procesos del sistema.
- Estudio de Django como Framework para desarrollo de la aplicación web.
- Estudio de jQuery como biblioteca JavaScript del lado del cliente.
- Elaboración, diseño e implementación de la herramienta según las características y requisitos identificados.
- Realización de pruebas de aceptación para comprobar el correcto funcionamiento de la herramienta.

Una vez terminado el desarrollo de la aplicación, se contará con una herramienta que ayudará a los usuarios a mejorar los procesos de prueba de software, permitiendo:

- la persistencia de los resultados de las pruebas en el tiempo.

- la gestión de las pruebas.
- la visualización en forma de gráficas de los datos recolectados.
- la generación de reportes históricos de comportamiento de los indicadores CPU y memoria.

Para el desarrollo de las tareas de investigación se combinan diferentes métodos en la búsqueda y procesamiento de la información como son:

Métodos Teóricos:

- **Analítico-Sintético:** Utilizado para analizar los documentos y conceptos fundamentales sobre las herramientas y tecnologías permitiendo la extracción de los elementos que ayuden en la implementación.

- **Análisis histórico-lógico:** Utilizado para conocer, con mayor profundidad los antecedentes y las tendencias actuales de las herramientas y tecnologías.

- **Modelación:** Utilizado para definir y representar gráficamente las funcionalidades del sistema usando el Lenguaje Unificado de Modelado (UML).

Método Empíricos:

- **Experimentos:** Empleado en la elaboración de prototipos funcionales, con el objetivo de comprobar la efectividad de la implementación de las funcionalidades.

Este documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas y anexos.

En el *Capítulo 1: "Fundamentación teórica y tecnologías para el desarrollo"*, se introducen los conceptos de pruebas de software, sistemas SCADA, aplicaciones de monitoreo. Además se definen las metodologías, lenguajes y tecnologías para el desarrollo de la herramienta para el monitoreo y análisis de los procesos del GALBA.

En el *Capítulo 2: "Descripción del sistema, planificación y diseño",* se describe el flujo actual de eventos, se elabora una propuesta del sistema y se especifican las historias de usuario. Se planifica el proceso de desarrollo de la herramienta, se expone el diseño del sistema, la arquitectura y los patrones de diseño.

En el *Capítulo 3: "Implementación y prueba"*, se define el estándar de codificación de python, el diagrama de despliegue, diagrama Entidad-Relación de la base de datos y las pruebas realizadas a la herramienta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA Y TECNOLOGÍAS PARA EL DESARROLLO.

Los sistemas SCADA desempeñan un papel fundamental en la automatización industrial. La tendencia actual de estos sistemas es a estar instalados de forma distribuida dentro de una red de computadoras. Esto dificulta la realización de pruebas para determinar la calidad del sistema. Existe un grupo de pruebas destinadas a comprobar el rendimiento de los sistemas que a través de aplicaciones para el monitoreo se puede observar el comportamiento de los procesos.

En el presente capítulo se tiene como objetivo realizar un estudio sobre las herramientas para el monitoreo de consumo de recursos en aplicaciones de software, caracterizar algunas de estas, así como realizar el estudio de la metodología, herramientas y lenguajes de programación a usar para el desarrollo de la solución.

1.1 Sistemas SCADA

En la actualidad los sistemas SCADA desempeñan un papel fundamental en la automatización industrial, estos comprenden las soluciones de aplicación que necesitan de la captura de información de un proceso o planta industrial, la cual es empleada para realizar análisis con los que se pueden obtener importantes indicadores que permiten una realimentación del proceso. De forma general se consideran como funciones básicas de un sistema SCADA la supervisión y control remoto de instalaciones, procesamiento de información, generación de reportes, gestión de alarmas, almacenamiento de información histórica, presentación de gráficos de tendencias y programación de eventos entre otros. (2)

Los primeros sistemas SCADA se caracterizaban por ser monolíticos, las generaciones actuales de estos sistemas tienden a ser distribuidos. Un SCADA distribuido es aquel sistema en que sus componentes tanto de hardware como de software se encuentran conectados en una red de área local, cada uno de estos, con una función específica dentro del sistema. Los componentes, mediante la comunicación y coordinación entre

sí, permiten que el sistema funcione como una entidad única, cuyo objetivo fundamental es lograr el control y supervisión de los procesos industriales.

1.1.1 Sistema SCADA Guardián del ALBA (GALBA)

El GALBA es un sistema distribuido en módulos que trabajan de manera conjunta posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un software para la distribución de los servicios en la red, conocido como “middleware” o software de comunicación entre aplicaciones. La distribución de los módulos existentes en el SCADA permite obtener configuraciones escalables en dependencia de los requisitos que presente cada aplicación. Es un sistema en tiempo real y presenta una arquitectura distribuida. Está dividido en varios subsistemas entre los que se encuentran (2):

- **Middleware:** Es la capa de software, que se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Este módulo tiene como finalidad proporcionar la capa de comunicación de alto nivel, tanto sincrónica, como asincrónica, para la comunicación de todos los módulos que conforman el sistema SCADA.

- **Adquisición:** Es el encargado de la adquisición, recepción, procesamiento y distribución de los datos provenientes del campo.

- **Configuración:** El servicio de configuración está formado por un grupo de componentes cada uno con una tarea específica y una base de datos que contendrá las configuraciones de cada uno de los módulos que conformen el proyecto activo. Es el encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA.

- **Almacenamiento de datos históricos:** Es el encargado de almacenar la información del sistema para que posteriormente pueda ser empleada, por ejemplo, en generación de reportes, tendencias o en gestión de producción. La

base de datos histórica (BDH) contendrá la información persistente de los datos recolectados de los dispositivos.

- **Seguridad:** Proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos, además brinda las herramientas necesarias para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos, tales como fallas de energía, problemas de red o servidores.

- **Visualización o HMI:** Se encarga de representar en un ordenador, los procesos que ocurren en el campo en tiempo real, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador diferentes niveles de control en dependencia de sus niveles de privilegios. Este módulo permite al operador el contacto directo con el sistema y realizar la supervisión y el control del proceso en general.

Este sistema debido a la distribución de sus componentes dentro de una red de computadoras añade dificultad a las pruebas de software, proceso necesario para determinar el estado de la calidad del software.

1.2 Pruebas de software

En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requisitos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante las técnicas de prueba. El proceso de pruebas, objetivos, métodos y técnicas usados se describen en el plan de prueba (3).

El proceso de pruebas no va orientado a demostrar la ausencia de fallos en el software, sino todo lo contrario: encontrar cuantos fallos existan, por escondidos que se encuentren o difícilmente reproducibles que sean.

Entre los diversos tipos de pruebas que se realizan al software se encuentran:

- **Prueba unitaria:** Es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

- **Prueba funcional:** Es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.

- **Pruebas de integración:** Son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba de todos los elementos unitarios que componen un proceso, hecha en conjunto.

- **Caja blanca:** Es un tipo de prueba de software que se realiza sobre las funciones internas de un módulo.

- **Caja negra:** Ejercitan los requisitos funcionales desde el exterior del módulo.

- **Prueba no funcional:** Se utilizan para verificar que la aplicación desarrollada cumple con los requisitos no funcionales.

1.2.1 Pruebas no funcionales

Es la verificación de los requisitos no funcionales de una aplicación. Las diferentes pruebas no funcionales son comúnmente confundidas y sus alcances tienden a sobreponerse (4).

Dentro de esta disciplina se pueden diferenciar los siguientes grupos:

- **Pruebas de disponibilidad:** Son las que verifican que sus procesos de negocio están funcionando de forma correcta, al margen del estado de sus recursos hardware.

- **Pruebas de seguridad:** Estas cubren el proceso de evaluar la seguridad de sus sistemas desde un punto de vista externo y sin conocimiento previo de los

mismos. Los sistemas y políticas de seguridad son analizados exhaustivamente con el fin de encontrar fallos de seguridad, tanto en el diseño, como en la implementación de la aplicación.

- **Pruebas OAT:** Las pruebas de Aceptación Operacional (OAT) se realizan como paso inmediatamente anterior a la puesta en producción de un sistema, una vez finalizadas las Pruebas de Aceptación del Usuario (UAT). El objetivo general de este tipo de pruebas es comprobar que la aplicación dispone de la fiabilidad y el soporte necesarios para su puesta en marcha en producción.

- **Monitorización de entornos:** Este servicio permite a las empresas detectar proactivamente los problemas (ya sean software o hardware) en sus entornos antes de que estos ocurran. Conociendo los problemas antes de que lo hagan los clientes es posible asignar prioridades a su resolución (en base al impacto en el negocio, los acuerdos de nivel de servicio).

- **Pruebas de Rendimiento:** Este tipo de pruebas no se realiza para encontrar defectos en una aplicación, sino para eliminar cuellos de botella y establecer una línea base que pueda ser utilizada en un futuro para comparar el incremento en el rendimiento de la aplicación bajo pruebas. Este servicio comúnmente incluye las pruebas de carga, estrés y resistencia.

1.2.1.1 Pruebas de rendimiento

Las pruebas de rendimiento se ejecutan tanto para determinar cómo responde un sistema ante una cierta carga, como para validar otros atributos relacionados con la calidad, como pueden ser la escalabilidad o el uso de recursos entre otros.

Las pruebas de rendimiento pueden tener distintos propósitos. Por ejemplo, pueden demostrar que el sistema cumple los criterios de rendimiento o pueden medir qué partes del sistema o qué carga hacen que el sistema rinda de forma incorrecta (5).

Existen distintos tipos de pruebas de rendimiento como son:

- **Pruebas de carga:** Una prueba de carga se ejecuta para comprender el comportamiento de una aplicación ante una carga determinada. Esta carga puede ser el número de usuarios esperado en producción o un número de transacciones durante un tiempo determinado. El resultado de esta prueba dará el tiempo de respuesta de todas las transacciones críticas.

- **Pruebas de estrés:** Estas pruebas son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento. Tiene como finalidad determinar la robustez de una aplicación cuando la carga es extrema.

- **Pruebas de resistencia (SOAK):** Esta prueba se realiza con el fin de determinar si la aplicación puede mantener la carga esperada de manera continua y durante un largo tiempo. El objetivo principal de este tipo de pruebas es verificar que no existen fugas de memoria o procesos que pierdan rendimiento tras un cierto período de tiempo.

- **Pruebas de picos:** Tal y como el nombre sugiere, este tipo de pruebas se realizan insertando la carga en el sistema en forma de “picos” que se irán lanzando en distintos momentos de la prueba y que permitirán comprender el comportamiento de la aplicación ante cambios bruscos de carga.

Uno de los indicadores que se necesita monitorear para las pruebas de rendimiento es el consumo de los recursos de los procesos relacionados con el software en ejecución.

1.3 Monitoreo de procesos

El contexto de un programa que está en ejecución es lo que se llama un proceso. Este contexto puede ser más procesos hijos que se hayan generado del principal (proceso padre), los recursos del sistema que esté consumiendo, sus atributos de seguridad (tales como su propietario y permisos de archivos así como roles), entre otros (6).

El monitoreo de procesos, a rasgos generales, consiste en la observación del curso de uno o más parámetros para detectar eventuales anomalías en el comportamiento de los procesos. Algunos de los parámetros son el identificador, el usuario que ejecuta el

proceso, el comando que lanzó el proceso, el consumo de los recursos del sistema como la memoria (memoria virtual y memoria residente en el sistema), porciento de uso del procesador, entre otros.

