Universidad de las Ciencias Informáticas

Facultad 5

Servidor de integración continua y pruebas para código fuente

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

Autor: Jesús Vilches Pupo

Tutor: Roberto Alejandro Espí Muñoz

La Habana, Junio 2012
"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informática Industrial de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Firma del autor	**Firma del tutor**
Jesús Vilches Pupo	Roberto Alejandro Espí Muñoz

Dedicatoria

Para mis padres, Mery y Pedrito, por hacer de mí el hombre que soy hoy en día.

Para mis abuelos, Silvia y Emilio, como pequeño tributo a toda una vida dedicada a mí.

Agradecimientos

A mis papás, Mery y Pedrito, quienes nunca me han fallado y a quienes les debo lo que soy.

A mi abuela Silvia, a quien nunca voy a poder agradecerle ni retribuirle completamente todo lo que ha hecho por mí.

A mi abuelo Pupo, que en paz descanse, mi modesto regalo en retribución de una vida dedicada a mi educación y formación de hombre de bien.

A mi amigo y hermano Alejandro, por su incondicional apoyo y amistad a través de todos estos años.

A mi amigo Karel, por brindarme su amistad y sabio consejo en todo momento.

A mi tutor Roberto Espí, por confiar en mí para desarrollar su idea.

A la Universidad de las Ciencias Informáticas, por colaborar con el sueño de convertirme en ingeniero.

Resumen

Sherlock es una aplicación web desarrollada en el año 2010 con el objetivo de automatizar el proceso de integración y pruebas de código fuente en la Línea de Integración y Despliegue del Centro de Informática Industrial (CEDIN). La aplicación desde su creación sólo ha sido utilizada en fase piloto debido a que presenta errores de diseño y programación, además de no contar con un número suficiente de funcionalidades.

La presente investigación se planteó implementar una nueva versión de la aplicación con el objetivo de corregir las deficiencias existentes e incorporarle nuevas funcionalidades que permitan el empleo de Sherlock en un entorno real de producción.

Para el desarrollo de la solución, primeramente se efectuó un estudio de las tecnologías candidatas y se realizó la selección de las mismas. Fueron definidos los puntos de mejora y optimización a cubrir, se generaron los artefactos producidos durante el desarrollo de la herramienta y se recopilaron los datos arrojados mediante las pruebas efectuadas a la aplicación.

La nueva distribución fue desarrollada con el empleo del marco de trabajo Symfony2, aprovechando diversas características que le aportan a la herramienta mantenibilidad, escalabilidad, organización, seguridad y otras.

Palabras clave: escalabilidad, integración, marco de trabajo, pruebas, seguridad, Sherlock, Symfony2, web.

TABLA DE CONTENIDOS

Índice de figuras

Índice de tablas

Introducción

Con el paso del tiempo y la constante evolución tecnológica por la que atraviesa la sociedad contemporánea, se ha comprobado la necesidad y la utilidad de la red de redes. Internet ha jugado un papel importante en el aumento del nivel de vida y ha contribuido al desarrollo de diversas esferas de la sociedad.

La forma principal de interactuar con Internet es mediante los sitios web. Para su construcción, existen aspectos significativos a tomar en cuenta, entre los cuales se pueden mencionar el diseño gráfico, la organización estructural del contenido y la seguridad. Es interesante observar cómo en los últimos años ha evolucionado la forma en que estos sistemas son implementados, donde el empleo de *frameworks* de desarrollo es considerado por muchos programadores la manera más segura, ágil y viable de implementar este tipo de aplicaciones.

Los servidores web de integración continua son un ejemplo fehaciente de la diversidad y extensión de los sitios web soportando la tendencia actual de la automatización constante de complejos procesos.

La Línea de Integración y Despliegue (en lo adelante LID), perteneciente al Centro de Informática Industrial (en lo adelante CEDIN) de la Universidad de Ciencias Informáticas (UCI) realiza entre sus procesos la integración y prueba a código fuente. En el año 2010, se realizó una investigación que obtuvo como resultado, la creación de Sherlock, una aplicación web con el objetivo de brindarle apoyo a la LID mediante la automatización de los procesos antes mencionados.

La **situación problémica** que respalda el desarrollo de la presente investigación reside en que actualmente Sherlock no posee una estructura sustentada en patrones arquitectónicos y de diseño ni aplicados de manera correcta. Esto provoca que los desarrolladores y mantenedores de la herramienta cuenten con un código poco ordenado y complejo de actualizar. El sistema, además, no ofrece un conjunto de funcionalidades de prueba y verificación de métricas de calidad necesarias para la Línea de Integración y Despliegue. Otro aspecto relevante es que la aplicación no trata de manera puntual aspectos de seguridad como la prevención de inyecciones SQL[1]. Estas dificultades limitan la capacidad de extender

la herramienta de manera ágil y segura. Es interés del CEDIN mantener y estabilizar una herramienta propia para los procesos de integración y prueba.

Las deficiencias antes expuestas, entre otras que se detallan en el epígrafe **2.1** de la investigación, provocan que la aplicación no alcance la estabilidad requerida para superar la fase piloto en la que se encuentra desde el año 2010.

El análisis de la situación problémica permite formular el siguiente **problema de investigación:** ¿Cómo mejorar la mantenibilidad, la extensibilidad (1), aumentar la disponibilidad e incorporar nuevos módulos y funcionalidades a Sherlock? La presente investigación propone como **objetivo general** desarrollar una nueva versión de la herramienta Sherlock.

El objetivo general planteado será desglosado en **objetivos específicos**, los cuales se enuncian a continuación:

- Implementar la nueva versión de Sherlock, utilizando para ello un *framework* PHP.
- Incorporar nuevos módulos de prueba y verificación de métricas de calidad de software.

En aras de alcanzar los objetivos propuestos, es necesario establecer el área del conocimiento sobre el cual se trabajará definiendo como **objeto de estudio** las tecnologías y herramientas de desarrollo web dinámico, así como las pruebas y métricas de calidad de software aplicadas al código fuente, enmarcando el **campo de acción** al empleo de *frameworks* PHP para el desarrollo de aplicaciones web, así como módulos de prueba y métricas de calidad de software para fomentar el uso de Sherlock dentro de la UCI.

Para dar cumplimiento a los objetivos propuestos, la investigación será organizada a través de las siguientes tareas:

- Estudio de la herramienta Sherlock para comprender el funcionamiento de la suite de pruebas y métricas de calidad de software, que permita valorar puntos de mejora.
- Estudio de los diferentes *frameworks* PHP de desarrollo para crear la base de conocimiento necesaria a la hora de seleccionar el conjunto de tecnologías, herramientas y lenguajes de programación a usar para dar respuesta al problema de la investigación.

- Selección de módulos de pruebas a incluir en Sherlock para aumentar la funcionalidad de la herramienta.
- Generación de los artefactos de ingeniería de software que se correspondan con la metodología seleccionada para garantizar una correcta estructura y documentación de la solución propuesta.
- Implementación de la solución para cumplir con las funcionalidades requeridas.
- Validación de las nuevas funcionalidades incorporadas al software a través de pruebas de caja negra.

Con la elaboración del presente trabajo, a partir de las tareas de la investigación definidas, se pretenden alcanzar los siguientes resultados:

- Versión 0.3 de Sherlock con estructura estándar proporcionada por el empleo de un *framework* PHP que brinde a los desarrolladores rapidez y reducción de los costos de extensión y mantenimiento.
- Nuevos módulos de prueba y validación de métricas de calidad de software para Sherlock que aumenten la funcionalidad de la herramienta.

Para la realización de la investigación se utilizará el **Método de Análisis-Síntesis** para estudiar las tecnologías y herramientas de desarrollo web dinámico así como de los módulos de pruebas que serán integrados a Sherlock. Posteriormente, el análisis efectuado mediante el estudio de bibliografía especializada, brindará la posibilidad de plasmar brevemente las características distinguidas, permitiendo una mayor comprensión de los elementos constituyentes del objeto de estudio.

Para una mejor compresión de la investigación se determinó la siguiente estructura: Resumen, Introducción y tres capítulos que constituyen el cuerpo de la tesis, Conclusiones, Recomendaciones, Referencias bibliográficas, Bibliografía consultada, Anexos y Glosario de Términos.

Capítulo 1 Fundamentación teórica

La finalidad del presente capítulo es presentar los elementos teóricos que sirven de soporte a la investigación del problema planteado, al mismo tiempo contribuyen a su mejor comprensión. Se brinda además, una síntesis descriptiva de la metodología, tecnologías y herramientas que componen el objeto de estudio.

Se aborda el tema de la programación web dinámica: definiciones y características de las tecnologías y herramientas empleadas para su desarrollo. Se definen además los conceptos de prueba, métrica, así como distintos tipos de prueba y métricas de calidad aplicadas al código fuente.

1.1 Programación web dinámica y del lado servidor

Dentro de la programación web, cualquier página, sitio o aplicación puede ser caracterizado como estático o dinámico; la característica distintiva de pertenecer a esta última clasificación radica en que la mayor parte de su contenido se genera a partir de lo que un usuario introduce o selecciona por medio de una petición en la web de su navegador, a través de formularios. La web dinámica, además, puede brindar múltiples funcionalidades: manipulación de información mediante gestores de bases de datos, foros, consultas en línea, correos inteligentes, entre otras.

El orden lógico general al que están sujetos los sistemas dinámicos es el siguiente:

- El navegador efectúa la petición de una página.
- El servidor recibe la petición y determina el intérprete adecuado.
- El intérprete del lenguaje de programación usado del lado del servidor es invocado.
- El intérprete ejecuta los *scripts*, métodos o aplicaciones necesarias y devuelve al servidor el documento o recurso generado.
- El servidor envía el documento resultante en formato HTML.

- El documento es interpretado por el navegador, se ejecutan los *scripts* del lado cliente y se presenta el resultado en pantalla.

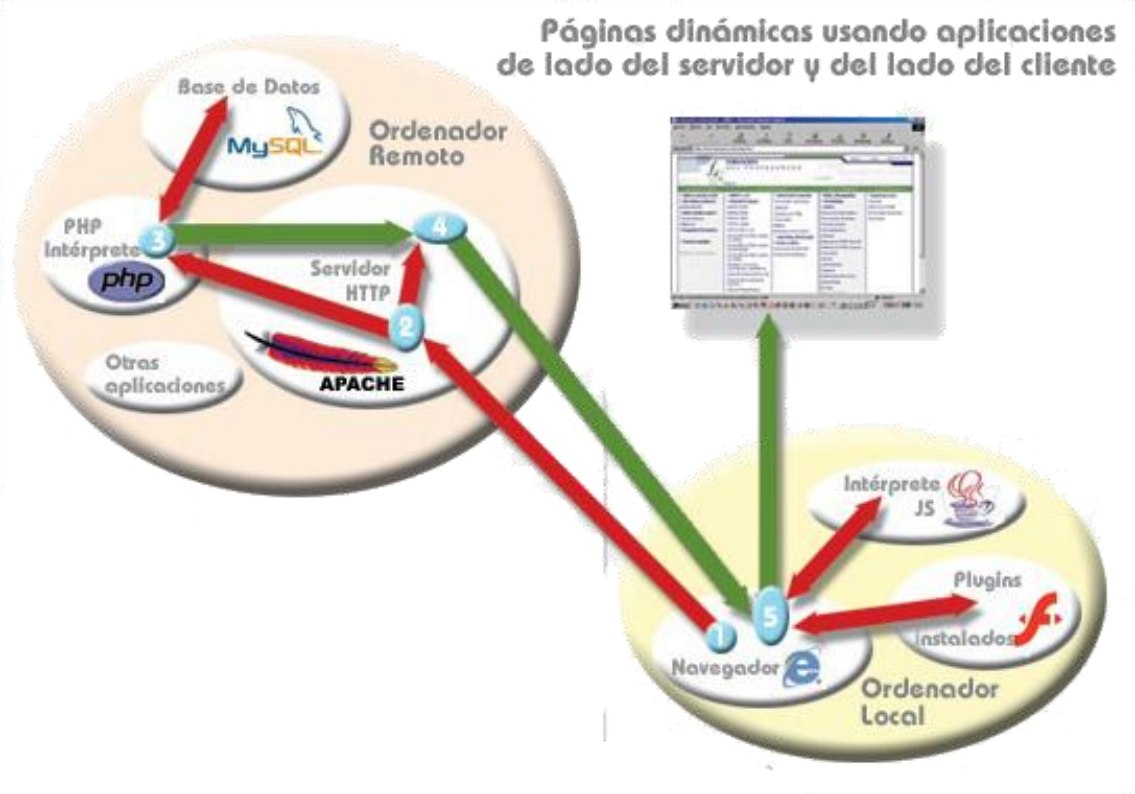

Fig1. Procesamiento de una petición y su respuesta.

En esta secuencia se evidencia cómo la presencia de tecnología del lado servidor, dígase herramientas o lenguajes de programación, es imprescindible para lograr páginas web dinámicas, a la vez que enriquece la interacción final del sistema web con el usuario.

1.2 Tecnologías y lenguajes de programación del lado servidor

El empleo de tecnologías, herramientas y lenguajes de programación del lado servidor es un factor fundamental a la hora de emprender un proyecto de desarrollo web dinámico. Es por medio de estos que la petición del usuario puede ser manejada correctamente para posteriormente presentar una respuesta coherente a lo que el usuario espera. Estas herramientas son el puente entre el *host*[2] del cliente y el *host*

servidor, ofreciendo la comunicación necesaria para brindar los distintos servicios solicitados, los cuales pueden ser desde la visualización de un documento hasta la descarga de un archivo de video.

Existen varias tecnologías y lenguajes de programación del lado servidor usados para el desarrollo de aplicaciones web dinámicas tales como los CGI, ASP, PHP, JSP entre otros, de los cuáles se ofrece una breve descripción a continuación.

1.2.1 Interfaz de Salida Común

Sus siglas **CGI** identifican a la tecnología ***Common Gateway Interface***, nacida en 1993 y primera en ser utilizada como puente para comunicar el servidor con programas externos que implementan funcionalidades requeridas para generar la respuesta a la petición del usuario. Simons Cozens en su libro "Beginning Perl" (2000) enuncia que las aplicaciones que maneja el CGI pueden estar compiladas en diferentes lenguajes de programación, siendo el más popular Perl, aunque también se pueden mencionar C, C++ y Java. Los CGI son sistemas que operan sobre una gran variedad de servidores web y plataformas, sin embargo, no son soluciones fáciles de escribir y requieren de complejas habilidades de programación y diseño. Debido al desconocimiento por parte del servidor HTTP[3] de la manera en que un CGI es implementado, su interacción se vuelve limitada en cuando a manejo de memoria, hilos de ejecución etc. Poseen además, el inconveniente de ser programas muy lentos, dado que necesitan cargar en memoria tantas copias de sí como peticiones HTTP hayan que involucren la presencia de los CGI, provocando lentitud en el servidor web y un posible colapso del sistema si el número de peticiones aumenta considerablemente.

1.2.2 Páginas Activas del Servidor

ASP son las iniciales que definen a las ***Active Server Pages*** según sus siglas en inglés. Es una tecnología desarrollada a mediados de los años 90, convirtiéndose en el primer motor de script del lado servidor de la empresa Microsoft para la generación de sitios o aplicaciones web dinámicas. Los lenguajes usados para el desarrollo de este tipo de páginas suelen ser Visual Basic Script o Javascript, aunque otros pueden ser utilizados. La aparición del marco de trabajo .NET, acarreó un moderno ambiente de programación, con código mucho más reutilizable y optimizado, lo cual propinó que ASP evolucionara de su versión 3.0 a ASP.NET. Es necesario destacar como principal desventaja que ASP, tanto en su forma

clásica como en .NET, es una tecnología propietaria de Microsoft y que su empleo implica generalmente el uso de los productos de Microsoft.

1.2.3 Páginas de Servidor en Java (JSP)

JSP es el acrónimo de ***Java Server Pages*** que es la tecnología de la empresa *Sun Microsystems* para incluir contenido HTML dinámico generado con java en páginas HTML estáticas; algo así como el equivalente Java a ASP y PHP. La especificación JSP 1.2 fue la primera que se liberó en 1999 y en la actualidad está disponible la especificación JSP 2.1. Tomando como referencia el libro "Java Server Pages" (2000), su autor Hans Bergsten define que esta infraestructura permite crear aplicaciones que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es, en esencia, un lenguaje multiplataforma.

