



UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS  
FACULTAD #4

# HERRAMIENTA INFORMÁTICA PARA EL MONITOREO DE ERRORES DE LAS APLICACIONES WEB EN EL CENTRO FORTES

*TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE  
INGENIERO EN CIENCIAS INFORMÁTICAS.*

AUTOR: Danis Carlos Chaviano Jiménez

TUTOR: Ing. Arcel Labrada Batista  
COTUTOR: MSc. Roberto López Desagües

La Habana, Junio de 2014  
“Año 56 de la Revolución”

## **DECLARACIÓN DE AUTORÍA**

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del 2014.

Autor:

---

Danis Carlos Chaviano Jiménez

Tutor:

---

Ing. Arcel Labrada Batista

Cotutor:

---

MSc. Roberto López Desagües



*"Solo podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer."*

*Alan Turing*

**A** todos mis compañeros en estos años de tristezas, alegrías y sacrificios, a los que de una forma u otra han hecho posible este momento. A nuestro Comandante en Jefe Fidel Castro Ruz por crear una magnífica Universidad del Futuro. A la UCI Por enseñarme muchas verdades de la vida y enseñarme a vencer cualquier reto, por formarme como un Ingeniero capaz de afrontar cualquier tarea.

A mis amigos, por convertirse en mi familia cuando me encontraba lejos de la mía, por compartir conmigo en momentos de alegría, trabajo o sufrimiento.

A Geovelsi, Roberto, Edgar, Jeiser, Lianne, Yordanis, etc. por formar parte de mi familia.

A todos los que de una forma u otra me ayudaron en la realización de este trabajo, en especial a Dani por toda su ayuda.

A mi familia por apoyarme siempre y creer en mí.

A mis tutores por toda su ayuda brindada, a pesar de su trabajo o su tesis de doctorado siempre estuvieron presente.

A todos los profesores que he tenido en mi trayectoria como estudiante, cada uno de ellos me ha enseñado y han hecho posible este sueño.

Al Tuti y a mi prima Alicita por todo su apoyo y ayuda que me dieron en todo momento.

A todos los de Ranchuelo que están aquí en la universidad luchando por cumplir su sueño, por su ayuda siempre que hacía falta, en especial a Isenith y mi primo Lijandy.

**A** mi mamá por ser todo para mí, por estar ahí en todo momento. Por su apoyo, dedicación y cariño que me ha dado. Por ayudarme a convertirme en el hombre que soy. Por ser mi ejemplo en la vida. Por indicarme el camino bueno, pero dejarme elegir por mí mismo. Por hacerme sentir orgulloso cada vez que hablo de ella.

*A mis hermanos por darme todo su apoyo y ayuda incondicional.*

*A mis abuelos maternos, que aunque ya no están con nosotros siempre tendrán un lugarcito en mi corazón.*

*A mis sobrinitas que son mi vida.*

*A mi padrastro Jorgito que ha sido una persona importante en toda mi formación.*

*A todo el que no pensó que fuera posible, le dedico este trabajo para que vean que sí se puede.*

## RESUMEN

Las aplicaciones web han alcanzado gran popularidad e importancia en los últimos años. Permiten el intercambio y consulta de información de los usuarios. La calidad con la que se brinda un servicio, es un factor clave mediante el cual una empresa puede ganar o perder clientes. La importancia de detectar errores a tiempo puede prevenir pérdida o adulteración de datos, el uso excesivo de recursos, incluso un mal funcionamiento de componentes de la aplicación.

La investigación que se presenta, aborda los principales aspectos que se desarrollaron con el fin de realizar la solución informática de una herramienta para el monitoreo de los errores en las aplicaciones web. Se realiza el estudio de las principales herramientas y tecnologías utilizadas en el proceso de creación de la solución. El documento de esta investigación, deja constancia de la metodología empleada la cual permitió cumplir el objetivo general propuesto y satisfacer las necesidades del cliente. Por último, se especifican las pruebas a las que fue sometida la solución elaborada, quedando demostrado que la misma cumple con la calidad y funcionalidades requeridas por el cliente.

Por el resultado alcanzado, la investigación se considera un aporte a la ciencia, debido a que auxilia a los desarrolladores que realizan el monitoreo de las aplicaciones web para el control de la calidad de las mismas.

