Universidad de las Ciencias Informáticas
Facultad 1

Título

Aplicación centralizada para la gestión de SPEX

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

Autor
Yosbel Palacios Ferrer

Tutores
Ing. Rolando León Dueñas
Ing. Danis Carlos Chaviano Jiménez

Junio, 2017
La Habana “ Año 59 de la Revolución”

Declaración de autoría

Declaro por este medio que yo Yosbel Palacios Ferrer, con carné de identidad 93092900407 soy el autor principal del trabajo titulado “Aplicación Centralizada para la gestión de SPEX” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente declaración jurada de autoría en La Habana a los _____ días del mes de _____ del año _____.

_____

Firma del autor

_____

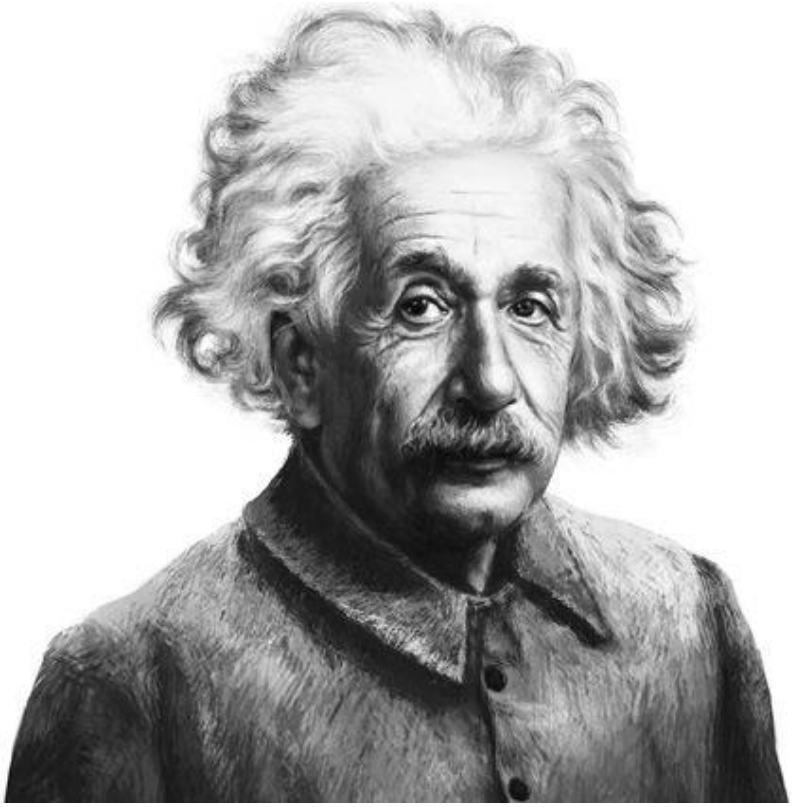
Firma del tutor

Ing. Rolando León Dueñas

_____

Firma del tutor

Ing. Danis Carlos Chaviano Jiménez



"La ciencia no ha sido y nunca será un libro terminado. Cada éxito importante trae nuevos interrogantes".

Albert Einstein

Dedicatoria

A mis padres por su apoyo incondicional.

A mi novia, quien es mi fuente de inspiración y el motor que me ayuda a seguir adelante.

A mis hermanos.

A mis tíos, tías, todos mis primos.

A toda mi familia en general y las personas que de alguna manera estuvieron involucrados en mi desarrollo profesional.

Agradecimientos

A mis padres Mercedes y Juan Pablo, principalmente por traerme a la vida, por estar en todo el proceso de mi crecimiento, dando lo mejor de sí para que hoy sea quien soy.

A mi novia Jessyca Rodas, por ser una fuente vital de Energía Positiva y darme ese aliento y esa fuerza necesarias para seguir siempre adelante con el afán de cumplir mis metas y nunca rendirme.

A mis tutores Danis Carlos y Rolando León, por el asesoramiento y ayuda de su parte, en este camino de alcanzar la categoría de Ingeniero en Ciencias Informáticas.

A mis tíos Estrella, Dora, Ana Ferrer y Reynaldo Molina, su participación en mi desempeño como estudiante fue crucial, asumiendo papeles de padres y madres para mí.

A la Revolución Cubana, Fidel Castro y Raúl Castro, por crear la Universidad de las Ciencias Informáticas y permitirme cursar estudios en ella.

A todos los que de una manera u otra fueron partícipes en mi proceso de Formación Profesional.

Resumen

Las tecnologías de la información muestran un crecimiento constante con el transcurso del tiempo, evidenciándose en el desarrollo del sector empresarial. Con necesidad de equipamiento tecnológico capaz de cumplir sus funciones, la empresa Gente de Mérito se encarga de asegurar que las computadoras que se comercializan cumplan con las restricciones de desempeño en condiciones extremas de trabajo. Actualmente en dicha empresa no se cuenta con un sistema que centralice los procesos de pruebas y estrés del hardware. El presente trabajo propone el desarrollo de un sistema centralizado de pruebas y estrés de hardware, donde se obtiene como resultado una aplicación web que permite gestionar el proceso de pruebas y estrés que desarrolla la empresa. La propuesta de solución cuenta con una interfaz web, donde los técnicos establecen las diferentes configuraciones para realizar las pruebas. Con los reportes generados por el sistema SPEX se generan diferentes vistas de la información. Se realizan pruebas usando el método de caja negra y caja blanca a la solución, llegando a resultados satisfactorios en todas las funcionalidades.

Palabras clave:

Pruebas y estrés de hardware, sistema, reporte, SPEX.

Índice de contenido

Índice de ilustraciones

Índice de tablas

APLICACIÓN CENTRALIZADA PARA LA GESTIÓN DE SPEX

Introducción

Hoy en día, la tecnología es parte del sistema de vida de todas las sociedades, dado que se están sumando a la voluntad social y política de controlar el propio destino de la humanidad. Posee un papel muy importante en el desarrollo de la esfera de las comunicaciones y en el desarrollo de sistemas y softwares informáticos que brinden la posibilidad de satisfacer las necesidades humanas de una manera cada vez mejor y eficiente. Este auge tecnológico se evidencia en gran medida en el sector empresarial, donde es necesario contar con equipamiento tecnológico que permita realizar las tareas que se deseen. Es por eso que para el correcto funcionamiento de las computadoras se hace necesario la utilización de componentes de hardware de calidad, independientemente del uso para la cual son diseñados. Existen situaciones en las que el hardware, según sus capacidades técnicas, es puesto bajo una gran presión y es cuando se conoce qué tan eficiente es y cuánto puede rendir en la realización de tareas que requieran mayor cantidad de recursos computacionales.

En la Universidad de las Ciencias Informáticas (UCI) se encuentra el Centro de Software Libre (CESOL), el cual tiene entre sus funciones desarrollar productos de software relacionados con la migración a software libre. En este centro radica el departamento de Servicios Integrales de Migración, Asesoría y Soporte (SIMAYS), donde fue creado el Sistema de Pruebas y Estrés (SPEX), a pedido de la Empresa Industrial para la Informática, las Comunicaciones y la Electrónica. Dicha empresa se dedica a desarrollar, producir, comercializar de forma mayorista y brindar servicios técnicos a productos de las tecnologías de la información, las telecomunicaciones, la electrónica y otros productos electrotécnicos.

En la empresa se ensamblan computadoras que posteriormente se comercializan. Una vez ensambladas se lleva a cabo un proceso de pruebas rigurosas, para comprobar el rendimiento y las capacidades presentes en los equipos, y además verificar que cumplan con las restricciones definidas por la empresa. Es ahí donde se evidencia el propósito de la creación de SPEX, ya que es el software encargado de llevar a cabo el proceso de pruebas y estrés del hardware de dichos equipos tecnológicos. SPEX necesita varias configuraciones manuales para realizar las pruebas. Estas configuraciones se hacen por cada componente de hardware de la computadora, como es el caso del disco duro, la memoria RAM (*Random Access Memory*, Memoria de acceso aleatorio), el procesador, hardware de video y hardware de audio. En esas configuraciones se puede definir el tipo de prueba que se le va a realizar a cada uno de esos componentes, y también la carga o presión con la que se va a ejecutar. Es posible también establecer el

tiempo que van a durar las pruebas y si se va a probar cada componente de la computadora de manera independiente o todos al mismo tiempo, así como el tiempo de actualización en las tablas de SPEX que muestran la información en tiempo real de lo que está ocurriendo.

Posteriormente a las configuraciones, se ejecuta el sistema SPEX para realizar las pruebas rigurosas, y al finalizar su ejecución, genera un reporte con el resultado de todo lo ocurrido en el proceso. Teniendo en cuenta que en la empresa las pruebas se realizan a un lote, el cual puede contar con un gran número de computadoras, todas con las mismas características, es necesario realizar todo el proceso anterior en cada una de ellas.

Actualmente las pruebas y estrés de hardware se realizan de manera manual y secuencial, una computadora a la vez. Esto implica que el trabajo sea lento y engorroso dado que el operario debe interactuar con cada una de las computadoras para realizar las configuraciones antes mencionadas, ejecutar SPEX y obtener el reporte generado. La forma en que se realiza todo este proceso puede requerir mucho tiempo en dependencia de la cantidad de computadoras. En caso de que un lote cuente con mil computadoras, será esa la cantidad de veces que el operario debe interactuar con SPEX para configurar, ejecutar y obtener el reporte. Además, que en la empresa existe más de un lote de equipos ensamblados. Mientras mayor sea la cantidad, mayor será el tiempo requerido para llevar a cabo todo el proceso y mayor necesidad de fuerza de trabajo.

Después de mencionar las anteriores deficiencias se define el siguiente **problema de investigación:** ¿Cómo facilitar el proceso de configuración, ejecución y obtención de los reportes del sistema SPEX?

Se define como **objeto de estudio** el proceso de administración remota de sistemas de pruebas de hardware, y como **campo de acción** las herramientas y librerías para la administración remota de sistemas de pruebas de hardware desde la web.

Para dar solución al problema de investigación se declara como **objetivo general** desarrollar una aplicación que permita gestionar la configuración, ejecución y los reportes de SPEX simultáneamente, para facilitar el proceso de pruebas y estrés del hardware. Teniendo como **objetivos específicos:**

- Fundamentar la investigación mediante la elaboración del marco teórico para sustentar los conceptos y la propuesta de desarrollo de la Aplicación Centralizada para la gestión de SPEX.
- Realizar el análisis y diseño de la propuesta de solución.
- Implementar las funcionalidades necesarias para la Aplicación Centralizada para la gestión de SPEX.

- Realizar las pruebas pertinentes para probar la Aplicación Centralizada para la gestión de SPEX.

Para guiar la investigación se plantean las siguientes **preguntas científicas**:

1. ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de administración remota de sistemas de pruebas de hardware?
2. ¿Cuáles son las herramientas y tecnologías más adecuadas para desarrollar una aplicación centralizada que permita gestionar la configuración, ejecución y los reportes de SPEX simultáneamente?
3. ¿Qué técnicas y pruebas aplicar durante la evaluación de la configuración y ejecución de los reportes de SPEX simultáneamente?

Posibles resultados:

Una herramienta centralizada para facilitar la realización de pruebas y estrés del hardware de las computadoras ensambladas en la Empresa Industrial para la Informática, las Comunicaciones y la Electrónica

Los **métodos de investigación** empleados son:

Los métodos teóricos y empíricos. Los **métodos teóricos** para realizar un estudio del arte de herramientas y prácticas usadas, con el apoyo del método *Histórico-Lógico* para llegar a soluciones similares a la propuesta en este trabajo investigativo, también el *Analítico-Sintético*, mediante este método se identifican conceptos y definiciones importantes relacionadas con el tema, permitiendo generar una propuesta adecuada a la situación planteada. Como **métodos empíricos** la *Observación,* para acceder a la información directa relacionada con el objeto de investigación y hacer un diagnóstico del problema a investigar. Se usa durante todo el desarrollo del trabajo permitiendo evaluar el proceso y los avances en la elaboración de la solución propuesta.

También se aplicó como técnica la **tormenta de ideas** para obtener información significativa sobre las necesidades funcionales que debe cumplir la propuesta de solución. Esta técnica se aplicó en el Centro de Software Libre (*Anexo 1 Técnica tormenta de ideas*).

Estructuración del trabajo de diploma

Capítulo 1: Plantea la fundamentación teórica de la investigación, e incluye un estudio del estado del arte del tema que se investiga, además, las tecnologías y herramientas en las que se apoya la solución al problema.

Capítulo 2: Describe el proceso de análisis y diseño, el cual comienza con el planteamiento y planificación

de las historias de usuario que describen las funcionalidades a desarrollar. Detalla la arquitectura de la solución y sus principales características.

Capítulo 3: Describe las fases de implementación y pruebas y los componentes que fue necesaria su utilización en la solución. Se implementan todas las funcionalidades identificadas, logrando un sistema que satisface las principales necesidades del cliente. Se detallan también las pruebas que se le realizaron al sistema ya finalizado, con el objetivo de asegurar la calidad y eficiencia de la solución.

Capítulo 1. Fundamentación Teórica

En este capítulo se lleva a cabo la primera fase de la metodología de desarrollo, se presentan los resultados del análisis documental realizado sobre los sistemas informáticos existentes vinculados al campo de acción, que poseen similares características o que pretenden desempeñar una función semejante al sistema que se desea desarrollar, además de mejoras potenciales que se puedan agregar a la solución propuesta. También se realiza un estudio sobre las herramientas y tecnologías, que justifican la solución empleada en la construcción de la Aplicación Centralizada.

1.1 ¿Por qué hacer una prueba de estrés?

Para garantizar la fiabilidad y la estabilidad de un sistema. Incluso si una computadora enciende y funciona bien en condiciones de uso normal, el hardware voluble puede causar problemas cuando realizas tareas más pesadas, tales como juegos o edición de vídeo.

Realizar estas pruebas ayuda a identificar cuellos de botella, reducir el riesgo de caídas del sistema, aprovechar los recursos de manera más eficiente, conocer los límites que soporta el sistema. Permite tomar decisiones sobre configuraciones de hardware, ajustes de software y selección de arquitecturas (Ibarra, 2013). Los fallos por estos motivos suelen ser muy costosos por tanto existen diferentes tipos de pruebas que se pueden hacer en una computadora, arrojando resultados diferentes.

1.2 Las Pruebas

El proceso de realizar pruebas consiste en someter al sistema a altas cargas de trabajo, simulando la actividad real de los futuros usuarios del sistema. Estas pruebas ayudan a predecir el comportamiento de un sistema y conocer el grado en que realiza las funciones para las que ha sido diseñado sin pérdidas de servicio y con un tiempo de respuesta óptimo y estable. Se planifican, preparan y ejecutan en base a objetivos firmemente establecidos, que permitirán decidir qué tipo de prueba es adecuada para el caso y determinar, en términos medibles, qué resultados son aceptables y cuáles no. Se ejecutarán diferentes tipos de pruebas, dependiendo del objetivo (Software Quality System S.A, 2017). Estas pruebas ayudan a conocer que tan eficiente puede llegar a ser el sistema, permiten conocer hasta qué punto se puede contar con un correcto desempeño del mismo.

1.2.1 Prueba de carga.

Son la simulación de las cargas de trabajo típicas de múltiples usuarios realizando procesos de negocio típicos. Indican y validan la respuesta de la aplicación al ser sometida a una carga de usuarios y/o transacciones esperadas, una vez en el ambiente de producción. Se centran en medir la eficacia de cada uno de los componentes individuales del sistema. Sirven para medir la respuesta de la aplicación ante distintos volúmenes de carga esperados (Software Quality System S.A, 2017). Teniendo en cuenta lo que expresa *Software Quality System S.A,* se agrega que las pruebas de cargan generan un entorno de trabajo pesado para un sistema, de modo que se someta a este a una carga de trabajo con el fin de verificar la respuesta ante tal situación.

1.2.3 Pruebas de estrés.

Se trata de cargar el sistema o los componentes del mismo hasta que llegan a los límites de funcionamiento. Permiten encontrar el volumen de datos o el tiempo en que la aplicación deja de ser capaz de responder a las peticiones como se espera (Software Quality System S.A, 2017). Somete al sistema a condiciones extremas de trabajo, de modo que tenga que trabajar muy cerca del límite de sus capacidades o sobre este. Permite verificar hasta qué punto de estrés de trabajo el sistema es capaz de responder correctamente. Esta es una prueba con la cual se puede medir el nivel de eficiencia de un sistema.