1.2.4 Pre-procesador de Hipertexto

PHP inicialmente respondía al vocablo ***Personal Home Pages***, es un lenguaje interpretado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Fue creado originalmente por Rasmus Lerdorf en 1994. El gran parecido que tiene con los lenguajes estructurados como C y Perl, permiten a los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. Este lenguaje puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

A través de los años, PHP, conocido en la actualidad como **PHP *Hypertext Pre-processor***, se ha ido perfeccionando y su actual versión, la 5.3 lanzada el 30 de junio de 2009, ofrece tal número de ventajas que lo hace presente en más de 15 millones de dominios. Algunas de estas ventajas se muestran a continuación, tomadas de la página oficial de PHP y del libro "*PHP Essentials*" de Julie Meloni (2003):

- Es un lenguaje fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta el paradigma orientado a objetos.
- Es un lenguaje multiplataforma.

- Presenta la capacidad de conectarse con la mayoría de los manejadores de base de datos.
- Posee amplia documentación y dispone de una página oficial la cual incluye la descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para los usuarios.

1.3 Mecanismos de ayuda para la programación web dinámica

A raíz del incremento y la gran aceptación que ha ganado la programación web a través de los años, una tendencia ha ido acaparando la atención de los programadores y organizaciones: la necesidad de minimizar el tiempo de desarrollo de proyectos, reutilizar código, garantizar seguridad, mantenibilidad y por ende, eficiencia en las soluciones. Como respuesta, un grupo de conceptos y herramientas bases se han ido incorporando en la programación web en función de servir de apoyo a esta rama de la informática. Algunos de estos se mencionan a continuación.

1.3.1 Biblioteca

Una biblioteca representa una colección de métodos y subrutinas utilizados para desarrollar software y que no necesitan ser modificados. Contienen código y datos, que proporcionan servicios a programas independientes, pasando a formar parte de estos.

1.3.2 *Snippet*

El término *snnipet*, proviene del idioma inglés y es utilizado en programación para referirse a pequeñas partes reusables de código fuente, código binario o texto. Comúnmente son definidas como unidades o métodos funcionales que se pueden integrar fácilmente en módulos más grandes aportando funcionalidad. La palabra también se utiliza para referirse a la práctica de minimizar el código repetido mediante el empleo de un solo método que pueda ser reutilizado.

1.3.3 Complemento

El complemento o *plugin* responde a un grupo de componentes de software que se relacionan con otros programas de mayor envergadura para aportarle una función nueva y generalmente muy específica. El

plugin es ejecutado por la aplicación principal e interactúan generalmente mediante una interfaz de programación de aplicaciones (API). Los complementos permiten que los desarrolladores externos colaboren con el software principal extendiendo sus funciones, reduciendo el tamaño de la aplicación entre otras ventajas.

1.3.4 Interfaz de Programación de Aplicaciones

La Interfaz de Programación de Aplicaciones, comúnmente conocida como API o ***Application Programming Interface***, es el conjunto de funciones, procedimientos o métodos, que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de propósito general, de esta forma, los programadores se benefician de las ventajas de la API mediante el empleo de dichas funcionalidades, evitándose el trabajo de programar todo desde el principio. (2)

1.4 *Framework*

Un *framework* es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software, en base a la cual un proyecto puede ser organizado y desarrollado de forma sencilla. Típicamente, puede incluir funcionalidades como soporte de programas, bibliotecas y un lenguaje interpretado, para así ayudar a desarrollar y unir los componentes de un proyecto. (3)

A pesar de la agilidad en la programación, la reutilización de código, entre otras ventajas que ofrece el empleo de bibliotecas, *snippets*, complementos, entre otras herramientas, un proyecto web generalmente va a estar sujeto a restricciones de seguridad, organización, estructura, lógica, empleo de patrones de arquitectura, por mencionar algunos aspectos. Es aquí donde el empleo de un *framework* web reduce al mínimo estas deficiencias.

En dependencia del tipo de aplicación que se esté desarrollando, ya sea de escritorio, web u otra, el programador tendrá la posibilidad de apoyarse en un *framework* si así lo desea, por mencionar algunos ejemplos se pueden destacar dentro de la aplicaciones de escritorio los *frameworks* .NET, GTK y GNOME. En el ámbito de desarrollo de aplicaciones para teléfonos celulares se pueden encontrar plataformas de trabajo como PhoneGap, Spine y jQuery Mobile. Para desarrollar aplicaciones web existen

frameworks como Django, Spring, Jo, jQuery, CodeIgniter y muchos más. La presente investigación centra sus estudios sobre los *frameworks* de desarrollo web específicamente los referentes a PHP.

1.4.1 *Framework* de aplicaciones web

El concepto de *framework* de aplicaciones web, según Krzysztof Cwalina y Brad Abrams (2008), define un programa compuesto por clases, descriptores, bibliotecas, archivos de configuración así como otras funcionalidades. Estos marcos de trabajo están diseñados para dar soporte al desarrollo de sitios web dinámicos, aplicaciones y servicios web, el cual cuenta con una estructura reutilizable que facilita y agiliza el desarrollo de estos sistemas.

Actualmente, existe un gran número de *frameworks* de desarrollo de sitios y aplicaciones web, entre los cuales se destacan los que utilizan lenguajes de programación como JavaScript, Python, CSS, Ruby, HTML y PHP.

1.4.2 *Framework* PHP

Como fue enunciado anteriormente, la investigación será enmarcada en los *frameworks* PHP. Elegir un marco de trabajo PHP, en concordancia con lo descrito por Javier Eguiluz (2011) en su libro “Desarrollo ágil de aplicaciones con Symfony2”:

- Permite el uso del patrón Modelo Vista Controlador, ya que viene integrado al propio *framework*.
- Separa la lógica de negocio y la presentación de la información, facilitando la distribución de tareas.
- Abstrae al programador del tipo de base de datos empleado.
- Permite también la proyección del modelo de datos a clases del esquema relacional.
- Optimiza las consultas para el Sistema Gestor de Bases de Datos utilizado.
- Soporta la generación automática de formularios a partir del modelo y las validaciones y tipos de datos predefinidos como fechas, direcciones de correo electrónico y URLs[4].

- Poseen seguridad integrada ante contra amenazas como inyecciones SQL, CSRF[5] (*Cross-Site Request Forgery*) entre otras.

Una gran variedad de herramientas de este tipo coexisten actualmente, entre los que destacan: Zend, CakePHP, CodeIgniter y Symfony.

1.4.2.1 Zend *Framework*

Zend es un *framework* desarrollado por la empresa del mismo nombre que respalda comercialmente a PHP. Es un *framework* de código abierto diseñado para desarrollar aplicaciones y servicios web con PHP 5, el cual dispone de una documentación confiable. Zend *Framework* es una implementación que usa código orientado a objetos. La estructura de cada componente está construida con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.

Zend *Framework* ofrece un gran rendimiento y una robusta implementación Modelo Vista Controlador (en lo adelante MVC), una abstracción de base de datos que balancea el mapeo objeto relacional (en lo adelante ORM[6]) con eficiencia y simplicidad, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado de datos. Cuenta con módulos para manejar archivos PDF, canales RSS, entre otros. Existen también módulos que implementan bibliotecas de cliente para acceder de forma sencilla a los servicios web más populares. Empresas como Google, Microsoft y StrikeIron se han asociado con Zend para proporcionar interfaces de servicios web y otras tecnologías que desean poner a disposición de los desarrolladores de Zend.

Entre sus desventajas se encuentran la de poseer pocos *plugins* disponibles a la comunidad. A pesar de ser el marco de trabajo de la empresa que está detrás de PHP, la comunidad es algo escasa, por tanto su documentación también.

1.4.2.2 CakePHP

CakePHP es un *framework* de código abierto para el desarrollo rápido de aplicaciones web. Fue creado en el 2005 usando muchos de los conceptos de otro popular marco de trabajo, Ruby OnRails. CakePHP

utiliza patrones como el MVC y ORM, es orientado a objetos y se distribuye bajo la licencia *Massachusetts Institute of Technology* (MIT).

Este *framework* posee componentes para el tratamiento sencillo de seguridad, sesiones, correos electrónicos, cookies y muchas otras características. Es extremadamente simple, necesitando de una mínima configuración.

Entre algunos de los inconvenientes que se pueden mencionar acerca de CakePHP se encuentra la escasa documentación que ofrece en relación a otros marcos de trabajo, presenta problemas al tratar de integrarlo con otras aplicaciones, por considerarse intrusivo en la metodología de desarrollo. Con respecto al manejo de CakePHP sobre los datos relacionales cabe destacar la ausencia de alguna capa de abstracción de bases de datos, lo que hace que el empleo del ORM sea complejo.

1.4.2.3 CodeIgniter

Se trata de un marco de trabajo desarrollado por Rick Ellis, director ejecutivo de Ellislab Inc., para la creación rápida de aplicaciones web bajo PHP. Es un producto de código libre registrado bajo la licencia Apache/BSD.

Este *framework* dispone de clases muy completas para el trabajo con bases de datos soportadas en múltiples plataformas, además de *Active Record* para el ORM. CodeIgniter implementa mecanismos para temas de seguridad como el filtrado contra XSS[7] (*Cross-Site Scripting*), así como también el trabajo con formularios y la validación de datos. Dispone además de varias bibliotecas útiles para la administración de sesiones, clases para el envío de correo electrónico, subida y encriptación de datos, entre otras características.

CodeIgniter presenta, como puede suceder en otras herramientas, inconvenientes o desventajas, una de ellas es no poseer un número suficiente de bibliotecas. Puede mencionarse también el hecho de no incorporar ningún ORM, aunque el problema es resuelto mediante soluciones de terceros como Active Record. CodeIgniter además no dispone de una interfaz de línea de comandos o algún otro mecanismo para automatizar operaciones como la generación dinámica de módulos, bases de datos y otras.

1.4.2.4 Symfony2

Framework PHP creado en el año 2011 por Fabien Potencier, director ejecutivo de la empresa SensioLabs. Symfony2 es ciento por ciento código libre y está basado en buenas prácticas, cuenta con una comunidad muy importante y dispone de excelente documentación. La versión 1.0 del *framework* fue liberada en enero de 2007 y requería PHP 5.0. La última versión estable es la 2.0.6, introducida al mercado en noviembre de 2011 con soporte para PHP 5.3.

Symfony2 es un marco de trabajo relativamente fácil de aprender, que aplica principios como DRY[8] (*Don't Repeat Yourself*), KISS[9] (*Keep It Small and Simple*) entre otros. Lo mencionado anteriormente convierte a Symfony2 en un marco flexible, sencillo, pero a la vez completo. Cuenta con una consola de línea de comandos para realizar tareas como limpiar o calentar la caché, generar entidades de la base de datos, ejecutar consultas mediante el lenguaje de consultas doctrine y otras. Fabien Potencier (2011), creador de este *framework*, en la página oficial de Symfony2 destaca otras características que son mencionadas a continuación:

- Todo se maneja a través de *bundles*, incluido el propio framework. Un *bundle* es un paquete con una estructura definida e implementa una funcionalidad. Permite utilizar *bundles* de terceros y compartir los *bundles* propios entre distintos proyectos. (4)
- La capa de presentación utiliza plantillas que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework o de programación gracias al motor de plantillas Twig.
- Las funciones incluidas permiten minimizar el código utilizado en la capa de presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
- Los formularios incluyen validación automatizada y relleno dinámico de datos, asegurando la obtención de información correcta y mejorando la experiencia del usuario.
- La gestión de la memoria caché reduce el ancho de banda utilizado y la carga del servidor.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.

- Algunos de los *bundles* más importantes que incluye son Monolog, para la gestión de logs, SwiftMailer para el envío de correo electrónico, Doctrine, para el ORM entre otros.

1.5 Pruebas de software.

En el libro “Ingeniería de Software, un enfoque práctico”, Pressman (2002) hace alusión a algunos atributos y características que definen el término de prueba de software:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

- Una prueba tiene éxito si descubre un error no detectado anteriormente. (5)

1.5.1 Métodos de prueba

Los métodos de prueba de software tienen el objetivo de diseñar test que descubran diferentes tipos de errores con el menor tiempo y esfuerzo. Los dos métodos de prueba más utilizados son el método de prueba de caja blanca (en lo adelante CB), transparente o de cristal y el método de prueba de caja negra (en lo adelante CN). La diferencia entre ambos radica en el hecho de que con la CB es necesario conocer el código y estar relacionado con él para saber exactamente cuál es la lógica interna de lo que se va a probar, sin embargo para la CN solo basta conocer las posibles entradas y salidas del programa.

Prueba de caja blanca: “Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba”. (5) Las mismas, están orientadas a comprobar que las operaciones internas del programa se ajusten a las especificaciones y garantiza que los componentes internos se prueben adecuadamente.

Prueba de caja negra: también denominada prueba de comportamiento, se centra principalmente en los requisitos funcionales del software. Las pruebas de caja negra, permiten obtener un conjunto de condiciones de entrada que ejerciten los requisitos funcionales de un programa. (5) Estas pruebas

permiten encontrar funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a las Bases de Datos externas, entre otros.

1.6 Métricas de software

Diversas organizaciones de software implementan mediciones como parte de sus actividades cotidianas, pues estas brindan la información objetiva necesaria para la toma de decisiones y que tendrá un impacto efectivo en el negocio y desempeño en la ingeniería. Para asegurar que un proceso o sus productos resultantes son de calidad o puedan ser comparados, es necesario asignar valores, descriptores, indicadores o algún otro mecanismo mediante el cual se pueda llevar a cabo dicha comparación. (6) Por tanto, una métrica se puede definir como: “medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado”. (5)

Existen innumerables métricas con propósitos diferentes que reflejan o describen la conducta del software, estas pueden medir entre otros aspectos la competencia, calidad, desempeño y la complejidad del software contribuyendo a establecer de una manera sistemática y objetiva una visión tanto interna como externa del trabajo, mejorando así la calidad del producto.

1.7 Tendencias tecnológicas seleccionadas

Debido a que el objetivo del presente trabajo es mejorar la herramienta Sherlock y no proponer una solución nueva para resolver la automatización de los procesos de integración y pruebas en el Departamento de Integración y Despliegue, el empleo de las siguientes tendencias tecnológicas para el desarrollo de la nueva versión está parcialmente condicionado, ya que el uso de las mismas fue definido por la LID y empleado en la versión actual de Sherlock: **JavaScript**, **CSS**, **HTML**, **MySQL Server**, **HTTP Apache y PHP**.

No obstante, se incorporan nuevas tecnologías y herramientas que permitirán que el nuevo producto supere los inconvenientes de su predecesor, derivando en un sistema de calidad superior.

1.7.1 Lenguajes de programación

1.7.1.1 Javascript

JavaScript es un lenguaje de programación interpretado, desarrollado originalmente por Brendan Eich en 1995. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. (7)

1.7.1.2 Hojas de estilo en cascada

El nombre hojas de estilo en cascada viene del término en inglés *Cascading Style Sheets*, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "<*style*>". (8)

1.7.1.3 Lenguaje de marcado de hipertexto HTML5

Hypertext Markup Language, mejor conocido por sus siglas en inglés como HTML, es el lenguaje empleado para describir la estructura de las páginas web. Según el *World Wide Web Consortium,* HTML brinda a los autores las vías para publicar documentos en línea que contengan cabeceras, texto, tablas, listas, fotos, entre otras características. Permite además devolver información *online* mediante hipervínculos, incluir hojas de cálculo, videos, archivos de música directamente en el documento entre muchas otras funcionalidades. (8)

1.7.1.4 Anotaciones

Las anotaciones son un mecanismo ampliamente utilizado en lenguajes de programación como Java. Técnicamente las anotaciones no son más que comentarios incluidos en el propio código fuente de la aplicación. A diferencia de los comentarios normales, las anotaciones no sólo no se ignoran, sino que se tienen en cuenta y pueden influir en la ejecución del código. Aunque la versión 5.3 de PHP todavía no

soporta anotaciones, las aplicaciones Symfony2 hacen uso de ellas para definir el tipo de dato de los registros de una base de datos, restricciones y validaciones de datos entre otras funcionalidades. Estas características definen la inclusión de las anotaciones dentro de los lenguajes a usar para dar respuesta a la solución de la problemática de esta investigación.