**Palabras claves:** aplicaciones web, errores HTTP, monitoreo, servicio REST

# ÍNDICE

<b>INTRODUCCIÓN .....</b>	<b>1</b>
<b>CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA .....</b>	<b>5</b>
1.1 CONCEPTOS ASOCIADOS AL DOMINIO DEL PROBLEMA .....	5
1.2 ESTUDIO COMPARATIVO DE HERRAMIENTAS CON SOLUCIONES SIMILARES .....	7
1.3 METODOLOGÍA DE DESARROLLO DE <i>SOFTWARE</i> .....	9
1.4 HERRAMIENTAS Y TECNOLOGÍAS .....	15
1.5 SERVICIOS WEB .....	23
1.6 CONCLUSIONES DEL CAPÍTULO.....	24
<b>CAPÍTULO 2 ANÁLISIS Y DISEÑO.....</b>	<b>26</b>
2.1 MODELO DE DOMINIO.....	26
2.2 PROPUESTA DE SOLUCIÓN .....	27
2.3 PERSONAS RELACIONADAS CON EL SISTEMA .....	29
2.4 HISTORIAS DE USUARIOS PLANIFICADAS .....	29
2.5 FASE DE PLANIFICACIÓN .....	33
2.6 TARJETAS CRC .....	36
2.7 MODELO DE DATOS .....	38
2.8 CONCLUSIONES DEL CAPÍTULO.....	39
<b>CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS .....</b>	<b>40</b>
3.1 PATRONES DE DISEÑO .....	40
3.2 PATRONES ARQUITECTÓNICOS.....	44
3.3 USO DE COMPONENTES DE TERCEROS .....	45
3.4 DESARROLLO DE LAS ITERACIONES .....	46
3.5 PRUEBAS DE <i>SOFTWARE</i> .....	51
3.6 RESULTADOS DE LAS PRUEBAS .....	57
3.7 RESULTADOS OBTENIDOS .....	57
3.8 CONCLUSIONES DEL CAPÍTULO.....	57
<b>CONCLUSIONES GENERALES .....</b>	<b>58</b>
<b>RECOMENDACIONES .....</b>	<b>59</b>
<b>REFERENCIAS BIBLIOGRÁFICAS .....</b>	<b>60</b>
<b>BIBLIOGRAFÍA .....</b>	<b>63</b>
<b>ANEXOS .....</b>	<b>67</b>
ANEXO 1: PROTOTIPOS DE INTERFAZ DE USUARIO.....	67
ANEXO 2: GUÍA DE OBSERVACIÓN .....	70

## INTRODUCCIÓN

Las aplicaciones web han alcanzado gran popularidad e importancia en los últimos años. Permiten la consulta e intercambio de información de los usuarios, como también alta disponibilidad a partir de la posibilidad de ofrecer servicios desde múltiples localizaciones, lo cual asegura su continuidad. Puede contener además, elementos que permiten una comunicación activa entre el usuario y la información.

Luego del despliegue de las aplicaciones web, pueden presentarse errores o vulnerabilidades que no son depurados en el proceso de desarrollo. Corregirlos conlleva a una pérdida de tiempo, pues con frecuencia se desconoce cuándo ocurrió la excepción. Esto se debe entre otros factores a que no se poseen herramientas que permitan un monitoreo constante de los mismos. La importancia de detectar estos errores, previene la pérdida o adulteración de datos, el uso excesivo de recursos, mal funcionamiento de sus componentes y la aparición de errores inesperados provocados por ataques de denegación de servicio a través de *software* malicioso.

Es importante además, porque el uso de la Web cada día se extiende de forma vertiginosa a la mayoría de los ámbitos de la sociedad y la vida cotidiana. Es cada vez más utilizada para la información, el comercio y en especial en la educación a distancia. En este último aspecto su uso es dirigido fundamentalmente a apoyar el Proceso de Enseñanza y Aprendizaje (PEA), en el sentido de administrar, distribuir y controlar los procesos en línea, además de proporcionar funciones administrativas y de seguimiento necesarias para posibilitar y controlar el acceso a los contenidos, implementar recursos de comunicación y llevar a cabo el seguimiento de quienes utilizan la herramienta.

Cuba no está exenta del empleo de estas tecnologías para el desarrollo de *software* educativo, por lo que desarrolla los hiperentornos educativos o de aprendizaje como concepción pedagógica. La Universidad de las Ciencias Informáticas (UCI), específicamente el Centro de Tecnologías para la Formación (FORTES) perteneciente a la Facultad 4, desarrolla tecnologías que permiten ofrecer servicios y productos para la implementación de soluciones de formación, tanto para los niveles curriculares de la enseñanza inicial, media o preuniversitaria, así como tecnologías de formación a distancia y semi-presencial, utilizados mayormente en el nivel superior y de postgrado.

Muchos de estos productos se desarrollan sobre un marco de trabajo (*framework*) o un Sistema de Administración de Contenido (CMS por sus siglas del inglés *Content Management System*). En ambos casos se lleva un control de los errores, mediante eventos que se generan guardando la información del error en archivos de registro de sucesos HTTP (Protocolo de Transferencia de Hipertexto o *Hypertext Transfer Protocol*), o también conocidos como archivos log. En estos



archivos se guarda toda la información referente a los eventos generados por las aplicaciones web, lo cual hace más engorroso el trabajo.