En una computadora, las aplicaciones de software utilizan diversos recursos del sistema como el procesador y la memoria, para su ejecución y buen funcionamiento.

1.3.1 Consumo del procesador.

La Unidad Central de Procesamiento, en inglés: Central Processing Unit (CPU) o simplemente el procesador, es el componente de una computadora y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

El consumo del procesador es un término utilizado para describir cuánto está trabajando el procesador. Puede variar dependiendo de los tipos de tareas que estén siendo ejecutadas por el procesador. Este comportamiento puede ser monitoreado para ver cuánto de la capacidad del procesador está en uso. Cuando el uso de la CPU alcanza el 100% no hay más capacidad de espacio para ejecutar otros programas. Cuando el porcentaje de uso de la CPU empieza a superar el 100% se requiere entonces de la toma de acciones adicionales (7).

1.3.2 Consumo de la memoria

La Memoria de Acceso Aleatorio, en inglés: Random Access Memory (RAM), es donde la computadora guarda los datos que está utilizando en el momento. El almacenamiento es considerado temporal porque los datos y programas permanecen en ella mientras que la computadora este encendida o no sea reiniciada. Como lo indica su nombre, es posible acceder a cualquier ubicación de ella aleatoria y rápidamente siempre en un mismo intervalo de tiempo, normalmente pequeño (8).

Uno de los problemas más comunes de las aplicaciones es la fuga de memoria, que ocurre cuando un bloque de memoria reservada no es liberado. Cuando se produce una pérdida de referencia hay una fuga de memoria: se solicita memoria al ordenador y

no es liberada cuando se deja de usar. Un programa con fugas de memoria corre el riesgo de consumir toda la memoria disponible en el ordenador. (9)

1.4 Aplicaciones de monitoreo

Para el monitoreo del comportamiento de los procesos de un sistema, existen varias aplicaciones y comandos para realizar esta labor, algunos brindan más información que otros en dependencias del objetivo para el cual fueron diseñados.

1.4.1 Módulo psutil

Este módulo brinda una interfaz para la recolección de información de todos los procesos en ejecución y la utilización del sistema (procesador, memoria, red) de una manera sencilla utilizando Python, con funcionalidades ofrecidas por herramientas como *ps*, *top*, *nice*, entre otras. Actualmente tiene soporte para los sistemas operativos Windows, GNU/Linux, OSX y Free-BSD (10).

1.4.2 Process snapshot (ps)

Ps permite informar sobre el estado de los procesos, está basado en el sistema de archivos */proc*, es decir, lee directamente la información de los archivos que se encuentran en este directorio. Tiene una gran cantidad de opciones, incluso estas opciones varían dependiendo del estilo en que se use el comando (6).
Estas variaciones sobre el uso de *ps* son las siguientes:

- Estilo UNIX, donde las opciones van precedidas por un guión -
- Estilo BSD, donde las opciones no llevan guión
- Estilo GNU, donde se utilizan nombres de opciones largas y van precedidas por doble guión --

```
#> ps aux    (formato BSD sin guión, u muestra usuarios y demás columnas)
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1648   556 ?        Ss   16:59   0:01 init [5]
root         2  0.0  0.0      0     0 ?        S<   16:59   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   16:59   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   16:59   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   16:59   0:00 [migration/1]
```

Figura 1: Salida del comando ps.

1.4.3 Top

Top es una herramienta muy usada y útil para el monitoreo en tiempo real del estado de los procesos y de otras variantes del sistema, se ejecuta desde la línea de comandos, es interactivo y por defecto se actualiza cada 3 segundos (6).

```
$> top
top - 13:07:30 up 8 days,  6:44,  4 users,  load average: 0.11, 0.08, 0.08
Tasks: 133 total,   1 running, 131 sleeping,   0 stopped,   1 zombie
Cpu(s):  0.0%us,  0.2%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:    497356k total,   472352k used,    25004k free,    21500k buffers
Swap:  1156640k total,   257088k used,   899552k free,    60420k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
26156 sergon    15   0  2160 1016  784 R    1  0.2   0:00.93 top
    1 root      15   0  2012  616  584 S    0  0.1   0:00.98 init
    2 root      RT   0     0    0    0 S    0  0.0   0:00.29 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
    5 root      RT   0     0    0    0 S    0  0.0   0:00.38 migration/1
```

Figura 2: Salida de comando top

1.4.4 Htop

Htop es un interactivo visor de procesos hecho para GNU/Linux. Está diseñado para reemplazar a *top*. Muestra una lista actualizada de los procesos que se ejecutan en una computadora, generalmente ordenados por la cantidad de uso de la CPU. *Htop* ofrece una lista completa de los procesos en ejecución, proporciona información visual acerca del estado del procesador, de la memoria swap y la memoria RAM (11).

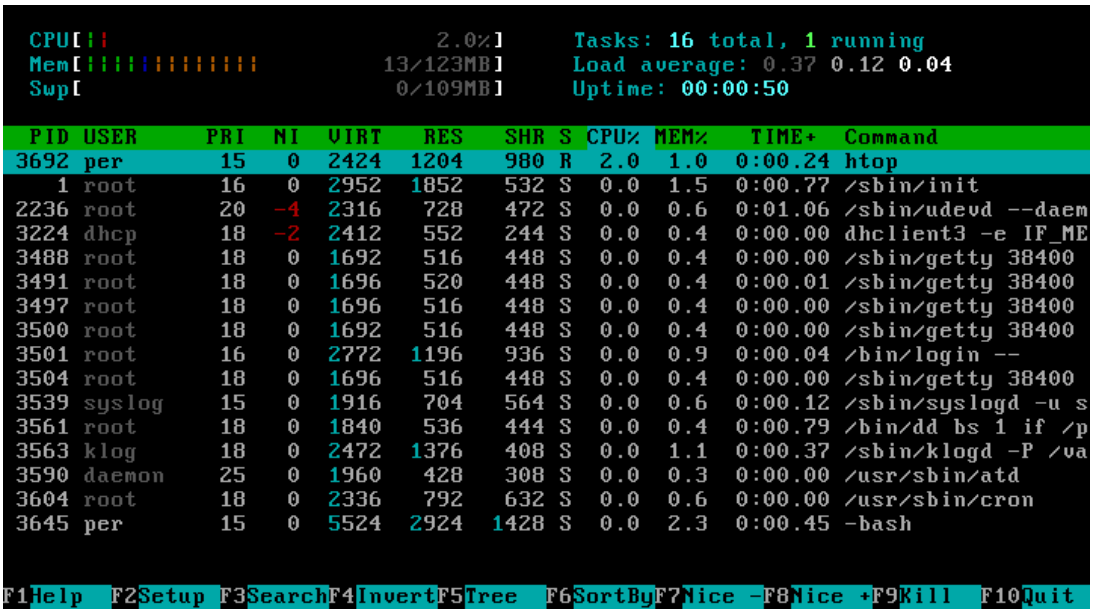

Figura 3: Salida comando htop

1.4.5 Gnome System Monitor

Gnome System Monitor es un visor de procesos y monitor del sistema para el escritorio GNOME de GNU/Linux con una interfaz agradable y fácil de usar. Tiene algunas características interesantes, como una vista de árbol para las dependencias entre procesos, iconos para procesos, la habilidad para ocultar procesos que no quieres ver, historial gráfico del tiempo de uso de CPU/memoria/swap, la habilidad para terminar/priorizar procesos (sólo usuario root), así como las características estándar que podrían esperarse de un visor de procesos (12).

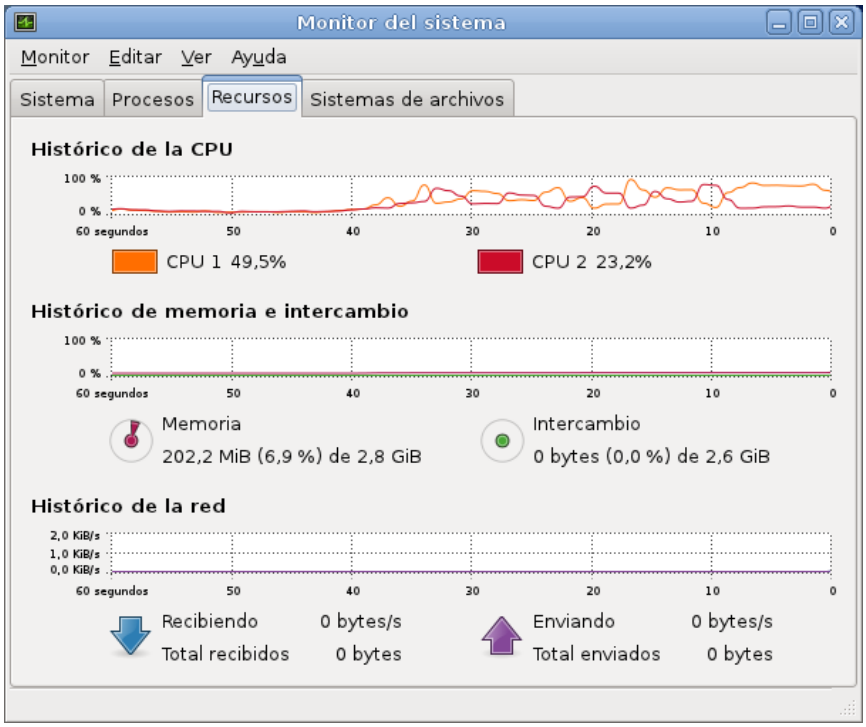

Figura 4: Imagen de la herramienta *Gnome System Monitor.*

1.5. Evaluación de aplicaciones de monitoreo para el desarrollo de la herramienta.

En este acápite se realiza un análisis de las aplicaciones de monitoreo con el objetivo de seleccionar la que mayor funcionalidades brinde y que pueda ser utilizada como fuente de datos para la herramienta de monitoreo y análisis a desarrollar.

1.5.1. Criterios de evaluación

Se definen a continuación un conjunto de características que debe soportar la herramienta seleccionada:

- **Funcionalidades que brinda:** Se necesita una herramienta que sea capaz de brindar todos los datos o indicadores a utilizar en la herramienta de análisis y monitoreo para almacenar en tiempo real y poder graficar dicha información.

- **Implementación con herramientas libres y código abierto:** Con el propósito de utilizar al máximo, funcionalidades de las herramientas existentes para el desarrollo

de las aplicaciones, es importante la utilización de las herramientas y tecnologías bajo la Licencia Pública General (GPL).

- **Facilidad de uso:** Es importante que la herramienta seleccionada brinde los datos de manera sencilla, sean fáciles de utilizar y que la interacción del desarrollador con el sistema o producto sea rápida y efectiva.
- **Multiplataforma:** Debe ser capaz de soportar varios sistemas operativos, principalmente Linux y Windows.
- **Persistencia de datos:** La herramienta debe permitir que los datos obtenidos puedan ser almacenados en un esquema de persistencia para su posterior utilización.

1.6. Módulo propuesto para el desarrollo de la herramienta.

Como consecuencia del estudio y análisis de los resultados arrojados tras la evaluación de cada una de las aplicaciones de monitoreo abordadas previamente en el modelo de decisión (ver Figura 5), se determina que el más apropiado para el desarrollo de la herramienta de monitoreo y análisis, es el módulo *psutil*.