1.7.1.5 YAML no es otro lenguaje de marcado

Acrónimo recursivo que significa "YAML *Ain't Another Markup Language*. Es un formato de serialización diseñado en el año 2001 que permite presentar los datos como combinaciones de listas, relaciones y datos escalares. La sintaxis es relativamente sencilla con lo intención de que sea legible y sencilla de relacionar con los tipos de datos más comunes en la mayoría de los lenguajes de alto nivel. Symfony2 emplea YAML como una de las vías para escribir los archivos de configuración de seguridad, de rutas entre otros.

1.7.1.6 Lenguaje de Consultas Doctrine

Doctrine Query Language (DQL), es la opción que brinda Doctrine de escribir las consultas a base de datos en un dialecto SQL propio orientado a objetos el cual está inspirado en Hibernate HQL. DQL además abstrae considerablemente la asignación entre las filas de la base de datos y los objetos, permitiendo a los desarrolladores escribir poderosas consultas de una manera sencilla y flexible. Este lenguaje es empleado para la solución de la investigación por ser el lenguaje por defecto que usa la biblioteca Doctrine2, la cual está integrada de forma predeterminada a Symfony2.

1.7.1.7 Lenguaje de etiquetado extensible

XML (*Extensible Markup Language*), es un lenguaje utilizado para estructurar la información en un documento o en general en cualquier fichero que contenga texto. Propuesto en 1996 por el *Worl Wide Web Consortium*, la primera especificación apareció en 1998.

La forma en que XML representa la información es a través de los símbolos mayor y menor que, los cuales se usan para delimitar las marcas que dan la estructura al documento. Cada marca tiene un nombre; ejemplo<*figura*>, que puede tener uno o más atributos:<*figura fichero="foto1.jpg" tipo="jpeg"*>. Los atributos toman valores que tienen que estar entre comillas o entre apóstrofes. Ninguna marca se

puede dejar abierta; o sea, por cada marca<*figura*>, deberá existir una marca<*/figura*>correspondiente que indica dónde termina el contenido de la marca. (9)

La decisión de utilizar este lenguaje radica en que una de las funcionalidades que se pretende agregar con la nueva versión de Sherlock es graficar reportes. Para esta tarea es necesario leer los archivos de reportes generados. Existen módulos que permiten crear sus archivos con formato XML y mediante la clase SimpleXML que brinda PHP, la manipulación de estos ficheros para extraer la información a graficar puede ser realizada de forma rápida y sencilla.

1.7.1.8 Python

Python es un lenguaje de programación de alto nivel, creado en 1991 por Guido van Rossum cuya filosofía hace hincapié en una sintaxis clara que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Este lenguaje es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License. Este lenguaje es empleado para controlar el funcionamiento de Buildbot, este software, es el encargado de la ejecución del proceso de integración continua dentro de Sherlock.

1.7.2 Herramientas

1.7.2.1 Twig

Twig es un motor de plantillas para PHP, moderno, flexible, rápido y seguro, utilizado, entre otras herramientas por Symfony2, la cual ayuda al programador en la generación de vistas concisas, amigables y legibles. Una plantilla Twig permite representar diversos formatos basados en texto (HTML, XML, CSV, LaTeX).

1.7.2.2 Geany

Geany es un pequeño y ligero Ambiente Integrado de Desarrollo (IDE) creado por Enrico Tröger en el año 2005. Fue diseñado con el objetivo de proveer a los desarrolladores un pequeño y rápido IDE, con solo unas pocas dependencias de otros paquetes. Geany, ofrece múltiples funcionalidades: soporta varios

lenguajes de programación como C, Java, PHP, Python entre otros, es multiplataforma y fue creado bajo los términos de la Licencia Pública General de GNU. Debido a los pocos recursos de sistema que consume, la utilización de este IDE de desarrollo se basa fundamentalmente en las características limitadas del hardware donde va a ser desarrollada la nueva versión de Sherlock, además del soporte que brinda Geany para gran parte de los lenguajes que serán empleados en la implementación de la solución.

1.7.2.3 JQuery

JQuery es una biblioteca de JavaScript, creada por John Resig en el año 2006, la cual simplifica la manera de interactuar con los documentos HTML. Ofrece, entre otras funcionalidades, interactividad y modificaciones del árbol DOM[10], eventos, efectos y animaciones personalizadas de forma simple, que de otra manera requerirían mucho más código, logrando grandes resultados en menos tiempo y con menores costos de desarrollo.

1.7.2.4 Symfony2

Es un robusto *framework* PHP diseñado para el desarrollo de aplicaciones web de pequeña, mediana y gran envergadura. Implementa el patrón MVC, además de estar basado en buenas prácticas. Cumple con principios de programación como DRY e implementa patrones de diseño. Emplea la biblioteca Doctrine2 para la proyección de un modelo de objetos a otro relacional además de integrar diversas bibliotecas para validación de datos, tratamiento de seguridad, pruebas unitarias entre otras.

La selección de Symfony2 como marco de trabajo para desarrollar la solución de la problemática se basa, además de las ventajas analizadas anteriormente, en los conocimientos previos de la tecnología por parte del autor de la presente investigación, tornando rápida la curva de aprendizaje, con excepción de algunos temas puntuales como el tratamiento de seguridad, la generación automática de formularios entre otros. También es importante destacar la diversa documentación con la que cuenta el framework y la extensa comunidad existente en la UCI, con un foro de debate específico para tratar temas de esta tecnología.

1.7.2.5 PhpMyAdmin

Esta aplicación es una potente herramienta libre y de código abierto creada en 1998 por el proyecto de desarrolladores de phpMyAdmin, la cual maneja la administración de MySQL a través de un navegador

web. Permite crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en varios idiomas incluyendo el inglés y el español. PhpMyAdmin se encuentra disponible bajo la licencia GPL. (10)

1.7.2.6 MySQL Server

MySQL Server es un Sistema Gestor de Base de Datos (SGBD) ampliamente utilizado. Es una aplicación de código abierto, desarrollada y proporcionada por MySQL AB bajo los términos de la licencia GNU GPL. El servidor fue implementado originalmente para manejar grandes bases de datos de manera rápida y ha sido usado exitosamente en ambientes de producción por varios años.

Una de las principales ventajas de MySQL es ser multiplataforma, posee también una estabilidad comprobada, gran rapidez a la hora de ejecutar las consultas, conectividad segura y una excelente integración con PHP. Podemos destacar además que soporta hasta 32 índices por tabla, una gran cantidad de tipos de datos para las columnas. Este SGBD aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo y posee gran portabilidad entre sistemas.

1.7.2.7 Servidor web Apache

HTTP Apache (Apache) es un servidor web de código abierto para plataformas Unix (BSD, GNU/Linux), Microsoft Windows, Macintosh, entre otras, que implementa el protocolo HTTP/1.1 y el concepto de sitio virtual. Apache se desarrolla dentro del proyecto HTTP Server de la Apache Software Foundation.

Este servidor es utilizado principalmente para enviar páginas web estáticas y dinámicas en Internet. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web. También es usado cuando el contenido de las páginas necesita ser puesto a disposición en una forma segura y confiable.

1.7.2.8 Doctrine2

Doctrine2 básicamente es una biblioteca que emplea la técnica ORM para mapear los objetos usados en el modelo de la aplicación a una base de datos relacional.

Las bases de datos relacionales generalmente solo permiten guardar tipos de datos primitivos (dígase enteros, cadenas de texto y otros) por lo que no se puede guardar de forma directa los objetos de la aplicación en las tablas, sino que estos se deben convertir antes en registros, que por lo general afectan a varias tablas. En el momento de volver a recuperar los datos, es necesario efectuar el proceso inverso, se deben convertirlos registros en objetos. Doctrine se encarga de forma automática de este proceso, simulando así tener una base de datos orientada a objetos. (11)

1.8 Metodologías de Desarrollo

Una vez analizadas las principales tendencias tecnológicas, resulta necesario establecer una guía de orientación para el ciclo de vida del proyecto, es decir, una metodología. Una metodología de desarrollo de software es un conjunto de pasos, procedimientos, técnicas y herramientas que ayudan a los desarrolladores a realizar un nuevo software. Es un marco de trabajo empleado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. La finalidad es garantizar la eficacia logrando con ello el cumplimiento de los requisitos iniciales y minimizar las pérdidas de tiempo en el proceso de generación de un software. Alrededor de la década de los 90, comenzaron a surgir metodologías con la finalidad de utilizarlas para la documentación del ciclo de vida de un software, entre ellas: Proceso Unificado de Desarrollo o *Rational Unified Process* RUP (1999), Scrum (fin de los 90), Programación Extrema o *Extreme Programming* XP (1999), Método Dinámico de Desarrollo de Sistemas o *Dynamic Systems Development Method* (1995), *Crystal Clear* (inicios de 1990) y otras.

Las metodologías pueden ser clasificadas en tradicionales o ágiles. Las primeras están guiadas por una fuerte planificación durante todo el proceso de desarrollo, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema pero no ofrecen una buena solución para proyectos donde los requisitos no se conocen con exactitud, porque no están pensadas para trabajar con incertidumbre. Por su parte, las metodologías ágiles buscan un punto medio entre ninguno y demasiados procesos, proporcionando los suficientes. Minimizan el riesgo, desarrollando el software en pequeñas iteraciones. Promueven la comunicación directa entre los desarrolladores y el cliente, y no a través de documentos, por lo que produce muy poca documentación como resultado positivo.

1.8.1 Programación Extrema

XP, según la definición que ofrece su autor Kent Beck en el libro, *Extreme Programming Explained* (1999), es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, orientada al aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, en la comunicación fluida entre los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Esta metodología es adecuada para proyectos con requisitos imprecisos y variables, donde existe un alto riesgo técnico. (12)

Al igual que otras metodologías ágiles de desarrollo, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

1.8.1.1 Las Historias de Usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas, en formato duro o digital en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. (13)

Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas, siendo descompuestas en tareas de programación y asignadas a los programadores para ser desarrolladas durante una iteración.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Insertar Proyecto
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.5
Nivel de Complejidad: Media	**Puntos Reales:** 0.5
Descripción: El sistema debe permitir al usuario insertar un proyecto. Los datos asociados son: Nombre, SCM, Url, Rama, Directorio del código fuente, Lenguaje, Herramienta de construcción, Planificadores.	
Observaciones:	

Fig2. Ejemplo de historia de usuario

1.8.1.2 Las Tarjetas CRC (Clases Responsabilidades Colaboraciones)

Cada tarjeta representa un objeto. En la cabecera se pone el nombre de la clase que representará dicha tarjeta. En el cuerpo, las responsabilidades se ponen en la parte izquierda y las clases que colaboran con dicha responsabilidad se ubican en la parte derecha. En la práctica conviene tener pequeñas tarjetas que se llenarán y que son mostradas al cliente para que las valide. (13)

Clase	
Responsabilidades	Colaboradores

Fig3. Modelo de tarjeta CRC

1.8.1.3 Proceso XP

El ciclo de desarrollo de XP, enunciado igualmente por Beck (1999), consiste, a grandes rasgos en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.

2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor de negocio.

5. Vuelve al paso 1.

Una vez analizadas las principales características de XP, se decide escoger esta metodología para el desarrollo de la solución, ya que además de ser la metodología seleccionada en el trabajo de investigación precedente a este, es recomendable su empleo en tareas de corto plazo y equipos pequeños como es el caso del presente proyecto.

Conclusiones parciales

En este capítulo fue descrito el proceso de petición y respuesta por el que atraviesa cualquier usuario que interactúa con una página, sitio o aplicación web dinámica. Fueron caracterizados algunos de los lenguajes y herramientas más importantes del lado del servidor que permiten desarrollar soluciones web dinámicas, evidenciando como el empleo de bibliotecas, *plugins*, *snippets* y los *frameworks* PHP influyen positivamente en la rapidez, agilidad de desarrollo, seguridad y calidad final del producto.

Se realizó, además, el análisis de las tendencias tecnológicas que contribuirán a la implementación de la solución propuesta, siendo PHP, el lenguaje de programación del lado servidor escogido para esta tarea, dada su sencillez de uso y las múltiples ventajas. El desarrollo de la solución será guiado por la metodología XP, dado que la misma cumple con las condiciones y necesidades de la problemática en cuestión.

El *framework* PHP seleccionado fue Symfony2. Además, fueron utilizadas otras herramientas para la ejecución de operaciones específicas dentro del marco de trabajo: YAML para las tareas de configuración de las rutas, el archivo de seguridad y los archivos de configuración del propio *framework.* Las Anotaciones fueron escogidas como intérprete entre Symfony2 y el esquema relacional para el formato y la validación de los datos y Doctrine2 como biblioteca para aplicar ORM entre el modelo de objetos y la base de datos. Esta biblioteca a su vez utiliza DQL para las consultas al modelo de datos.

Capítulo 2 Diseño e implementación del sistema

En el presente capítulo se describe las principales funcionalidades de Sherlock versión 0.2b, haciendo énfasis en las deficiencias detectadas y que fueron corregidas en la nueva versión Sherlock. Se muestran las historias de usuarios que permitieron recoger los requisitos a partir de los cuales la solución de esta investigación fue implementada. Fue plasmada, como parte del diseño de la aplicación, la arquitectura que sustenta el desarrollo de la nueva herramienta además de definir tres iteraciones por los programadores para organizar el desarrollo de los requisitos planteados.

2.1 Estado de Sherlock versión 0.2b

Sherlock 0.2b ofrece varias funcionalidades para cumplir con los objetivos para los cuáles fue creado. Esta herramienta cuenta con una base de datos en la cual son agregados los proyectos a los cuáles el usuario quiere aplicarle el proceso de integración y pruebas. Una vez adicionado el proyecto, el mecanismo de integración (Buildbot) se dispara de forma automática a partir de los datos ofrecidos. Posteriormente son ejecutadas las pruebas sobre el código, cuyos resultados son almacenados en el directorio de Sherlock, generándose un reporte con estos resultados y mostrándolos en la aplicación para el usuario. Además de este flujo principal, el usuario puede efectuar otras operaciones como modificar los datos de los proyectos agregados, eliminarlos entre otras.

Sin embargo, la aplicación actual no es del completamente funcional, presenta errores y algunas carencias que hacen que la aplicación no sea explotada al máximo y se use solamente en la actualidad sobre proyectos pilotos. Posee una arquitectura de 3 capas y en cada una de ellas existen deficiencias que son mencionadas a continuación para así poder comprender la relevancia de que la nueva versión sea desarrollada sobre Symfony2.

Primeramente, en la capa de presentación, se observa una interfaz visual poco llamativa e intuitiva, donde se manifiesta el escaso empleo de hojas de estilo para tratar de decorar la aplicación, manteniendo la sencillez de la interfaz. Las vistas están diseñadas fundamentalmente empleando tablas, método que en la actualidad es poco usado y recomendado. El diseño de las vistas utilizando el método mencionado anteriormente contribuye con la lentitud de carga del sitio, además de dificultar la tarea de mantenimiento

ya que no es sencillo para el desarrollador leer el código de tablas anidadas. Por último, pero no menos importante, la utilización excesiva de programación PHP en esta capa afecta el rendimiento de la aplicación, entorpece el mantenimiento y la capacidad de extenderse de la misma, además de incumplir con la filosofía de programación del patrón multicapa.

Posteriormente, en la capa de lógica de la aplicación se destaca el empleo redundante de clases, atributos y otras estructuras, consumiendo recursos y memoria del sistema de forma innecesaria ya que dichas estructuras pueden ser integradas a otros conceptos.

La tercera capa, denominada de almacenamiento o de datos, contempla varias faltas, que, aunque no inmediatas, pueden representar futuros problemas para los desarrolladores de la aplicación. La primera es que la gestión de datos, dígase creación de tablas y consultas, está desarrollada sobre el SGBD MySQL de forma directa, aspecto que dificulta la migración hacia otro SGBD de ser necesario, elevando el costo del proceso, ya que habría que reescribir casi toda la configuración desde cero. Por otro lado, la base de datos presenta atributos poco descriptivos y tablas innecesarias, las cuales son utilizadas en la lógica de la aplicación, sin embargo, contribuyen a la excesiva consulta a la base de datos y ejecución de funcionalidades.