La realidad constata problemas de organización en el proceso de almacenamiento en los archivos de tipo log. De una parte no se puede mostrar, de forma clasificada, los errores que son guardados por cada evento en los archivos log de las aplicaciones web. De otra, se necesita de la presencia de los administradores en el servidor para poder acceder a ellos. Además, los desarrolladores para poder llevar el control de los errores, necesitan otras informaciones que faciliten el control y servir de ayuda para descartar cualquier problema de compatibilidad que exista en la aplicación, sin embargo, tampoco son guardadas en este tipo de archivo. Se destacan los datos del usuario que generó el error, navegador web, sistema operativo, entre otros.

En el centro FORTES, los mecanismos existentes no son suficientes para lograr agilidad por el equipo de desarrollo en el entendimiento de las excepciones lanzadas por las distintas aplicaciones, pues al no ser guardadas en las bases de datos no se cuenta con un registro histórico de los mismos. Otra deficiencia encontrada, es que los reportes enviados se encuentran en distintos formatos lo que dificulta el proceso de unificación del contenido de los errores.

Las insuficiencias mencionadas indican que se está en presencia del siguiente **Problema a resolver**: ¿Cómo mostrar a los desarrolladores los errores que se generan en las aplicaciones web del centro FORTES para facilitar su solución?

Teniendo como **objetivo general**: Desarrollar una herramienta informática que permita mostrar de forma automática a los desarrolladores del centro FORTES los errores que generan las aplicaciones web y facilitar así su solución.

La parte de la ciencia que será **objeto de estudio** es los procesos de monitorización de las aplicaciones web.

Definiéndose como **campo de acción** las herramientas para el monitoreo y gestión de errores en las aplicaciones web desarrolladas en el centro FORTES.

Teniendo como **objetivos específicos**:

- Fundamentar la teoría y metodología que soporta la investigación.
- Realizar el análisis y el diseño de la solución de *software* propuesta.
- Implementar la solución propuesta para el monitoreo de errores en las aplicaciones web.

Las insuficiencias señaladas en la monitorización de errores en las aplicaciones web del centro FORTES, indican una **idea a defender** y no una hipótesis a demostrar, que parte de la idea de que si se desarrolla una herramienta capaz de centralizar la información referente a los errores en las aplicaciones web desarrolladas en el centro FORTES, permitirá a los desarrolladores tener un mejor control de los mismos.

**Posibles resultados:**

- Documentación de acuerdo a la metodología de desarrollo de *software* seleccionada.
- Una herramienta que sea configurable, con el fin de realizar la vigilancia, visualización y seguimiento de errores en las aplicaciones web, que permita generar reportes estadísticos sobre los mismos y facilitar la toma de decisiones al respecto.

**Tareas a cumplir:**

1. Elaboración de los diseños teóricos y metodológicos de la investigación.
2. Realización de análisis críticos y valorativos de los sistemas informáticos que realizan tareas de monitorización web.
3. Evaluación de las herramientas o componentes que se utilizarán para el análisis y diseño del sistema.
4. Definición de la arquitectura para el desarrollo de la herramienta para el monitoreo de aplicaciones web.
5. Identificación y descripción de los aspectos funcionales del *software*.
6. Especificación de las estrategias de codificación, los estándares, estilos y métodos de validación a utilizar en el proceso de desarrollo.
7. Implementación de las funcionalidades definidas en los aspectos funcionales del *software*.

Los **métodos de investigación** que soportan la investigación son la combinación dialéctica de **métodos teóricos** y **empíricos**. Entre los primeros se emplearon el Histórico-Lógico, para la realización del estudio del estado del arte de los estándares y sistemas que presentan soluciones similares, así como la evolución y desarrollo que han experimentado, las metodologías de desarrollo de *software*, marcos de trabajo, lenguajes y herramientas de desarrollo necesarias para cumplir el objetivo general propuesto.

Analítico-Sintético, para examinar las partes fundamentales relacionadas con el objeto de estudio, comprender su funcionamiento y complementarlo con su utilización en el campo de acción de la investigación.

**Métodos empíricos:**

La observación: se empleó para determinar la evolución de los procesos, en específico en las validaciones de las funcionalidades de la solución. Se apoyó en las técnicas de recopilación de información: cuestionario y entrevista de tipo no estructurada.

Además de los métodos científicos anteriormente expuestos también se utilizó:

El criterio de expertos: permitió obtener opiniones entre diferentes expertos para verificar que los reportes generados por la herramienta de monitoreo de errores, basado en los reportes de las

excepciones lanzadas por las Aplicaciones del centro FORTES permiten apoyar la toma de decisiones.