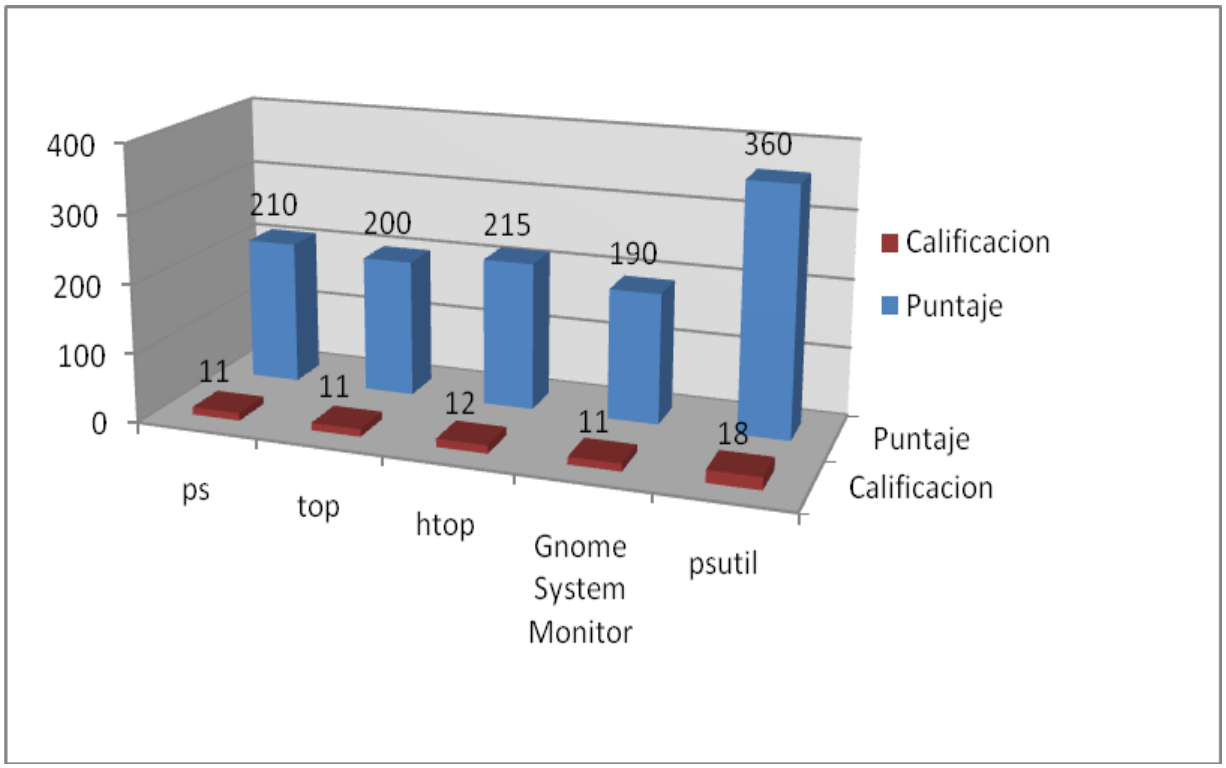

Figura 5: Matriz de decisión de aplicaciones de monitoreo para la implementación de la herramienta de monitoreo y análisis.

1.7 Metodología, lenguajes y tecnologías para el desarrollo.

La calidad de un producto de software es determinada en gran medida por la calidad del proceso utilizado para desarrollarlo y mantenerlo, por lo que se dedica esta sección a la selección de la metodología de desarrollo, lenguaje de programación, herramienta y lenguaje de modelado, así como la herramienta de desarrollo a utilizar en la herramienta para el monitoreo y análisis de los procesos del GALBA.

1.7.1 Metodología de desarrollo

En un proyecto de desarrollo de software la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. Una metodología es un proceso. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.

Una de las metodologías de desarrollo de software utilizada en la actualidad es la *Extreme Programing* (XP), utilizada para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto (13).

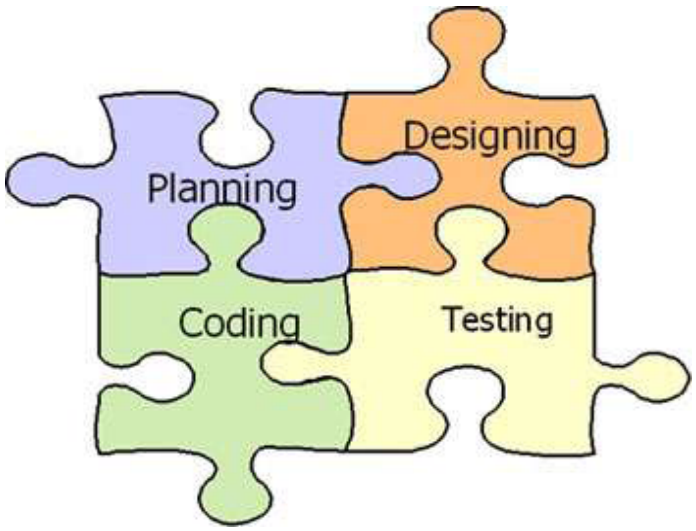

Figura 6: Metodología *Extreme Programing*

Características de la metodología XP:

- *Pruebas Unitarias:* Se basa en las pruebas realizadas a los principales procesos, de tal manera que se puedan hacer pruebas de las fallas que pudieran ocurrir.
- *Refabricación:* Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- *Programación en pares:* Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.

- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Lo fundamental en este tipo de metodología es:

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

1.7.2 Lenguaje de Modelado

UML (*Unified Modeling Language*) es desde finales de 1997 un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software (14).

Es válido destacar que UML es un lenguaje de modelado, no un método o un proceso; actualmente se publicó la versión 2.0, que proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les posibilita aprovechar mejor los modelos y generar así una mayor cantidad de código reduciendo en gran medida el ciclo de desarrollo de sus aplicaciones.

UML constituye un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, no brinda el total éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos al posibilitar una nueva y fuerte integración entre las herramientas, los procesos y los dominios. Además UML posee una corrección fiable de errores en todas las etapas de la construcción del software y es aplicable para tratar asuntos en sistemas complejos de misión crítica, tiempo real y cliente/servidor (15).

1.7.3 Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*), en español Ingeniería de Software Asistida por Computadora, son aplicaciones informáticas utilizadas en el proceso de desarrollo de software en tareas como: realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente a partir del diseño, compilación automática, documentación o detección de errores entre otras (16).

Visual Paradigm es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste (17).

Las características principales de esta herramienta son las que a continuación se enuncian:

- Soporte de UML.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad-Relación.
- Editor de figuras.

1.7.4 Framework de desarrollo

Un framework, en el desarrollo del software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio (18).

Django es un framework de alto nivel, escrito en *Python* que facilita el desarrollo de aplicaciones web dinámicas. Es un framework que abstrae a los programadores de los problemas comunes del desarrollo web y acelera las tareas más frecuentes en la programación. Proporciona un método de mapear las URLs ejecutando un código en especial para cada una. Permite mostrar y validar formularios de manera muy simple, manipulando el código del formulario y adaptándolo a las necesidades de la aplicación.

El mismo convierte los datos enviados por los usuarios en estructuras de datos que pueden ser manipuladas fácilmente. A través de plantillas ayuda a separar el contenido de la presentación evitando tener que manipular la lógica de negocio cuando se necesite realizar cambios de apariencia en la página. Permite lidiar con trescientas peticiones web por segundo. Su ORM es muy poderoso, permitiendo definir los modelos de datos enteramente en Python. Además provee soporte para el uso de plantillas, filtros, etiquetas y formularios (19).

1.7.5 Bibliotecas gráficas

1.7.5.1 JQuery

Es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (Document Object Model), manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX (*Asynchronous JavaScript And XML*) a páginas web.

JQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia del *Massachusetts Institute of Technology (MIT)* y la Licencia Pública General de GNU v2 (GPL), permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (20).

1.7.5.2 HighCharts

Permite añadir gráficas animadas a un sitio web en una gran variedad de formatos (spline, área, barra, columna, pastel, entre otras). Está implementado en Javascript por lo que es sencillo de incorporar a un sitio web. Es compatible con cualquier navegador que soporte Javascript, por lo que funciona en Chrome, Firefox, Safari, Internet Explorer ya sea con Linux, Windows, OSX, Android, entre otros (21).

1.7.6 Lenguajes de programación.

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevados a cabo por las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una computadora (22). Por las características de la aplicación, es necesario el uso del lenguaje Python para el desarrollo con el framework Django y el lenguaje JavaScript para realizar validaciones del lado del cliente y el trabajo con las gráficas.

1.7.6.1 Python

Es un lenguaje muy expresivo, es decir, los programas Python son muy compactos: un programa Python suele ser más corto que su equivalente en lenguajes como C. Python llega a ser considerado por muchos un lenguaje de programación de muy alto nivel. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si se utilizaran otros lenguajes de programación.

Python ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje. Su entorno de ejecución detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información para detectarlos y corregirlos (22).

1.7.6.2 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. Es un lenguaje que se integra directamente en páginas HTML. Tiene como características principales las siguientes (23):

- Es interpretado por el cliente.
- Está basado en objetos.
- No es necesario declarar los tipos de variables que van a utilizarse.
- Las referencias a objetos se comprueban en tiempo de ejecución.
- No puede escribir automáticamente al disco duro.

1.7.7 Entorno de desarrollo

Eclipse es un entorno de desarrollo integrado (IDE) multiplataforma, fue desarrollado por IBM y en la actualidad lo mantiene la Fundación Eclipse. En sus inicios este IDE se desarrolló para los programadores que utilizaban el lenguaje Java. Actualmente emplea extensiones (*plugins*) para agregar funcionalidades en dependencia de las necesidades del desarrollador, gracias a estas extensiones se ha extendido el soporte de Eclipse hasta lenguajes como C/C++, *Python*, PHP, JavaScript y otros.

Además permite utilizar lenguajes de procesado de texto, aplicaciones de red y Sistemas de Gestión de Bases de Datos. Brinda soporte para Sistemas de Control de Versiones e incluye extensiones para realizar pruebas de unidad (24).

1.7.8 Sistema Gestor de Base de Datos (SGBD)

PostgreSQL es el gestor de bases de datos objeto-relacional de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión (permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros), soportando casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Delphi, Python, Perl, PHP y Bash.), además de proveer una arquitectura que ha ganado una fuerte reputación por su fiabilidad e integridad de sus datos.

Permite bloqueos de tabla y filas para controlar el acceso concurrente a los datos en tablas. Algunos de estos modos de bloqueo los adquiere PostgreSQL automáticamente antes de la ejecución de una declaración, mientras que otros son proporcionados para ser usados por las aplicaciones. Brinda soporte de tipos y funciones de usuario, incorpora una estructura de datos Array: Conectividad TCP/IP, JDBC y ODBC. Por último señalar que es multiplataforma, estando disponible en casi cualquier sistema Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows (25).

1.8 Conclusiones parciales.

En este capítulo se definieron las herramientas y tecnologías a utilizar, como metodología de desarrollo de software se eligió *Extreme Programming* y para realizar el modelado del sistema la herramienta case *Visual Paradigm*, apoyándose en el Lenguaje Unificado de Modelado (UML). Después de un proceso de comparación utilizando una matriz de decisión de calificación contra puntajes se seleccionó el módulo *psutil* para obtener los datos a gestionar en el sistema. Se seleccionó *Eclipse* como IDE, *Python* y *JavaScript* como lenguajes de programación y como framework Django; finalmente se escogió el gestor de base de datos *PostgreSQL* para el tratamiento de los datos.