Sherlock 0.2b posee además una jerarquía de carpetas no organizadas correctamente, además de un código poco comentado y ordenado. Los aspectos analizados anteriormente, además de no contar con un número importante de módulos de prueba, derivan en una aplicación poco amigable y objetiva para el usuario, al tiempo que para los desarrolladores el costo de mantenerla y extenderla es muy elevado. Las deficiencias descritas anteriormente, resultan en que no se valore a Sherlock como la solución automatizada al proceso de integración y pruebas en el CEDIN.

2.2 Diseño e implementación de Sherlock versión 0.3

Con el desarrollo de la nueva versión de Sherlock, serán corregidos los inconvenientes existentes para conseguir un producto de mayor robustez y calidad, además, las funcionalidades existentes se modificarán para optimizar sus resultados y finalmente, tomando en cuenta las recomendaciones de la investigación anterior, se incluirán nuevas características como parte de la nueva versión.

La versión 0.3 de Sherlock, incluye en su interfaz una vista para la administración. La misma, es necesario aclarar que no es el tipo de administración propuesta en las recomendaciones de la investigación anterior, ya que el sistema no precisa gestionar información de sus usuarios más allá de conocer a sus administradores, el panel de administración, brindará por tanto opciones de configuración del Buildbot. Dichas opciones permitirán a los administradores interactuar de manera rápida y flexible con los principales parámetros de configuración de la herramienta encargada de la integración continua de código fuente.

Con la idea de lograr un software altamente funcional y que sea tomado en cuenta para su empleo sobre un entorno real de producción, se realizó el análisis de un amplio grupo de herramientas de prueba (ver Anexo 1). A partir de dicho análisis y de acuerdo con las necesidades de la LID, teniendo en cuenta parámetros como pruebas a código fuente de diversos lenguajes, que realicen la búsqueda de una alta gama de errores además de la generación automática de documentación, le serán incorporados a Sherlock los siguientes módulos:

Doxygen es un sistema de documentación libre creado bajo la licencia GPL para lenguajes de programación como C++, C, Java, Objective C, Python, Fortran, PHP, C# entre otros lenguajes. Permite la generación en línea de documentación de un proyecto vía web, y/o un manual de referencia (en Latex).

Doxygen puede ser configurado para extraer código estructurado a partir de archivos fuente. Permite visualizar de manera automática gráficos de dependencia a partir de la relación entre varios elementos, gráficos de herencia, diagramas de colaboración, entre otros.

Findbugs es un programa el cual, mediante análisis estático localiza errores en código escrito en Java. Es un software libre, distribuido bajo la Licencia Pública Menor GNU. Requiere para su uso del entorno de ejecución de Java JRE o del Kit de Desarrollo en Java JDK 1.5.0 o superior, sin embargo, es capaz de analizar programas compilados en cualquier versión de Java y de detectar más de 150 variedades de errores cometidos a la hora de programar. La versión actual de esta aplicación es la 2.0.1-rc2, liberada el 11 de mayo de 2012.

CppCheck es una herramienta de código abierto para análisis estático de los lenguajes de programación C y C++. Su primera versión fue liberada en el año 2009 por su creador Daniel Marjamäki, es un programa

multiplataforma, distribuido bajo GPL. Entre alguna de sus principales características se pueden encontrar el chequeo de fuera de rango, de fuga de memoria, el uso incorrecto de la *Standard Library* (STL), la advertencia de uso de funciones obsoletas, entre otras.

Pixy es una herramienta multiplataforma de código abierto desarrollada por el Laboratorio de Sistemas de Seguridad de la Universidad Tecnológica de Viena. Su principal objetivo es escanear código fuente PHP para detectar dos de las principales vulnerabilidades de las aplicaciones web: las inyecciones SQL y los ataques XSS. Permite además la detección automática de errores en la inclusión de archivos y también la generación de gráficos de dependencia que ayuden a entender a los usuarios las causas de las vulnerabilidades encontradas.

Como parte del desarrollo de la solución y tomando en cuenta las recomendaciones de la investigación precedente (14) de emplear nuevas tecnologías sobre la aplicación, la nueva versión de Sherlock fue reescrita prácticamente en su totalidad debido a la migración del sistema al framework Synfony2, aprovechando las múltiples ventajas que ofrece en cuanto a flexibilidad y mantenimiento.

El empleo de este framework mejora el rendimiento de la herramienta a la hora de cargar su contenido, ya que el punto de entrada a la aplicación es único y desde este se redirecciona al resto de las páginas, evitando así tener que cargar completamente las rutas cada vez que son solicitadas. Permite, además, la aplicación de herencia para las vistas, contribuyendo a no tener que generar elementos repetidos en estas, sino que hereda los componentes comunes y solo carga el contenido específico, también aporta legibilidad y organización al código.

En términos de la lógica de negocio se mejoró la estructura y organización del contenido, separando los componentes controladores de la aplicación, de las vistas y del modelo de datos, ya que el framework ofrece una jerarquía de carpetas sencilla con la cual los nuevos desarrolladores que contribuyan con la herramienta se sentirán cómodos y podrán adaptarse al entorno de trabajo en poco tiempo.

El modelo de datos también fue objeto de cambio y mejoras. El proceso de validación de datos provenientes de las vistas y efectuado en los controladores fue sustituido por el mecanismo de validación automática de Symfony2. Con el uso de esta técnica la validación de datos se efectuó directamente en la definición de las entidades (tablas) mientras que el controlador únicamente se encarga de invocar la

función *isValid*() para desencadenar el proceso. Otra de las características agregadas a la aplicación fue incluirle compatibilidad con otros SGBD mediante el uso de Doctrine2 para el manejo del modelo de datos. Además, el empleo del lenguaje de consultas DQL, permitió el tratamiento de inyecciones SQL, XSS y otros aspectos de seguridad importantes en aplicaciones web que no son cubiertos totalmente por Sherlock 0.2b.

2.3 Modelo arquitectónico

En la construcción de software existen diferentes componentes que proporcionan una plantilla de trabajo que unifica la manera en la que los miembros del equipo ven el sistema y además impone una transformación al diseño del mismo. Estos componentes son los estilos arquitectónicos, los patrones arquitectónicos y los patrones de diseño, los cuales están estrechamente relacionados, formando parte de un todo conocido como arquitectura de software.

En su libro "*UML y Patrones",* Craig Larman (2010) concuerda con Eugenia Bahit, cuando definen a la arquitectura de software como la forma en la cual los componentes de un sistema, se organizan, interactúan relacionan entre sí y con el contexto, aplicando normas y principios de diseño y calidad, que fortalecen y fomentan la usabilidad a la vez que dejan preparado el sistema, para su propia evolución. (15)

2.3.1 Estilo arquitectónico

"Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para los componentes del mismo". (5)

Por la forma en que está concebida la aplicación, fueron aplicados dos estilos arquitectónicos: **la arquitectura programa principal/subprograma y la arquitectura orientada a objetos**.

La primera arquitectura corresponde a un sub-estilo de la denominada **arquitectura llamada y retorno.** Este sub-estilo fue aplicado por la necesidad existente de separar las funciones de la aplicación en una jerarquía de control donde el denominado programa principal invoca a varios componentes de programa, los cuales a su vez pueden invocar a otros componentes.

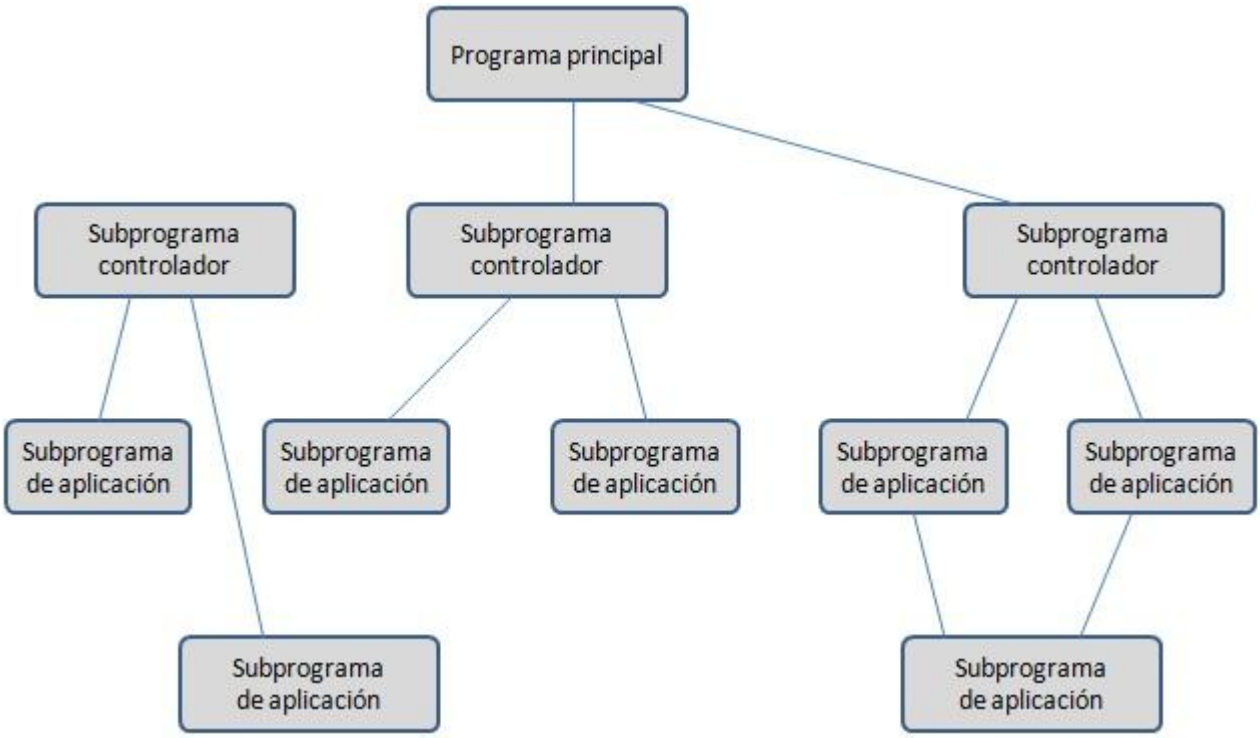

Fig4. Arquitectura orientada a objetos

El segundo de los estilos, la **arquitectura orientada a objetos**, está diseñada sobre la idea de que los componentes de un sistema encapsulen los datos y las operaciones que deben aplicarse para manipular los datos. La comunicación y coordinación entre los componentes se consigue mediante el paso de mensajes. (5)

2.3.2 Patrón Arquitectónico.

El MVC es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y uno de los más utilizados para aplicaciones web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite “no mezclar lenguajes de programación en el mismo código”. (15)

MVC divide las aplicaciones en tres niveles de abstracción. El **modelo**, representa la lógica de negocios siendo el responsable de acceder de forma directa a los datos actuando como intermediario con la base

de datos. La **vista**, es la encargada de mostrar la información al usuario de forma gráfica y legible mientras que el **controlador**, actúa como intermediario entre la vista y el modelo; es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista.

El funcionamiento básico del patrón MVC puede resumirse en:

1. El usuario realiza una petición.
2. El controlador captura el evento y efectúa la llamada al modelo efectuando las modificaciones y operaciones pertinentes sobre el mismo.
3. El modelo será el encargado de interactuar con la base de datos, ya sea en forma directa, con una capa de abstracción para ello, un *web service*, etc. y retornará esta información al controlador.
4. El controlador recibe la información y la envía a la vista.
5. La vista, procesa esta información creando una capa de abstracción para la lógica que se encargará de procesar los datos y otra para el diseño de la interfaz gráfica.
6. La lógica de la vista, una vez procesados los datos, los acomodará en base al diseño de la interfaz gráfica y los entregará al usuario.

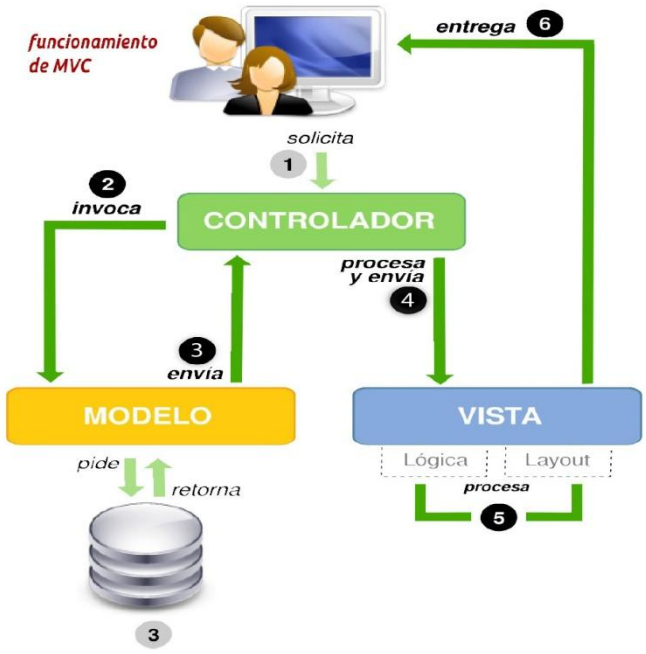

Fig5. Patrón arquitectónico Modelo Vista Controlador

2.3.3 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interfaces. La principal característica que identifica a estos patrones es que deben resolver problemas similares a otros ocurridos con anterioridad en distintas circunstancias.

A partir del estudio del libro "UML y Patrones" (Craig Larman, 2010), se puede observar una gran variedad de patrones de diseño en el campo de la programación. Estos patrones son conocidos como patrones *Gang of Four* (en lo adelante GoF). Existen otros patrones conocidos como GRASP (*General Responsibility Assignment Software Patterns)* o Patrones de Principios Generales para Asignar Responsabilidades). Los patrones de diseño ayudan a elegir las clases adecuadas y a decidir cómo estas deben interactuar. La siguiente sección de la investigación recoge los patrones empleados en la solución.

2.3.3.1 Patrones GoF

- **Singleton**

  La utilización de este patrón asegura la creación de una sola instancia de clase y provee un punto de acceso global para accederla. Se crea una clase que gestiona una sola instancia de ella misma y cada vez que se necesite, simplemente se consulta la clase y esta devolverá esa misma instancia.

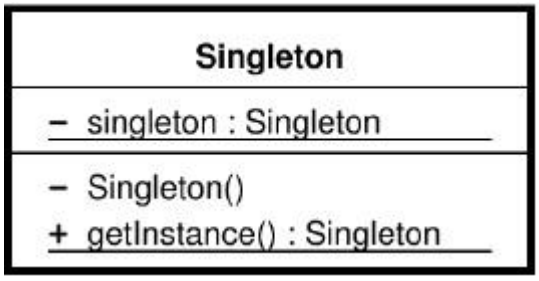

Fig6. Patrón singleton

2.3.3.2 Patrones GRASP

- **Bajo acoplamiento**

  Asigna una responsabilidad de manera que el acoplamiento permanezca bajo. El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. Un elemento con bajo acoplamiento no depende de demasiados otros elementos. (16)

- **Alta cohesión**

  Asignar una responsabilidad de manera que la cohesión permanezca alta. En cuanto al diseño de objetos, la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. (16)

- **Experto**

  Patrón utilizado para asignar responsabilidades principalmente en el diseño orientado a objetos. Su puesta en práctica conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

- **Controlador**

  Asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. (17)

- **Creador**

  Se asigna la responsabilidad de que una clase B cree un objeto de la clase A si se cumple uno o más de los casos siguientes:

  - B agrega objetos de A.
  - B contiene objetos de A.
  - B registra instancias de objetos de A.
  - B utiliza más estrechamente objetos de A.
  - B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea. (16)

2.4 Exploración

Una vez determinada la arquitectura de software del sistema, se inició la construcción de la solución. Para ello, y como fue descrito en la fundamentación teórica de la investigación, fue seleccionada la metodología XP como guía del proceso. La misma dictamina la exploración como la fase inicial del desarrollo de la aplicación. En esta fase, fueron definidas las historias de usuario al tiempo que el equipo de desarrollo comenzó a familiarizarse con las herramientas, tecnologías y prácticas que se utilizaron en el proyecto.