### **Estructuración del trabajo de diploma**

El primer capítulo describe la fundamentación teórica de dicha investigación, el cual incluye un estudio del estado del arte del tema. Se explican y justifican las tendencias, tecnologías y herramientas en las que se apoya la solución al problema. Posteriormente se incluye un capítulo que describe el proceso de análisis y diseño, el cual comienza con la descripción, priorización y planificación de las historias de usuario que describen las funcionalidades a implementar, pasando luego a detallar la arquitectura de la solución y sus principales características. Por último aparece un tercer capítulo donde se describen las fases de implementación y pruebas, se comienza con la descripción de los componentes de terceros utilizados en la solución. Se implementan todas las funcionalidades identificadas, logrando un sistema que satisface las principales necesidades del cliente. Se detallan también las pruebas que se le realizaron al sistema, con el objetivo de asegurar la calidad y eficiencia de la solución. Seguidamente se encuentran las referencias bibliográficas y la bibliografía que están formadas por documentos y artículos de interés para los autores de la investigación y se concluye con los Anexos donde se agregan algunos artefactos generados en el transcurso del desarrollo de la solución.

## Capítulo 1 FUNDAMENTACIÓN TEÓRICA

Los sistemas de gestión de aprendizaje amplían el espacio y el tiempo de la clase ya que facilitan el acceso a lecturas, ejercicios y material educativo, el envío de tareas y trabajos y la comunicación sincrónica o asincrónica entre estudiantes y profesores. En varias ocasiones sucede que posterior a su despliegue, son detectados errores que dificultan su solución y trasciende en otras consecuencias relacionadas con el plan de negocio de un proyecto.

Desarrollar una herramienta informática para su solución significa, ante todo, realizar un levantamiento de experiencias anteriores tanto fuera, como dentro de Cuba, así como examinar los elementos que permitan su desarrollo. Es el objetivo que se propone este capítulo, dividido en cinco epígrafes y sub-epígrafes.

### 1.1 Conceptos asociados al dominio del problema

Los conceptos asociados al dominio del problema esclarecen el significado que para la investigación que se realiza tienen todas las categorías y términos fundamentales empleados en el planteamiento del problema y el objetivo general. Su definición conceptual se corresponde con la idea que sustenta el investigador. En este sentido, se puntualiza el significado del término Tecnologías de la Información y las Comunicaciones (TIC), *software* educativo, hiperentornos de aprendizaje y errores HTTP.

#### 1.1.1 Tecnologías de la Información y las Comunicaciones

A partir de la bibliografía consultada se estudiaron varios conceptos sobre las TIC, uno de los más completo es el del autor A. M. Cabrera que define las TIC como "*...el conjunto de procesos y productos derivados de las nuevas herramientas (hardware y software), soportes de la información y canales de comunicación relacionados con el almacenamiento, procesamiento y transmisión digitalizados de los datos*" (1).

Del estudio realizado por Cabrera, sobre las características de las TIC, se resumen las siguientes:

- Inmaterialidad, ya que su materia prima es la información, e información en múltiples códigos y formas: visuales, auditivas, audiovisuales, textuales y de datos.
- Interconexión, aunque suelen presentarse de forma independiente, ofrece grandes posibilidades para que puedan combinarse y ampliar de esta forma sus potencialidades y extensiones.
- Interactividad, es una de las características que le permiten adquirir un sentido pleno en el terreno de la formación, permite una interacción sujeto-máquina y la adaptación de esta a las características educativas y cognitivas de la persona, facilitando de esta forma que los sujetos no sean meros receptores pasivos de información sino procesadores activos y conscientes de la misma.

- La instantaneidad, facilita la rapidez al acceso e intercambio de información, rompiendo las barreras espacio temporales que han influido durante bastante tiempo en la organización de actividades formativas.
- Su mayor influencia sobre los procesos que sobre los productos.
- La creación de nuevos lenguajes expresivos que permiten nuevas realidades expresivas como es el caso de los multimedia e hipertextos, que al mismo tiempo llevarán a la necesidad de adquirir nuevos dominios alfabéticos y la posibilidad de la interconexión de las mismas.
- La tendencia progresiva a la automatización, es decir, a la realización de sus actividades controladas desde dentro del propio sistema (1).

### 1.1.2 Software educativo

Varios autores hacen alusión al tema del software educativo. No obstante se asume el definido por Sonia Morejón que lo considera como: “...cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar – aprender y administrar” (2).

De igual forma, Sonia identifica las características del software educativo, las que se pueden resumir en la idea de que propicia la creación de un contexto adecuado para la construcción y transmisión de conocimiento en el momento que se integran en el proceso educativo propicio. Su evolución a través de la historia ha aportado cada vez más prestaciones y facilidades al proceso educativo.