CAPÍTULO 2 DESCRIPCIÓN DEL SISTEMA. PLANIFICACIÓN Y DISEÑO.

La planificación y el diseño del software son etapas importantes en su ciclo de desarrollo. Es clave durante su desarrollo que se logre una buena planificación antes de entrar en el diseño del mismo. Las aplicaciones deben estar soportadas por una correcta selección de su arquitectura para organizar y comprender mejor el sistema. El presente capítulo aborda los temas referentes a cada una de estas etapas.

2.1 Flujo actual de eventos

El departamento de construcción de componentes es el encargado de desarrollar los diferentes módulos que conforman el sistema SCADA GALBA. Luego pasan al departamento de integración y despliegue el cual es el encargado de integrar todos estos módulos y los pone a disposición del equipo de pruebas. Este equipo le realiza las distintas pruebas a los módulos, ya sean pruebas funcionales, pruebas de caja negra, pruebas no funcionales, entre otras.

Para ver el comportamiento de los procesos del GALBA bajo ciertas condiciones, el probador utiliza por lo general herramientas como el monitor del sistema, comandos como *top* y *ps*, que les permite ver el consumo de los diferentes recursos del sistema por parte de los procesos, pero esta información no persiste, evitando así un posible análisis de dicho comportamiento y detectar posibles errores.

2.2 Propuesta del sistema

Se implementará una herramienta que permita el monitoreo y análisis del consumo de procesador y memoria de los procesos del GALBA y que la información que se recolecte persista en una base de datos para su posterior análisis.

La herramienta estará compuesta por dos módulos, uno podrá ser ejecutado en cualquier computadora donde esté instalado al menos un módulo del GALBA, encargado de recolectar la información de cada proceso y hacerla persistente. El otro módulo será una aplicación web para que pueda ser accedida desde cualquier lugar y es el encargado de la visualización en forma de tendencias de dicha información.

También brindará la posibilidad de mostrar las tendencias en ciertos intervalos de tiempo, de manera que pueda ser analizada posteriormente.

Otro de los aspectos de relevancia es que se pueden generar reportes y así tener constancia para una futura comparación. Mediante el framework Django se logrará que todo esté bien organizado a través de su sistema de URL, vistas y plantillas. Igualmente cuenta con una interfaz de administrador bien robusta que permite gestionar las computadoras y procesos que serán monitoreados.

2.2.1 Personal relacionado con el sistema

El personal relacionado con el sistema es todo aquel que interactúe con la aplicación y que obtiene un resultado de valor de uno o varios procesos que se ejecutan en el mismo.

Personal relacionado con el sistema	Descripción
Probador	Hace uso de la aplicación, beneficiándose de sus funcionalidades, es el responsable de gestionar las computadoras y procesos además de visualizar los comportamientos de las pruebas.
Administrador	Realiza las acciones del usuario Probador además de gestionar los usuarios del sistema.

Tabla 1 Personal relacionado con el sistema

2.3 Historias de Usuario

Las historias de usuario (HU) reflejan los requisitos del cliente. Tienen el mismo propósito que los casos de uso, con la excepción de que no se limitan a la descripción de la interfaz de usuario. Se utilizan en la creación de las pruebas, para verificar si el programa cumple con lo que especifica cada HU. La principal diferencia entre estas y la tradicional especificación de requisitos, es el nivel de detalle. Las HU solamente proporcionarán los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha HU (26).

Historias de usuario	
Número: 1	**Nombre Historia de Usuario:** Autenticar Usuario
Actor: Probador	**Iteración Asignada:** 1
Prioridad de Negocio: Media	**Puntos Estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Se debe permitir a cualquier usuario que acceda al sistema, la opción de introducir sus datos (usuario y contraseña) para de esta manera realizar una verificación y entonces asignarle los permisos que le competen en la aplicación.	
Observaciones: El usuario estará previamente definido por el departamento de prueba. Desde el módulo de administración que provee el framework Django, se creará el usuario con el cual se trabajará.	

Tabla 2 HU Autenticar usuario

Historias de usuario	
Número: 2	**Nombre Historia de Usuario:** Gestionar usuario
Actor: Administrador	**Iteración Asignada:** 1
Prioridad de Negocio: Media	**Puntos Estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: El administrador podrá crear, modificar, listar o eliminar todos los usuarios que harán uso del sistema.	
Observaciones:	

Tabla 3 HU Gestionar usuario

Historias de usuario	
Número: 3	**Nombre Historia de Usuario:** Gestionar computadora
Actor: Probador	**Iteración Asignada:** 1
Prioridad de Negocio: Media	**Puntos Estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: El probador podrá crear, modificar, listar o eliminar todas las computadoras en las cuales se van a desarrollar las pruebas.	

Observaciones: El nombre de la computadora debe coincidir con el nombre real de la computadora donde se realizarán las prueba, la dirección IP será única en cada caso.

Tabla 4 HU Gestionar computadora

Historias de usuario	
Número: 4	**Nombre Historia de Usuario:** Gestionar proceso
Actor: Probador	**Iteración Asignada:** 1
Prioridad de Negocio: Media	**Puntos Estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: El probador podrá crear, modificar, listar o eliminar todos los procesos a los cuales se van a realizar las pruebas.	
Observaciones: Las acciones de crear, modificar o eliminar, se realizarán cuando se edite una computadora.	

Tabla 5 HU Gestionar proceso

Historias de usuario	
Número: 5	**Nombre Historia de Usuario:** Monitorear proceso
Actor: Probador	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 3
Riesgo en desarrollo: Medio	**Puntos Reales:** 3
Descripción: El probador podrá ejecutar en la PC con los servicios del GALBA la aplicación encargada de monitorear los procesos utilizando los datos de las computadoras y procesos configurados anteriormente.	
Observaciones: Es indispensable tener configurado como mínimo una computadora con un proceso para poder realizar esta historia de usuario.	

Tabla 6 HU Monitorear proceso

Historias de usuario	
Número: 6	**Nombre Historia de Usuario:** Visualizar comportamiento de la prueba.

Actor: Probador	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 3
Riesgo en desarrollo: Alto	**Puntos Reales:** 3
Descripción: **Seleccionar computadora** El probador selecciona la computadora a la cual quiere ver los detalles de las pruebas realizadas. **Seleccionar prueba** El probador selecciona una de las pruebas que está almacenada en la base de datos para ver los valores recolectados de uno de los procesos que se le especifique. **Seleccionar proceso** El probador selecciona el proceso al cual quiere ver los detalles de las pruebas, ya sea ver un proceso al que se le está realizando una prueba en ese instante o de una prueba ya almacenada en la base de datos. **Graficar valores** El sistema grafica los valores recolectados en las pruebas en forma de tendencias. - Los valores de una prueba terminada serán mostrados en su totalidad en una gráfica y podrán ser visualizadas por intervalos de tiempo en otra gráfica para su mayor entendimiento. - De una prueba en ejecución serán mostrados los últimos valores recolectados hasta ese instante.	
Observaciones:	

Tabla 7 HU Visualizar comportamiento de la prueba.

Historias de usuario	
Número: 7	**Nombre Historia de Usuario:** Generar reporte
Actor: Probador	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 3
Riesgo en desarrollo: Alto	**Puntos Reales:** 3
Descripción: Se obtiene un reporte de la gráfica que muestra un intervalo de tiempo dado.	
Observaciones:	

Tabla 8 HU Generar reporte

Historias de usuario	
Número: 8	**Nombre Historia de Usuario:** Requisitos de software
Descripción: Se debe tener instalado una computadora con el sistema operativo GNU/Linux Debian Squeeze con los siguientes paquetes instalados: • Django 1.3 • PostgreSQL >=8.4 • Python >=2.6 • Psutil • Google Chrome >=14	
Observaciones:	

Tabla 9: HU Requisitos de software

Historias de usuario	
Número: 9	**Nombre Historia de Usuario:** Requisitos de Hardware
Descripción: Para la ejecución de la aplicación web se debe tener una computadora con las siguientes prestaciones: • Procesador Intel Pentium IV • Frecuencia del CPU: 2.4GHz • Memoria RAM: 512MB Para la ejecución del módulo de monitoreo los Requisitos de Hardware dependerán de los del GALBA.	
Observaciones:	

Tabla 10: HU Requisitos de Hardware.

Historias de usuario	
Número: 10	**Nombre Historia de Usuario:** Restricciones en el diseño e implementación.
Descripción: Para el desarrollo de la solución se definen una serie de restricciones:	

• Lenguaje de programación: Python y JavaScript • Entorno integrado de desarrollo (IDE): Eclipse Indigo • Framework: Django
Observaciones:

Tabla 11: HU Restricciones en el diseño e implementación.

Historias de usuario	
Número: 11	**Nombre Historia de Usuario:** Requisitos de apariencia o interfaz externa.
Descripción: Se desarrollará una aplicación web que permita a los usuarios acceder a la herramienta mediante un navegador. La interfaz debe ser sencilla, de fácil uso, e intuitiva en cuanto a los pasos a seguir a la hora de probar.	
Observaciones:	

Tabla 12: HU Requisitos de apariencia o interfaz externa.

2.4 Planificación y entrega

La planificación es una fase corta en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (27).

Los programadores realizan una estimación del esfuerzo necesario de cada una de las HU en dependencia de la prioridad establecida por el cliente. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

2.4.1 Plan de entrega

Una vez que el cliente culmina la elaboración de las HU, se comienza con la creación del Plan de Entregas. El mismo se hace con la intención de que los programadores obtengan una estimación de dichas historias en cuanto al nivel de detalle, o sea, para fijar el período de tiempo que se puede tardar en la implementación de cada una.

Este plan recoge las historias que serán agrupadas para conformar una entrega, y el orden de las mismas. El plan se realiza en base de las estimaciones de esfuerzos de desarrollo realizadas por los programadores. Las estimaciones de esfuerzos asociadas a la implementación de las historias tendrán como medida el punto. Un punto, se corresponde a una semana de programación ideal. Las historias deben poder ser programadas en un tiempo entre una y tres semanas.

La planificación se puede realizar basándose en el tiempo o en el alcance. La velocidad del proyecto es aprovechada para establecer cuantas historias de usuario se pueden hacer antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

Historia de usuario	Final 1ra Iteración 23/03/2012	Final 2da Iteración 18/05/2012
Gestionar computadora Gestionar proceso Gestionar usuario Autenticar usuario	4	Finalizado
Monitorear proceso Visualizar comportamiento de la prueba Generar reporte	No empezado	8

Tabla 13 Plan de entrega

Historias de usuario	Tiempo estimado	Iteración	Tiempo real
Gestionar computadora Gestionar proceso Gestionar usuario	4	1	4

Autenticar usuario			
Monitorear proceso Visualizar comportamiento de la prueba Generar reporte	8	2	8

Tabla 14 Estimación del esfuerzo por historia de usuarios

2.4.2 Plan de iteraciones

Luego de estimar el esfuerzo con el cual serán desarrolladas las historias de usuario y contar con un plan de entrega, se da paso a la planificación de la etapa de implementación del sistema. En este plan se recogen las historias de usuario que serán implementadas en cada iteración, de acuerdo al orden preestablecido. De acuerdo con lo antes mencionado se decide realizar el sistema en dos iteraciones, las cuales se especifican a continuación:

Iteración 1

Esta primera iteración tiene como meta implementar las historias de usuarios con prioridad media para el cliente, las cuales son: autenticar usuario, gestionar usuario, gestionar computadora y gestionar proceso. Estas historias de usuario constituyen la estructura básica del sistema. Con la finalización de esta iteración se obtendrá la primera entrega del sistema con el propósito de mostrar al cliente lo realizado y recibir retroalimentación del mismo.