1.2.4 Prueba de Volumen.

Consisten en la simulación de un gran volumen de datos a lo largo de un periodo largo de tiempo (Software Quality System S.A, 2017). De esta forma se somete al sistema a trabajar con grandes cantidades de información, el objetivo es verificar si un sistema es estable durante un largo período de tiempo.

1.3 Sistema SPEX

Actualmente el sistema SPEX se encuentra desplegado y funcionando en la Empresa Industrial para la Informática, las Comunicaciones y la Electrónica. Su función es someter a los diferentes componentes que comprende el hardware de la computadora a trabajos rigurosos o forzados, llevando así el hardware a trabajar muy cerca del límite de sus capacidades o sobre éste. También se conoce su función como estrés

del hardware. Los componentes que son puesto a prueba por dicho sistema son el procesador, la memoria RAM (*Ramdon Access Memory*, Memoria de Acceso Aleatorio), disco duro, hardware de video y hardware de audio. SPEX es capaz de generar un reporte con el resultado de las pruebas realizadas, así como su categoría (satisfactorias o con errores), en caso de presencia de errores, el sistema brinda la información relacionada a los mismos y una opción de apagar la computadora cuando acaben las pruebas. El modo de funcionamiento es manual, ya que el operario debe establecer la configuración interactuando directamente con el software, y una vez terminada la ejecución de las pruebas, el operario debe interactuar nuevamente con SPEX para tener acceso al reporte que éste genera. Todo este proceso sucede de forma secuencial, una computadora a la vez. Brinda opciones tales como:

- Cambiar lenguaje.
- Cambiar estilo y tema visual.
- Configurar a gusto cada prueba que se realizan.
- Guardar cada uno de los cambios realizados en un fichero de configuración, para cargarlos cuando inicie el software nuevamente.

El (

Anexo 3 SPEX*)* muestra el sistema SPEX en ejecución, en él se aprecia un resumen en forma de tabla donde se puede ver lo que está ocurriendo internamente en las pruebas, además de las ventanas tabuladas con información en forma de gráficas.

1.4 Tecnologías

Las tecnologías que se emplean para desarrollar la propuesta de solución fueron seleccionadas por las ventajas aprovechables para el desarrollo de la Aplicación Centralizada para la gestión de SPEX, teniendo en cuenta la integración y compatibilidad entre ellas, y también mencionar que son tecnologías utilizables en un entorno de desarrollo libre.

1.4.1 Framework PHP Symfony 2.

Un framework simplifica el desarrollo de las aplicaciones, ya que automatiza muchos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al

desarrollador a crear código más legible y más fácil de mantener. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. También, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (*Symfony Documentation*, 2016).

Se decide seleccionar el Framework Symfony en la versión 2 para desarrollar la propuesta de solución debido a que presentan características de mucha utilidad que facilitan en gran medida la realización de la aplicación. A esto se suma la orientación por parte del Jefe del Centro CESOL de utilizar el proyecto Bosón como parte del marco de trabajo de Symfony 2. La estructura interna que posee Symfony permite separar cada componente que forma parte de la aplicación, lo que permite un fácil acceso dichos componentes, tanto al desarrollador como a la persona encargada de ofrecer mantenimiento al software. También es muy favorable la facilidad con la que se puede desplegar la aplicación creada desde el entorno de desarrollo hacia el entorno de producción. Las características siguientes apoyan la decisión de la selección de Symfony como *framework* para el desarrollo de la propuesta de solución:

- Fácil de instalar y configurar en la mayoría de plataformas con la garantía de que funciona correctamente en los sistemas Windows y Unix.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a

largo plazo.

- Código fácil de leer que incluye comentarios que permite un mantenimiento muy sencillo
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros (Symfony Documentation, 2016).

1.4.2 Lenguaje PHP 5.4

PHP 5.4 (*Hypertext Pre-Processor*, Lenguaje de Programación Interpretado) es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML (*HyperText Markup Language*, lenguaje de marcas de hipertexto). Código abierto significa que es de uso libre y gratuito para todos los programadores que quieran usarlo. Incrustado en HTML significa que en un mismo archivo se puede combinar código PHP 5.4 con código HTML, siguiendo reglas. PHP 5.4 se utiliza para generar páginas web dinámicas, aquellas cuyo contenido no es el mismo siempre. Por ejemplo, los contenidos pueden cambiar en base a los cambios que haya en una base de datos, de búsquedas o aportaciones de los usuarios. (Manual de PHP, 2017). Lo que llevo a la selección de PHP como lenguaje de programación, además de ser el lenguaje en que esta desarrollar Bosón y Symfony, es la ventaja de ser un lengujae del lado del servidor, lo que se entiende como un lenguaje transparente al usuario. Al estar embebido o incrustado en el código HTML le proporciona a este el dinamismo y operabilidad deseados. Permite una potente y sencilla interacción con bases de datos mediante Doctrine, proporcionado por Symfony y que se encarga de tratar el acceso a los datos persistentes. Es importante destacar que PHP 5.4 posee librerias que le confieren al lenguaje una excelente integración con el protocolo SSH (*Secure Shell*, Intérprete de órdenes seguro), el cual permite una conexión remota de manera rápida y segura, en este caso entre las computadoras.

PHP posee características que favorecen en gran medida el desarrollo de la aplicación propuesta, las cuales son:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el

que se encarga de ejecutar el código y enviar su resultado HTML al navegador.

- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad.
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- También permite el manejo de excepciones.

Librerías de PHP 5.4 utilizadas

Entre las librerías presentes en el lenguaje de programación PHP 5.4, algunas se destacan en la implementación de la solución de la investigación planteada, ya que proporcionan la integración entre el *Framework* Symfony y el protocolo de conexión remota SSH. Estas permitirán gestionar las configuraciones, ejecución y reportes de SPEX remotamente. Estas librerías son:

- ***ssh2_connect*:** Establece una conexión a un servidor SSH remoto. Una vez conectado, el cliente debería verificar la clave de host del servidor, y entonces, autenticarse usando la contraseña o la clave pública.
- ***ssh2_auth_password*:** Autentica la conexión sobre SSH usando contraseña en texto plano.
- ***ssh2_exec*:** Ejecuta un comando en el servidor remoto.
- ***ssh2_scp_send*:** Copia un fichero desde el sistema de ficheros local a un servidor remoto usando el protocolo SCP (*Syntax for Secure Copy,* Sintaxis para copia segura).
- ***stream_set_blocking*:** Establece el modo bloqueo/no-bloqueo en un flujo. (Manual de PHP, 2017).

1.4.3 JavaScript 1.6

Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con este lenguaje se pueden crear diferentes efectos e interactuar con usuarios o los diferentes componentes de una página web. Posee varias características, es basado en acciones que posee menos restricciones y también es multiplataforma. Gran parte de la programación está centrada en describir

objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros. Es necesario resaltar que hay dos tipos de *JavaScript*: por un lado, está el que se ejecuta en el cliente, este es el *JavaScript* propiamente dicho, aunque técnicamente se denomina Navegador JavaScript. Pero también existe un JavaScript que se ejecuta en el servidor, es más reciente y se denomina *LiveWire JavaScript* (Valdés, 2007). En adición a lo que plantea Valdés, el funcionamiento de JavaScript se evidencia en gran medida en las acciones del usuario en una aplicación web ya que permite validar datos introducidos por él, mostrar mensajes relacionados con la acción solicitadas e incluso posibilita modificar la manera en que es presentada la información en las vistas.

1.4.4 CSS 3

CSS 3 (*Cascading Stylesheets*, Hojas de Estilo en Cascada) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y *XHTML*. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Al crear una página web, se utiliza en primer lugar el lenguaje *HTML/XHTML* para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, entre otros. Una vez creados los contenidos, se utiliza el lenguaje CSS 3 para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página (Eguiluz, 2017). Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos *HTML/XHTML* bien definidos y con significado completo también llamados documentos semánticos. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. CSS actúa directamente en las vistas o plantillas que se le muestran al cliente y con las cuales éste interactúa. Modifica el aspecto visual del contenido haciéndolo más entendible o estándar. Adicionar también que se pueden seguir pautas de diseño a la hora de modificar el estilo de las vistas al hacer uso de este lenguaje.

1.4.5 Servicio SSH 7.2

SSH 7.2 (*Secure Shell*, Intérprete de órdenes seguro) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios

conectarse a un host[1] remotamente. A diferencia de otros protocolos de comunicación remota tales como *FTP* (*File Transfer Protocol*, Protocolo de transferencia de archivos) o Telnet (*Telecommunication Network,* Red de Telecomunicaciones), SSH 7.2 encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

Está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la Shell de comando. Ya que las aplicaciones antiguas no encriptan contraseñas entre el cliente y el servidor. El uso de métodos seguros para registrarse remotamente a otros sistemas reduce los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto. La utilización de este protocolo en el desarrollo de la solución propuesta es crucial ya que proporciona una vía de acceso rápida y segura entre equipos mediante una conexión de red. Se utiliza por las grandes ventajas que posee, las cuales fueron tomadas en cuenta para decidir utilizar esta tecnología:

- Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.
- El cliente transmite su información de autenticación al servidor usando una encriptación robusta de 128 bits.
- Todos los datos enviados y recibidos durante la sesión se transfieren por medio de encriptación de 128 bits, lo cual los hacen extremamente difícil de descifrar y leer.
- El cliente tiene la posibilidad de reenviar aplicaciones X11[2] desde el servidor. Esta técnica, llamada reenvío por X11, proporciona un medio seguro para usar aplicaciones gráficas sobre una red (Red Hat, 2005).

1.5 Herramientas homólogas de administración remota

Se realiza un estudio del arte de las herramientas capaces de realizar administración remota de sistemas o aplicaciones de pruebas de hardware o software. Las herramientas estudiadas son las siguientes:

Applications Manager V13

[1] ***Host*** *o anfitrión se usa en informática para referirse a las computadoras u otros dispositivos conectados a una red.*

[2] *Software que fue desarrollado para dotar de una interfaz gráfica a los sistemas Unix, X es el encargado de mostrar la información gráfica de forma totalmente independiente del sistema operativo. Generalmente se refiere a la versión 11 de este protocolo.*

Es una solución completa de monitorización de aplicaciones de negocio, que ayuda a evitar cortes y degradaciones de servicio de sus aplicaciones críticas a través de una monitorización proactiva de todos sus parámetros. Permite monitorizar sistemas y aplicaciones críticas de negocio tales como servidores de aplicaciones, bases de datos, servicios y servidores web. Ofrece al administrador herramientas potentes de diagnóstico que ayuden a identificar y diagnosticar los problemas antes de que impacten en los usuarios. Además, permite monitorizar la experiencia de usuario final y el tiempo de respuesta de cada transacción dentro de diferentes aplicaciones.

Principales características:

- Descubrimiento automático de aplicaciones.
- Más de mil indicadores claves de rendimiento de las aplicaciones.
- Vista holística[3] de las aplicaciones de negocio y servicios de red en una única entidad.
- Análisis de causa y raíz de los problemas.
- Emisión de alertas en base a superación de umbrales o detección de anomalías.
- Gestión de fallos con múltiples posibilidades de notificación y acciones.
- Potentes capacidades de reportes.
- Interfaz de usuario habilitada para teléfonos inteligentes.
- REST API para integración con otras herramientas de monitorización
- Integración de Vistas de Negocio en Google Maps (Mapas de Google), (IREO, 2017).

KACE K1000

El dispositivo de administración de sistemas K1000 proporciona herramientas de monitoreo y administración de servidores integrales como parte de sus capacidades de administración de sistemas, incluidos inventarios sin agente, monitoreo de registros de sistema, distribución de software y capacidades de servicio al usuario para sistemas *Windows*, *Linux*, *UNIX* y *Mac*.

Cuenta con funciones tales como:

- Monitoreo de servidores: Experimenta un nivel más alto de monitoreo de registro de servidor para plataformas *Windows*, *Mac*, *Linux* y *UNIX*, que es fácil de configurar y está completamente

[3] Tendencia o corriente que analiza los eventos desde el punto de vista de las múltiples interacciones que los caracterizan. Supone que todas las propiedades de un sistema no pueden ser determinadas o explicadas como la suma de sus componentes. Considera que el sistema completo se comporta de un modo distinto que la suma de sus partes.

integrado en la mesa de servicio del dispositivo *K1000* y la aplicación móvil

- Mesa de servicio: La mesa de servicio del dispositivo *K1000* es parte de una sola UI (*User Interface,* Interfaz de usuario) completamente integrada en las capacidades de administración del servidor y activos. Disfrute de un acceso fácil a información relativa según el contexto en todos sus equipos y servidores, o proporcione soporte remoto a los clientes directamente desde la mesa de servicio del dispositivo K1000.
- Inventario sin agentes de las plataformas *Windows, Unix, Linux y Mac*: El dispositivo K1000 proporciona un inventario sin agentes para los servidores y las computadoras con Windows, al igual que para sistemas *Linux, UNIX y Mac.* Puede asegurarse de que no se desperdicien recursos para rastrear el entorno del servidor y evitar posibles problemas de cumplimento con la adhesión a las reglamentaciones aplicables.
- Generación de informes: La interfaz web simple del dispositivo K1000 permite generar informes precisos en un instante. Todo su inventario de hardware, software y sistema operativo puede verse rápidamente en la ficha de inventario, o desde la sección de alertas e informes, y la generación de informes basada en un asistente del dispositivo K1000 facilita más que nunca la llegada de esa información a las manos correctas (Quest Software Inc., 2017).

UltraVNC

Es una aplicación para Windows que permite controlar otros equipos de forma remota desde el protocolo VNC (*Virtual Network Computing*, Computación Virtual en Red), mediante la parte cliente, y que otros equipos accedan al mismo de forma remota mediante la parte servidor. Permite visualizar el escritorio de otra computadora con cualquier sistema operativo de forma remota desde el propio equipo, siempre que éste disponga de un servidor que implemente el protocolo VNC, además de tomar el control del mismo si así se desea. Esta funcionalidad facilita la asistencia técnica a distancia, entre otros aspectos. Dispone de una opción de configuración por la cual adaptará automáticamente la calidad gráfica de los datos recibidos desde el equipo remoto al que se está accediendo, resolviendo problemas de excesivo consumo de ancho de banda o de latencia[4] en la transmisión de información.

[4] *En redes informáticas de datos se denomina **latencia** a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.*

La herramienta dispone de una ventana que muestra la información del estado actual de la conexión establecida, pudiendo monitorizar el ancho de banda consumido durante la visualización del escritorio remoto. Permite configurar una amplia variedad de parámetros que influirán en el establecimiento de la conexión con un equipo remoto. De esta forma, se podrá adaptar dicha conexión a las necesidades del administrador y establecer restricciones durante la misma (Centro de Apoyo Tecnológico a Emprendedores, 2012).

Remmina

Es una alternativa para disponer de un software de escritorio remoto en el sistema operativo Linux. Este software permite conectarse de forma remota a un ordenador utilizando diferentes protocolos para ello y sin la necesidad de instalar más software adicional en el ordenador cliente.

Remmina soporta los siguientes protocolos:

- RDP (*Remote Desktop Protocol,* Protocolo de escritorio remoto).
- VNC (*Virtual Network Computing*, Computación Virtual en Re).
- XDMACP (*X Display Manager Control Protocol*, Protocolo de Control de Administrador de la Pantalla X).
- SSH (*Secure Shell*, Intérprete de órdenes seguro) (Velasco, 2014).

Adicionar también que Remmina no se limita a conectarse a otros sistemas idénticos, sino que permite también la conexión entre sistemas operativos diferentes siempre y cuando cumplan con los estándares de protocolo, por ejemplo, aunque Linux por defecto no disponga de un servidor RDP, sí podrá conectarse a un ordenador con Windows desde este software para Linux.

Después de realizado el estudio de las herramientas mencionadas anteriormente se resume que, existen disimiles herramientas cuyo propósito es acceder y compartir escritorios de manera remota, ya sea del sistema operativo Windows o de un sistema operativo de Linux. Por tanto, no son reflejadas en el estudio realizado, ya que no guardan relación con el objetivo de la presente investigación.