### 1.1.3 Hiperentornos de aprendizaje

El método de síntesis permitió puntualizar el término hiperentornos de aprendizaje, definido en estos estudios como, sistemas informáticos basados en tecnologías hipermedia y que contienen una mezcla de elementos representativos de diversas tipologías de software educativo. Es una “mezcla de tutoriales, entrenadores simuladores, juegos y evaluadores o como: tutoriales, tutoriales inteligentes, sistemas hipermedias, simulaciones y micro-mundos” (2).

### 1.1.4 Errores en aplicaciones web

Según el estudio bibliográfico realizado en internet<sup>1</sup> sobre el protocolo de transferencia de hipertexto, o HTTP, se define que es el protocolo usado en cada transacción de la *World Wide Web* (www) y administrado por el Consorcio *World Wide Web Consortium* (W3C). Uno de los aspectos fundamentales del HTTP es el permitir a los navegadores obtener información en un formato simple (HTML, siglas del inglés *HyperText Markup Language*) de un servidor. Sin embargo, para poder desplegar esta información, el navegador o el cliente necesitan saber qué

---

<sup>1</sup> <http://www.hasheado.com/>

tipo de información es la que va a recibir, si una imagen, un texto, un documento de alguna aplicación, etc. También necesita saber si va a recibir la información, o si hubo algún error durante la realización del proceso. Para esto, el protocolo HTTP cuenta con ciertos códigos de respuesta estándar. Hay varios tipos de respuesta:

- *“Un código de respuesta 1xx Informativa. Se recibe la petición y se continúa con el proceso. Los códigos en este rango indican respuestas provisionales. Los servidores web no deben enviar mensajes 1xx al cliente HTTP excepto bajo condiciones experimentales.*
- *Un código de respuesta 2xx Éxito. Esta clase de códigos indican que la petición del cliente fue recibida, entendida, aceptada y procesada exitosamente.*
- *Un código de respuesta 3xx Redireccionamiento<sup>2</sup>. Para estos códigos el cliente debe realizar acciones adicionales para completar la petición. La acción requerida debe ser portada por el agente de usuario sin la interacción del usuario si y solo si el método usado en la segunda petición es de tipo GET o HEAD. El agente de usuario no debería redireccionar automáticamente más de 5 veces, sino se considera un bucle infinito.*
- *Un código de respuesta 4xx Error en el Cliente. Indica que hubo un error del lado del cliente.*
- *Un código de respuesta 5xx Errores de Servidor. Indica que el error fue del lado del servidor”*  
(3).

## 1.2 Estudio comparativo de herramientas con soluciones similares

Luego de un estudio parcial de las herramientas usadas en internet para conocer el tráfico de una página y que una empresa pueda mediante estas, llevar las tareas de monitoreo sobre los mismos, a continuación se presenta un análisis de las herramientas más usadas para realizar este proceso<sup>3</sup>:

### 1.2.1 Compete

Con el servicio de [www.compete.com](http://www.compete.com) se puede realizar un análisis exhaustivo de la competencia. Con esta herramienta se puede encontrar puntualmente el tráfico de la página de una empresa y el de los competidores para realizar exhaustivas comparaciones. Compete toma en cuenta las estadísticas del tráfico en la red para realizar los análisis de forma gráfica mostrando visualmente los datos.

Este servicio se considera una buena referencia para el mercadeo en internet, ya que brinda un análisis de los mercados principales, palabras claves y los portales en la web. Entre los análisis

---

<sup>2</sup> Redireccionar: acceder a una determinada dirección mediante otra.

<http://tecnologia.glosario.net/terminos-viricos/redireccionar-9828.html>

<sup>3</sup> Para mayor información consultar el “AlmacenPlantillasWeb”, en

<http://almacenplantillasweb.es/herramientas/herramientas-seo/herramientas-para-analisis-del-trafico-de-una-web/>.

que realiza se encuentra: hacer una comparación numérica de los sitios, análisis del comportamiento y mantener un seguimiento.

Compete ofrece además una gráfica de análisis de páginas web con mayor índice de visitas. Se puede observar en una gráfica por periodo de tiempo y en el tiempo total que los usuarios permanecieron durante su visita (4).

### 1.2.2 Alexa

Alexa es un medidor que provee información histórica del tráfico de una web o la cantidad de visitas que recibe. Presenta estadísticas de qué países generan más tráfico y de qué países provienen los usuarios de ese sitio web.

La información que presenta Alexa acerca de una web son: la imagen de la portada de un sitio web, el rango de tráfico de la página, páginas que visitan las mismas personas que acceden a la página actual, velocidad del servidor, etc.