Iteración 2

En esta iteración se implementarán las historias de usuario que tienen prioridad alta para el cliente, las cuales son: monitorear proceso, visualizar comportamiento de la prueba y generar reporte. La versión de prueba referente a esta iteración junto con las implementaciones anteriores, serán mostradas al cliente con el objetivo de realizar cambios en base a la opinión del mismo. Al finalizar esta iteración, luego de que haya

pasado satisfactoriamente por la etapa de pruebas se dispondrá de la aplicación con todas las funcionalidades descritas por el cliente.

2.4.3 Plan de duración de las iteraciones

Para una mayor organización del trabajo y como parte del ciclo de vida de un proyecto utilizando la metodología XP, se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con el cual se cuenta. Este plan tiene como finalidad reflejar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas, lo que ayuda a obtener una idea aproximada del tiempo que durará la confección del sistema en su totalidad.

Iteraciones	Orden de las HU a implementar	Duración total de las iteraciones
Iteración 1	Gestionar usuario Autenticar usuario Gestionar computadora Gestionar proceso	4 semanas
Iteración 2	Monitorear proceso Visualizar comportamiento de la prueba Generar reporte	8 semanas

Tabla 15 Plan de duración de las iteraciones

2.4.4 Historias de usuario divididas en tareas

Las historias de usuario están embebidas dentro de tareas de programación, que serán definidas en cada iteración. Estas tareas son escritas en fichas como las historias de usuario, pero en el lenguaje de los programadores. Cada tarea debería ser estimada en un período de uno a tres días de desarrollo.

2.4.4.1 Iteración 1

Historia de usuario	Tareas
Autenticar usuario	- Crear plantilla para formulario de autenticar. - Validar los campos del formulario. - Verificar los datos insertados. - Definir los privilegios del usuario
Gestionar usuario	- Definir tabla de usuarios
Gestionar computadora	- Definir la tabla de computadoras.
Gestionar proceso	- Definir tabla de procesos.

Tabla 16 Tareas abordadas en la primera iteración

No. Historia de usuario	Tarea
1,2,3,4	- Sincronizar con la base de datos

Tabla 17 Tarea genérica

Las tareas están descritas en los Anexos (Anexo 1)

2.4.4.2 Iteración 2

Historia de usuario	Tareas
Monitorear proceso	- Definir tabla de pruebas. - Definir tabla de configuración de procesos. - Definir gestor de base de datos a utilizar. - Definir consultas sql. - Definir contexto de ejecución de prueba.

Visualizar comportamiento de la prueba	- Crear plantilla de la vista principal. - Visualizar listado de computadoras. - Visualizar listado de procesos. - Visualizar listado de pruebas. - Consultar valores en la base de datos. - Consultar últimos valores en la base de datos. - Crear gráfica para visualizar todos los valores. - Crear gráfica para visualizar un intervalo.
Generar reporte	- Exportar datos a mostrar - Crear plantilla para el reporte. - Ajustar datos del reporte. - Guardar reporte en formato pdf.

Tabla 18 Tareas abordadas en la segunda iteración

Las tareas mencionadas anteriormente relacionadas con las historias de usuarios están descritas en el Anexo 2

2.5. Diseño del sistema

La metodología XP no requiere la descripción del sistema por medio de diagramas de clase utilizando notación UML, sino que se guía por técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Esto no implica que no se utilicen los diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando su complejidad no sea alta y defina información importante.

2.5.1 Tarjetas CRC (Contenido o Clase, Responsabilidad y Colaboración)

Las características más sobresalientes de las tarjetas CRC son su simpleza y ductilidad. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema (28).

Estas tarjetas se utilizan para estructurar las clases y a su vez definir las responsabilidades sobre las mismas. El nombre de la clase se escribe en la parte superior de la tarjeta a modo de título, las responsabilidades de esta clase son escritas en el lado izquierdo y la colaboración con otras clases se lista a la derecha de cada responsabilidad.

Clase	
Responsabilidades	Colaboradores

Tabla 19 Modelo de Tarjeta CRC.

La aplicación a desarrollar consta de disimiles objetos, por lo cual está formada por las siguientes clases:

- DBObject
- DBPostgresql
- DBSQLite
- ExecutePMonitorQuery
- TestThread
- TreeViewAdmin
- ChartViewAdmin
- ConvertSVG2PNG
- Report

Las tarjetas CRC del sistema, relacionadas con las clases mencionadas anteriormente, están definidas en el Anexo 3.

2.5.2 Arquitectura del sistema

El proceso en el cual se define una solución para los requisitos técnicos y operacionales de un sistema es conocido como diseño de arquitectura. Donde se definen qué componentes conforman el sistema, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada.

La arquitectura definida como parte de la solución de la aplicación a desarrollar es el Modelo Vista Controlador (MVC), ya que permite la separación de los datos de la aplicación, la interfaz de usuario y la lógica del control en tres componentes distintos. Además es la arquitectura del framework utilizado para el desarrollo de la aplicación.

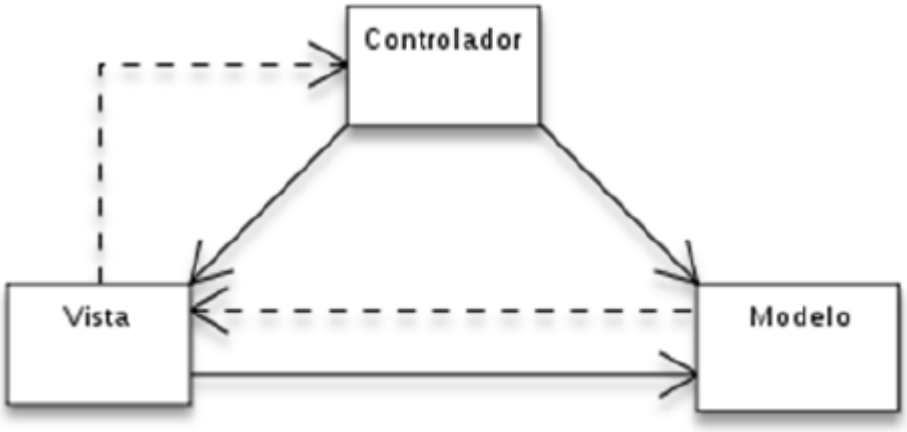

Figura 7: Modelo Vista Controlador (MVC)

A continuación se hace referencia a las tres capas que componen a la arquitectura definida:

El *Modelo* incorpora la capa de dominio y persistencia, es el encargado de guardar los datos en un medio persistente (ya sea una base de datos o un archivo de texto). Se refiere a la lógica del negocio o servicio y los datos asociados con la aplicación.

La *Vista* se encarga de presentar la interfaz al usuario, en sistemas Web, esto es típicamente HTML, aunque pueden existir otros tipos de vistas. En la vista, sólo se deben realizar operaciones simples. Es la encargada de la presentación de los datos.

El *Controlador* es el que captura los cambios en la vista y los envía al modelo, este último retorna los datos a la vista. Además es el que atiende las peticiones y componentes para la toma de decisiones de la aplicación.

2.5.3 Patrones de diseño

Los patrones son soluciones comunes a problemas de diseño de software orientado a objetos y que además poseen ciertas características de efectividad para resolver ese problema. Son reusables ya que pueden ser aplicados en otros diseños o problemas (29).

En el desarrollo de la aplicación se hará el uso del patrón Singleton en las clases ChartViewAdmin y TreeViewAdmin para garantizar que sólo exista una instancia de cada una, ya que el mismo está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. También se usará el patrón Decorator que permitirá añadir dinámicamente una funcionalidad a un objeto.

2.6. Conclusiones parciales

En este capítulo se describió una propuesta del sistema que se desea implementar obteniéndose el plan de entrega como parte de la fase de planificación. Se dejaron claras las iteraciones por las cuales va a transitar la aplicación, señalando las historias de usuario que serán implementadas en cada una de ellas. Como parte del diseño del sistema se describieron las tarjetas CRC, las cuales representan cada clase del sistema. Se aplicaron los patrones de diseño *Singleton* y *Decorator* y se utilizó el Modelo Vista Controlador en la arquitectura del sistema.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA

En la etapa de implementación de un software, se transforman las clases y objetos en ficheros fuente, binarios y ejecutables. El resultado final, es un sistema ejecutable que en la etapa de pruebas, es evaluada su calidad y desempeño como producto de software. En esta etapa se detectan y corrigen errores para la posterior aceptación del producto.

El presente capítulo abordará los temas relacionados con la implementación y pruebas realizadas al sistema. Se definen los estándares de codificación, que posibilitan la organización, limpieza y claridad en la escritura del código en el proceso de desarrollo del software y se describen los artefactos generados en estas fases.

3.1 Estándares de codificación

El lenguaje Python, utilizado para la implementación de la aplicación cuenta con un estilo de código que mejora su lectura. Para un mayor entendimiento, se muestran los pasos que se siguieron para escribir el código.

3.1.1 Definiciones generales

Las definiciones se realizarán de manera descriptiva, evitando las abreviaturas y los nombres cortos.

Clases:

Las clases deben comenzar con mayúscula y en caso de estar conformada por palabras compuestas, la definición debe ser continua y cada palabra debe iniciar con mayúscula siguiendo el estilo determinado.

Ejemplo:

```
class ExecutePMonitorQuery(object):
```

Métodos:

Los métodos deben empezar con minúsculas, escribiéndose de manera seguida en caso de estar conformados por palabras compuestas. Se escoge iniciar las funcionalidades con minúsculas para diferenciarlas de las clases, pero en caso de ser compuestas, estarán separadas por un guion bajo.

Ejemplos:

```
def process(self):
```

```
def insert_test(self, start_time, end_time):
```

Variables:

Las variables comenzarán con minúsculas, aquellas que sean compuestas estarán separadas por un guión bajo.

Ejemplos:

```
datetime_isos = datetime.fromtimestamp(start_time)
```

```
hostname = options.hostname
```

3.1.2 Comentarios

Se utilizan comentarios para especificar clases o funciones. La utilización de los mismos no estará presente en todo el código, solo en aquellos casos que sea necesario.

Ficheros:

Al inicio de cada fichero, especificando las clases o funciones que comprende.

Ejemplo:

```
'''
Interfaz de usuario del recolector
'''
```

Clases y funciones:

Aquellas clases o funciones en las que se utilicen comentarios para especificar su papel en el código, deben tener dicha aclaración seguida de su definición, buscando una mayor interrelación con el código y evitando que se confunda con el código que le antecede.

Ejemplo:

```
class ExecutePMonitorQuery(object):
    '''
    Consultas a la Base de Datos
    '''
```

3.2 Diagrama de despliegue

Los diagramas favorecen a un mejor entendimiento de la implementación del software. Estos permiten comprender con claridad lo que se ha implementado. Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. La figura 8 muestra el despliegue de los diferentes nodos físicos y la forma de conexión entre ellos.