2.4.1 Actores del sistema

Sherlock0.3 modificó los actores del sistema de su antecesor, eliminando el actor Usuario Restringido, ya que no es preciso una especialización del actor Usuario para restringir los privilegios de estos sobre la gestión de proyectos, un usuario podrá tener privilegios para gestionar los proyectos solo cuando éste sea el creador de dicho proyecto.

Actores	Descripción
Usuario	Tiene acceso a las funcionalidades del sistema con privilegios determinados, no puede manipular proyectos que no sean creados por él ni acceder a las opciones de administración.
Administrador	Es una especialización de usuario, utiliza la herramienta con total acceso a todas las funcionalidades del sistema.
Buildbot	Agente externo que realiza la integración continua del código.
Sistema	Ejecuta los módulos de prueba de manera automática.

Tabla 1. Actores del Sistema

2.4.2 Historias de usuario de Sherlock 0.3

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, describen brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. (12)

A continuación se muestran las historias de usuario descritas a partir de los requisitos detectados, donde fueron recogidas las principales funcionalidades con las que debía contar el sistema.

Historia de usuario	
Número: 1	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Migrar Sherlock a un *framework* PHP.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 5	**Iteración asignada:** 1
Programador responsable: Jesús Vilches Pupo	
Descripción: Implementar una aplicación con una nueva interfaz gráfica y que le brinde a los desarrolladores el rápido entendimiento de la estructura del software. Además, debe contribuir a la disminución del costo de mantenimiento y extensión, así como al aumento de la rapidez de carga del sitio.	
Observaciones: Reescribir o eliminar clases que conforman la arquitectura de tres capas de Sherlock 0.2b para llevar la aplicación a la arquitectura Modelo Vista Controlador y a la compatibilidad con Symfony2.	

Tabla 2. Historia de Usuario No.1

Historia de usuario	
Número: 2	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Diseño e implementación de la base de datos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Jesús Vilches Pupo	
Descripción: Crear y configurar el modelo actualizado de base de datos teniendo en cuenta la información de los proyectos y de los reportes con los resultados de las pruebas. Garantizar el correcto formato y validación de la información.	
Observaciones:	

Tabla 3. Historia de Usuario No.2

Historia de usuario

Número: 3	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Configurar mecanismo de integración.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Jesús Vilches Pupo	
Descripción: Reescribir las clases SherlockConf y Builder para corregir las deficiencias encontradas y migrarlas hacia un *framework* PHP.	
Observaciones:	

Tabla 4. Historia de Usuario No.3

	Historia de usuario
Número: 4	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Gestionar dependencias.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Jesús Vilches Pupo	
Descripción: Verificar antes de ejecutar cada módulo de prueba que sus dependencias estén correctamente instaladas.	
Observaciones:	

Tabla 5. Historia de Usuario No.4

	Historia de usuario
Número: 5	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Filtrar proyectos mediante un rango de fechas.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Jesús Vilches Pupo	
Descripción: Mostrar los resultados de las pruebas, permitiendo el filtrado de proyectos mediante la	

selección en el navegador de un rango de fechas mostrando sólo los proyectos a los que se le hayan ejecutado pruebas bajo este criterio de búsqueda.
Observaciones:

Tabla 6. Historia de Usuario No.5

Historia de usuario	
Número: 6	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Modificar horario de integración.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Jesús Vilches Pupo	
Descripción: La herramienta debe ofrecer a los administradores del sistema la posibilidad de modificar el horario de integración de un proyecto de forma dinámica a través de la interfaz visual de Sherlock.	
Observaciones:	

Tabla 7. Historia de Usuario No.6

Historia de usuario	
Número: 7	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Autenticar usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Jesús Vilches Pupo	
Descripción: Migrar el proceso de autenticación y autorización de usuarios hacia mecanismos de seguridad más completos y robustos.	
Observaciones:	

Tabla 8. Historia de Usuario No.7

Historia de usuario	
Número: 8	**Usuario:** Línea de Integración y Despliegue
Nombre historia: Graficar reportes.	

Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 4	**Iteración asignada:** 3
Programador responsable: Jesús Vilches Pupo	
Descripción: La herramienta debe poder mostrar gráficas de algunos reportes con los resultados de las pruebas ejecutadas sobre los proyectos procesados por Sherlock.	
Observaciones:	

Tabla 9. Historia de Usuario No.8

		Historia de usuario
Número: 9	**Usuario:** Línea de Integración y Despliegue	
Nombre historia: Integrar nuevos módulos.		
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto	
Puntos estimados: 3	**Iteración asignada:** 3	
Programador responsable: Jesús Vilches Pupo		
Descripción: Incorporar al sistema los módulos de definidos por la LID.		
Observaciones:		

Tabla 10. Historia de Usuario No.9

2.5 Plan de Iteraciones

El plan de iteraciones está compuesto por la relación de historias de usuarios que fueron definidas con anterioridad. Las historias de usuarios son ejecutadas en las diferentes iteraciones de acuerdo con la prioridad establecida por el cliente para cada una de ellas. La duración de las mismas equivale al tiempo transcurrido en la creación de la funcionalidad que se describe, y la duración total es obtenida a través de la suma general del tiempo en que demoró la realización de la historia de usuario, este total se expresa en semanas. A continuación se muestra el plan de iteraciones establecido para la aplicación.

No. Iteración	No. de HU a implementar	Duración de la HU	Pruebas realizadas a las HU	Duración total
1ra	HU1	5	HU1	**8 semanas**

	HU2	1	HU2	
	HU3	2	HU3	
2da	HU 4	1	HU4	**4 semanas**
	HU 5	1	HU5	
	HU 6	2	HU6	
3ra	HU 7	2	HU7	**9 semanas**
	HU 8	4	HU8	
	HU 9	3	HU9	
Total: 21 semanas equivalentes a 5 meses con 1 semana				

Tabla 11. Plan de Iteraciones

2.5.1 Tareas de programación

Las Historias de Usuario de Sherlock agrupadas en cada iteración fueron implementadas durante el transcurso de la iteración a la cual pertenecen. Al principio de cada iteración se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. El trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable.

Tarea	
Número: 1	**Número historia:** 1
Nombre tarea: Organizar el proyecto en una estructura MVC.	
Tipo de tarea: Investigación, Diseño, Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 02/01/2012	**Fecha fin:** 07/01/2012
Programador responsable: Jesús Vilches Pupo.	
Descripción: La estructura de carpetas deberá acomodarse según el orden que brinda Symfony2 para que sea un orden genérico que permita a los desarrolladores nuevos comprender dicha estructura poco tiempo después de insertarse en el proyecto. Crear todas las rutas en el archivo routing.yml para que cada vista sea procesada por un controlador determinado.	

Tabla 12. Tarea de Programación No.1 para HU1

Tarea	
Número: 2	**Número historia:** 1
Nombre tarea: Diseño de interfaces.	

Tipo de tarea: Diseño, Desarrollo	**Puntos estimados:** 1
Fecha inicio: 09/01/2012	**Fecha fin:** 20/01/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Crear y diseñar la plantilla **sherlock_plantilla.html.twig** de la cual heredan el resto de las vistas. En el directorio **views** (directorio establecido por Symfony2 para ubicar las vistas), crear, diseñar y programar las diferentes interfaces gráficas correspondientes a la sección de proyectos.	
Observaciones: El resto de las interfaces visuales correspondientes a las demás secciones de la aplicación son implementadas en posteriores tareas de programación pertenecientes a otras historias de usuario.	

Tabla 13. Tarea de Programación No.1 para HU3

Tarea	
Número: 3	**Número historia:** 1
Nombre tarea: Implementar lógica de la aplicación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 21/01/2012	**Fecha fin:** 31/01/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: La clase controladora *ProjectController* es la que ejecuta toda la lógica del *Create*, *Reading*, *Update*, *Delete* (CRUD) del proyecto. En la vista de mostrar proyecto si se presiona la opción **agregar proyecto** se ejecutará la función de mismo nombre donde se mostrará un formulario para adicionar los datos necesarios con sus respectivas validaciones. Si se presiona sobre la imagen del lápiz ejecutar la opción de **modificar** el proyecto seleccionado y si es presionada la imagen que representa una cruz roja **eliminar** el proyecto. Sólo podrán ejecutar las acciones de eliminar y modificar un proyecto determinado los administradores del sistema y los creadores de dicho proyecto.	
Observaciones: El resto de las clases controladoras correspondientes a las demás funcionalidades de la aplicación son implementadas en posteriores tareas de programación pertenecientes a otras historias de usuario.	

Tabla 14. Tarea de Programación No.2 para HU3

Tarea	
Número: 1	**Número historia:** 2
Nombre tarea: Generar entidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha inicio: 01/02/2012	**Fecha fin:** 08/02/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Crear tres clases: Proyecto, Reporte y Usuario. Configurar las validaciones de cada atributo y establecer las relaciones entre entidades. La relación entre Proyecto y Reporte es de **uno a muchos**. Para eliminar un proyecto definir en la clase Proyecto por medio de anotaciones la opción *onDelete="CASCADE"* y en la clase Reporte *onDelete="SET NULL"*.	

Tabla 15. Tarea de Programación No.1 para HU2

Tarea	
Número: 2	**Número historia:** 2
Nombre tarea: Mapear entidades y crear repositorios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 09/02/2012	**Fecha fin:** 12/02/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Una vez creadas las entidades, se deben asociar a tablas relacionales compatibles con MySQL y crear repositorios para cada una de las entidades los cuales contendrán consultas específicas para sus datos.	

Tabla 16. Tarea de Programación No.2 para HU2

Tarea	
Número: 1	**Número historia:** 3
Nombre tarea: Configurar mecanismo de integración.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13/02/2012	**Fecha fin:** 25/02/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Reescribir las clases *SherlockConf* y *Builder* para corregir las deficiencias encontradas, migrarlas hacia Symfony2 para poder ejecutar el Buildbot.	

Tabla 17. Tarea de Programación No.1 para HU4

Tarea	
Número: 1	**Número historia:** 4
Nombre tarea: Gestionar dependencias de módulos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27/02/2012	**Fecha fin:** 03/03/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Cada módulo tiene una clase (*Module*) que encapsula sus propiedades entre las cuales se encuentra un *array* con los paquetes de los que dependen dichos módulos. Los mismos, contienen también la función *CheckModuleDeps*(), la cual devuelve verdadero o falso dependiendo si el ordenador que ejecuta las pruebas tiene instaladas todas las dependencias del módulo. La función *start*() de cada módulo contiene la lógica de cómo éste se debe ejecutar, al hacerlo, invocar la función *checkModuloDeps*() y en caso de que algún modulo no tenga resueltas todas sus dependencias no ejecutar el módulo (devolver *false*) e informárselo al usuario. En caso contrario (devolver *true*) ejecutar el módulo de forma normal.	

Tabla 18. Tarea de Programación No.1 para HU5

Tarea	
Número: 1	**Número historia:** 5
Nombre tarea: Diseño de la interfaz gráfica.	
Tipo de tarea: Diseño, Desarrollo	**Puntos estimados:** 0.8
Fecha inicio: 05/03/2012	**Fecha fin:** 07/03/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Diseñar y programar las interfaces gráficas para mostrar y poder filtrar los proyectos que posean reportes de resultados a partir de las pruebas y métricas que se le hayan aplicado.	

Tabla 19. Tarea de Programación No.1 para HU6

Tarea	
Número: 2	**Número historia:** 5
Nombre tarea: Programar la lógica de la funcionalidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 08/03/2012	**Fecha fin:** 09/03/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: La clase controladora *Report* es la encargada de procesar toda la información proveniente de las vistas, filtrar los proyectos mediante el criterio seleccionado realizando consultas al modelo de datos y mostrar los resultados correspondientes.	

Tabla 20. Tarea de Programación No.2 para HU6

Tarea	
Número: 1	**Número historia:** 6
Nombre tarea: Diseñar interfaz de administración.	
Tipo de tarea: Diseño, Desarrollo	**Puntos estimados:** 0.7
Fecha inicio: 12/03/2012	**Fecha fin:** 15/03/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Generar la vista de administración donde se podrá realizar la modificación de la hora de integración al archivo de configuración de Buildbot.	

Tabla 21. Tarea de Programación No.1 para HU7

Tarea	
Número: 2	**Número historia:** 6
Nombre tarea: Programar la lógica de las operaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha inicio: 16/02/2012	**Fecha fin:** 18/03/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Se crea la clase controladora *AdminController*, la cual implementará la función *changeHour*().	

Esta función escribirá sobre el archivo master.py, que es el archivo de configuración del Buildbot para modificar el parámetro de la hora de ejecución del proceso de integración continua.

Tabla 22. Tarea de Programación No.2 para HU7

	Tarea
Número: 1	**Número historia:** 7
Nombre tarea: Diseño de la interfaz.	
Tipo de tarea: Diseño, Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 26/03/2012	**Fecha fin:** 29/03/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Crear una vista con un formulario donde el usuario pueda introducir el nombre de usuario y la contraseña. En caso de que las credenciales no sean correctas enviar al usuario un mensaje de notificación.	

Tabla 23. Tarea de Programación No.1 para HU8

	Tarea
Número: 2	**Número historia:** 7
Nombre tarea: Implantar parámetros de configuración de seguridad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 30/03/2012	**Fecha fin:** 04/04/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Configurar la seguridad para la aplicación estableciendo los parámetros correspondientes en el archivo security.yml como el *firewall*, las listas de control de acceso y el proveedor de datos, entre otros.	

Tabla 24. Tarea de Programación No.2 para HU8

	Tarea
Número: 3	**Número historia:** 7
Nombre tarea: Implementar proveedor de datos personalizado.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 09/04/2012	**Fecha fin:** 13/04/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Dado que los usuarios que se autentican en la aplicación, son usuarios obtenidos de un servicio web (LDAP[11]), es necesario definir una clase (*User*) redefiniendo para ello la clase *UserInterface* y la clase *Serializable*. Estas clases contienen los datos de los usuarios que necesita gestionar la seguridad de Symfony.	

Tabla 25. Tarea de Programación No.3 para HU8

	Tarea
Número: 1	**Número historia:** 8
Nombre tarea: Inicializar los resultados.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/04/2012	**Fecha fin:** 21/04/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: La ejecución de las pruebas genera reportes guardados en una carpeta con el nombre de cada módulo. En la clase que encapsula la información de cada módulo se debe interpretar la información contenida en el fichero XML creado por la generación de los resultados de los módulos de prueba y guardarlos en una variable.	

Tabla 26. Tarea de Programación No.1 para HU9

	Tarea
Número: 2	**Número historia:** 8
Nombre tarea: Graficar reportes.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 26/04/2012	**Fecha fin:** 04/05/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Se crea un objeto de tipo **graficoX**, siendo la X el tipo de gráfico a generar (columna, pastel, barra u otro).La información salvada de la lectura del fichero XML, se pasa como parámetro en la creación del objeto. El objeto ejecuta la función paint(), la cual devuelve un *string* con el código para graficar el reporte. Luego, se crea un archivo en la dirección del módulo específico y con la función *file_put_contents*() de PHP, se guarda el *string* retornado por la función paint().	

Tabla 27. Tarea de Programación No.2 para HU9

	Tarea
Número: 1	**Número historia:** 9
Nombre tarea: Integrar y ejecutar los módulos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 07/05/2012	**Fecha fin:** 16/05/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Mediante la clase *Module* programar la interfaz de ejecución de cada uno de los módulos dentro del ambiente de ejecución de Sherlock.	

Tabla 28. Tarea de Programación No.1 para HU10

	Tarea
Número: 2	**Número historia:** 9
Nombre tarea: Mostrar resultados.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 17/05/2012	**Fecha fin:** 27/05/2012
Programador responsable: Jesús Vilches Pupo	
Descripción: Diseñar y programar la interfaz de salida de resultados de los módulos de prueba.	

Tabla 29. Tarea de Programación No.2 para HU10

2.6 Tarjetas Clases Responsabilidades Colaboraciones

Las tarjetas CRC son una técnica de modelado orientado a objeto que permite identificar clases y sus responsabilidades, su principal finalidad es obtener un diseño simple y evitar la implementación de funcionalidades innecesarias, ayudando de esta manera al refinamiento de las clases.