Alexa también calcula el tráfico de los sitios, los clasifica en atención a su importancia y determina su posición en el ranking mundial (5).

### 1.2.3 Site24x7

Site24x7 es una herramienta para monitorear sitios web, pero además ofrece otras ventajas, por ejemplo:

- Es una herramienta totalmente gratis.
- Monitorea transacciones de una web, almacenando la secuencia de peticiones http (tanto métodos post como get).
- Monitorea si se ha sido víctima de un ataque de desfiguración (ataque para desfigurar o dañar las páginas del sitio).
- Monitoreo continuo en intervalos desde 5 minutos hasta 24 horas (6).

Además de otras características.

### 1.2.4 Netvibes

Es un servicio web que actúa a modo de escritorio virtual personalizado, similar a la Página Principal Personalizada de Google (iGoogle), MSN Live.

Visualmente está organizada en solapas o pestañas (*tabs*), donde cada solapa por lo general es en sí un para agregar diversos módulos y *widgets* desplazables previamente definidos por el usuario. Estos módulos, a su vez, actúan como pequeñas ventanas cuyo contenido es generado por otro servicio web o ser mini-aplicaciones.

Desde el punto de vista comercial, están los llamados Universos, que son páginas creadas principalmente por empresas o grupos musicales y en que se muestran diversas fuentes web, imágenes y otros materiales relacionados con el creador (7).

### 1.2.5 OpManager

Características del monitoreo de URL de OpManager:

- *“Monitoreo de URL, hosts virtuales y la Intranet.*
- *OpManager revisa las URLs para asegurar que tengan acceso y estén sirviendo páginas, esta es una forma más fiable de monitoreo de sitios web que dependen de pings ICMP (Protocolo de Mensajes de Control de Internet, por sus siglas en inglés de Internet Control Message Protocol) o chequeo de puertos TCP (Protocolo de Control de Transmisión, por sus siglas en inglés Transmission Control Protocol) en el puerto 80.*
- *Monitorea todos sus sitios web públicos, así como la disponibilidad y el buen estado de las aplicaciones basadas en Web y que se encuentren en su Intranet.*
- *OpManager proporciona también las tendencias de la disponibilidad y del rendimiento de su sitio web” (8).*

El análisis de las soluciones similares permitió observar algunas fortalezas que pueden ser reutilizadas de desarrollo como son: el uso de servicio web y la vista de los datos obtenidos del proceso de monitoreo. Independientemente de las características que con anterioridad han sido expuestas, ninguna de ellas cumple con los requerimientos que dan solución al problema de la investigación que se presenta en estos estudios. Entre otros motivos porque todas tienen una tarea en común que consiste en el monitoreo del tráfico de la red, sin embargo lo que se necesita es el monitoreo de los errores. Esta conclusión es la que condiciona que el autor se proponga el desarrollo del objetivo general propuesto.

### 1.3 Metodología de desarrollo de software

El empleo de una metodología durante el desarrollo de un *software*, le confiere a este y a la investigación que se le asocia, transparencia y calidad, elementos muy perseguidos por el cliente y necesarios para el producto.

La aplicación correcta de una metodología ayuda a mejorar el tiempo de desarrollo del *software*; definir con exactitud el personal que requiere el proyecto, las herramientas a utilizar, conocimientos concretos sobre el problema a resolver, entre otros aspectos que deben ser chequeados y controlados.

El análisis de la bibliografía consultada, muestra que existen dos tipos de metodologías: ágiles y tradicionales, dejando al equipo de desarrollo de un producto, tomar la decisión de cuál es la que más se ajusta a sus características.

Las metodologías pueden ser tradicionales o pesadas, que según la filosofía de desarrollo son las que hacen mayor énfasis en la planificación y control del proyecto, y necesitan de una especificación precisa de aspectos funcionales y modelado.



Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del *software*, con el fin de conseguir un *software* más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.

Según autores consultados en la Ecured plantean que las metodologías ágiles se encargan de valorar al individuo y promover las iteraciones del equipo más que a las herramientas o los procesos utilizados. Plantean además, que se hace mucho más importante crear un producto de *software* que funcione, que escribir mucha documentación (9). Esta metodología propone que el cliente esté en todo momento colaborando en el proyecto. A diferencia de otras metodologías es más importante la capacidad de respuesta ante un cambio realizado, que el seguimiento estricto de un plan (10).

Por los recursos, el personal y tiempo para realizar el trabajo se opta por usar una metodología ágil. Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos.

En lo adelante se profundizará en las metodologías SCRUM<sup>4</sup> y Programación Extrema (XP) por ser metodologías ágiles que se basan en el trabajo orientado directamente al objetivo.