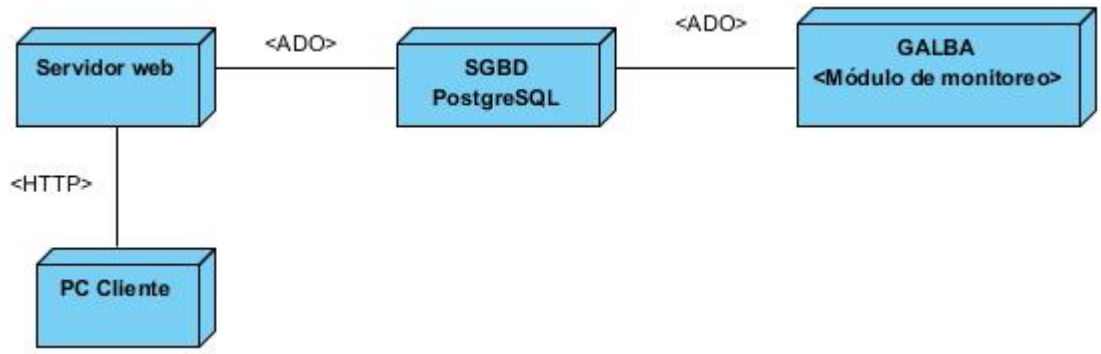

Figura 8: Diagrama de despliegue

A continuación se expone una breve descripción de los nodos y sus conexiones:

PC Cliente: Este nodo representa todas las PC clientes que se pueden conectar para acceder a los servicios que brinda la aplicación. Accede al Servidor web mediante el protocolo HTTP.

Servidor web: Nodo que funcionará como servidor donde estará instalada la aplicación web. Se conecta al SGBD PostgreSQL mediante una interfaz ADO con el objetivo de obtener la información necesaria para graficar tendencias y generar reportes.

SGBD PostgreSQL: PC donde estará la base de datos, se encargará de permitir gestionar los datos con que trabajará la aplicación.

GALBA <Módulo de monitoreo>: Este nodo representa la computadora donde se ejecuta al menos un servicio del GALBA y está instalada la aplicación encargada de monitorear la información de los procesos. Se conecta con el SGBD PostgreSQL mediante una interfaz ADO.

3.3 Diagrama Entidad-Relación

Mediante un diagrama Entidad-Relación se modela todo lo referente a la disposición de la base de datos que se ha creado.

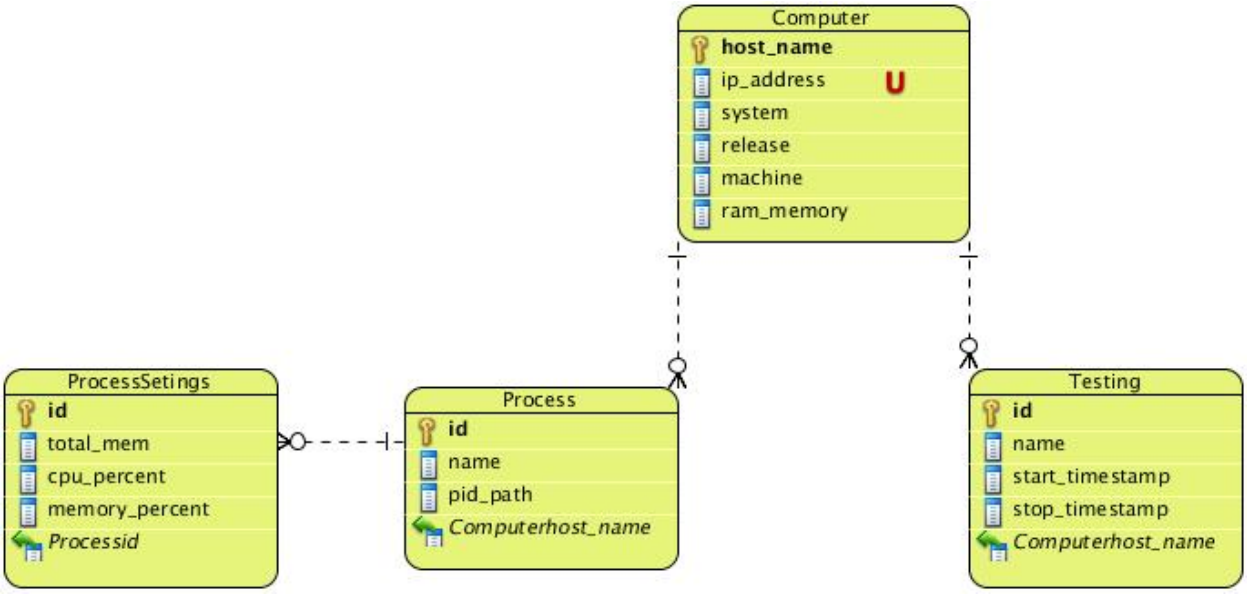

Figura 9: Diagrama Entidad-Relación

3.4 Prueba

Una vez que se ha culminado con la implementación de cualquier sistema, es de vital importancia realizarle algunas pruebas para comprobar su correcto funcionamiento una vez que se ponga a disposición de los usuarios finales. Es muy importante que se pruebe constantemente el software. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

La metodología XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

3.4.1 Pruebas de aceptación

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación. Estas son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación.

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas.

A continuación se muestran las pruebas de aceptación propuestas a realizarse por iteración para una mayor organización.

3.4.1.1 Iteración 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Autenticar	
Descripción: Prueba para la funcionalidad de autenticar	
Condiciones de ejecución: No puede haber usuario autenticado	
Entradas/Pasos de ejecución: Insertar nombre de usuario y clave para acceder al sistema	
Resultado esperado: Si los datos son correctos accede al sistema y dependiendo de los permisos son las acciones que puede realizar, en caso contrario muestra el mensaje de error “Por favor, entre un usuario y una contraseña correcta.”	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 20 Prueba 1 HU Autenticar usuario

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Adicionar un usuario	
Descripción: Prueba para la funcionalidad de adicionar un usuario	
Condiciones de ejecución: El administrador debe estar autenticado. Se utilizará un usuario con datos válidos	
Entradas/Pasos de ejecución: Para añadir un usuario en el sistema se insertan los datos válidos del mismo y luego se le define el rol que le corresponde.	
Resultado esperado: El usuario es añadido sin errores, de lo contrario muestra un mensaje señalando que el usuario ya existe.	

Evaluación de la prueba: Prueba satisfactoria

Tabla 21 Prueba 1 HU Gestionar usuario

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Modificar un usuario	
Descripción: Prueba para la funcionalidad de modificar un usuario	
Condiciones de ejecución: El administrador debe estar autenticado. Se usará un usuario con datos válidos. El usuario que se quiere modificar debe estar inscrito en la aplicación.	
Entradas/Pasos de ejecución: Para modificar a uno de los usuarios del sistema se dará paso a escoger al usuario previamente inscrito en él, y luego se procederá a modificar dicho usuario.	
Resultado esperado: El usuario es modificado sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 22 Prueba 2 HU Gestionar usuario

Caso de prueba de aceptación	
Código: HU2_P3	**Historia de usuario:** 2
Nombre: Eliminar un usuario	
Descripción: Prueba para la funcionalidad de eliminar un usuario	
Condiciones de ejecución: El administrador debe estar autenticado. Se usará un usuario con datos válidos. El usuario que se quiere eliminar debe estar inscrito en la aplicación.	
Entradas/Pasos de ejecución: Para eliminar a uno de los usuarios del sistema se dará paso a escoger el usuario previamente inscrito en él, y luego se procederá a eliminar dicho usuario.	

Resultado esperado: El usuario es eliminado sin generar errores.
Evaluación de la prueba: Prueba satisfactoria

Tabla 23 Prueba 3 HU Gestionar usuario

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Adicionar una computador	
Descripción: Prueba para la funcionalidad de adicionar una computadora	
Condiciones de ejecución: El administrador o el probador deben estar autenticados. Se utilizará una computadora con datos válidos	
Entradas/Pasos de ejecución: Para añadir una computadora en el sistema se insertan los datos válidos de la misma.	
Resultado esperado: La computadora es añadida sin errores, de lo contrario muestra un mensaje señalando que la computadora ya existe.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 24 Prueba 1 HU Gestionar computadora

Caso de prueba de aceptación	
Código: HU3_P2	**Historia de usuario:** 3
Nombre: Modificar una computadora	
Descripción: Prueba para la funcionalidad de modificar una computadora	
Condiciones de ejecución: El administrador o el probador deben estar autenticados. Se usará una computadora con datos válidos. La computadora que se quiere modificar debe estar inscrita en la aplicación.	
Entradas/Pasos de ejecución:	

Para modificar a una de las computadoras del sistema se dará paso a escoger la computadora previamente inscrita en él, y luego se procederá a modificar dicha computadora.
Resultado esperado: La computadora es modificada sin generar errores.
Evaluación de la prueba: Prueba satisfactoria

Tabla 25 Prueba 2 HU Gestionar computadora

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3
Nombre: Eliminar una computadora	
Descripción: Prueba para la funcionalidad de eliminar una computadora	
Condiciones de ejecución: El administrador o probador deben estar autenticados. Se usará un usuario con datos válidos. La computadora que se quiere eliminar debe estar inscrita en la aplicación.	
Entradas/Pasos de ejecución: Para eliminar a una de las computadoras del sistema se dará paso a escoger la computadora previamente inscrita en él, y luego se procederá a eliminar dicha computadora.	
Resultado esperado: La computadora es eliminada sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 26 Prueba 3 HU Gestionar computadora

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Adicionar un proceso	
Descripción: Prueba para la funcionalidad de adicionar un proceso	
Condiciones de ejecución:	

El administrador o probador deben estar autenticados. Se utilizará un proceso con datos válidos
Entradas/Pasos de ejecución: Para añadir un proceso en el sistema se insertan los datos válidos del mismo.
Resultado esperado: El proceso es añadido sin errores.
Evaluación de la prueba: Prueba satisfactoria

Tabla 27 Prueba 1 HU Gestionar proceso

Caso de prueba de aceptación	
Código: HU4_P2	**Historia de usuario:** 4
Nombre: Modificar un proceso	
Descripción: Prueba para la funcionalidad de modificar un proceso	
Condiciones de ejecución: El administrador o probador deben estar autenticados. Se usará un proceso con datos válidos. El proceso que se quiere modificar debe estar inscrito en la aplicación.	
Entradas/Pasos de ejecución: Para modificar a uno de los procesos del sistema se dará paso a escoger al proceso previamente inscrito en él, y luego se procederá a modificar dicho proceso.	
Resultado esperado: El proceso es modificado sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 28 Prueba 2 HU Gestionar proceso

Caso de prueba de aceptación	
Código: HU4_P3	**Historia de usuario:** 4
Nombre: Eliminar un proceso	
Descripción: Prueba para la funcionalidad de eliminar un proceso	

Condiciones de ejecución: El administrador o probador deben estar autenticados. Se usará un proceso con datos válidos. El proceso que se quiere eliminar debe estar inscrito en la aplicación.
Entradas/Pasos de ejecución: Para eliminar a uno de los procesos del sistema se dará paso a escoger el proceso previamente inscrito en él, y luego se procederá a eliminar dicho proceso.
Resultado esperado: El proceso es eliminado sin generar errores.
Evaluación de la prueba: Prueba satisfactoria