Las herramientas que se estudiaron cuentan con diferentes características, las cuales conducen a que cada una realice su función correctamente. Entre ellas existen herramientas tanto multiplataforma como software privativo, que no cumplen con la política de utilización de software libre implementada en el país, además de no trabajar con el rigor requerido para las necesidades planteadas. En la solución para el problema de la presente investigación se utilizará software libre o software de código abierto, en apoyo al proceso de migración del país. Las herramientas anteriores no cuentan con un sistema que permita

interactuar directamente con el hardware físico de la computadora, para así gestionar las configuraciones que requiere cada componente para la ejecución de SPEX. También se encuentra como deficiencia la no utilización del protocolo SSH establecer conexiones remotas. Por las razones anteriores se decide desarrollar un sistema propio que permita interactuar directamente con el hardware, y sea capaz de centralizar los procesos que realiza SPEX. Particularmente los procesos de configuración, ejecución y obtención del reporte de pruebas que se genera al final del proceso de pruebas.

1.6 Metodología para el desarrollo de software

La metodología de software a utilizar en la presente investigación será la variación de AUP para la UCI (AUP-UCI), debido a que, por resolución, es esta la metodología a utilizar en la línea de productos elaborados en la Universidad de las Ciencias Informáticas.

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición), AUP-UCI mantiene la fase de Inicio, pero modifica el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la fase de Ejecución y se agrega la fase de Cierre de la siguiente forma:

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Sánchez, 2015).

Conclusiones parciales

Se realizó un estudio que demostró la necesidad de desarrollar una aplicación nueva para gestionar los procesos del sistema SPEX. Se describieron las tecnologías que se utilizaran en la fase de implementación.

Capítulo 2. Análisis y Diseño

2.1 Introducción

En el presente capítulo se describen las características de la solución propuesta de manera que facilite un mayor entendimiento de la misma. De esta forma se describe la propuesta de solución para el desarrollo del sistema SPEX. Se definen los principales conceptos asociados al mismo, se especifican los requisitos funcionales y no funcionales. Se realiza el diseño de las clases y se definen los patrones a utilizar.

2.2 Propuesta de solución

Se propone desarrollar una solución informática que se encargue de centralizar los procesos del sistema SPEX, que permita definir las configuraciones de los diferentes componentes de hardware como disco duro, memoria RAM (*Ramdon Access Memory,* memoria de acceso aleatorio), sistema de video, procesador y los dispositivos de entrada y salida de audio de manera remota. Además, debe ser capaz de ejecutar el sistema SPEX en las computadoras y obtener posterior a la ejecución un reporte con el resultado de las pruebas realizadas. Esta solución contará con una interfaz de fácil interacción. Entre sus funciones estará, realizar búsquedas de los lotes existentes mediante filtrado por nombre o por fecha, mostrar gráficos generados a partir de la información registrada en la base de datos, relacionada con los lotes de computadoras y las pruebas realizadas. También será posible cambiar el idioma de presentación en las vistas que se muestran al usuario. Contará con una opción para decidir si apagar las computadoras que ya fueron objeto de prueba, una vez terminado el proceso. En caso de presencia de errores, se puede optar por la opción apagar o mantener encendida las computadoras.

2.3 Requisitos

Los requisitos de un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos o requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de algún dispositivo, hacer un pedido o encontrar información.

Técnica de obtención de requisitos.

Se utilizó la técnica **tormenta de ideas** para obtener los requisitos funcionales de la Aplicación Centralizada que se propone como solución al problema de la presente investigación. Esta es una técnica

que consiste en dar oportunidad, a todos los miembros de un grupo reunido, de opinar o sugerir sobre un determinado asunto que se estudia, ya sea un problema, un plan de mejoramiento u otra cosa, de esta forma se aprovecha la capacidad creativa de los participantes *(Anexo 1 Técnica tormenta de ideas*).

2.3.1 Requisitos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos los requerimientos funcionales tienen la posibilidad de declarar lo que el sistema no puede hacer. Dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque tomado al redactar los requisitos (Pressman, 2010).

Se considera también que los requisitos funcionales son aquellas funciones que el software debe ser capaz de desarrollar, y por tanto cada uno de ellos tiene que ser descritos de manera correcta y concreta, de forma tal que se facilite el entendimiento de los mismos.

Se estima la prioridad de cada requisito de acuerdo a importancia que representan para el negocio, de manera que la prioridad alta se refiere a los requisitos más imprescindibles.

A continuación, se muestra una tabla que muestra los requisitos o requerimientos de software obtenidos mediante la técnica aplicada:

Tabla 1. Requisitos funcionales

(Fuente: elaboración propia)

Prioridad Baja	Prioridad Media	Prioridad Alta
1. Adicionar lote. 2. Eliminar lote. 3. Editar lote. 4. Listar lotes. 5. Adicionar computadora. 6. Eliminar computadora.	13. Filtrar lote. 14. Modificar computadora. 15. Configurar disco duro. 16. Configurar memoria. 17. Configurar procesador. 18. Configurar video. 19. Configurar audio. 20. Configurar ejecución de	21. Mostrar gráfico de lotes por fecha 22. Mostrar gráfico de computadoras por fechas. 23. Mostrar gráfico de pruebas por lote. 24. Mostrar gráfico de errores por lote.

7. Listar computadoras. 8. Adicionar prueba. 9. Listar prueba. 10. Adicionar tipo de error. 11. Listar tipo de error. 12. Cambiar idioma.	SPEX.	25. Generar fichero de configuración. 26. Ejecutar aplicación SPEX.

Para validar los requisitos se decidió emplear la técnica de los **prototipos de interfaz de usuario** que permite obtener la descripción de requisitos en prototipos, que, al ser mostrados al cliente, éste puede hacerse una idea de la estructura de la interfaz del sistema. Con esta técnica se debe asegurar que el usuario entienda que lo que está viendo es un prototipo y no el sistema final.

Los prototipos de sistema permiten a los usuarios experimentar para ver cómo éste ayuda a su trabajo. Fomentan el desarrollo de ideas que desembocan en requerimientos. Además de permitir a los usuarios mejorar las especificaciones de requerimientos, el desarrollo de un prototipo tiene otras ventajas:

- Al demostrar las funciones del sistema se identifican las discrepancias entre los desarrolladores y los usuarios.
- Durante el desarrollo del prototipo, el personal del desarrollo de software puede darse cuenta de que los requerimientos son inconsistentes o están incompletos.
- Se dispone rápidamente de un sistema que funciona y demuestra la factibilidad y usabilidad de la aplicación a administrar.
- El prototipo se utiliza como base para escribir la especificación para la producción.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que realiza el software, sino a las propiedades emergentes de éste. El autor de la presente investigación define los requisitos no funcionales como las restricciones que deben cumplirse para lograr que el sistema desarrollado sea capaz de realizar su función. Estas restricciones pueden ser de diferentes tipos. A continuación, se muestran los siguientes requisitos no funcionales:

Restricciones de lenguaje

- **RNF 1**: El lenguaje de programación a utilizar para desarrollar la Aplicación Centralizada para la gestión de SPEX tiene que ser PHP 5.4, ya que por orientación de Jefe de Centro se debe usar Bosón (Framework base para los proyectos desarrollados en PHP en la UCI), el cual es un proyecto para Symfony que también está desarrollado totalmente en PHP.

Restricciones de software

- **RNF 2**: En el entorno donde se va a desplegar la Aplicación Centralizada para la gestión de SPEX, tiene que estar instalado *PostgreSQL* [5] en la versión 9.3 o superior como gestor de base de datos y Apache 2.4 o superior como servidor web.
- **RNF 3**: El sistema SPEX 1.0 tiene que estar instalado en las computadoras donde se desplegará la Aplicación Centralizada para la gestión de SPEX.
- **RNF 4**: Las computadoras que van a ser objeto de prueba tienen que tener instalado un servidor SSH en la versión 7.2 o superior, para aceptar las conexiones que provienen de la Aplicación Centralizada para la gestión de SPEX.

2.4 Historias de usuario

Las historias de usuario (HU) son requerimientos ya que expresan el problema que el sistema o producto software debe resolver. Son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad (Izaurralde, 2013). Se tiene en cuenta lo expresado por Isaurralde al describir las historias de usuario y se agrega que son artefactos que posibilitan entender de una manera más detallada el funcionamiento de cada uno de los requerimientos funcionales de un software. Describe su funcionamiento y también condiciones que deben cumplirse para dar solución a un requisito. Posibilitan además al programador que se designe para realizar mantenimiento a la aplicación desarrollada, nutrirse de conocimiento relacionado al funcionamiento interno de cada funcionalidad. Se decidió utilizar el artefacto HU para describir los requisitos de software, haciendo uso del escenario número cuatro de la metodología AUP-UCI y se

[5] Sistema de gestión de bases de datos relacional orientado a objetos; software libre, publicado bajo la licencia *PostgreSQL*

describe una HU por cada requisito funcional. A continuación, se describen nueve HU, las restantes serán descritas en el (*Anexo 2 Historias de usuario*).

Tabla 2. Historia de usuario: Adicionar lote.

(Fuente: elaboración propia)

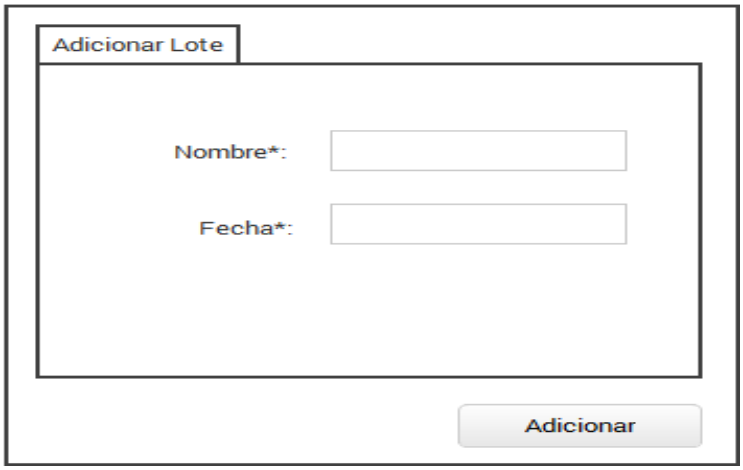
Historia de Usuario	
Nombre: Adicionar lote.	**Número:** HU_1
Modificaciones: Ninguna.	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 1
Prioridad en el negocio: Alto.	**Puntos estimados:** 2
Riesgo en desarrollo: Medio.	**Puntos Reales:** 2
Descripción: Permite adicionar un lote en el sistema. Para ello se introduce el nombre del lote y su fecha de creación. Una vez adicionado el lote, será mostrado en una lista, desde donde se podrá acceder a las opciones del lote, las cuales son ver reporte, abrir o cerrar el lote, acceder a la configuración del lote y por último eliminarlo.	
Observaciones:	
Prototipo de interfaz de usuario: 	

Tabla 3. Historia de usuario: Filtrar lote

(Fuente: elaboración propia)

Historia de Usuario	
Nombre: Filtrar lote	**Número:** HU_2
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 1
Prioridad en el negocio: Bajo	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite al usuario buscar un lote por su nombre o por la fecha de creación. Al escribir en el campo de búsqueda se ira filtrando en el listado de lotes los que cumplan con la expresión de búsqueda escrita.	
Observaciones:	
Prototipo de interfaz de usuario: 	

Tabla 4. Historia de usuario: Configurar procesador

(Fuente: elaboración propia)

Historia de Usuario	
Nombre: Configurar procesador	**Número:** HU_3
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 1
Prioridad en el negocio: Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos Reales:** 1
Descripción: Posibilita al usuario definir la configuración del procesador para la ejecución de las	

pruebas. Al mostrarse el formulario para realizar esta funcionalidad, se cargan las características del procesador de las computadoras, en conjunto con una serie de opciones para seleccionar el tipo de prueba específica que se le va a realizar a este componente de hardware.

Observaciones: Debe estar adicionada como mínimo una computadora en el lote.

Prototipo de interfaz de usuario:

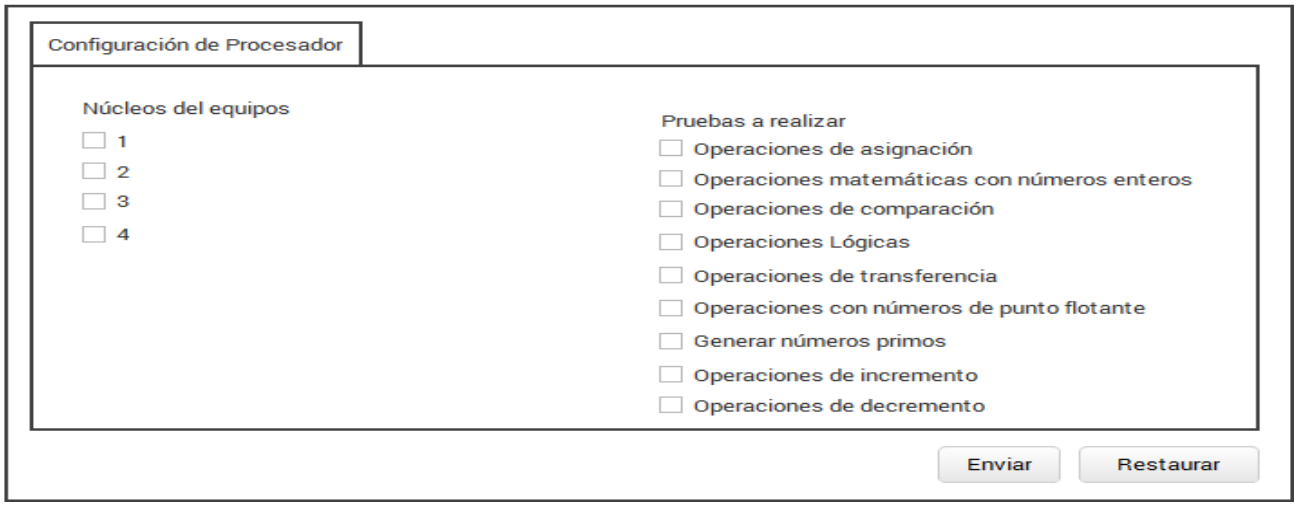


Tabla 5. Historia de usuario: Listar computadora

(Fuente: elaboración propia)

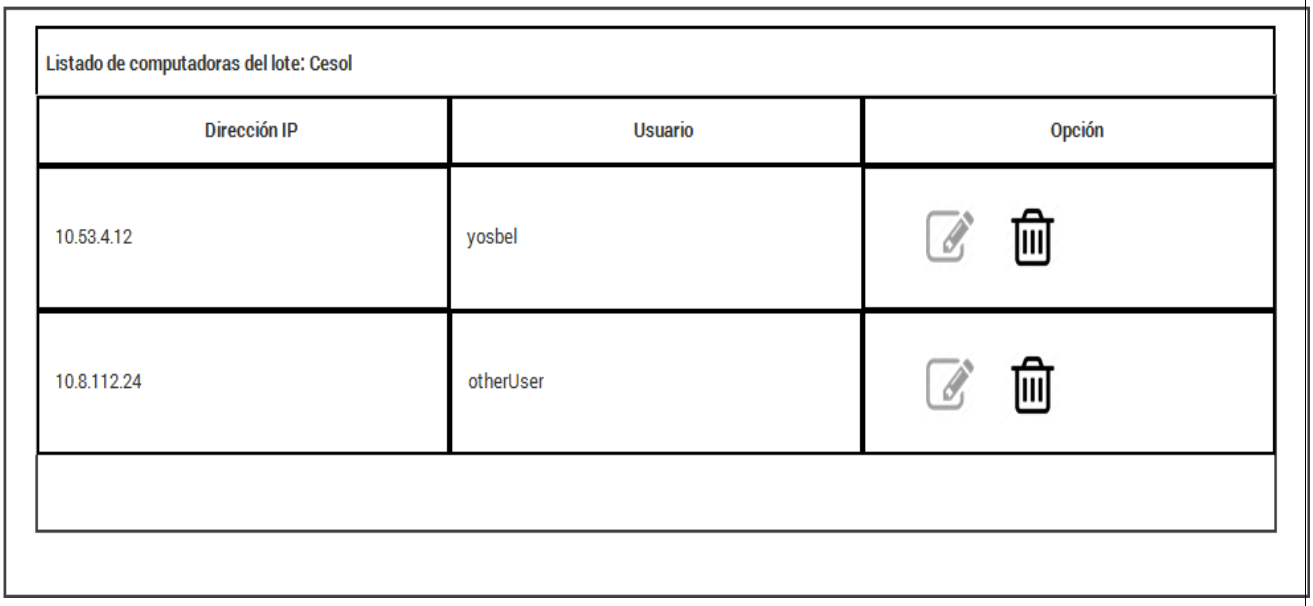
Historia de Usuario	
Nombre: Listar computadora	**Número:** HU_4
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 1
Prioridad en el negocio: Medio	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Muestra todas las computadoras que pertenecen a un lote y en el listado existen opciones para cada una de ellas, donde se puede editar o eliminar la computadora que se desee.	
Observaciones:	
Prototipo de interfaz de usuario: 	