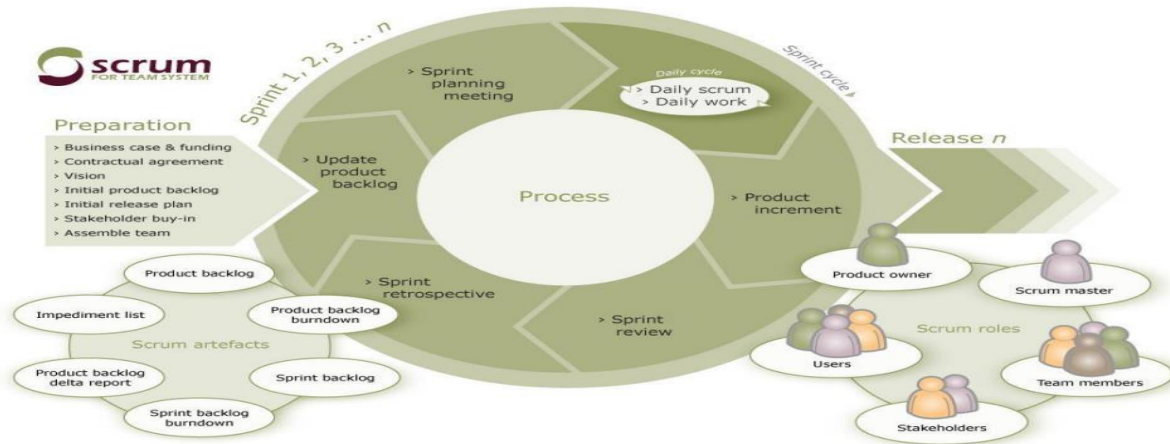
1. SCRUM.
2. *Extreme Programming*.

### 1.3.1 Scrum

En artículo publicado por Mike Cohen define Scrum como un marco para la gestión de proyectos (11). Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: el desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días, donde el resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente; la segunda característica son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (12).

---

<sup>4</sup> **SCRUM:** Término en rugby, que hace referencia a como se devuelve un balón que ha salido fuera del campo, al terreno de juego de una manera colectiva, la traducción al castellano sería melé.



**Figura 1.1** Ciclo de vida de la metodología SCRUM

Según Palacios es posible identificar tres fases durante el ciclo de vida en SCRUM: 1) planificación del *sprint*, 2) seguimiento del *sprint*, y 3) revisión del *sprint*. Estas fases pueden encontrarse como: fase antes del juego, fase del juego o desarrollo, fase después del juego (13). Dentro de los principios de SCRUM, se pueden mencionar:

- Los equipos son auto-gestionados.
- Se realizan reuniones diarias en las que los miembros del equipo se plantean 3 cuestiones:
  - ¿Qué has hecho desde la última revisión?
  - ¿Qué obstáculos te impiden cumplir la meta?
  - ¿Qué vas a hacer antes de la próxima reunión?
- Las iteraciones de desarrollo tienen una frecuencia inferior a un mes, al final de las cuales se presenta el resultado a los externos del equipo de desarrollo, y se realiza una planificación de la siguiente iteración, guiada por el cliente (14).

Sin embargo requiere confiar responsabilidades al equipo, incluso permite fallar si es necesario, además los miembros del equipo de trabajo programan de forma individual.

Esta metodología es apropiada para entornos ligeros donde se tiene una útil realimentación de los usuarios, en cada iteración se definen cuáles son los objetivos de la siguiente, tiene una planificación más transparente para los clientes al estar diseñada para el cambio.

### 1.3.2 XP

La literatura constata varias definiciones de esta metodología. Programación Extrema (XP por sus siglas del inglés *eXtreme Programming*), es definida por varios actores, el desarrollo de esta investigación se guió por los conceptos definidos por el autor Ian Sommerville que especifica:

XP es posiblemente el método ágil más conocido y ampliamente utilizado. El nombre fue acuñado por Beck (15), debido a que el enfoque fue desarrollado utilizando buenas prácticas reconocidas, como el desarrollo iterativo, y con la participación del cliente en niveles extremos.

En la programación extrema todos los requerimientos se expresan como escenarios llamados historias de usuario los cuales se implementan directamente como una serie de tareas. Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código. Todas las pruebas se deben ejecutar satisfactoriamente cuando el código nuevo se integre al sistema. Existe un pequeño espacio de tiempo entre las entregas del sistema (16).