Tabla 29 Prueba 3 HU Gestionar proceso

3.4.1.2 Iteración 2

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** 5
Nombre: Monitorear proceso	
Descripción: Prueba para la funcionalidad de monitorear proceso.	
Condiciones de ejecución: Debe estar creada la base de datos con las tablas correspondiente. Se utilizará ip, puerto, nombre de base de datos, usuario y clave del servidor de base de datos válidos. Debe estar ejecutándose al menos un proceso del GALBA	
Entradas/Pasos de ejecución: En la línea de comandos se ejecutará la aplicación pasándoles los parámetros (ip, puerto, nombre base de datos, usuario y clave) correctamente y a continuación se presionará la tecla *Enter*	
Resultado esperado: Mostrará el mensaje “Presionar Enter para terminar” indicando que se ha ejecutado correctamente.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 30 Prueba 1 HU Monitorear proceso

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Seleccionar computadora	
Descripción: Prueba para la funcionalidad de seleccionar computadora.	
Condiciones de ejecución: El administrador o probador deben estar autenticados. Debe haber como mínimo una computadora con un proceso inscrito en la base de datos.	
Entradas/Pasos de ejecución: Desplegar la lista de computadoras. Desplegar la lista de procesos de la computadora. Seleccionar la computadora.	
Resultado esperado: Muestra un dialogo con la posibilidad de seleccionar una prueba y un proceso. En caso de que no haber pruebas realizadas mostrará el mensaje “Error: No hay pruebas realizadas aún.”	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 31 Prueba 1 HU Visualizar comportamiento de la prueba

Caso de prueba de aceptación	
Código: HU6_P2	**Historia de usuario:** 6
Nombre: Seleccionar proceso	
Descripción: Prueba para la funcionalidad de seleccionar proceso	
Condiciones de ejecución: El administrador o probador deben estar autenticados. Debe haber como mínimo una computadora con un proceso inscrito en la base de datos. Debe estar en ejecución la aplicación de monitoreo de procesos	

Entradas/Pasos de ejecución: Desplegar la lista de computadoras. Desplegar la lista de procesos de una computadora. Seleccionar el proceso.
Resultado esperado: Muestra la gráfica de dicho proceso con el último valor almacenado. En caso de no haber realizado ninguna prueba aún mostrará el mensaje “Error: No hay pruebas realizadas aún.”
Evaluación de la prueba: Prueba satisfactoria

Tabla 32 Prueba 2 HU Visualizar comportamiento de la prueba

Caso de prueba de aceptación	
Código: HU6_P3	**Historia de usuario:** 6
Nombre: Visualizar comportamiento de la prueba.	
Descripción: Prueba para la funcionalidad de visualizar comportamiento de la prueba.	
Condiciones de ejecución: El administrador o probador deben estar autenticados. Debe haber como mínimo una computadora con un proceso inscrito en la base de datos. Debe haberse realizado al menos una prueba.	
Entradas/Pasos de ejecución: Desplegar la lista de computadoras. Desplegar la lista de procesos de una computadora. Seleccionar la computadora Seleccionar la prueba. Seleccionar el proceso. Dar clic en el botón *Aceptar*	
Resultado esperado:	

Muestra la gráfica histórica correspondiente al proceso seleccionado.
Evaluación de la prueba: Prueba satisfactoria

Tabla 33 Prueba 3 HU Visualizar comportamiento de la prueba

Caso de prueba de aceptación	
Código: HU7_P1	**Historia de usuario:** 7
Nombre: Generar reporte	
Descripción: Prueba para la funcionalidad de generar reporte.	
Condiciones de ejecución: El administrador o probador deben estar autenticados. Debe estar visualizándose el comportamiento de una prueba.	
Entradas/Pasos de ejecución: Dar clic en el vínculo “Generar reporte”	
Resultado esperado: Se genera el reporte, se lanza el dialogo de “Guardar”.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 34 Prueba 1 HU Generar reporte

3.5 Conclusiones parciales

En este capítulo se registran las bases necesarias para la implementación de la aplicación. Con la especificación de los estándares de codificación se abarca todo lo referente a la estructura que contendrá el desarrollo del sistema. Los diagrama posibilitan un mejor entendimiento de cómo quedará el sistema. Por otra parte se realizaron las pruebas de aceptación a cada historia de usuario, que ofrecerán al cliente conformidad y seguridad ante las funcionalidades del sistema.

CONCLUSIONES GENERALES

Al término de la presente investigación, “Herramienta para el monitoreo y análisis de los procesos del sistema SCADA Guardián del ALBA”, se llegaron a las siguientes conclusiones:

- Se diseñó una herramienta para el monitoreo y análisis de los procesos del sistema SCADA Guardián del ALBA que puede ser utilizada en cualquier sistema que requiera su uso, logrando una independencia total del sistema.
- La herramienta brinda un entorno que facilita el monitoreo y análisis de los procesos del GALBA por parte del equipo de pruebas.
- La ejecución exitosa de las pruebas de aceptación demostraron que la herramienta está lista para su uso.

RECOMENDACIONES

Al finalizar la investigación se recomienda:

- Personalizar los reportes, creando una vista para introducir datos que le interesen al usuario.
- Optimizar el proceso de extracción de los datos a graficar, haciendo mejor uso de la base de datos y evitar una sobrecarga de la aplicación.
- Optimizar la implementación de la vista para que pueda ser utilizada en todos los navegadores web.

REFERENCIAS BIBLIOGRÁFICAS

1. Portal Centro de Informática Industrial (CEDIN). Misión. [Online] 2010. [Cited: marzo 08, 2012.] http://portal.cedin.prod.uci.cu/.

2. **Herrera Vázquez, Moisés.** *Introducción a la Arquitectura del Guardián del ALBA.* 2008.

3. Las pruebas de software. [Online] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

4. Pruebas no funcionales. [Online] http://www.es.testhouse.net/pruebas-no-funcionales/.

5. Pruebas de rendimiento. [Online] http://www.es.testhouse.net/pruebas-de-rendimiento/.

6. **González Durán, Sergio.** Manual básico de administración de procesos en Linux. [Online] http://www.linuxtotal.com.mx/index.php?cont=info_admon_012.

7. **Easey, Cameron.** Definition of CPU Usage. [Online] 2012. [Cited: Marzo 8, 2012.] http://www.ehow.com/about_5076431_definition-cpu-usage.html.

8. **Clark Scott, J.** What is RAM (random access memory)? [Online] Septiembre 2005. [Cited: Marzo 8, 2012.] http://searchmobilecomputing.techtarget.com/definition/RAM.

9. **Marzal Varó, Andrés and Gracia Luengo, Isabel.** *Introducción a la programación con C.*

10. **Rodola, Giampaolo and Loden, Jay.** psutil - A cross-platform process and system utilities module for Python - Google Project Hosting. [Online] [Cited: 03 08, 2012.] http://code.google.com/p/psutil/.

11. **Lazalde, Alan.** Comando Linux htop: administra interactivamente los procesos del sistema. [Online] Marzo 31, 2010. [Cited: Marzo 8, 2012.]

http://bitelia.com/2010/03/comando-linux-htop-administra-interactivamente-los-procesos-del-sistema.

12. **Bacher, Sebastien.** Gnome System Monitor. [Online] Marzo 14, 2003. [Cited: Marzo 8, 2012.] http://freecode.com/projects/gnome-system-monitor.

13. **Mendoza Sanchez, María A.** *Metodologías De Desarrollo De Software.*

14. **BOOCH, GRADY, JACOBSON, IVAR and RUMBAUGH, JAMES.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : ADDISON-WESLEY, 2007. 978-84-7829-087-1.

15. **Lafuente, Guillermo Javier.** UML Unified Modeling Languaje. [Online] Febrero 2002. [Cited: Marzo 8, 2012.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html.

16. **Menéndez-Barzanallana Asensio, Rafael.** Herramientas CASE. Ingenieria del software. Informatica Aplicada a la gestión Pública. Universidad de Murcia. [Online] Mayo 23, 2011. [Cited: Marzo 8, 2012.] http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

17. **Hernandis, J. A.** Why Visual Paradigm for UML? [Online] 2005. [Cited: Marzo 8, 2012.] http://www.visual-paradigm.com/product/vpuml/.

18. **J. Gutiérrez, Javier.** Departamento de Lenguajes y Sistemas Informáticos. [Online] [Cited: Marzo 8, 2012.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

19. **Holovaty, Adrian and Kaplan-Moss, Jacob.** *El libro de Django.* 2007.

20. **jQuery Project, The.** jQuery: The Write Less, Do More, JavaScript Library. [Online] 2010. [Cited: marzo 08, 2012.] http://jquery.com/.

21. **Leyva, Francisco I.** Agrega gráficas interactivas a tu sitio web con Highcharts [Javascript]. [Online] 01 31, 2012. [Cited: 03 08, 2012.]

http://www.panchosoft.com/blog/2012/01/31/agrega-graficas-interactivas-a-tu-sitio-web-con-highcharts-javascript/.

22. **Marzal Varó, Andrés and Gracia Luengo, Isabel.** *Introducción a la programación con Python.* 2003.

23. **Flanagan, David.** *JavaScript: The Definitive Guide.* s.l. : O'Reilly Media, Inc., 2011. 978-0-596-80552-4.

24. **Montero Garrido, Jesús Manuel.** *Plataforma Eclipse. Introducción Técnica.*

25. **Pecos, Daniel.** PostgreSQL_MySQL. [Online] [Cited: marzo 08, 2012.] http://danielpecos.com/docs/mysql_postgres/x15.html.

26. **Fernández Escribano, Gerardo.** *Introducción a Extreme Programming.*

27. **Joskowicz, José.** Reglas y Prácticas en eXtreme Programming. [Online] 2008. http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

28. **Casas, Sandra and Reinaga, Héctor.** Identificación y Modelado de Aspectos Tempranos dirigido por Tarjetas de Responsabilidades y Colaboraciones. [Online] 2008. http://www.oocities.org/espanol/profeprog2/INVPAPER25.pdf.

29. **Rojas, Mc and Olivares, Juan Carlos.** Patrones de Diseño. [Online] Mayo 2007. http://antares.itmorelia.edu.mx/~jcolivar/courses/dp07b/patrones.pdf.

ANEXOS

Anexo 1: Tareas abordadas en la primera iteración

Tarea	
Número de la tarea: 1	**Número de HU:** 1,2,3,4
Nombre de la tarea: Sincronizar con la base de datos	
Tipo de tarea: Configuración	
Fecha de inicio: 27/02/2012	**Fecha de fin:** 27/02/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se van a sincronizar los datos con la base de datos.	

Tabla 35 Tarea 1: Sincronizar con la base de datos

Tarea	
Número de la tarea: 2	**Número de HU:** 1
Nombre de la tarea: Crear plantilla para formulario de autenticar.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 27/02/2012	**Fecha de fin:** 28/02/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se crea un archivo con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para autenticar los usuarios.	

Tabla 36 Tarea 2: Crear plantilla para formulario de autenticar.

Tarea	
Número de la tarea: 3	**Número de HU:** 1
Nombre de la tarea: Validar los campos del formulario.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 29/02/2012	**Fecha de fin:** 29/02/2012

Programador responsable: José Manuel Batista Viltre
Descripción: Se validan que los campos del formulario no estén vacíos.

Tabla 37 Tarea 3: Validar los campos del formulario.