Tabla 6. Historia de usuario: Configurar memoria

(Fuente: elaboración propia)

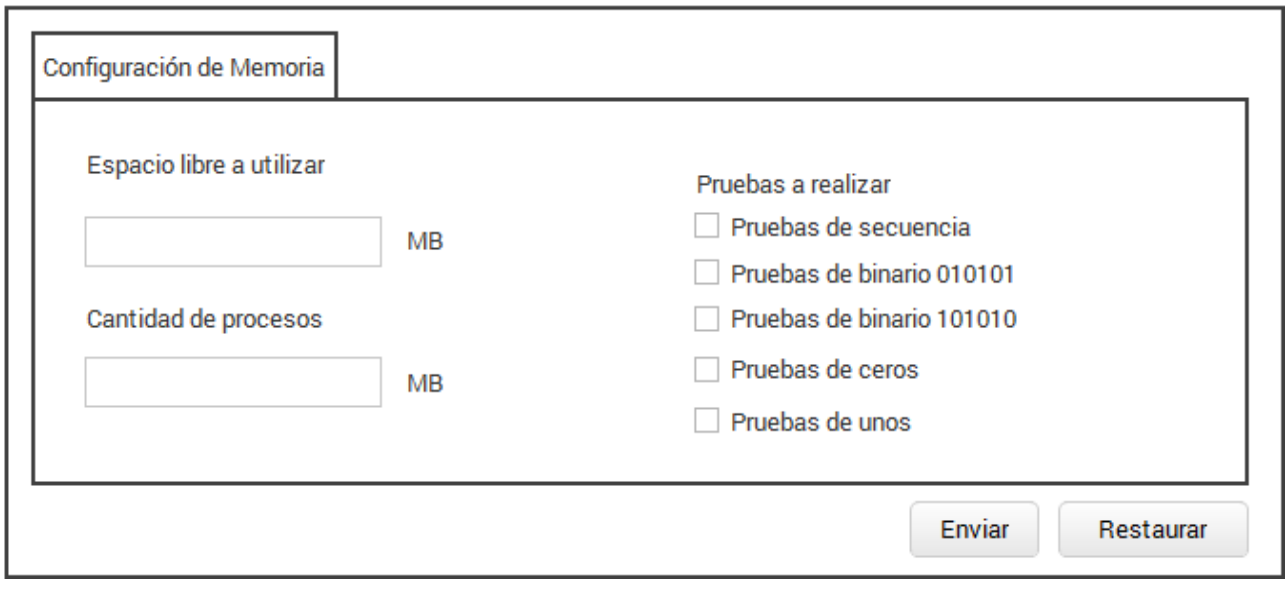
Historia de Usuario	
Nombre: Configurar memoria.	**Número:** HU_5
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 1
Prioridad en el negocio: Alto	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos Reales:** 1
Descripción: Permite establecer la configuración de la memoria RAM para el proceso de pruebas. Al mostrarse el formulario para esta funcionalidad, esta muestra la cantidad de espacio libre en la computadora. Muestra opciones para seleccionar los tipos de pruebas específicos que se le harán a la memoria durante la ejecución de SPEX, y también se puede definir la cantidad de procesos que se ejecutarán.	
Observaciones:	
Prototipo de interfaz de usuario: 	

Tabla 7. Historia de usuario: Adicionar Computadora

(Fuente: elaboración propia)

Historia de Usuario	
Nombre: Adicionar computadora	**Número:** HU_6
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite adicionar una computadora a un lote existente en el sistema. La computadora se crea con parámetros como la dirección IP, el usuario y contraseña de la misma.	
Observaciones: El lote debe estar previamente creado. El campo para introducir la dirección IP cuenta con método de validación escrito en JavaScript, su modo de funcionamiento es permanecer en un color rojo hasta tanto la dirección IP introducida por el usuario cumpla con el formato correcto, de lo cual el usuario se dará cuenta al cambiar el color rojo por el color negro.	
Prototipo de interfaz de usuario	

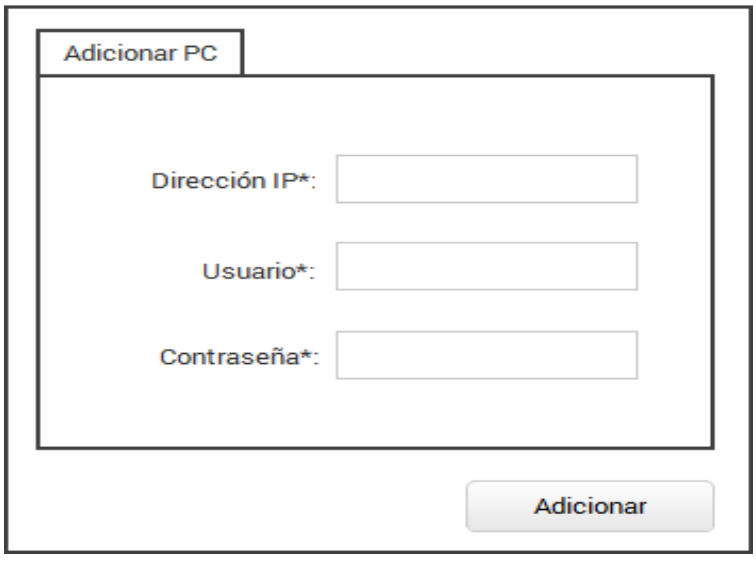


Tabla 8. Historia de usuario: Adicionar test

(Fuente: elaboración propia)

Historia de Usuario	
Nombre: Adicionar prueba	**Número:** HU_7
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos Reales:** 1
Descripción: Brinda la posibilidad de adicionar una prueba realizada. Las pruebas son enviadas como parte del reporte que genera SPEX al finalizar su ejecución. Este reporte llega con formato de objeto JSON que es interpretado por la Aplicación Centralizada para la gestión de SPEX.	
Observaciones: Esta HU no cuenta con un prototipo de interfaz, debido a que esta funcionalidad se ejecuta desde una petición realizada por el sistema SPEX al terminar su ejecución.	

Tabla 9. Historia de usuario: Generar fichero de configuración.

(Fuente: elaboración propia)

Historia de Usuario	
Nombre: Generar fichero de configuración	**Número:** HU_8
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos Reales:** 1
Descripción: Permite generar el fichero de configuración y enviarlo a las computadoras que se van a probar. Para realizar esta función se tiene en cuenta la configuración de disco, de audio, de memoria, de video y de procesador. También se integra en dicho fichero la configuración relacionada con la ejecución de SPEX, donde se definen opciones de configuración como por	

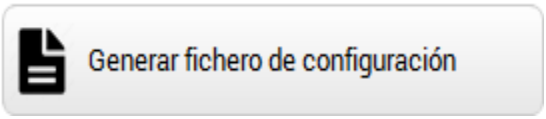
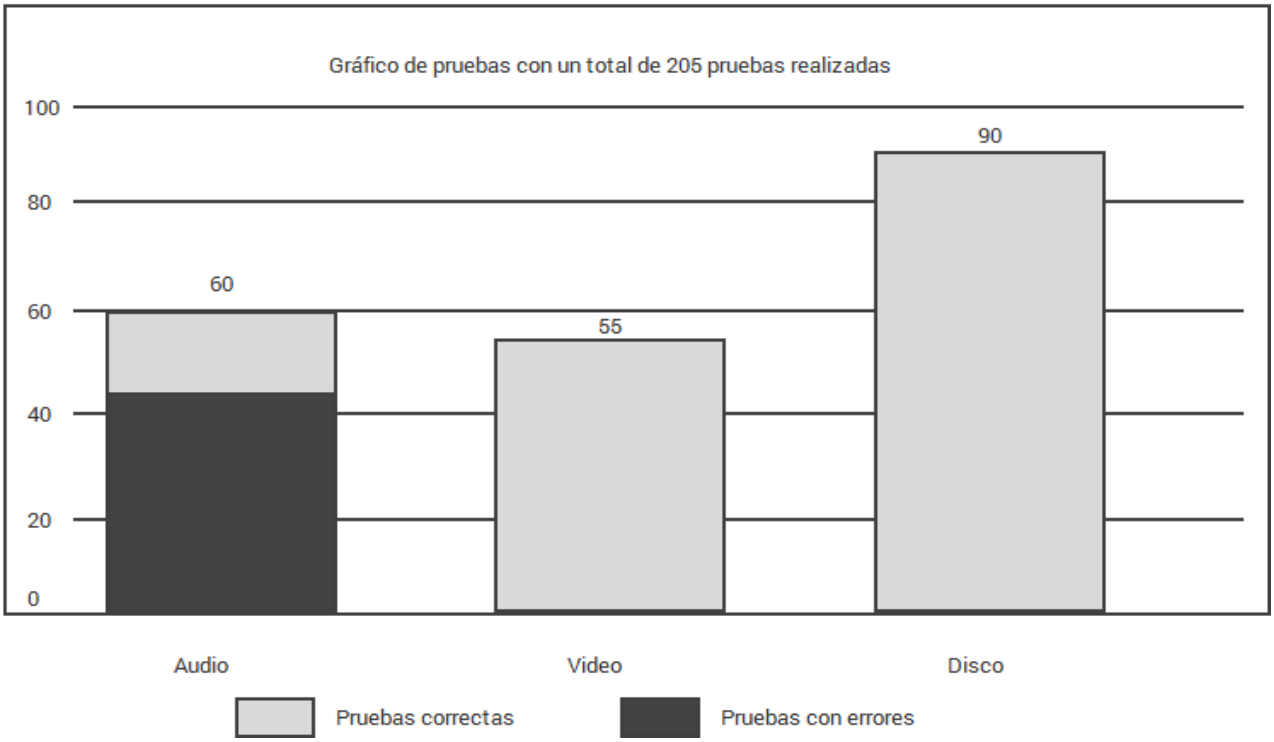
ejemplo el tiempo que durara el proceso de pruebas, el tiempo con que se actualizarán las diferentes vistas que posee SPEX y el orden de realización de las pruebas a los diferentes componentes de hardware antes mencionados. Una vez generado el fichero será copiado en cada una de las computadoras del lote.
Observaciones: Antes de generar el fichero, previamente deben estar establecidas las configuraciones de todos los componentes de hardware (disco duro, memoria, video, procesador, audio) y también la configuración relacionada con la ejecución de SPEX. Este fichero va a tener toda la configuración detallada, donde en cada línea habrá un elemento de la misma. Este fichero es el que dirige todo el proceso de ejecución de SPEX.
Prototipo de interfaz de usuario: 

Tabla 10. Historia de usuario: Mostrar gráfico de pruebas por lote

(Fuente: elaboración propia)

Historia de Usuario	
Nombre: Mostrar gráfico de pruebas por lote	**Número:** HU_9
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos Reales:** 1
Descripción: Muestra una gráfica donde se resume el total de pruebas que se le realizó a un lote de computadoras, tanto la cantidad de pruebas correctas como la cantidad de pruebas con errores. Para generar la gráfica se usan los datos existentes en la base de datos de la aplicación.	
Observaciones:	

Prototipo de interfaz de usuario:



2.5 Arquitectura del sistema

La arquitectura seleccionada en la presente investigación para el desarrollo de la propuesta de solución es Modelo-Vista-Controlador (*MVC*), ya que es el que utiliza el *Framework* de desarrollo Symfony. Debido a que el sistema será implementado usando este *Framework*, fue necesario mantener la arquitectura que éste ya utiliza. Entre sus ventajas está la posibilidad de separar la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Potencier, 2017). Symfony implementa la arquitectura MVC

de la siguiente forma:

- La capa del Modelo
  - Abstracción de la base de datos
  - Acceso a los datos
- La capa de la Vista
  - Vista
  - Plantilla
  - Layout[6]
- La capa del Controlador
  - Controlador frontal
  - Acción

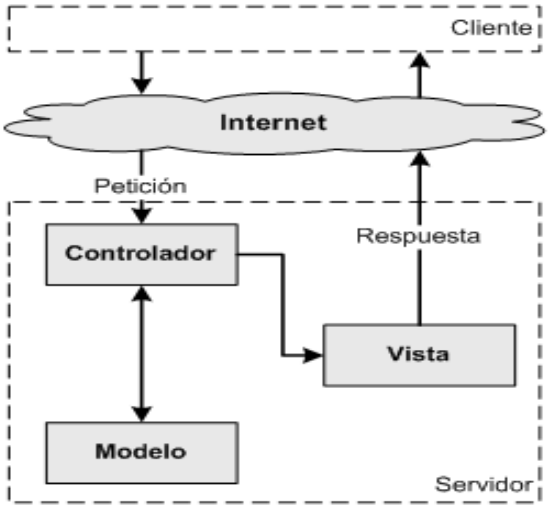


Figura 1. Arquitectura de software

(Fuente: Documentación de Symfony)
(Potencier, 2017)

[6] Cuadrícula imaginaria que divide en espacios o campos la página que se diseña para facilitar la distribución de elementos como textos o gráficos en la misma

En total son siete scripts, lo que parecen muchos archivos para abrir y modificar cada vez que se crea una página. Afortunadamente, Symfony simplifica este proceso, obtiene lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. En primer lugar, el controlador frontal y el *layout* son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios *layouts*, pero solamente es obligatorio tener uno de cada uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática (Potencier, 2017).

2.6 Patrones de diseño

Los patrones de diseño son propuestas generales planteadas en término de una colaboración de objetos mediante las que se resuelven una gran cantidad de problemas que aparecen una y otra vez en el desarrollo de aplicaciones informáticas, se consideran una serie de buenas prácticas de aplicación recomendable en el diseño de software (Pressman, 2010).

2.6.1 Patrones GRASP

GRASP es un acrónimo por sus siglas en inglés (*General Responsibility Asignment Software Patterns*, Patrones Generales de Software para Asignación de Responsabilidades*).* La asignación eficiente de responsabilidades a los componentes del software es la actividad más importante en el análisis y diseño orientado a objetos. Entre los patrones existentes se seleccionaron los que se describen a continuación, ya que fundamentan cuestiones y aspectos importantes del diseño en la propuesta de solución.

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema (Mora, 2005):

- Lógica de negocio.
- Persistencia a la base de datos.
- Interfaz de usuario

Su uso es evidente en todas las clases, ya que cada una conoce su información e implementa sus

funcionalidades, como es el caso de la clase *loteController*.

Creador: Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando (Mora, 2005):

1. B contiene a A.
2. B es una agregación (o composición) de A.
3. B almacena a A.
4. B tiene los datos de inicialización de A (datos que requiere su constructor).
5. B usa a A.

Se evidencia este uso en la clase *lotePCController* en el método *createAction*, al crear instancias de la clase entidad *lotePC*, como se muestra en la *Figura 3*.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema (Mora, 2005). En la solución plantada se evidencia este patrón en todos los métodos de las clases controladoras.

Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto identificable. Ejemplos de una baja cohesión son clases que hacen demasiadas cosas. En todas las metodologías se considera la refactorización. Uno de los elementos a refactorizar son las clases saturadas de métodos. Éste funcionamiento está presente en todas las clases del sistema, como por ejemplo en la *lotePCController*.

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia (Mora, 2005). Éste patrón se evidencia en la poca relación existente entre las clases que conforman el componente, por tanto, está presente en la clase *spexController*.