Tarjeta CRC	
Número: 1	**Nombre clase:** *ModulesController*
Responsabilidades: Mostrar información acerca de los módulos de prueba.	
Colaboraciones: *ObjectRepository*, *SherlockConf*	

Tabla 30. Tarjeta CRC No.1

Tarjeta CRC	
Número: 2	**Nombre clase:** *ProjectController*
Responsabilidades: Ejecuta las operaciones de adicionar, editar, mostrar y eliminar proyectos.	
Colaboraciones: Proyecto, *ProjectType*, *ProjectRepository*, *Builder*, *SherlockConf*	

Tabla 31. Tarjeta CRC No.2

Tarjeta CRC	
Número: 3	**Nombre clase:** *ReportController*
Responsabilidades: Mostrar los proyectos a los que se le hayan ejecutado pruebas, permitir filtrar estos proyectos por rango de fechas, por una fecha exacta o mediante un módulo de prueba.	
Colaboraciones: Proyecto, *Report*, *ProjectRepository*	

Tabla 32. Tarjeta CRC No.3

Tarjeta CRC	
Número: 4	**Nombre clase:** *RunController*
Responsabilidades: Iniciar la ejecución de pruebas sobre los proyecto.	
Colaboraciones: Proyecto, *SherlockConf*, *ObjectRepository*	

Tabla 33. Tarjeta CRC No.4

Tarjeta CRC	
Número: 5	**Nombre clase:** *SecurityController*
Responsabilidades: Manejar la autenticación y la autorización de la aplicación.	
Colaboraciones: *SecurityContext*, *Controller*	

Tabla 34. Tarjeta CRC No.5

Tarjeta CRC	
Número: 6	**Nombre clase:** *ReportController*
Responsabilidades: Mostrar los proyectos a los que se le hayan ejecutado pruebas, permitir filtrar estos proyectos por rango de fechas, por una fecha exacta o mediante un módulo de prueba.	
Colaboraciones: Proyecto, *Report*, *ProjectRepository*	

Tabla 35. Tarjeta CRC No.6

Tarjeta CRC	
Número:7	**Nombre clase:** *AdminController*
Responsabilidades: Permitir gestionar esclavos, planificadores, modificar el horario de integración entre otros aspectos del archivo de configuración del Buildbot.	
Colaboraciones: *Builder*, *SherlockConf*	

Tabla 36. Tarjeta CRC No.7

Tarjeta CRC	
Número: 8	**Nombre clase:** Proyecto
Responsabilidades: Representa la entidad proyecto la cual será mapeada a tabla relacional de la base de datos de la aplicación.	
Colaboraciones:	

Tabla 37. Tarjeta CRC No.8

Tarjeta CRC	
Número: 9	**Nombre clase:** Reporte
Responsabilidades: Representa la entidad proyecto la cual será mapeada a tabla relacional de la base de datos de la aplicación.	
Colaboraciones:	

Tabla 38. Tarjeta CRC No.9

Tarjeta CRC	
Número: 10	**Nombre clase:** *ProjectType*
Responsabilidades: Esta es la clase que permite pintar en la vista el formulario proyecto a la hora de adicionar o modificar un proyecto.	

Colaboraciones:

Tabla 39. Tarjeta CRC No.10

	Tarjeta CRC
Número: 11	**Nombre clase:** *ProjectRepository*
Responsabilidades: Esta clase es la que permite consultar y poner datos en la tabla proyecto de la base de datos.	
Colaboraciones: Proyecto	

Tabla 40. Tarjeta CRC No.11

	Tarjeta CRC
Número: 12	**Nombre clase:** *Builder*
Responsabilidades: Clase que establece la comunicación con el Buildbot	
Colaboraciones: Proyecto, *SherlockConf*	

Tabla 41. Tarjeta CRC No.12

	Tarjeta CRC
Número: 13	**Nombre clase:** *SherlockConf*
Responsabilidades: Almacenar datos de configuración de Sherlock	
Colaboraciones:	

Tabla 42. Tarjeta CRC No.13

	Tarjeta CRC
Número: 14	**Nombre clase:** *ObjectRepository*
Responsabilidades: Encargada de que durante la ejecución de la aplicación exista un único arreglo de los módulos de prueba y métricas.	
Colaboraciones:	

Tabla 43. Tarjeta CRC No.14

	Tarjeta CRC
Número: 15	**Nombre clase:** *Module*
Responsabilidades: Clase que permite integrar módulos de prueba y métricas desarrollados por terceros a la aplicación.	
Colaboraciones: *Extension*, *ObjectRepository*, Proyecto	

Tabla 44. Tarjeta CRC No.15

	Tarjeta CRC
Número: 16	**Nombre clase:** *Extension*
Responsabilidades: Clase padre que brinda las características generales para las clases especializadas	

Module.
Colaboraciones: Proyecto

Tabla 45. Tarjeta CRC No.16

Tarjeta CRC	
Número: 17	**Nombre clase:** *Graphic*
Responsabilidades: Clase genérica que contiene la información general de los tipos de gráficos que serán visualizados en la aplicación.	
Colaboraciones:	

Tabla 46. Tarjeta CRC No.17

2.7 Diagrama de despliegue

La metodología XP propone, para una mejor visión del proyecto y de su desarrollo, el empleo de diagramas siempre y cuando su creación no implique mayor esfuerzo que la implementación del mismo. El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. (18)

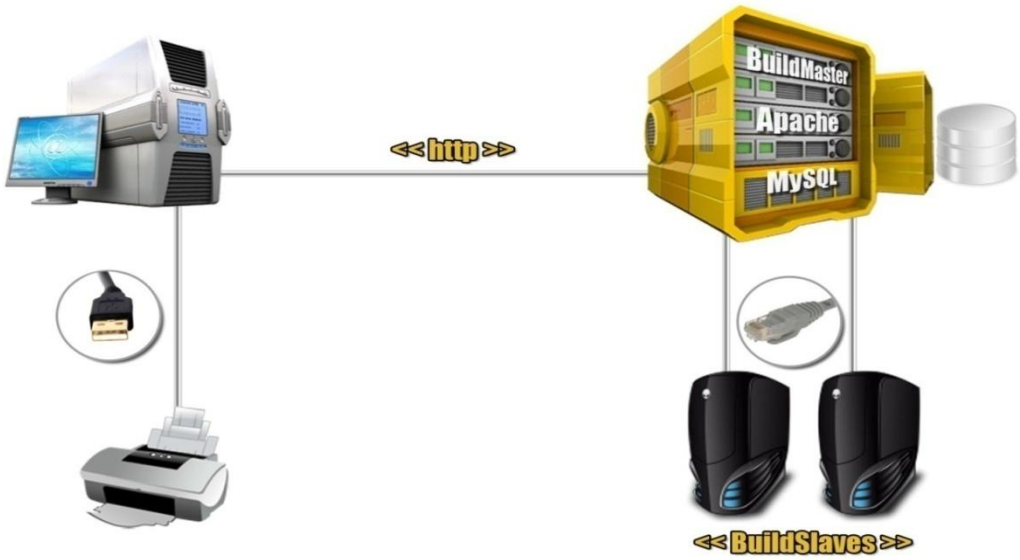

Fig7. Diagrama de despliegue

Conclusiones parciales

Luego de haber detectado los puntos de optimización de la versión anterior, y ofrecer una propuesta de la solución a la problemática actual, quedó definido el diseño del sistema. Se estableció un modelo arquitectónico que proporciona una plantilla de trabajo unificando así la manera en que los miembros del equipo ven el sistema.

Se efectuó además, el levantamiento de requisitos mediante las historias de usuario, las cuales fueron divididas en tareas de programación para poder repartir de forma organizada y lo más sencilla posible la forma en que será implementada la solución. Estas actividades quedaron planificadas mediante un plan de iteraciones efectuando de esta manera una estimación del tiempo que empleado en darle cumplimiento a los objetivos propuestos.

Capítulo 3 Validación de la solución propuesta

El presente capítulo cubre las pruebas realizadas a la aplicación que demuestran que la misma se encuentra en condiciones de ser explotada en un entorno real de desarrollo. Fueron realizados los diseños de casos de prueba correspondientes a cada una de las iteraciones y se ejecutaron, para finalmente recolectar y evaluar los datos resultantes dando cumplimiento con las entregas establecidas en el plan de iteraciones.

3.1 Pruebas de software

La prueba de software, es definida por Pressman como una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, donde los resultados son observados y registrados para realizar posteriormente algún tipo de evaluación.

Las pruebas permiten determinar si el software satisface los requerimientos del usuario y si el software en una determinada fase del desarrollo cumple las condiciones impuestas al inicio de la etapa, proceso conocido como validación y verificación del software. En esencia, *"Todo código que pueda fallar debe tener una prueba"*. (19)

Otra de las razones por las que se realizan las pruebas es para comprobar si existe un error o algún retraso importante que pueda afectar el funcionamiento del sistema, ya que se requiere que sea rápido, confiable y estable.

3.2 Pruebas realizadas

Para la realización de las pruebas a la herramienta, fueron diseñados casos de prueba de funcionalidad y de seguridad, en aras de depurar alguna no conformidad encontrada o dar por culminada cada una de las iteraciones planificadas en el desarrollo de la solución.

3.2.1 Pruebas de funcionalidad

Las pruebas de funcionalidad tienen como objetivo la detección de errores en la implementación de los requerimientos. Su meta, es la verificación del procesamiento, recuperación y desarrollo adecuado de las reglas de negocio y los requisitos funcionales. Mediante la técnica de caja negra, son ejecutadas dichas pruebas, en espera de la apropiada aceptación de datos y obtención de resultados cuando se usen datos válidos, o del despliegue de mensajes apropiados de error y/o precaución cuando se usen datos inválidos.

3.2.2 Pruebas de seguridad

Las pruebas de seguridad son un prototipo de pruebas de sistema las cuales van encaminadas a comprobar que los mecanismos de protección integrados al sistema respondan satisfactoriamente ante un intento de irrupción inapropiada, además del acceso a información confidencial o sección restringida. La seguridad del sistema se puede medir además mediante la capacidad que posea el mismo de prever alguna acción destinada a provocar algún daño ya sea de forma casual o premeditada.

3.3 Diseño de casos de pruebas

Para la documentación de los casos de prueba del sistema, en el cuerpo del documento fue plasmado por cada iteración un caso de prueba, mientras que el resto puede ser examinado en la sección de Anexos de la investigación.

3.3.1 Primera iteración

Caso de prueba de funcionalidad: Añadir proyecto.

Descripción general: el caso de uso se inicia cuando el usuario necesita efectuar el proceso de integración y pruebas a las fuentes de algún producto software. Accede a la sección de insertar un proyecto e introduce los datos correspondientes.

Condiciones de ejecución: *el usuario debe estar autenticado.*

Historias de usuario validadas en el caso de prueba: *1, 2, 3.*

Escenario	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1 Se añade un proyecto de forma satisfactoria.	**1.1** Situarse en la opción de "Proyecto" del menú principal. **1.2** Al desplegarse el menú escoger la opción de "Adicionar". **1.3** Proveer los campos con los datos pedidos de manera correcta. **1.4** Pulsar el botón actualizar “Agregar/Modificar”.	**N/A**	El proyecto se registra en la base de datos, se notifica al usuario del éxito de la operación y automáticamente se despliega un listado de los proyectos registrados en el sistema.	**1.1** El controlador *createProjectAction*() es invocado, el cual, mediante la función *buildbot_get_schedulers*() obtiene del archivo de configuración del Buildbot los planificadores disponibles para la integración. **1.2** Con la inyección de dependencias se obtiene el “*form.factory*”, se pinta la vista de agregar proyecto y se le hace un *render* a la vista para mostrarla en el navegador. **1.3** El usuario introduce los datos del proyecto, selecciona el planificador y pulsa el botón “Adicionar”. **1.4** Se vuelve a invocar a *createProjectAction* (), el cual recoge los datos y se validan. **1.5** Seguidamente con la función *buildbot_configurator*(), se escribe en el fichero de configuración del Buildbot la información pertinente. **1.6** Se mapea y persiste la nueva entidad proyecto a la base de datos. **1.7** Se reinicia el servicio del Buildbot con la función *buildbot_restart_service*() para chequear si el fichero quedó correctamente configurado. **1.8** Se renderiza la vista de mostrar proyectos. **1.9** En la vista se muestra un mensaje de operación exitosa y se muestra el listado de proyectos.
EC 2 Introducir datos incorrectos para agregar un proyecto.	**2.1** Situarse en la opción de "Proyecto" del menú principal. 2.2 Al desplegarse el menú escoger la opción de "Adicionar". 2.3 Proveer los campos con los datos pedidos de manera incorrecta. 2.4 (Opcional) Pulsar el botón actualizar “Agregar/Modificar”.	**N/A**	Se muestran mensajes de error informando sobre el error cometido.	**2.1** Se mantiene el mismo flujo central del escenario 1 hasta la acción 1.3, en este punto antes de pulsar el botón “Adicionar”, son validados los campos y los que no estén correctamente escritos se le devolverá un mensaje de error al usuario para que corrija la información.

Tabla 47. Diseño de caso de prueba/1ra iteración/Añadir proyecto

3.3.2 Segunda iteración

Caso de prueba de funcionalidad: Modificar horario de integración.

Descripción general: el caso de uso se inicia cuando el administrador del sistema necesita modificar la hora en que se iniciará el proceso de integración.

Condiciones de ejecución: *el usuario debe poseer rol de administrador.*

Historias de usuario validadas en el caso de prueba: *7.*

Escenario	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1 Modificar la hora de integración introduciendo una hora válida.	**1.1** Situarse en la opción de "Administración" del menú principal. **1.2** Al desplegarse el menú escoger la opción de "Modificar integración". **1.3** Proveer los campos con una hora válida. **1.4** Pulsar el botón actualizar.	**V** **22:12**	Internamente se modifica el horario de integración y se notifica al usuario del éxito de la operación.	**1.1** El valor de la hora es enviado al controlador "AdminController" el cual crea una instancia del objeto "Builder". **1.2** Con esta instancia se ejecuta la función *changeHourAction*() pasándole como parámetro la hora obtenida de la vista. Esta función abre el archivo de configuración del Buildbot master.py en modo lectura y escritura y parsea el fichero hasta llegar al lugar indicado. Una vez que el puntero está ubicado en la posición deseada se reemplaza el segmento de texto anterior por uno nuevo con la nueva hora de integración.

Tabla 48. Diseño de caso de prueba/2da iteración/Modificar horario de integración

3.3.3 Tercera iteración

Caso de prueba de funcionalidad: Visualizar reporte.

Descripción general: el caso de uso se inicia cuando el usuario precisa verificar los resultados obtenidos a partir de las pruebas aplicadas al su código fuente.

Condiciones de ejecución: *debe haberse ejecutado al menos una vez el proceso de integración y prueba sobre el proyecto del cual el usuario desea observar los resultados.*

Historias de usuario validadas en el caso de prueba: 9, 10.