Tarea	
Número de la tarea: 4	**Número de HU:** 1
Nombre de la tarea: Verificar los datos insertados.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 01/03/2012	**Fecha de fin:** 01/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se verifica en la base de datos que los datos insertados sean correctos.	

Tabla 38 Tarea 4: Verificar los datos insertados.

Tarea	
Número de la tarea: 5	**Número de HU:** 1
Nombre de la tarea: Definir los privilegios del usuario	
Tipo de tarea: Desarrollo	
Fecha de inicio: 02/03/2012	**Fecha de fin:** 02/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: En el módulo de administración que provee Django se gestionarán los privilegios de cada usuario.	

Tabla 39 Tarea 5: Definir los privilegios del usuario

Tarea	
Número de la tarea: 6	**Número de HU:** 2
Nombre de la tarea: Definir tabla de usuarios.	
Tipo de tarea: Desarrollo	

Fecha de inicio: 05/03/2012	**Fecha de fin:** 09/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se utilizará la misma tabla brindada por el framework Django.	

Tabla 40 Tarea 6: Definir tabla de usuarios.

Tarea	
Número de la tarea: 7	**Número de HU:** 3
Nombre de la tarea: Definir la tabla de computadoras	
Tipo de tarea: Desarrollo	
Fecha de inicio: 12/03/2012	**Fecha de fin:** 16/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se va a definir la clase computadora en el archivo model.py del framework Django, que se corresponde con la entidad de igual nombre en la base de datos.	

Tabla 41 Tarea 7: Definir la tabla de computadoras

Tarea	
Número de la tarea: 8	**Número de HU:** 4
Nombre de la tarea: Definir tabla de procesos.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 19/03/2012	**Fecha de fin:** 23/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se va a definir la clase proceso en el archivo model.py del framework Django, que se corresponde con la entidad de igual nombre en la base de datos.	

Tabla 42 Tarea 8: Definir tabla de procesos.

Anexo 2: Tareas abordadas en la segunda iteración

Tarea	
Número de la tarea: 9	**Número de HU:** 5
Nombre de la tarea: Definir tabla de pruebas.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 26/03/2012	**Fecha de fin:** 27/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se va a definir la clase prueba en el archivo model.py del framework Django, que se corresponde con la entidad de igual nombre en la base de datos.	

Tabla 43 Tarea 9: Definir tabla de pruebas.

Tarea	
Número de la tarea: 10	**Número de HU:** 5
Nombre de la tarea: Definir tabla de configuración de procesos.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 28/03/2012	**Fecha de fin:** 30/03/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se va a definir la clase de configuración de proceso en el archivo model.py del framework Django, que se corresponde con la entidad de igual nombre en la base de datos.	

Tabla 44 Tarea 10: Definir tabla de configuración de procesos.

Tarea	
Número de la tarea: 11	**Número de HU:** 5
Nombre de la tarea: Definir gestor de base de datos a utilizar.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 02/04/2012	**Fecha de fin:** 04/04/2012
Programador responsable: José Manuel Batista Viltre	

Descripción: Se define una clase genérica DBObject de la cual heredan DBSQLite y DBPostgresql.

Tabla 45 Tarea 11: Definir gestor de base de datos a utilizar.

Tarea	
Número de la tarea: 12	**Número de HU:** 5
Nombre de la tarea: Definir consultas sql.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 05/04/2012	**Fecha de fin:** 06/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se definen las consultas a la base de datos para seleccionar los procesos, insertar las pruebas e insertar el resultado del monitoreo de procesos.	

Tabla 46 Tarea 12: Definir consultas sql.

Tarea	
Número de la tarea: 13	**Número de HU:** 5
Nombre de la tarea: Definir contexto de ejecución de prueba.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 09/04/2012	**Fecha de fin:** 13/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se define todo lo relacionado con el contexto en que se ejecutarán las pruebas, como el tipo de información que almacenará, el tiempo en que lo realizará entre otros parámetros.	

Tabla 47 Tarea 13: Definir contexto de ejecución de prueba.

Tarea	
Número de la tarea: 14	**Número de HU:** 6
Nombre de la tarea: Crear plantilla de la vista principal.	

Tipo de tarea: Desarrollo	
Fecha de inicio: 16/04/2012	**Fecha de fin:** 18/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se crea un archivo con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar.	

Tabla 48 Tarea 14: Crear plantilla de la vista principal.

Tarea	
Número de la tarea: 15	**Número de HU:** 6
Nombre de la tarea: Visualizar listado de computadoras.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 19/04/2012	**Fecha de fin:** 19/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se muestran todas las computadoras configuradas en la base de datos.	

Tabla 49 Tarea 15: Visualizar listado de computadoras.

Tarea	
Número de la tarea: 16	**Número de HU:** 6
Nombre de la tarea: Visualizar listado de procesos.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 20/04/2012	**Fecha de fin:** 20/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se muestran todos los procesos dado una computadora.	

Tabla 50 Tarea 16: Visualizar listado de procesos.

Tarea	
Número de la tarea: 17	**Número de HU:** 6
Nombre de la tarea: Visualizar listado de pruebas.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 23/04/2012	**Fecha de fin:** 23/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se muestran todas las pruebas realizadas hasta el momento.	

Tabla 51 Tarea 17: Visualizar listado de pruebas.

Tarea	
Número de la tarea: 18	**Número de HU:** 6
Nombre de la tarea: Consultar valores en la base de datos.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 24/04/2012	**Fecha de fin:** 25/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se consultan los valores que se van a graficar y se ajustan a la gráfica.	

Tabla 52 Tarea 18: Consultar valores en la base de datos.

Tarea	
Número de la tarea: 19	**Número de HU:** 6
Nombre de la tarea: Consultar últimos valores en la base de datos.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 26/04/2012	**Fecha de fin:** 27/04/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se consultan los últimos valores almacenados en la base de datos y se ajustan a la gráfica.	

Tabla 53 Tarea 19: Consultar últimos valores en la base de datos.

Tarea	
Número de la tarea: 20	**Número de HU:** 6
Nombre de la tarea: Crear gráfica para visualizar todos los valores.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 30/04/2012	**Fecha de fin:** 02/05/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se crea la gráfica donde se mostrarán todos los valores recolectados de una prueba.	

Tabla 54 Tarea 20: Crear gráfica para visualizar todos los valores.

Tarea	
Número de la tarea: 21	**Número de HU:** 6
Nombre de la tarea: Crear gráfica para visualizar un intervalo.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 02/05/2012	**Fecha de fin:** 04/05/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se crea la gráfica donde se mostrará el intervalo seleccionado de la gráfica general o se mostrará los últimos valores de una prueba en ejecución.	

Tabla 55 Tarea 21: Crear gráfica para visualizar un intervalo.

Tarea	
Número de la tarea: 22	**Número de HU:** 7
Nombre de la tarea: Exportar datos a mostrar.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 07/05/2012	**Fecha de fin:** 08/05/2012
Programador responsable: José Manuel Batista Viltre	

Descripción: Se exportan los datos de interés a poner en el reporte.

Tabla 56 Tarea 22: Exportar datos a mostrar.

Tarea	
Número de la tarea: 23	**Número de HU:** 7
Nombre de la tarea: Crear plantilla para el reporte.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 09/05/2012	**Fecha de fin:** 11/05/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se diseña la plantilla del reporte.	

Tabla 57 Tarea 23: Crear plantilla para el reporte.

Tarea	
Número de la tarea: 24	**Número de HU:** 7
Nombre de la tarea: Ajustar datos del reporte.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 14/05/2012	**Fecha de fin:** 16/05/2012
Programador responsable: José Manuel Batista Viltre	
Descripción: Se ajustan los datos recibidos a la plantilla.	

Tabla 58 Tarea 24: Ajustar datos del reporte.

Tarea	
Número de la tarea: 25	**Número de HU:** 7
Nombre de la tarea: Guardar reporte en formato pdf.	
Tipo de tarea: Desarrollo	
Fecha de inicio: 17/05/2012	**Fecha de fin:** 18/05/2012
Programador responsable: José Manuel Batista Viltre	

Descripción: Se genera el reporte en formato pdf.

Tabla 59 Tarea 25: Guardar reporte en formato pdf.

Anexo 3: Tarjetas C.R.C del sistema

DBObject	
Define los atributos del sistema gestor de base de datos que utilizara la aplicación de recolección.	

Tabla 60 Tarjeta CRC clase DBObject

DBPostgresql	
Hereda de DBObject.	DBObject

Tabla 61 Tarjeta CRC clase DBPostgresql

DBSQLite	
Hereda de DBObject.	DBObject

Tabla 62 Tarjeta CRC clase DBSQLite

ExecutePMonitorQuery	
Definir las consultas a la base de datos.	DBPostgresql

Tabla 63 Tarjeta CRC clase ExecutePMonitorQuery

TestThread	
Define las propiedades de la prueba. Definir contexto de ejecución de prueba.	ExecutePMonitorQuery

Tabla 64 Tarjeta CRC clase TestThread

TreeViewAdmin	
Maneja la vista del árbol	

Tabla 65 Tarjeta CRC clase TreeViewAdmin

ChartViewAdmin	
Maneja los datos que se van a graficar	

Tabla 66 Tarjeta CRC clase ChartViewAdmin

ConvertSVG2PNG	
Convierte imágenes de tipo svg a imágenes de tipo png	

Tabla 67 Tarjeta CRC clase ConvertSVG2PNG

Report	
Genera los reportes.	ConvertSVG2PNG

Tabla 68 Tarjeta CRC clase Report

Anexo 4: Elementos a tener en cuenta en el desarrollo del modelo de decisión elaborado para la selección de la aplicación de monitoreo a utilizar en la implementación de la herramienta de monitoreo y análisis.

Una matriz de decisión es una herramienta de decisiones utilizada por los arquitectos de software como parte de sus herramientas a la hora de seleccionar una tecnología o aplicación a utilizar. En la siguiente tabla se detalla la información que posee la matriz de decisión.

Criterios:	Opciones:	Importancia:	Puntaje:
Jerarquía de criterios de decisión, también conocido como modelo de decisión.	Opciones a seleccionar, también llamado soluciones o alternativas.	Valor de cada criterio sobre la base de su importancia en la decisión final, es un valor comprendido entre 0 y 100.	Tasa de cada opción en una relación de escala mediante la calificación a cada criterio.

Tabla 69: Elementos de la matriz de decisión.

Calificación	Descripción
0	No Aceptable
1	Poco Aceptable
2	Aceptable.
3	Sobresaliente.
4	Excelente.

Tabla 70: Tabla que muestra cómo calificar una opción.

Modelo de Decision		ALTERNATIVAS									
		ps		top		htop		Gnome System Monitor		psutil	
Criterios	Importancia	Calificación	Puntaje	Calificación	Puntaje	Calificación	Puntaje	Calificación	Puntaje	Calificación	Puntaje
Funcionalidades que brinda	30	3	90	3	90	3	90	3	90	4	120
Implementación con herramientas libres y código abierto.	10	4	40	4	40	4	40	4	40	4	40
Facilidad de uso	15	2	30	3	45	4	60	4	60	3	45
Multiplataforma	20	0	0	0	0	0	0	0	0	4	80
Persistencia de datos	25	2	50	1	25	1	25	0	0	3	75
Total	100	11	210	11	200	12	215	11	190	18	360

Tabla 71: Modelo de decisión