2.6.2 Patrones GOF (Gang of Four)

Los patrones GOF describen soluciones simples y elegantes a problemas específicos en el diseño de software. Se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de

comportamiento.

Decorador: Permite añadir dinámicamente nuevas responsabilidades a un objeto o clase, se aplica con la intención de proporcionar una forma flexible de introducir o eliminar funcionalidad de un componente sin modificar su apariencia externa o su función, se evidencia en la clase *pcController* a la hora de modificar los atributos de un objeto clase.

Registry: Es un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Es muy útil su utilización en la Programación Orientada a Objetos, su uso está presente en la comunicación entre la clase *lotePCController* y las plantillas mediante las rutas de direcciones con parámetros asociados.

Fachada: Define una interfaz que hace que el componente sea más fácil de usar. Propone la definición de un único punto de conexión con un componente, este objeto fachado presenta una interfaz unificada y es responsable de colaborar con otros componentes.

2.7 Tarjetas Clase Responsabilidad Colaborador (CRC)

El modelado CRC proporciona una manera sencilla de identificación y organización de las clases que son relevantes para los requerimientos de un sistema o producto. Se describe de la siguiente forma:

CRC en realidad es un conjunto de tarjetas índice estándar que representan clases. Las tarjetas se dividen en tres secciones (Pressman, 2010).

- En la parte superior de la tarjeta se escribe el nombre de la **clase**.
- En la parte izquierda del cuerpo se enlistan las **responsabilidades** de la clase.
- En la derecha, los **colaboradores**.

CRC hace uso de tarjetas índice reales o virtuales. El objetivo es desarrollar una representación organizada de las **clases**. Las **responsabilidades** son los atributos y operaciones relevantes para la clase, es decir, cualquier cosa que la clase sepa o haga. Los **colaboradores** son aquellas clases que se requieren para dar a una clase la información necesaria a fin de completar una responsabilidad, implica una solicitud de información o de cierta acción (Pressman, 2010).

Clases. La taxonomía de tipos de clase puede ampliarse con las siguientes categorías (Pressman, 2010):

- **Clases de entidad:** Se extraen directamente del enunciado del problema. Es común que estas clases representen cosas almacenadas en una base de datos y que persistan mientras dure la aplicación.

- **Clases de frontera:** se utilizan para crear la interfaz que el usuario mira y con la que interactúa cuando utiliza el software. Se diseñan con la responsabilidad de administrar la forma en la que se presentan a los usuarios los objetos de entidad.
- **Clases de controlador:** administran una unidad de trabajo de principio a fin. Están diseñadas para administrar la creación o actualización de objetos de entidad, las instancias de los objetos de frontera en tanto obtienen información de los objetos de entidad, la comunicación compleja entre conjuntos de objetos y la validación de datos comunicados entre objetos o entre el usuario y la aplicación.

Responsabilidades. Existen cinco lineamientos para asignar responsabilidades a las clases (Pressman, 2010):

1. La inteligencia del sistema debe estar distribuida entre las clases para enfrentar mejor las necesidades del problema.
2. Cada responsabilidad debe enunciarse del modo más general posible.
3. La información y el comportamiento relacionado con ella deben residir dentro de la misma clase.
4. La información sobre una cosa debe localizarse con una sola clase, y no distribuirse a través de muchas.
5. Cuando sea apropiado, las responsabilidades deben compartirse entre clases relacionadas.

Colaboraciones. Una clase cumple sus responsabilidades en una de dos formas (Pressman, 2010):

1. Usa sus propias operaciones para manipular sus propios atributos, con lo que satisface una responsabilidad particular.
2. Colabora con otras clases.

A continuación, se presentan las tarjetas CRC asociadas a las clases del sistema:

Tabla 11. Tarjeta CRC: loteController

(Fuente: elaboración propia)

Clase: loteController	
Responsabilidades	**Colaboradores**
Listar lote	indexAction
Insertar lote	newAction createAction

Buscar lote	searchNameAction
Eliminar lote	deleteAction createDeleteForm refreshAction
Editar lote	editAction updateAction

Tabla 12. Tarjeta CRC: pcController

(Fuente: elaboración propia)

Clase: pcController	
Responsabilidades	**Colaboradores**
Listar computadora	showpcAction
Insertar computadora	newpcAction createpcAction
Eliminar computadora	delpcAction createDeleteForm refreshpcAction
Editar computadora	editpcAction updatepcAction createEditForm

Tabla 13. Tarjeta CRC: pluginsConfigController

(Fuente: elaboración propia)

Clase: pluginsConfigController	
Responsabilidades	**Colaboradores**
Establecer configuración de disco duro	discoConfAction
Establecer configuración de video	videoConfAction
Establecer configuración de procesador	procesadorConfAction
Establecer configuración de memoria RAM	memoriaConfAction

Establecer configuración de audio	audioConfAction

Tabla 14. Tarjeta CRC: pluginsController

(Fuente: elaboración propia)

Clase: pluginsController	
Responsabilidades	**Colaboradores**
Obtener características de disco duro	discoAction
Obtener características de video	videoAction
Obtener características de procesador	procesadorAction
Obtener características de memoria RAM	memoriaAction
Obtener características de audio	audioAction

Tabla 15. Tarjeta CRC: serviceTestController

(Fuente: elaboración propia)

Clase: serviceTestController	
Responsabilidades	**Colaboradores**
Obtener reporte generado en las pruebas	postReportAction

Tabla 16. Tarjeta CRC: spexController

(Fuente: elaboración propia)

Clase: spexController	
Responsabilidades	**Colaboradores**
Ejecutar el sistema SPEX	runSpexAction

Tabla 17. Tarjeta CRC: testPCController

(Fuente: elaboración propia)

Clase: testPCController	
Responsabilidades	**Colaboradores**

Listar prueba	indexAction showAction tabPcAction pruebasxPcAction erroresxPcAction
Insertar prueba	createAction createCreateFrom newAction
Eliminar prueba	deleteAction createDeleteForm
Editar prueba	editAction createEditForm updateAction

Tabla 18. Tarjeta CRC: typeErrorController

(Fuente: elaboración propia)

Clase: typeErrorController	
Responsabilidades	**Colaboradores**
Listar tipos de errores	indexAction showAction
Insertar tipo de error	createAction createCreateForm newAction
Eliminar tipo de error	deleteAction createDeleteForm
Editar tipo de error	editAction createEditForm updateAction

Tabla 19. Tarjeta CRC: typeTestController

(Fuente: elaboración propia)

Clase: typeTestController	
Responsabilidades	**Colaboradores**
Listar tipo de prueba	indexAction showAction
Insertar tipo de prueba	createAction createCreateForm newAction
Eliminar tipo de prueba	deleteAction createDeleteForm
Editar tipo de prueba	editAction createEditForm updateAction

Tabla 20. Tarjeta CRC: configController

(Fuente: elaboración propia)

Clase: configController	
Responsabilidades	**Colaboradores**
Definir configuración general de los plugins	configGeneralAction
Generar fichero de configuración	ficheroAction

2.8 Modelo de Datos

Un modelo de datos es una parte esencial en el proceso de modelado de una base de datos. Este es una estructura abstracta que documenta y organiza la información de los datos y las relaciones entre ellos. El propósito de un modelo de datos es, por una parte, representar los datos y por otra, ser comprensible. En él se describen los datos, las relaciones de datos y la semántica de los datos. En el (*Anexo 4 Modelo de datos)* referente al modelo de datos está ilustrado el *diagrama entidad relación (*DER*)* de la solución propuesta.

2.9 Diagrama de despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Los nodos se interconectan mediante soportes bidireccionales que pueden a su vez estereotiparse (Turmero, 2016). El siguiente diagrama fue elaborado haciendo uso de la herramienta de modelado Visual Paradigm en la versión 8.0.

Figura 2. Diagrama de Despliegue

(Fuente: elaboración propia)

El administrador, desde su estación de trabajo, podrá acceder al sistema a través de los protocolos *HTTP* o *HTTPS*, utilizando un navegador web Firefox en la versión 46 o superior, el sistema estará desplegado en un servidor de aplicaciones, con un servidor web Apache en la versión 2.4 o superior y un servidor de base de datos *PostgreSQL* en la versión 9.3 o superior. También tiene que estar instalado el sistema SPEX 1.0, quien es el encargado de realizar las pruebas. El servidor de aplicaciones establecerá conexiones a las estaciones de trabajo clientes, las cuales contarán con *GNU*/Linux Nova como sistema operativo instalado, además de un servidor SHH en la versión 7.2 o superior. El intercambio de información entre el servidor de aplicaciones y las estaciones de trabajo clientes será posible a través del protocolo SHH, viajando por el puerto 22.

Conclusiones

Mediante la aplicación de la metodología de software, el desarrollo de las historias de usuario, las tarjetas CRC y la descripción de la arquitectura de software, se realizó el análisis y diseño de la propuesta de solución para la presente investigación.

Capítulo 3. Implementación y Prueba

Introducción

Durante el desarrollo de un software o sistema informático, la etapa de implementación y prueba posee mucha importancia, ya que es donde se desarrolla cada componente del sistema y se realizan las pruebas para comprobar que se realizó la solución propuesta con los requerimientos necesarios. En el presente capítulo se abordan elementos referentes a la elaboración de la solución y su estructura. Se valida el diseño y la implementación de la solución mediante métricas y pruebas de caja blanca y caja negra.

3.1 Implementación de la solución.

Parte del estudio del arte realizado en el capítulo 1, con el apoyo del diseño de la propuesta de solución y los artefactos que sirvieron de ayuda para definir el mismo. Como resultado se obtendrá una aplicación que cumpla con las funciones necesarias para dar solución al problema existente en la Empresa Industrial para la Informática, las Comunicaciones y la Electrónica.

3.2 Estructura interna de la solución.

Para el sistema SPEX, la estructura interna esta complementada por el Framework de desarrollo web Symfony en su versión 2.

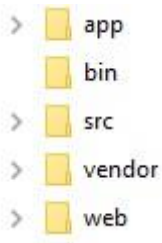


Figura 3. Directorio de Symfony

(Fuente: elaboración propia)

En la carpeta raíz de Symfony2 se encuentra el directorio **app/** el cual contiene los scripts encargados de proceso de carga del marco de trabajo, además de todo lo relacionado con la configuración general de la aplicación.

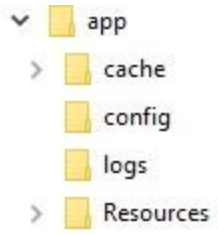


Figura 4. Directorio app/

(Fuente: elaboración propia)

Dentro de la carpeta **app/** se encuentra además la carpeta **cache/**, que contiene la información referente a los entornos de la aplicación, ya sea entorno de desarrollo, el cual se usa durante el proceso de desarrollo de la aplicación; y el entorno de producción, que es donde se despliega la aplicación una vez terminadas todas sus fases de desarrollo.

El directorio **config/** almacena los archivos de configuración de la aplicación para la conexión a la base de datos, las rutas y la seguridad. En la carpeta **logs/** se guardan los registros de la aplicación, y en la carpeta **Resources/** estarán los recursos que se utilizan, dentro existe una carpeta llamada **views/** que contiene la plantilla **base.html.twig** principal de la aplicación. Dentro de la carpeta **bin/** se encuentran scripts que están relacionados con el trabajo con Doctrine para la manipulación de la base de datos.

Figura 5. Directorio src/

(Fuente: elaboración propia)

Dentro de la carpeta **src/** se encuentra la carpeta **FOS/** que contiene el bundle **RestBundle/,** el cual se encarga de brindar el servicio para obtener el reporte generado en las pruebas realizadas. También reside en **src/** la carpeta **Pruebas/** y dentro de está el bundle **PruebasPCBundle/,** éste es el principal, contiene las funcionalidades y módulos necesarios para dar solución al problema de la presente investigación. A continuación, se explica el contenido de dicho bundle:

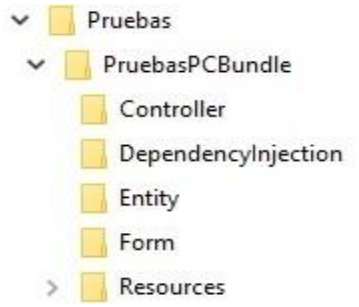


Figura 6. Directorio PruebasPCBundle

(Fuente: elaboración propia)

La carpeta **PruebasPCBundle/** es la que contiene toda la lógica del negocio y en su estructura contiene las siguientes carpetas:

- **Controller:** es la que contiene las clases controladoras del sistema, las cuales se encargan de gestionar las funcionalidades del mismo.
- **DependencyInjection:** contiene elementos relacionados con el contenedor de inyección de dependencias, además contiene las extensiones para las clases de inyección de dependencias.
- **Entity:** contiene las clases entidades que son convertidas en tablas físicas dentro de la base de datos de la aplicación.
- **Form:** contiene las clases que definen los tipos de formularios a emplear en las plantillas.
- **Resources:** dentro se encuentra la carpeta **config/**, donde se realizan todas las configuraciones específicas del bundle. Otra de las carpetas es **public/**, donde se aloja la subcarpeta **css/,** que contiene las clases css necesarias para darle la estructura gráfica a los formularios, plantillas y componentes que son mostrados al usuario, separando así el estilo del contenido. **images/,** en esta carpeta se encuentran las imágenes e iconos que se utilizan en el desarrollo de la solución**;** y **js/,** que almacena por cada funcionalidad los ficheros JavaScript necesarios para que el usuario interactúe con el sistema.

Las restantes carpetas que se encuentran en el directorio raíz son **Vendor/**: donde se alojan los bundles

de terceros, además, los componentes de Symfony2, el ORM Doctrine2 y el motor de plantillas **Twig**[7]. También el directorio **Web/**, que contiene el controlador frontal y todos los componentes estáticos de la aplicación como CSS, JavaScript e Imágenes.

3.3 Interfaces de usuario de la solución.

Dentro de las Interfaces de Usuario se distinguen básicamente dos tipos:

- Una interfaz de hardware, a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla visualizadora.
- Una interfaz de software, destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla.