Escenario	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1 Mostrar los resultados de las pruebas ejecutadas sobre un proyecto.	**1.1** Pulsar en el menú principal la opción de "Reportes". **1.2** En el panel de la izquierda de la ventana filtrar el proyecto deseado o seleccionarlo directamente sobre el listado en la sección de la derecha. **1.3** Aparece una lista en forma de *tabs* con cada uno de los módulos de prueba ejecutados sobre el código. **1.4** Pulsar sobre cada uno para mostrar el reporte correspondiente.	**V** **$idReporte**	Una vez localizado el proyecto deseado será mostrada una vista en forma de *tabs* con los módulos de prueba ejecutados sobre el código fuente del proyecto. Al navegar por cada uno de estos módulos será visualizado el reporte correspondiente a cada módulo en la misma vista.	**1.1** Al pulsar sobre el proyecto específico se ejecuta el método *showReportResultAction*() del controlador *ReportController*() al cual se le pasa como parámetro el identificador del reporte asociado al proyecto seleccionado. **1.2** Se guarda el proyecto relacionado con el reporte ejecutando una consulta a BD. **1.3** Se crea un arreglo y usando la función *scandir*() se llena con los nombres de los módulos ejecutados. **1.4** Se itera sobre el arreglo anterior y se crea un arreglo que guarda la ruta donde se encuentra el archivo html con los resultados de cada módulo ejecutado al proyecto escogido en la fecha seleccionada. **1.5** Ambos arreglos son enviados a la vista en la cual se itera sobre el primero para mostrar en forma de *tabs* los nombres de los módulos. **1.6** Al pulsar por cada uno de estos nombres se muestra abajo el archivo html correspondiente con los resultados de la prueba.
EC 2 Mostrar las gráficas	**2.1** Pulsar en el menú principal la	**V**	Al seleccionar la opción	**2.1** Al pulsar sobre el proyecto específico se ejecuta el método *showReportResultAction*() del

asociadas a los resultados creados a partir de las pruebas realizadas sobre el proyecto seleccionado.	opción de "Reportes". **2.2** En el panel de la izquierda de la ventana filtrar el proyecto deseado o seleccionarlo directamente sobre el listado en la sección de la derecha. **2.3** Aparece una lista en forma de *tabs* con cada uno de los módulos de prueba ejecutados sobre el código. **2.4** Pulsar sobre cada uno para mostrar los reporte correspondientes. **2.5** En la esquina superior de la visualización del reporte html pulsar sobre la opción "Mostrar gráfico".	**$idReporte**	"Mostrar gráfico" se visualizará el gráfico del reporte correspondiente	controlador *ReportController*() al cual se le pasa como parámetro el identificador del reporte asociado al proyecto seleccionado. **2.2** Se guarda el proyecto relacionado con el reporte ejecutando una consulta a BD. **2.3** Se crea un arreglo y usando la función *scandir*() se llena con los nombres de los módulos ejecutados. **2.4** Se itera sobre el arreglo anterior y se crea un arreglo que guarda la ruta donde se encuentra el archivo nombrado "grafico.html". **2.5** Ambos arreglos son enviados a la vista en la cual se itera sobre el primero para mostrar en forma de *tabs* los nombres de los módulos. **2.6** Al pulsar por cada uno de estos nombres se muestra abajo la opción mostrar gráfico la cual cuando se pulsa visualiza el gráfico.

Tabla 49. Diseño de caso de prueba/3ra iteración/Visualizar reporte

3.3.4 Pruebas de seguridad

Caso de prueba: Mostrar lista de proyectos.

Descripción general: el caso de uso se inicia cuando el usuario accede a la opción del sistema de visualizar los proyectos registrados actualmente en la aplicación.

Condiciones de ejecución: *deben existir proyectos en el sistema y el usuario debe estar autenticado.*

Historias de usuario validadas en el caso de prueba: *3, 8.*

Escenario	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1 El usuario accede a la opción de mostrar la lista de proyectos. En la sección de "Acciones" de la tabla los botones de "Editar" y "Eliminar" proyecto estarán disponibles (visibles).	**1.1** Situarse en la opción de "Proyecto" del menú principal. **1.2** Al desplegarse el menú escoger la opción de "Mostrar listado".	**N/A**	El sistema muestra la vista con todos los proyectos del sistema brindándole al usuario la oportunidad de editar o eliminar algún proyecto.	**1.1** La acción *showInfoAction*() del controlador *ProjectController*() es ejecutada. Se crea un objeto *SherlockConf* y se guardan en un arreglo sysAdmins los administradores del sistema. **1.2** Se registra en una variable $actualUser el usuario registrado en la sesión en ese momento. **1.3** Se itera sobre el arreglo sysAdmins y se pregunta si el $actualUser corresponde con alguno de los administradores del sistema guardando el resultado en una variable $admin. **1.4** Se ejecuta una consulta para guardar en una variable $result todos los proyectos registrados. **1.5** Se envía a la vista las variables $actualUser, $admin y $result. **1.6** En la vista se itera sobre $result mostrando las propiedades de cada proyecto en la tabla. **1.7** En el campo "Acciones" estarán visible las opciones de "Editar" y "Eliminar proyecto dado que mediante una comprobación de seguridad el usuario conectado es o bien administrador del sistema o el creador del proyecto".
EC 2 El usuario accede a la opción de mostrar la lista de proyectos. En la sección de "Acciones" de la tabla los botones de "Editar" y "Eliminar" proyecto no estarán disponibles (visibles).	**1.1** Situarse en la opción de "Proyecto" del menú principal. **1.2** Al desplegarse el menú escoger la opción de "Mostrar listado".	**N/A**	El sistema muestra la vista con todos los proyectos del sistema sin brindarle al usuario la oportunidad de editar o eliminar algún proyecto.	**1.1** La acción *showInfoAction*() del controlador *ProjectController*() es ejecutada. Se crea un objeto SherlockConf y se guardan en un arreglo sysAdmins los administradores del sistema. **1.2** Se registra en una variable $actualUser el usuario registrado en la sesión en ese momento. **1.3** Se itera sobre el arreglo sysAdmins y se pregunta si el $actualUser corresponde con alguno de los administradores del sistema guardando el resultado en una variable $admin. **1.4** Se ejecuta una consulta para guardar en una variable $result todos los proyectos registrados. **1.5** Se envía a la vista las variables $actualUser, $admin y $result. **1.6** En la vista se itera sobre $result mostrando las propiedades de cada proyecto en la tabla. **1.7** En el campo "Acciones" no estarán visible las opciones de "Editar" y "Eliminar proyecto dado que mediante una comprobación de seguridad el usuario conectado no califica como administrador del sistema o el creador del proyecto".

Tabla 50. Diseño de caso de prueba/3ra iteración/Mostrar lista de proyectos

Conclusiones parciales

El diseño y la ejecución de los casos de prueba abordados en esta sección contribuyeron a verificar y validar los requisitos identificados. Sin la aplicación de este paso necesario en la construcción del software, sería impredecible el comportamiento del mismo en manos del usuario final. Además, la ejecución de las pruebas permitió detectar y corregir fallas en la implementación de los requisitos, aspecto determinante para avanzar a través del plan de iteraciones y por consiguiente hacia el cumplimiento de la entrega del producto.

Conclusiones

El empleo de herramientas que apoyen las tareas de programación, en específico, el empleo de *frameworks* PHP para desarrollo web, sin representar la solución definitiva, dado que el factor decisivo siempre serán los conocimientos y las habilidades del programador, proporcionan numerosas ventajas y facilidades para el mismo. Le brinda al desarrollador rapidez, capacidad de extensión, reducción de tiempo de desarrollo entre otras ventajas. El empleo de un *framework* garantiza también en buena medida la seguridad de una aplicación, la mantenibilidad, la estructura y la organización.

Por otro lado, en relación a la ejecución de pruebas de software, se pudo constatar que un ciclo de pruebas de forma manual es un proceso largo y propenso a errores; sin embargo, la automatización del mismo, genera dentro de un equipo de desarrollo un ambiente de confianza soportado por los test, permite obtener retroalimentación de forma temprana y con frecuencia y de manera general, significa un importante punto de inflexión hacia la robustez, la seguridad y la calidad final del producto.

Al término del presente trabajo, luego de haber analizado la problemática que motivó el desarrollo de esta investigación, se arribaron a las siguientes conclusiones:

- A partir del análisis, el diseño y las tendencias tecnológicas descritas en el capítulo uno de la investigación, se logró implementar el servidor web de integración continua Sherlock, en su versión 0.3, el cual cuenta con una interfaz amigable, usable y funcional. Dicha implementación le proporciona a los desarrolladores reducción en los costos de tiempo de desarrollo y mantenimiento.

- Fueron incorporados al sistema siete módulos de prueba, para completar una suite de doce, aumentando de esta manera el espectro de pruebas realizadas a código fuente, contribuyendo a la generalización de la herramienta en el CEDIN.

- Con la implementación del software, fueron cubiertas en su mayoría las recomendaciones propuestas por la investigación anterior, factor que garantiza la continuidad y el soporte a la idea inicial de los creadores de la solución.

Recomendaciones

Una vez culminado el trabajo de investigación, quedan detalladas las siguientes recomendaciones para el soporte a la herramienta y la continua búsqueda de calidad sobre la aplicación:

- Implementar la funcionalidad de gestionar esclavos que permita a los administradores del sistema de manera sencilla y por medio de una interfaz visual administrar los esclavos sobre los cuales el Buildbot realiza el proceso de integración.

- Incluir un grupo de expertos en pruebas a la hora de seleccionar nuevos módulos, evitando el posible error de solapar funcionalidades y maximizar los resultados de los test, además que brinden información certera acerca de los gráficos que se puedan generar a partir de la ejecución de los módulos.

- La aplicación actualmente necesita almacenar información de los usuarios conectados. Esto incluye almacenar su contraseña, dato que se guarda en texto plano. Aunque Symfony2 y Doctrine2 implementan mecanismos que tratan este tipo de riesgo, es necesario tenerlo en cuenta y tomar medidas como la encriptación de la contraseña, para garantizar aún más la seguridad contra esta potencial vulnerabilidad.

- Implementar la funcionalidad de gestionar planificadores para diversificar y aumentar la cantidad y los momentos de ejecución del proceso de integración continua.

REFERENCIAS BIBLIOGRÁFICAS

1. Microsoft. [En línea] 2012. [Citado el: 19 de 06 de 2012.] http://msdn.microsoft.com/es-es/library/ms229028%28v=vs.80%29.aspx.

2. *Webopedia.* [En línea] [Citado el: 16 de Enero de 2012.] http://www.webopedia.com/TERM/A/API.html.

3. **Eguiluz, Javier.** *symfony.es Desarrollo web ágil con Symfony2.* [En línea] 29 de Diciembre de 2011. http://www.symfony.es/.

4. *Symfony.* [En línea] [Citado el: 17 de Enero de 2012.] http://www.symfony.com.

5. **Pressman, Roger S.** *Ingenieria del Software - Un Enfoque Practico 5b: Edicion (Spanish Edition).* s.l. : McGraw-Hill Companies, 2002. 8448132149.

6. **García, Félix.** *Proceso Software y Gestión del Conocimiento.* [En línea] 2008. [Citado el: 19 de Enero de 2012.] http://alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc-4a.pdf.

7. **Flanagan, David.** *JavaScript: The Definitive Guide.* s.l. : O'Reilly Media, Inc., 2006. 9780596101992.

8. **W3C.** World Wide Web Consortium. [En línea] 2012. [Citado el: 15 de 03 de 2012.] http://www.w3.org/standards/webdesign/htmlcss#whatcss.

9. ri5. [En línea] [Citado el: 12 de 03 de 2012.] http://www.ri5.com.ar/ayuda07.php.

10. Página Oficial de phpMyAdmin. [En línea] [Citado el: 15 de 03 de 2012.] http://www.phpmyadmin.net/home_page/index.php.

11. **Carrero, Angel.** *Programación en Castellano.* [En línea] [Citado el: 20 de Enero de 2012.] http://www.programacion.com/articulo/conceptos_basicos_de_orm_object_relational_mapping_349.

12. **Penadés, Patricio Letelier & Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*

13. *UNED.* [En línea] [Citado el: 25 de Enero de 2012.] http://www.ia.uned.es/ia/asignaturas/adms/GuiaDidADMS/node61.html.

14. **Pérez, Adisleidys Mirabal.** Herramienta de Apoyo a la Producción en el CEDIN. *Sitio virtual de la biblioteca de la UCI.* [En línea] 2011. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_04749_11.

15. **Bahit, Eugenia.** scribd.com. [En línea] 15 de Julio de 2011. [Citado el: 20 de abril de 2012.] http://es.scribd.com/doc/60975948/POO-y-MVC-en-PHP.

16. **Larman, Craig.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pearson Educacion S.A.; Edición: 2, 2010. 8420534382.

17. **Mora, Roberto Canales.** Arquitectura de software.Patrones de GRASP. [En línea] 2004. [Citado el: 25 de 04 de 2012.] file:///D:/Arquitectura%20de%20software/Patrones%20de%20GRASP/Patrones%20de%20GRASP.htm.

18. **Marca Huallpara Hugo Michael, Quisbert Limachi Nancy Susana.** virtual.usalesiana.edu.bo. [En línea] [Citado el: 12 de 04 de 2012.] virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

19. **Zarzuela, Gregorio Robles Martínez y Jorge Ferrer.** *Programación eXtrema y Software.* [Documento pdf] Madrid : s.n., 2002.

20. **ANOM.** Wikipedia. [En línea] 2012. [Citado el: 16 de 05 de 2012.] http://es.wikipedia.org/wiki/Host.

21. **Mitchell, Bradley.** Computing Wireless Networking. [En línea] 2012. [Citado el: 14 de 05 de 2012.] http://compnetworking.about.com/od/networkprotocols/g/bldef_http.htm.

22. PC Magazine Encyclopedia. [En línea] [Citado el: 14 de 05 de 2012.] http://www.pcmag.com/encyclopedia_term/0,1237,t=URL&i=53516,00.asp.

23. **Fogie, Seth.** *Cross Site Scripting Attacks: Xss Exploits and Defense.* 2007. 1-59749-154-3.

Bibliografía Consultada

- **Anom.** Gimpel Software. [En línea] 2011. [Citado el: 14 de 04 de 2012.] http://www.gimpel.com/html/products.htm.
- **Bergsten, Hans.***Java Server Pages.* s.l. : O'Reilly Media, 2000. 156592746X
- *Best Web Frameworks.* [En línea] [Citado el: 17 de Enero de 2012.] http://www.bestwebframeworks.com/compare-web-frameworks/php/
- *CakePHP.org.* [En línea] [Citado el: 17 de Enero de 2012.] http://book.cakephp.org/2.0/en/cakephp-overview.html
- *CodeIgniter.org.* [En línea] [Citado el: 17 de Enero de 2012.] http://codeigniter.com/user_guide/overview/features.html
- **Cortés, Oscar Hernando Guzmán.** WillyDev. [En línea] 20 de 04 de 2004. [Citado el: 10 de 05 de 2012.] http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf
- **Cozens, Simons.***Beginning Perl.* 1ra. s.l. : Wrox, 2000
- *DesarrolloWeb.com.* [En línea] [Citado el: 17 de Enero de 2012.] http://www.desarrolloweb.com/manuales/manual-codeigniter.html
- **Doria, Heidi González.***Las Métricas del Software y su Uso en la Región.* [En línea] 2001. [Citado el: 19 de Enero de 2012.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/portada.html
- Doxygen Web Page. [En línea] [Citado el: 15 de 03 de 2012.] http://www.stack.nl/~dimitri/doxygen/
- [En línea] [Citado el: 16 de Enero de 2012.] http://c2.com/cgi/wiki?ApplicationProgrammingInterface
- [En línea] [Citado el: 16 de Enero de 2012.] http://c2.com/cgi/wiki?ApiVsProtocol
- **Fogie, Seth.***Cross Site Scripting Attacks: Xss Exploits and Defense.* 2007. 1-59749-154-3.

- **Guardado, Iván.***web.Ontuts.* [En línea] 5 de Mayo de 2010. [Citado el: 20 de Enero de 2012.] http://web.ontuts.com/tutoriales/introduccion-a-object-relational-mapping-orm/
- **Hall, Marty.***Core Servlets and JavaServer Pages.* s.l. : Prentice Hall PTR, 2000. 0130893404
- **Krzysztof Cwalina, Brad Abrams.***Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries (2nd Edition).* s.l. : Addison-Wesley Professional, 2008. 0321545613
- **Lerman, Julia.***Programming Entity Framework.* s.l. : O'Reilly Media, 2009. 059652028X.
- *Mas Adelante.* [En línea] [Citado el: 16 de Enero de 2012.] http://www.masadelante.com/faqs/plug-in
- **Marcello Visconti, Hernán Astudillo.** Página de Inicio de Departamento de Informática. [En línea] [Citado el: 25 de 04 de 2012.] http://www.inf.utfsm.cl/
- **Ma. Argenis Gonzalez Castellanos y Wilson Rojas Pabón.***Comparación entre sistemas de gestión de bases de datos (SGBD) bajo licenciamiento libre y comercial.* [En línea] 2005. [Citado el: 25 de Enero de 2012.] http://www.ilustrados.com/documentos/sgbd.pdf
- **M, Johnny Zulca.***sitio Web mailxmail.* [En línea] 2 de Diciembre de 2008. [Citado el: 1 de Marzo de 2012.] http://www.mailxmail.com/curso-php-mysql-aplicaciones-web-1/programacion-cliente-servidor
- **Meloni, Julie.***PHP Essentials, 2nd Edition.* 2003 . 1931841349
- *Modelo Vista Controlador. Un patrón de arquitectura de software que separa los datos de una aplicación.* [En línea] [Citado el: 25 de Enero de 2012.] http://www.buenmaster.com/?a=536
- **Mullon, Robert.***sitio Web brighthub The History of ASP and ASP.NET.* [En línea] 2010. [Citado el: 20 de Enero de 2012.] http://www.brighthub.com/internet/web-development/articles/5009.aspx
- Página oficial de Findbugs. [En línea] [Citado el: 12 de 04 de 2012.] http://findbugs.sourceforge.net/findbugs2.html

- Página Oficial de Geany. [En línea] [Citado el: 14 de 03 de 2012.] http://www.geany.org/Main/About

  Página Oficial de jQuery. [En línea] [Citado el: 14 de 03 de 2012.] http://jquery.com/

- **Pizarro, Pablo.***Arquitectura de Software.* [En línea] 21 de Enero de 2007. [Citado el: 20 de Enero de 2012.] http://arquitectura-de-software.blogspot.com/2007/01/orm-object-relational-mapping-reloaded.html

- **Pugh, David H. Hovemeyer & William W.** Manual de Findbugs. [En línea] [Citado el: 12 de 04 de 2012.] http://findbugs.sourceforge.net/manual/index.html

- **Rodríguez, Ing Leover Armando González.***informatica-juridica.com.* [En línea] junio de 2011. [Citado el: 20 de Enero de 2012.] http://www.informaticajuridica.com/trabajos/Alternativas_para_el_desarrollo_de_Aplicaciones_Web.pdf

- segu-info.com.ar. [En línea] 06 de 10 de 2011. [Citado el: 14 de 05 de 2012.] http://blog.segu-info.com.ar/2011/10/prevenir-el-cross-site-request-forgery.html#axzz1wtuxMqXd

- *sitio Web WordPress.* [En línea] [Citado el: 16 de Enero de 2012.] http://codex.wordpress.org/Writing_a_Plugin

- *sitio Web EcuRed.* [En línea] 2011. [Citado el: 12 de Febrero de 2012.] http://www.ecured.cu/index.php/Pruebas_de_caja_negra

- *sitio Web de PHP.* [En línea] PHP Group. [Citado el: 15 de Enero de 2012.] http://www.php.net/

- *sitio Web de Anibal de la Torre Blog.* [En línea] 2012. [Citado el: 15 de Febrero de 2012.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

- *sitio Web de TechTerms.* [En línea] 15 de Enero de 2009. [Citado el: 15 de Enero de 2012.] http://www.techterms.com/definition/snippet

- *sitio oficial de Snippets.* [En línea] [Citado el: 16 de Enero de 2012.] http://www.snippets.org/

- Sitio Oficial de Python. [En línea] [Citado el: 15 de 05 de 2012.] http://www.python.org/

- *Social Compare.* [En línea] [Citado el: 17 de Enero de 2012.] http://socialcompare.com/en/comparison/php-frameworks-comparison
- Sourceforge.net. [En línea] [Citado el: 12 de 04 de 2012.] http://cppcheck.sourceforge.net/
- *StackOverflow.* [En línea] [Citado el: 16 de Enero de 2012.] http://stackoverflow.com/questions/1636259/difference-between-web-application-framework-and-a-content-management-system
- *Slideshare.com.* [En línea] Mobicules Technologies, 4 de Junio de 2010. [Citado el: 17 de Enero de 2012.]http://www.slideshare.net/mobicules/symfony-vs-ci
- *Slideshare.* [En línea] [Citado el: 16 de Enero de 2012.] http://www.slideshare.net/raulfraile/symfony2-framework-para-php5
- *Twig.* [En línea] [Citado el: 20 de Enero de 2012.] http://twig.sensiolabs.org/
- *Webopedia.* [En línea] [Citado el: 16 de Enero de 2012.] http://www.webopedia.com/TERM/A/API.html
- Wikipedia. [En línea] 09 de 04 de 2012. [Citado el: 14 de 05 de 2012.] http://es.wikipedia.org/wiki/Cross_Site_Request_Forgery
- Worl Wide Web Consortium. [En línea] 19 de 01 de 2005. [Citado el: 15 de 05 de 2012.] http://www.w3.org/DOM/
- *Zend Framework Des.com.* [En línea] 16 de Noviembre de 2011. [Citado el: 17 de Enero de 2012.] http://manual.zfdes.com/es/introduction.overview.html
- *Zend Framework.* [En línea] [Citado el: 17 de Enero de 2012.] http://framework.zend.com/manual/manual

Anexos

Anexo 1

Listado de herramientas de prueba analizadas.

Generales:

doxygen, bugzilla, gonzui, buildbot, bcpp, headache, qmtest, bitten, google-performance tools, ohcount, splint, munin, ganglia, funkload, acovea,ccmalloc, valgrind, python-profiler, hardening-wrapper, icheck, nana, ncc, pentium-builder, synopsis, debnest, debsigs,indent, bcpp, similarity-tester, uncrustify, universalindentgui, hyperestraier, simian, checkstyle, grinder, serenity, sonar, rcov, flog, findbugs, pmd, yasca, javascriptlint, pixy, phplint, cppcheck, sparse, uno, blast

Herramientas para C++:

astrée, BLAST, cppcheck, cpplint, clang, coccinelle, flawfinder, frama-C, flexeLint, Green Hills Software, doubleCheck, intel, lint, LDRA Testbed, Monoidics INFER, Parasoft C/C++ test, PC-Lint, polyspace, PVS-Studio, QA-C, Red Lizard's, goanna, sparse, splint, Closure Compiler, JSLint, objective, clang.

Herramientas para Perl:

Perl :: Critic, PerlTidy, Pychecker, Pylint

Anexo 2

Iteración: Primera

Caso de prueba de funcionalidad: Eliminar proyecto.

Descripción general: el caso de uso se inicia cuando el usuario desea eliminar un proyecto del sistema.

Condiciones de ejecución: *el usuario debe estar autenticado y ser el creador del proyecto o poseer privilegios de administración.*

Historias de usuario validadas en el caso de prueba: *1, 2, 3.*

Escenario	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1 El usuario selecciona un proyecto y lo elimina.	**1.1** Situarse en la opción de "Proyecto" del menú principal **1.2** Al desplegarse el menú escoger la opción de "Mostrar listado". **1.3** Localizar el proyecto deseado y en la sección de “Acciones” del listado pulsar sobre el icono de eliminar.	**N/A**	El proyecto es eliminado del sistema.	**1.1** El controlador *delProjectAction*() es invocado. **1.2** Se ejecuta la función *buildbot_delete_project*() la cual parsea el fichero master.py y elimina las referencias del proyecto en el archivo de configuración. **1.3** Se elimina la instancia del proyecto de la base de datos. **1.4** Se renderiza nuevamente la vista del listado de proyectos notificando al usuario del éxito de la operación.

Tabla 51. Diseño de caso de prueba/1ra iteración/Eliminar proyecto

Anexo 3

Iteración: Segunda

Caso de prueba de funcionalidad: Listar proyectos mediante rango de fechas.

Descripción general: el caso de uso se inicia cuando el usuario necesita visualizar reportes. Para ello primeramente es desplegado un listado con los proyectos a los cuales se les realizaron pruebas. Este listado puede se filtrado mediante un rango de fechas.

Condiciones de ejecución: *deben existir proyectos en el sistema y reportes de prueba creados.*

Historias de usuario validadas en el caso de prueba: *6.*

Escenario	Descripción	Variable1	Variable2	Respuesta del sistema	Flujo central
EC 1 Listar proyectos seleccionando	**1.1** Pulsar en el menú principal la opción de "Reportes" **1.2** En el panel de la derecha	**V**	**V**	Es desplegado en el panel	**1.1** Los valores de las fechas son enviados al

rango correcto de fechas.	de la ventana desplegar la lista de filtrado de proyecto y seleccionar "Rango de Fechas". **1.3** Seleccionar un rango de fechas válido y pulsar en el botón "Buscar".	**27/04/2012**	**30/04/2012**	de la derecha un listado con los proyectos a los cuales se les generó un reporte de pruebas dentro del intervalo de fecha seleccionado.	controlador el cual crea un objeto *Date*() con cada fecha. **1.2** Se ejecuta la consulta *filtrarReportDateR*() sobre el modelo, pasándole a la función como parámetros $fechaInit y $fechaFin para devolver en un arreglo los reportes de prueba generados en el intervalo de fechas escogido. **1.3** El controlador envía el arreglo $listaReporte a la vista, la cual itera sobre el mismo y va mostrando la información correspondiente en una tabla.
EC 2 Listar proyectos seleccionando rango incorrecto de fechas.	**2.1** Pulsar en el menú principal la opción de "Reportes". **2.2** En el panel de la derecha de la ventana desplegar la lista de filtrado de proyecto y seleccionar "Rango de Fechas". **2.3** Seleccionar una fecha en el campo "Fecha Inicial". **2.4** Seleccionar una fecha en el campo "Fecha Final", la cual debe ser inferior a la fecha inicial. **2.5** Pulsar el botón "Buscar".	**I** **30/04/2012**	**V** **27/--/2011**	Se muestra un mensaje de error informando que se debe seleccionar un rango válido de fechas.	**2.1** Se obtiene el valor del año de la fecha inicial a través del campo "Field0". **2.2** Se repite la operación para la fecha final introducida en el campo "Field3". **2.3** Son comparados los dos valores y si "Field0" es mayor que "Field3", se lanza un mensaje de error.
EC 3 Listar	**3.1** Pulsar en el menú principal	**N/A**	**N/A**	Se muestra	**3.1** Se evalúan

proyectos dejando en blanco alguna de las dos fechas.	la opción de "Reportes". **3.2** En el panel de la derecha de la ventana desplegar la lista de filtrado de proyecto y seleccionar "Rango de Fechas". **3.3** Seleccionar un intervalo de fecha dejando en blanco alguna de las dos fechas.			un mensaje de error informando los campos de "Fecha Inicial" y "Fecha Final" son obligatorios.	todos los elementos input correspondientes a las fechas. De haber alguno cuyo “value” sea igual a "", lanzar un mensaje de error.

Tabla 52. Diseño de caso de prueba/2da iteración/Listar proyecto mediante un rango de fechas

Anexo 4

Iteración: Segunda

Caso de prueba: Mostrar información de los módulos de prueba.

Descripción general: el caso de uso se inicia cuando el usuario desea revisar los módulos de prueba que se están ejecutando sobre el proyecto.

Condiciones de ejecución: *el usuario debe estar autenticado.*

Historias de usuario validadas en el caso de prueba: *5.*

Escenario	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1 Mostrar información sobre los módulos integrados a Sherlock de	**1.1** Pulsar en el menú principal la opción de Módulos".	**N/A**	El sistema mostrará en un panel de tipo "*slide*", uno a uno los proyectos que están integrados a Sherlock con una breve descripción de los	**1.1** La petición dispara al controlador *showModulesAction*(), el cual ejecuta la función estática *modulesLoad*() para cargar los módulos.**1.2** Una vez cargados, el objeto *ObjectRepository* se encarga de almacenar todos los módulos

forma correcta.			mismos.	en un arreglo. **1.3** El arreglo es enviado a la vista. Una vez en la vista se itera sobre el arreglo y se muestra la información correspondiente de cada módulo al usuario.
EC 2 Mostrar información sobre los módulos integrados a Sherlock y una alerta sobre aquellos módulos que no estén correctamente instalados.	**2.1.** Pulsar en el menú principal la opción de Módulos".	**N/A**	El sistema mostrará en un panel de tipo "*slide*", uno a uno los proyectos que están integrados a Sherlock con una breve descripción de los mismos. Aquellos proyectos, que no estén totalmente instalados serán marcados para evidenciar que no tienen todas sus dependencias instaladas.	**1.1** La petición dispara al controlador *showModulesAction*(), el cual ejecuta la función estática *modulesLoad*() para cargar los módulos. **1.2** Una vez cargados, el objeto *ObjectRepository* se encarga de almacenar todos los módulos en un arreglo. **1.3** En el controlador se itera por cada módulo de este arreglo ejecutando sobre cada uno la función *checkModuleDeps*() que verifica que estén gestionadas de manera correcta las dependencias de un módulo. **1.4** El resultado de está ejecución (Verdadero o Falso), se almacenará en un arreglo asociativo de la forma ('nombreModulo'=>true|false) el cual es enviado a la vista junto con el arreglo de módulos. En la vista se itera sobre el arreglo de módulos y se muestra la información correspondiente de cada módulo al usuario. **1.5** Se le informará al usuario si hay algún módulo que no tiene todas sus dependencias correctamente gestionadas.

Tabla 53. Diseño de caso de prueba/3ra iteración/Mostrar información de los módulos de prueba

Glosario de términos

[1]**SQL:** el lenguaje de consulta estructurado o SQL (por sus siglas en inglés *structured query language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas. Permite efectuar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre ella.

[2]***Host*:** el término *host* es empleado en informática para referirse a las computadoras conectadas a una red. Estas computadoras proveen y utilizan servicios como transferencia de archivos, conexión remota, servidores de base de datos, servidores web, entre otros. De forma general un anfitrión es todo equipo informático que posee una dirección IP y se encuentra interconectado con uno o más equipos. (20)

[3]**HTTP:** son las siglas que identifican al término *Hypertext Transfer Protocol* o Protocolo de Transferencia de Hipertexto. Este protocolo provee un estándar para la comunicación entre los navegadores web y los servidores. (21)

[4]**URL:** URL significa *Uniform Resource Locator.* Este término es empleado en informática para definir la ruta hacia un fichero ubicado en algún servidor web, servidor ftp, servidor de correo y otros. Es importante destacar que todos los archivos disponibles en la red, poseen una única dirección URL, la cual los identifica unívocamente. (22)

[5]**CSRF:** El CSRF (del inglés *Cross-Site Request Forgery* o falsificación de petición en sitios cruzados), es un tipo de *ataque* en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía. Al contrario del XSS que explota la confianza del usuario en un sitio en particular, el CSRF explota la confianza del sitio en un usuario en particular.

[6]**ORM:** las siglas ORM (*Object Relational Mapping*), responden a la técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional empleado en el desarrollo de nuestra aplicación.

[7]**XSS:** XSS, del inglés *Cross-Site Scripting* es un tipo de vulnerabilidad informática típica de las aplicaciones web, que permite a una tercera parte inyectar casual o intencionalmente código JavaScript en

páginas web vistas por el usuario pudiendo afectar el funcionamiento de dicho sistema. Estos errores pueden ser encontrados en cualquier aplicación que tenga como objetivo final presentar información en un navegador web. (23)

[8]**DRY:** DRY son las siglas que identifican el término *Don't Repeat Yourself*. Es considerado uno de los principios fundamentales de la programación. Formulado por Andy Hunt y Dave Thomas en su libro "The Pragmatic Programmer" en 1999. Soporta varias de las buenas prácticas de desarrollo de software y patrones de diseño. Este principio garantiza un código limpio, libre de repetición innecesaria mediante el reconocimiento de duplicación de código y métodos apropiados de abstracción.

[9]**KISS:** Kiss es el acrónimo para el principio de innovación definido por Kelly Johnson "*Keep It Small and Simple*", "Mantenlo Sencillo y Simple". El principio KISS recomienda el desarrollo de software implementando soluciones sencillas, comprensibles y con errores de fácil detección y corrección, rechazando lo entreverado e innecesario en el desarrollo de sistemas.

[10]**DOM:** según el W3C el DOM (*Document Object Model*) o Modelo de Objetos para Documentos, es una API que proporciona un conjunto estándar de objetos para representar documentos HTML y XML. Esta interfaz además permite combinar dichos objetos, accederlos y manipularlos.

[12]**LDAP:** LDAP son las siglas de *Lightweight Directory Access Protocol* (en español Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos a la que pueden realizarse consultas.