Figura 7. Interfaz de usuario: Listar Lotes

(Fuente: elaboración propia)

[7] *Twig es un motor de creación de plantillas para utilizar con PHP. Se ocupa de brindar una solución al tratamiento de las cuestiones visuales alrededor de una aplicación desarrollada en este lenguaje.*

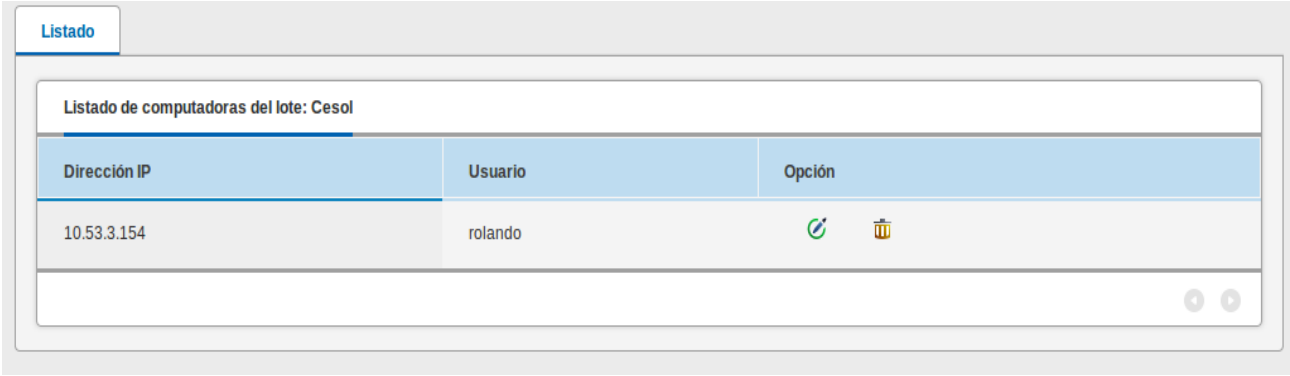


Figura 8. Interfaz de usuario: Listar Computadoras

(Fuente: elaboración propia)

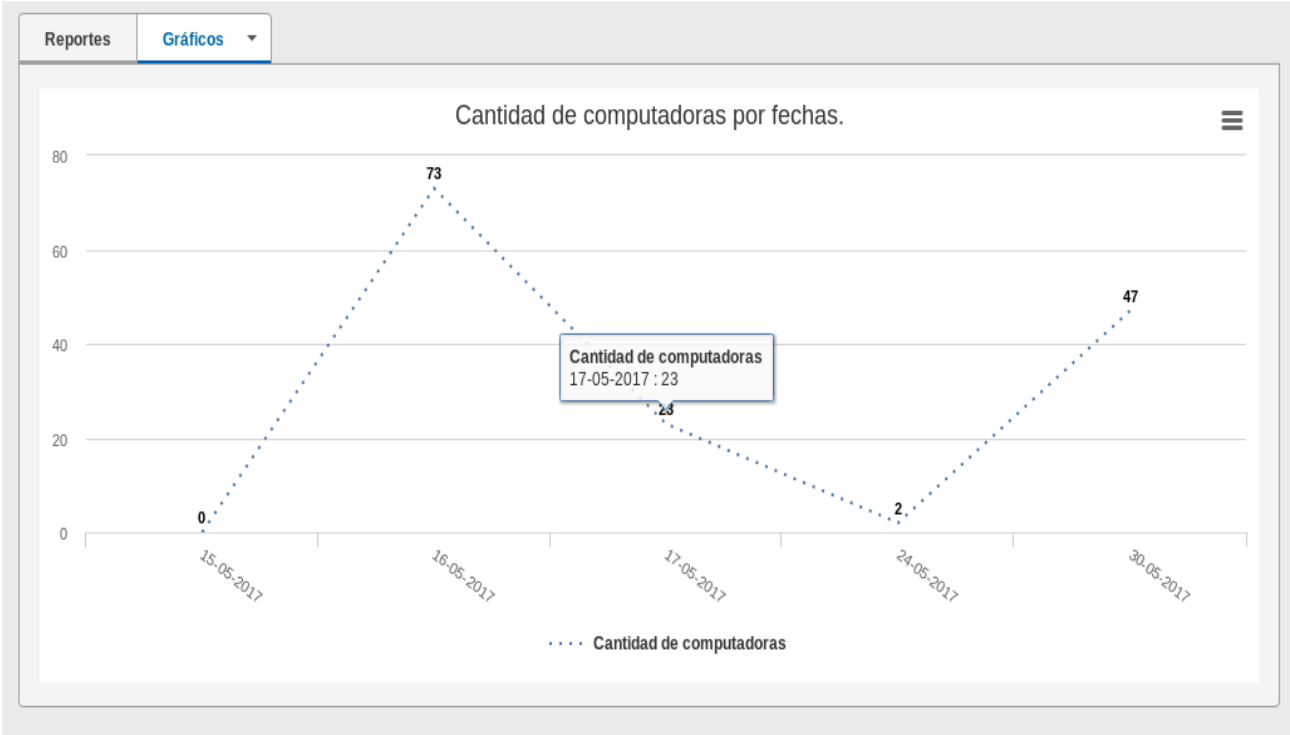


Figura 9. Interfaz de usuario: Mostrar gráfico de computadoras por fechas

(Fuente: elaboración propia)

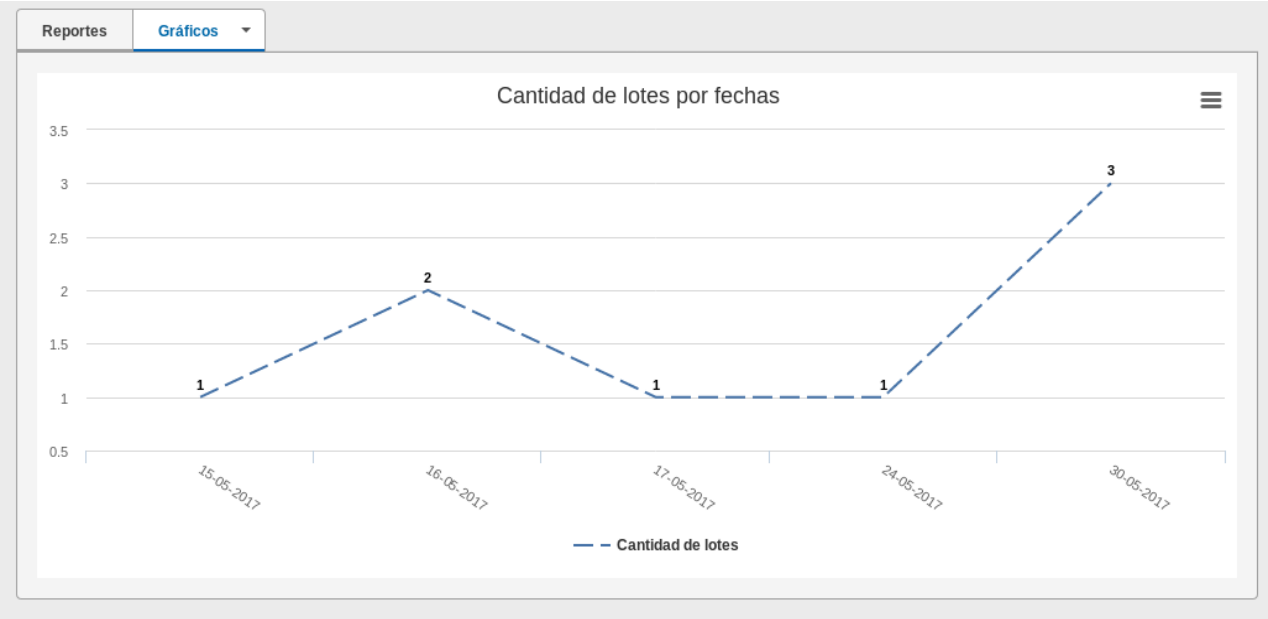


Figura 10. Interfaz de usuario: Mostrar gráfico de lotes por fechas

(Fuente: elaboración propia)

3.4 Estándares de codificación

El estándar de código se basa en la organización y aspecto físico de un software con el objetivo de facilitar la lectura, entendimiento, mantenimiento del código, reutilización a lo largo del proceso de desarrollo y no en la lógica del programa. Un estándar de programación no busca únicamente definir la nomenclatura de las variables, objetos, métodos y funciones, sino que además incluye reglas para el orden y legibilidad del código escrito. Es importante definir este estándar debido a que no siempre es el mismo programador quien realiza el mantenimiento del software. Además, es un aspecto a tener en cuenta para facilitar la reutilización del código. Partiendo de lo antes expuesto se definen los siguientes estándares principales:

Nomenclatura de las clases:

Clases controladoras: Los nombres de las clases controladoras comienzan con la primera letra de la palabra que identifica a la clase con minúscula, y acompañado de la palabra "*Controller*" con su primera letra en mayúscula. En caso de que sea un nombre compuesto se comienza con minúscula la primera palabra, y la segunda palabra tendrá su primera letra en mayúscula, acompañado de la palabra "*Controller*". **Ejemplo:** "*pluginsConfigController".*

Clases entidades: su nombre comienza con minúsculas y están ubicadas dentro del directorio *Entity* del

proyecto. En caso de tener un nombre compuesto cumplen con el mismo estándar de la nomenclatura de las clases Controladoras, con excepción de que no están acompañadas de la palabra *Controller*. Son las clases que crean el esquema de la base de datos según su contenido**. Ejemplo:** "*procesadorConf"*.

Nomenclatura de las funcionalidades o métodos:

El nombre de los métodos se escribe con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará notación ***CamelCasing***, que es un estilo de escritura que se aplica a frases o palabras compuestas, en este caso empleándose el tipo: ***lowerCamelCase***, el cual indica que las palabras compuestas que forman parte del nombre deben ir con mayúsculas exceptuando la primera palabra, **ejemplo**: "*discoConf".*

Además de cumplir con el estándar anterior, los nombres de los métodos irán seguidos de la palabra *"Action"* ***ejemplo:*** *"discoConfAction()".*

A continuación, se explica cómo estará estandarizada la estructura interna de las funcionalidades o métodos, a través del siguiente ejemplo con el método *"runSpexAction()"*:

Estilo de indentado o sangrado:

- La declaración de la clase debe estar en el borde izquierdo del área de trabajo.
- Debe haber una línea en blanco por debajo de la declaración de clases, también de las funcionalidades o métodos.
- La declaración del nombre de las funcionalidades o métodos debe comenzar después de 4 espacios del borde izquierdo.
- Las instrucciones que pertenecen a un mismo bloque, dígase una estructura *if-else, for, foreach, case,* iniciaran en la siguiente línea después de la declaración de la estructura y deben comenzar después de cuatro espacios partiendo de la declaración del bloque.
- Las llaves que abren un bloque de instrucciones deben ir en la misma línea en que se declara el mismo y después de un espacio en blanco.
- Las llaves que cierran un bloque de instrucciones deben ir en la línea siguiente a la culminación del mismo, y deben cumplir el mismo indentado de la declaración de dicho bloque.

Operadores:

- Los operadores deben tener un espacio a la izquierda y un espacio a la derecha.

Variables:

- El nombre de las variables o constantes debe comenzar con minúsculas.
- Si el nombre de las variables está compuesto por dos palabras o más, se empleará el tipo ***lowerCamelCase*** de la notación ***CamelCasing.***
- En las funciones que para su ejecución necesiten valores como parámetros, en caso de que sea más de un parámetro, estos deben ir separados por coma (,) y con un espacio después de la coma.
- Las variables *array* que se pasan como parámetros en muchas funcionalidades en la presente solución, deben escribirse en la siguiente línea del método o función a la que pertenece, y deben comenzar después de cuatro espacios partiendo de la declaración de dicho método.

```
class spexController extends Controller {

    public function runSpexAction($id){

        $em = $this->getDoctrine()->getManager();
        $lote = $em->getRepository('PruebasPCBundle:lotePC')->findOneById($id);
        $pc = $em->getRepository('PruebasPCBundle:PC')->findOneBy(
            array('pc_lote' => $id));
        $computadoras = $em->getRepository('PruebasPCBundle:PC')->findBy(
            array('pc_lote' => $id));
        foreach ($computadoras as $pc){
            $conexionSSH = ssh2_connect($pc->getIpaddress(), 22);
            ssh2_auth_password($conexionSSH, $pc->getUsuario(), $pc->getContrasenna());
            $codigo = "export DISPLAY=:0.0 && nohup spex --run > /dev/null 2>&1 &";
            ssh2_exec($conexionSSH, $codigo);
        }
        return $this->render('PruebasPCBundle:lotePC:tabConfig.html.twig', array(
            'pc_id' => $pc->getId(),
            'lote_id' => $lote->getId()
        ));
    }
}
```

Figura 11. Método runSpexAction() para estándar de codificación.

(Fuente: elaboración propia)

3.5 Pruebas funcionales.

Para probar la correcta funcionalidad de la Aplicación centralizada para SPEX se realizaron pruebas funcionales utilizando el método de prueba **caja negra**, en la cual la funcionalidad se verifica sin hacer énfasis en la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software. Estas pruebas se enfocan solamente en las entradas y salidas del sistema, sin preocuparse de tener conocimiento de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, se basan únicamente en los requerimientos de software y especificaciones funcionales. Al estar basadas en los requerimientos de software y en las entradas y salidas de cada funcionalidad, al definir una prueba de caja negra lo principal es identificar los datos de prueba (entradas) y el resultado esperado del sistema al ingresar esos datos, bien sean los datos de salida o algún comportamiento específico (Pressman, 2010).

Este tipo de pruebas permite encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores de rendimiento.

Dentro de la prueba de caja negra se incluyen varias técnicas de pruebas tales como:

- **Partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.
- **Análisis de valor de frontera:** El análisis de valor de frontera conduce a una selección de casos de prueba que revisan los valores de frontera (Pressman, 2010).

Partición de equivalencia

Se aplicó el método de caja negra a la solución utilizando específicamente la técnica partición de equivalencia con el siguiente caso de prueba y obteniendo como resultado la siguiente tabla:

Tabla 21. Caso de prueba: Adicionar computadora.

Nombre del requisito	Descripción	Escenarios de prueba	Flujo del escenario
Adicionar computadora	Permite adicionar una computadora a un lote.	EP 1.1 Adicionar computadora insertando dirección IP correcta, usuario y contraseña.	1. El usuario hace la petición al sistema para adicionar una nueva computadora. 2. En la clase controladora se activa la función para adicionar una computadora. 3. Se muestra la vent con el formulario para adicionar una computadora. 4. El usuario entra los datos requeridos para adicionar una computadora (dirección IP, usuario y contraseña). 5. El sistema valida que la dirección IP introducida presente un formato correcto. 6. El sistema almacena en la base de datos la nueva computadora creada. 7. El sistema muestra el listado de computadoras del lote.
		EP 1.2 Adicionar computadora insertando dirección IP con formato incorrecto, usuario y contraseña.	1. El usuario hace la petición al sistema para adicionar una nueva computadora. 2. En la clase controladora se activa la función para adicionar una computadora. 3. Se muestra la ventana con el formulario para adicionar una

			computadora. 4. El usuario entra los datos requeridos, insertando una dirección IP con formato incorrecto, usuario y contraseña. 5. El sistema muestra en color rojo la dirección IP indicando que esta incorrecto el formato actual de la misma. 6. El usuario debe corregir la dirección IP. 7. El sistema almacena en la base de datos la nueva computadora creada. 8. El sistema muestra el listado de computadoras del lote.

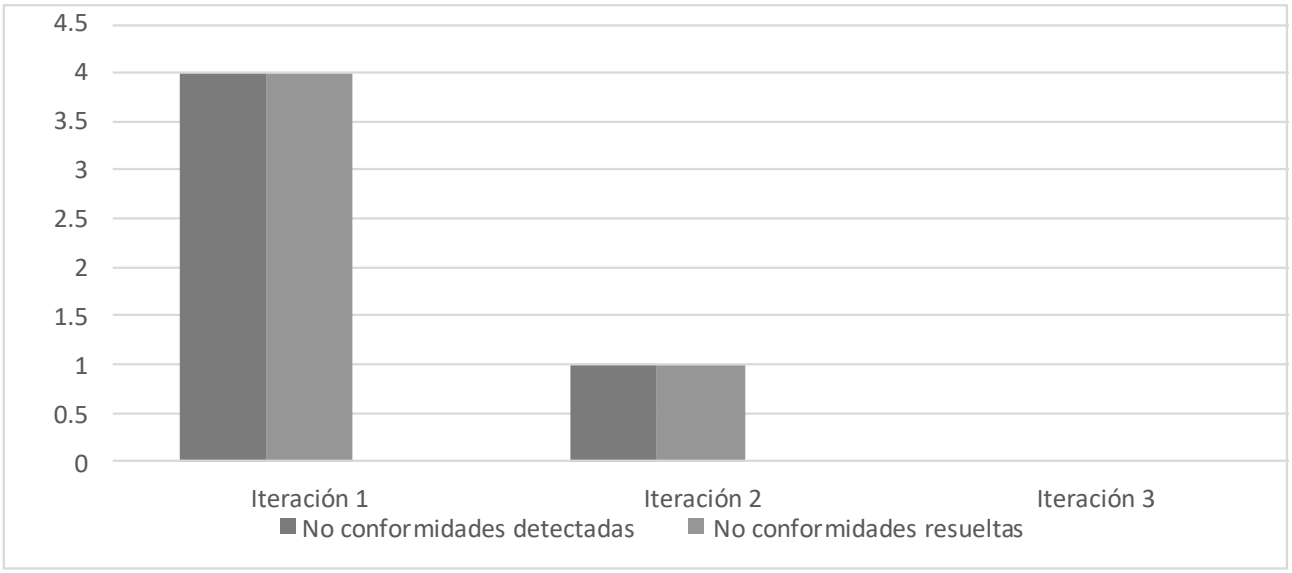
3.5.1 Resultado de las pruebas funcionales

Después de aplicado el método de caja negra para validar las funcionalidades implementadas en la solución se concluye que los resultados obtenido en cuanto a funcionalidad, fueron satisfactorios. Se les dio solución a las no conformidades detectadas durante la aplicación de las pruebas, obteniendo así un mejor funcionamiento de la aplicación.

En la siguiente tabla se muestra el resultado de las pruebas con las no conformidades detectadas por iteración:

Tabla 22. No conformidades de las pruebas de caja negra.

(Fuente: elaboración propia)



En la primera iteración se detectaron cuatro no conformidades, de ellas dos fueron de ortografía y otras dos de diseño. En la segunda iteración se detectaron dos no conformidades de funcionalidad. En la tercera iteración no se detectaron no conformidades.

3.6 Pruebas unitarias.

En la aplicación de las pruebas unitarias se utilizó la técnica de prueba **caja blanca**. Esta técnica se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Permite al ingeniero del software obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican al software, logrando como resultado que disminuya en un gran porciento el

número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad (Pressman, 2010).

Dentro del método se incluyen varias pruebas tales como:

- **La prueba del camino básico:** permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los pasos que se siguen para aplicar esta técnica son (Pressman, 2010):
    1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
    2. Se calcula la complejidad ciclomática del grafo.
    3. Se determina un conjunto básico de caminos independientes.
    4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.
- **La prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
- **La prueba de flujo de datos:** se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- **La prueba de bucles:** es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles (Pressman, 2010).

Dentro de la técnica caja blanca se emplea el método del camino básico para realizar las pruebas. La *Figura 9* muestra el código del método *newpcAction(),* que permite adicionar una nueva computadora a un lote dado su identificador.

```php
/**
 * Displays a form to create a new pc entity.
 *
 * @Route("/newpc", name="lotepc_newpc")
 * @Method("GET")
 * @Template()
 */
(1)  public function newpcAction($id, Request $request)
(1)     $entity = new PC();
(1)      $em = $this->getDoctrine()->getManager();
(1)      $lote = $em->getRepository('PruebasPCBundle:lotePC')->findOneById($id);
(2)      if (!$lote) {
(3)          throw $this->createNotFoundException('find lote entity.');
      }
(4)      $form = $this->createForm(new PCType(), $entity);
(4)      $form->handleRequest($request);
(5)     if ($form->isValid()) {
(6)          $entity->setPcLote($lote);
(6)          $lote->setCantPC();
(6)          $em->persist($entity);
(6)          $em->persist($lote);
(6)          $em->flush();
(6)          return $this->redirect($this->generateUrl('lotepc_showpc', array('pc_lote' => $lote->getId())));
     }
(7)     return array(
(7)          'id' => $lote->getId(),
(7)         'entity' => $entity,
(7)          'form' => $form->createView(),
      );
  }
```

Figura 12. Método para agregar una computadora a un lote

(Fuente: elaboración propia)

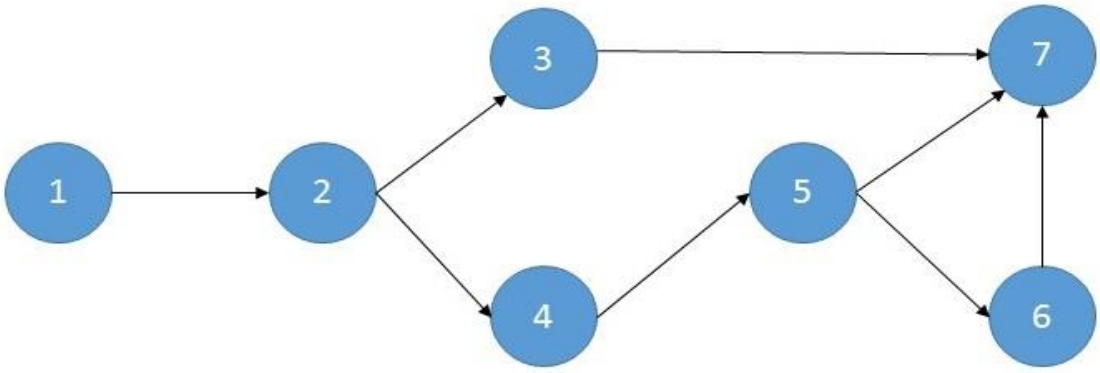


Figura 13. Grafo de flujo asociado al método newpcAction()

(Fuente: elaboración propia)

La complejidad ciclomática es una medición de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica o camino básico, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa. Brinda una cota superior para el número de pruebas que debe realizar, a fin de asegurar que todos los enunciados se ejecutaron al menos una vez. La complejidad ciclomática tiene fundamentos en la teoría de gráficos y proporciona una medición de software extremadamente útil. La complejidad se calcula en una de tres formas:

1. El número de regiones del gráfico de flujo corresponde a la complejidad ciclomática.
2. La complejidad ciclomática **V(G)** para un gráfico de flujo **G** se define como:
   - **V(G) = E - N + 2** donde E es el número de aristas del gráfico de flujo y **N** el número de nodos del gráfico de flujo.
   - **V(G) = P + 1** donde **P** es el número de nodos predicado8 contenidos en el gráfico de flujo **G** (Pressman, 2010).

En el gráfico de flujo anterior, la complejidad ciclomática puede calcularse usando cada uno de los algoritmos indicados:

1. El gráfico de flujo tiene tres regiones.
2. V(G) = 8 aristas - 7 nodos + 2 = 3.

[8] *Un nodo predicado es el que representa una condicional if o case, es decir, que de él salen varios caminos.*

3. V(G) = 2 nodos predicado + 1 = 3.

La complejidad ciclomática del gráfico de flujo asociado al método *newpcAction()* tiene valor 3. V(G) proporciona la cota superior sobre el número de rutas linealmente independientes a través de la estructura de control del programa. Por tanto, las rutas o caminos posibles son las siguientes:

Tabla 23. Rutas básicas del grafo de flujo

(Fuente: elaboración propia)

Numero de ruta	Rutas Básicas
1	1-2-3-7
2	1-2-4-5-7
3	1-2-4-5-6-7

Una vez determinadas las rutas o caminos básicos de flujo, se pasa a ejecutar los casos de pruebas para cada camino resultante. Los datos deben elegirse de modo que las condiciones en los nodos predicado se establezcan de manera adecuada conforme se prueba cada ruta. Cada caso de prueba se ejecuta y compara con los resultados esperados. Una vez completados todos los casos de prueba, el examinador puede estar seguro de que todos los enunciados del programa se ejecutaron al menos una vez.

Tabla 24. Caso de prueba Ruta básica 1

(Fuente: elaboración propia)

Ruta Básica 1: 1-2-3-7	
Descripción	Proceso para adicionar una nueva computadora a un lote.
Condición de ejecución	No está creado el lote.
Entrada	$ipaddress = “10.53.3.45” $usuario = “yosbel” $contraseñaa = “mypassword”
Resultado esperado	El sistema muestra el mensaje “No se encontró el lote”.
Resultado	Satisfactorio.

Tabla 25. Caso de prueba Ruta básica 2

(Fuente: elaboración propia)

Ruta Básica 2: 1-2-4-5-7	
Descripción	Proceso para adicionar una nueva computadora a un lote.
Condición de ejecución	Entradas en blanco en el formulario desde la vista de usuario.
Entrada	$ipaddress = “” $usuario = “rolando” $contrasñaa = “rolandopass”
Resultado esperado	El sistema no debe guardar la nueva computadora en la base de datos con parámetros nulos.
Resultado	Satisfactorio.

Tabla 26. Caso de prueba Ruta básica 3

(Fuente: elaboración propia)

Ruta Básica 3: 1-2-4-5-6-7	
Descripción	Proceso para adicionar una nueva computadora a un lote.
Condición de ejecución	Parámetros correctos.
Entrada	$ipaddress = “10.8.112.7” $usuario = “newuser” $contraseña = “newpass”
Resultado esperado	El sistema debe guardar la nueva computadora en la base de datos.
Resultado	Satisfactorio.

3.6.1 Resultado de las pruebas unitarias

Después de aplicado el método del camino básico como parte de las pruebas internas de caja blanca, se concluye que, de los 3 casos de pruebas realizados a los caminos generados anteriormente, todos tuvieron resultados satisfactorios, por tanto, la aplicación de la prueba fue satisfactoria en su totalidad.

3.7 Pruebas de integración

Se realizaron además pruebas de integración, donde se verificó y se comprobó que la Aplicación Centralizada para la gestión de SPEX se integra total y correctamente con SPEX, de manera que las configuraciones que se realizan en la Aplicación Centralizada llegan a SPEX, y a su vez, este envía satisfactoriamente el reporte generado a la Aplicación Centralizada.

Conclusiones parciales

El presente capítulo fue clave en el desarrollo de la solución propuesta. Se llevaron a cabo acciones que permitieron implementar las funcionalidades necesarias para la Aplicación Centralizada para la gestión de SPEX. Se realizaron pruebas donde se obtuvo resultados satisfactorios, por tanto, es factible la propuesta de solución.

Conclusiones

Durante el desarrollo del presente trabajo investigativo se llevaron a cabo distintas fases que permitieron dar cumplimiento al objetivo general. Teniendo en cuenta cada una de ellas y los resultados obtenidos, permite llegar a las siguientes conclusiones:

- Se elaboró la fundamentación teórica de la investigación que sustenta la propuesta de desarrollo de la Aplicación centralizada para la gestión de SPEX.
- Con el uso de los artefactos definidos por la metodología de software seleccionada, se realizó el análisis y diseño de la Aplicación centralizada para la gestión de SPEX.
- Se desarrolló la Aplicación Centralizada para la gestión de SPEX, la cual permite facilitar el proceso de pruebas y estrés de hardware.
- Se realizaron pruebas para verificar que los requisitos identificados cumplen correctamente sus funciones.

Recomendaciones

Como parte del proceso de investigación y desarrollo de la aplicación, surgieron ideas que son recomendables tener en cuenta para el futuro mejoramiento de la solución propuesta. Entre ellas se señala:

1. Agregar una función para generar un informe general de las pruebas realizadas en todos los lotes de computadoras, donde se pueda conocer estadísticamente la tendencia de ocurrencia de errores en las pruebas.

Referencias bibliográficas

2017. ¿Qué es PHP? y ¿Para qué sirve? [En línea] 2017. [Citado el: 8 de Abril de 2017.] http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=492:ique-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&catid=70&Itemid=193.

Centro de Apoyo Tecnológico a Emprendedores. 2012. Análisis de aplicación: UltraVNC. *Centro de Apoyo Tecnológico a Ciudadanos y Empresas.* [En línea] 2012. [Citado el: 11 de Abril de 2017.] https://www.bilib.es/recursos/catalogo-de-aplicaciones/analisis/doc/analisis-de-aplicacion-ultravnc/docctrl/show/Documento/.

Eguiluz, Javier. 2017. Capítulo 1. Introducción. *Libros del Web.* [En línea] 2017. [Citado el: 8 de Abril de 2017.] http://librosweb.es/libro/css/capitulo_1.html.

GEDEME. 2017. Nuestra Empresa. *GEDEME.* [En línea] 2017. http://www.gedeme.cu/es/nuestra-empresa.

Ibarra, Gilberto. 2013. Pruebas de estres. *Slideshare.* [En línea] 10 de Marzo de 2013. [Citado el: 4 de Abril de 2017.] https://es.slideshare.net/GilbertoIbarra/pruebas-de-estres.

IREO. 2017. ManageEngine Applications Manager. *Applications Manager.* [En línea] 2017. [Citado el: 11 de Abril de 2017.] http://www.ireo.com/fabricantes-y-productos/manageengine/applications-manager/resumen/.

Izaurralde, María Paula. 2013. *Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario.* Universidad Tecnológica Nacional , Córdova : 2013.

Jiménez, Danis Carlos Chaviano. 2016. Sistema de pruebas de hardware para computadoras ensambladas en GEDEME. *Monografías.* [En línea] 2016. [Citado el: 8 de Abril de 2017.] http://www.monografias.com/docs112/sistema-pruebas-hardware-computadoras-ensambladas-gedeme/sistema-pruebas-hardware-computadoras-ensambladas-gedeme.shtml.

Manual de PHP. 2017. ssh2_connect. *PHP: ssh2_connect.* [En línea] 2017. [Citado el: 8 de abril de 2017.] http://php.net/manual/es/function.ssh2-connect.php.

Mora, Roberto Canales. 2005. *¿Qué ofrece Autentia Real Business Solutions S.L?* Soporte a Desarrollo Informático, Avenida de Castilla,1 - Edificio Best Point - Oficina 21B 28830 San Fernando de Henares (Madrid) : 2005.

2015. Pmoinformatica.com. *La oficina de proyectos de informática.* [En línea] 2015. [Citado el: 23 de Abril de 2017.] http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html.

Potencier, François Zaninotto Fabien. 2017. El patrón MVC. *LIBROSWEB.* [En línea] 2017. [Citado el: 6 de Mayo de 2017.] https://librosweb.es/libro/symfony_1_2/capitulo_2/el_patron_mvc.html.

Pressman, Roger S. 2010. *Ingenieria de Software un enfoque práctico.* New York : Rogue S. Pressman, 2010. 978-607-15-0314-5.

Quest Software Inc. 2017. Monitoreo y administración de servidores. *Monitoreo y administración de servidores.* [En línea] Quest Software Inc., 2017. [Citado el: 11 de Abril de 2017.] https://www.quest.com/mx-es/products/kace-systems-management-appliance/server-management.aspx.

Red Hat. 2005. Manual de referencia. *Red Hat Enterprise Linux 4.* [En línea] 2005. [Citado el: 8 de Abril de 2017.] http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana. Cuba : Universidad de las Ciencias Informáticas, 2015.

Software Quality System S.A. 2017. Pruebas de Carga, Rendimiento y Estrés. *Laboratorio Acreditado.*

[En línea] 2017. [Citado el: 5 de Abril de 2017.] http://www.sqs.es/testing-laboratory.php.
Symfony Documentation. 2016. Symfony en pocas palabras . *Libros del Web.* [En línea] 2016. [Citado el: 8 de Abril de 2017.] http://librosweb.es/libro/symfony_1_4/capitulo_1/symfony_en_pocas_palabras.html.
Turmero, Pablo. 2016. Diagrama de interacción, estados, componentes, actividades y despliegue. *Monografias.* [En línea] 2016. [Citado el: 10 de Mayo de 2017.] http://www.monografias.com/trabajos107/diagrama-interaccion-estados-componentes-actividades-y-despliegue-modelado/diagrama-interaccion-estados-componentes-actividades-y-despliegue-modelado2.shtml.
Valdés, Damián Pérez. 2007. ¿Qué es Javascript? *¿Qué es Javascript?* [En línea] 3 de Julio de 2007. [Citado el: 8 de Abril de 2017.] http://www.maestrosdelweb.com/que-es-javascript/.
Velasco, Rubén. 2014. RedesZone. *Remmina, un cliente de escritorio remoto para Linux.* [En línea] 20 de Diciembre de 2014. [Citado el: 05 de Mayo de 2017.] https://www.redeszone.net/2014/12/20/remmina-un-cliente-de-escritorio-remoto-para-linux/.

Anexos

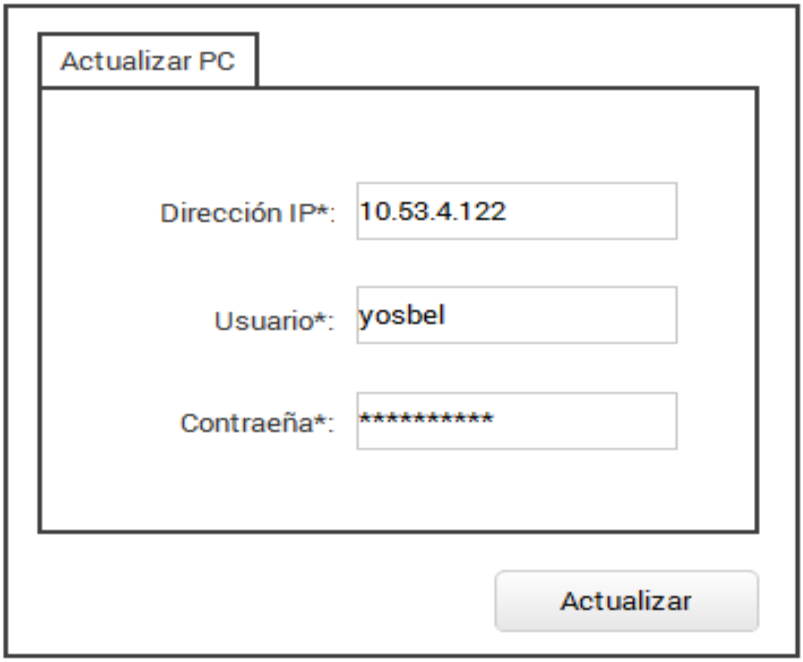
Anexo 1 Técnica tormenta de ideas

En la presente investigación se llevó a cabo esta técnica para obtener los requerimientos funcionales con los que debe cumplir la Aplicación Centralizada para la gestión de SPEX que se propuso como solución. Debido a que no se conocía la solución que se podía implementar se siguieron los siguientes pasos para realizar el proceso:

- Se nombró como moderador al Ingeniero Rolando León Dueñas, tutor del autor de la presente investigación.
- Cada miembro del equipo emitió su idea respecto al tema que se trató y se tuvo en cuenta que no se reiteraran las ideas expuestas. El equipo estuvo conformado por el autor de la presente investigación, los tutores y demás trabajadores del Centro de Software Libre.
- Las ideas expuestas por los miembros del equipo no fueron criticadas, ya que se tuvo en cuenta que cada una de ellas puede ser clave en el objetivo de aplicar la técnica.
- El proceso terminó una vez que se agotó el aporte de ideas por parte de los miembros del equipo reunido.
- Se agruparon todas las ideas propuestas por cada miembro y se seleccionaron de acuerdo a la capacidad de cada una de ellas de integrarse con las demás, para obtener una aplicación centralizada que dé respuesta al objetivo de la presente investigación.

Anexo 2 Historias de usuario

Historia de Usuario	
Nombre: Modificar computadora	**Número:** HU_10
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Medio	**Puntos estimados:** 2
Riesgo en desarrollo: Medio	**Puntos Reales:** 1

Descripción: Permite modificar los parámetros de una computadora.
Observaciones:
Prototipo de interfaz de usuario: 

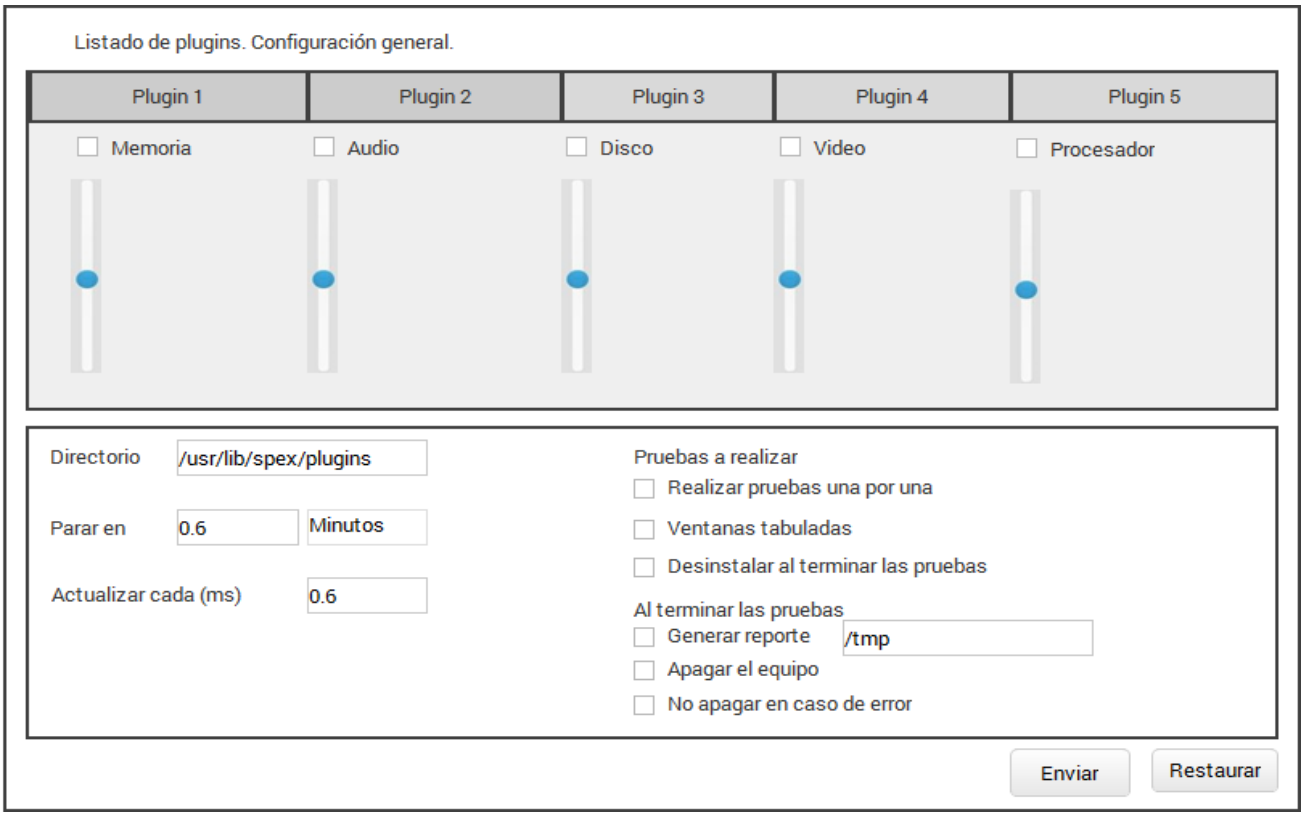
Anexo 2.1 Historia de usuario: Modificar computadora.

Historia de Usuario	
Nombre: Configurar ejecución de SPEX	**Número:** HU_11
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos Reales:** 1
Descripción: Permite definir la configuración de la ejecución de SPEX para ejecutar las pruebas, como el tiempo de duración del proceso de pruebas, el directorio donde estarán los plugins, la forma de presentar las ventanas de SPEX que muestran lo que está ocurriendo, la opción para apagar la	

computadora probada, desinstalar SPEX posterior a su ejecución y también la opción de probar cada componente de hardware al mismo tiempo o uno a uno.

Observaciones: Se podrá seleccionar cuales plugins o componentes de hardware serán objeto de pruebas. Después de establecer la configuración de ejecución será posible general el fichero.

Prototipo de interfaz de usuario:



Anexo 2.2 Historia de usuario: Definir configuración de ejecución.

Historia de Usuario	
Nombre: Eliminar lote	**Número:** HU_12
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Medio	**Puntos estimados:** 2

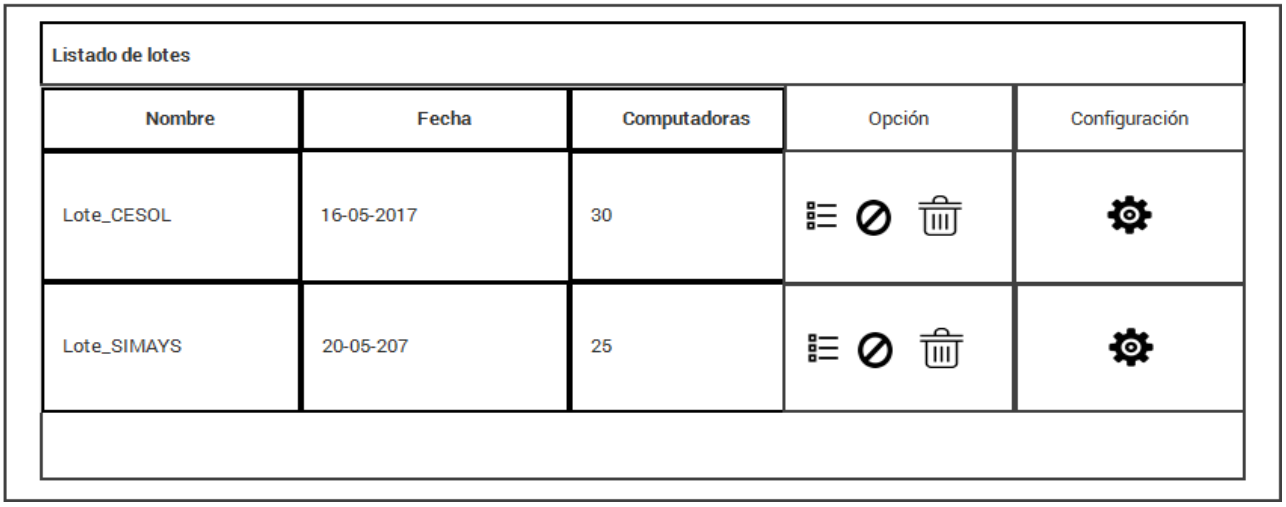
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite eliminar un lote del sistema.	
Observaciones: al eliminar un lote, se eliminan también las computadoras y los reportes de pruebas que pertenecen al mismo.	
Prototipo de interfaz de usuario: 	

Anexo 2.3 Historia de usuario: Eliminar lote

Historia de Usuario	
Nombre: Editar lote	**Número:** HU_13
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Medio	**Puntos estimados:** 2
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: permite editar un lote existente en el sistema	
Observaciones: 	

Anexo 2.4 Historia de usuario: Editar lote.

Historia de Usuario	
Nombre: Listar lote	**Número:** HU_14
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Medio	**Puntos estimados:** 2

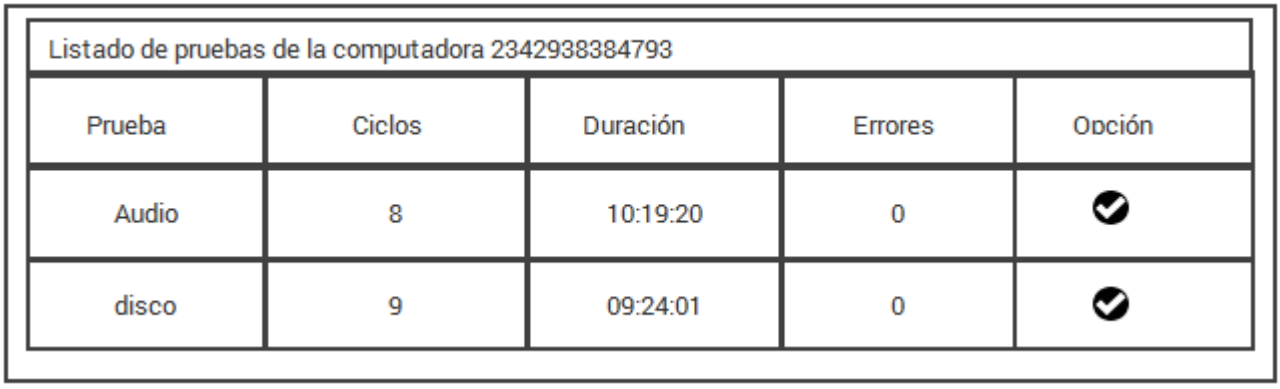
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Muestra en una vista los lotes existentes en el sistema	
Observaciones: Brinda la opción de eliminar o editar un lote, acceder a sus reportes y a su configuración.	
Prototipo de interfaz de usuario: 	

Anexo 2.4 Historia de usuario: Listar lote

Historia de Usuario	
Nombre: Eliminar computadora	**Número:** HU_15
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Medio	**Puntos estimados:** 2
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite eliminar una computadora de un lote en el sistema	
Observaciones: Al eliminar una computadora también se eliminan del sistema los reportes de	

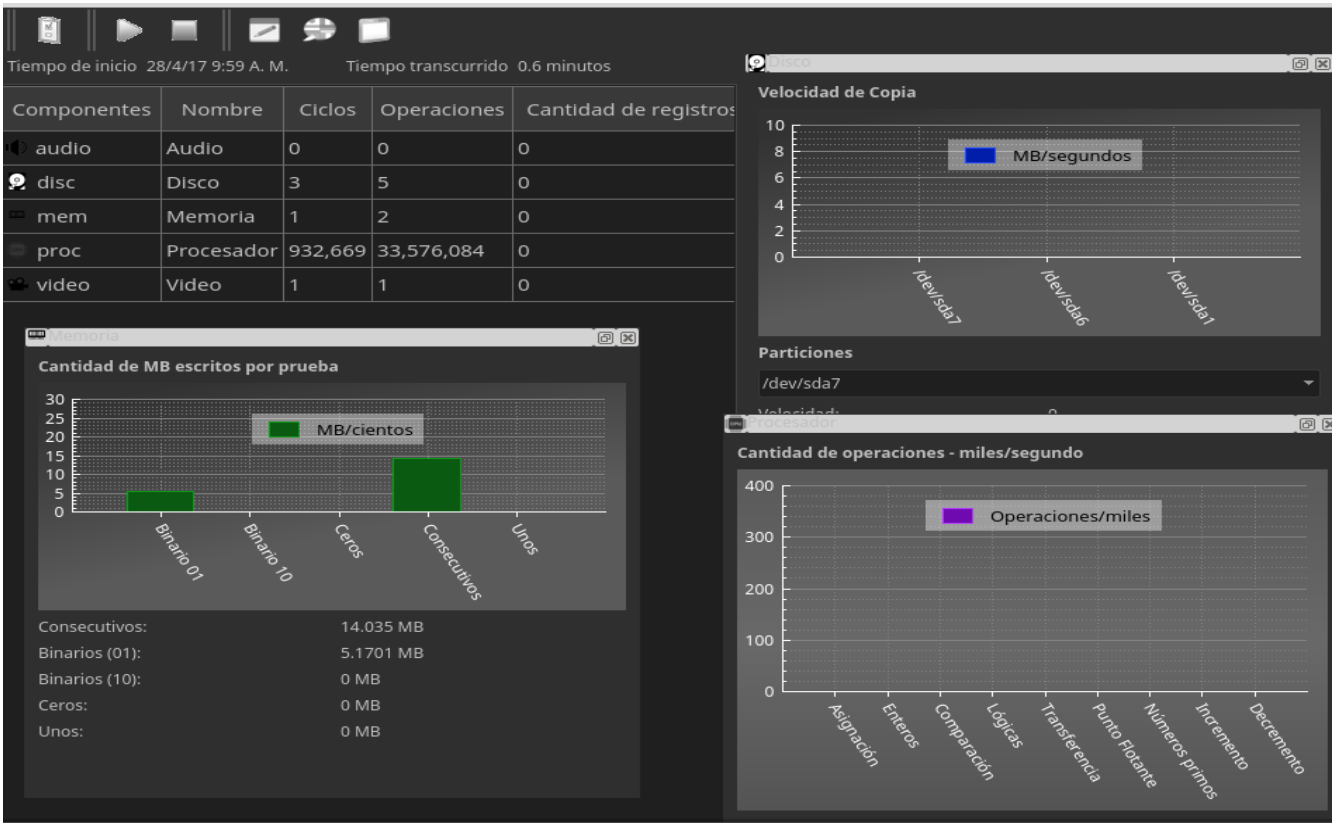
prueba de esa computadora.
Prototipo de interfaz de usuario: 

Anexo 2.5 Historia de usuario: Eliminar computadora

Historia de Usuario	
Nombre: Listar prueba	**Número:** HU_16
Modificaciones: Ninguna	
Programador: Yosbel Palacios Ferrer	**Iteración Asignada:** 2
Prioridad en el negocio: Medio	**Puntos estimados:** 2
Riesgo en desarrollo: Medio	**Puntos Reales:** 1
Descripción: permite mostrar las pruebas de un lote	
Observaciones:	
Prototipo de interfaz de usuario: 	

Anexo 2.6 Historia de usuario: Listar prueba.

Anexo 3 SPEX


Tiempo de inicio 28/4/17 9:59 A. M.
Tiempo transcurrido 0.6 minutos
Componentes | Nombre | Ciclos | Operaciones | Cantidad de registros
audio | Audio | 0 | 0 | 0
disc | Disco | 3 | 5 | 0
mem | Memoria | 1 | 2 | 0
proc | Procesador | 932,669 | 33,576,084 | 0
video | Video | 1 | 1 | 0
Velocidad de Copia
MB/segundos
/dev/sda7
/dev/sda6
/dev/sda1
Particiones
/dev/sda7
Cantidad de MB escritos por prueba
MB/cientos
Binario 01
Binario 10
Ceros
Consecutivos
Unos
Consecutivos: 14.035 MB
Binarios (01): 5.1701 MB
Binarios (10): 0 MB
Ceros: 0 MB
Unos: 0 MB
Cantidad de operaciones - miles/segundo
Operaciones/miles
Asignación
Enteros
Comparación
Lógicas
Transferencia
Punto Flotante
Números primos
Incremento
Decremento

Anexo 4 Modelo de datos