La programación extrema implica varias prácticas que se ajustan a los principios de los métodos ágiles<sup>5</sup>:

- El desarrollo incremental, se lleva a cabo a través de entregas del sistema, pequeñas y frecuentes, y por medio de un enfoque para la descripción de requerimientos basados en las historias de cliente o escenarios que pueden ser la base para el proceso de planificación.
- La participación del cliente se lleva a cabo a través del compromiso a tiempo completo del cliente en el equipo de desarrollo. Los representantes de los clientes participan en el desarrollo y son los responsables de definir las pruebas de aceptación del sistema.
- El interés en las personas, en vez de en los procesos, se lleva a cabo a través de la programación en parejas, la propiedad colectiva del código del sistema, y un proceso de desarrollo sostenible que no implique excesivas jornadas de trabajo.
- El cambio se lleva a cabo a través de las entregas regulares del sistema, un desarrollo previamente probado y la integración continua.
- El mantenimiento de la simplicidad se lleva a cabo a través de la refactorización constante para mejorar la calidad del código y la utilización de diseños sencillos que no prevén cambios futuros en el sistema (17).

#### **Personas que intervienen en la metodología XP:**

En un proceso XP, los clientes están fuertemente implicados en la especificación y establecimiento de prioridades de los requerimientos del sistema.

Los requerimientos no se especifican como una lista de funciones requeridas del sistema. Más bien, los clientes del sistema son parte del equipo de desarrollo y discuten escenarios con otros miembros del equipo. Desarrollan conjuntamente una tarjeta de historia que recoge las necesidades del cliente. El equipo de desarrollo intentará entonces implementar ese escenario en una entrega futura del *software*.

---

<sup>5</sup> La descripción del análisis de la metodología XP se basó fundamentalmente en (17).

La participación del cliente se lleva a cabo a través del compromiso a tiempo completo del cliente en el equipo de desarrollo. Los representantes de los clientes participan en el desarrollo y son los responsables de definir las pruebas de aceptación del sistema.

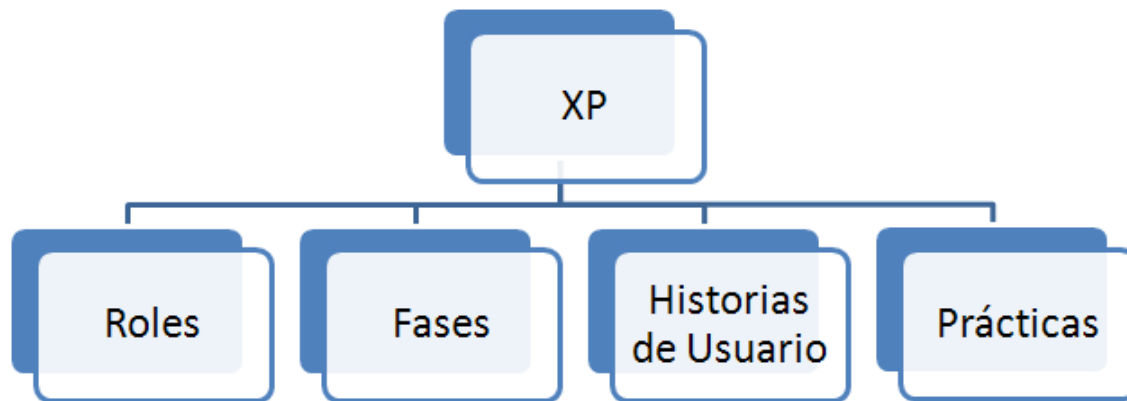


Figura 1.1 Principales elementos de la metodología XP

#### Los pasos de la metodología XP:

- Planificación incremental.

Los desarrolladores dividen estas historias en tareas de desarrollo.

- Entregas pequeñas.
- Diseño sencillo.
- Desarrollo previamente probado.
- Refactorización.

Esto conserva el código sencillo y fácil de mantener.

- Programación en parejas.
- Propiedad colectiva.
- Integración continua.

Después de la integración, se deben pasar al sistema todas las pruebas de unidad.

- Ritmo sostenible.
- Cliente presente.

El ciclo de vida ideal de XP consiste de seis fases, las mismas son (12):

**Exploración:** en esta etapa los clientes escriben las Historias de Usuario (HU) que quieren que sean incluidas en la aplicación, que describen las funcionalidades que serán añadidas al sistema.

**Planificación (Release):** se establece la prioridad de las HU y se acuerda el contenido de la primera entrega del proyecto, así como su estimación temporal.

**Iteraciones:** durante esta fase se decide cuáles serán las historias que se desarrollarán en cada iteración, así como las pruebas funcionales ejecutadas al final de cada iteración.

**Producción:** en esta fase se llevan a cabo un conjunto de pruebas extras, de rendimiento y funcionamiento que son necesarias antes de entregar el producto.

**Mantenimiento:** una vez sea liberada la primera versión a los usuarios, el sistema se debe mantener en el entorno de producción siempre y cuando aún hayan iteraciones en fase de producción.

**Cierre del proyecto o Muerte del Proyecto:** cuando ya no hay más HU que deban ser implementadas, las necesidades del cliente han sido satisfechas, así como la documentación (17).

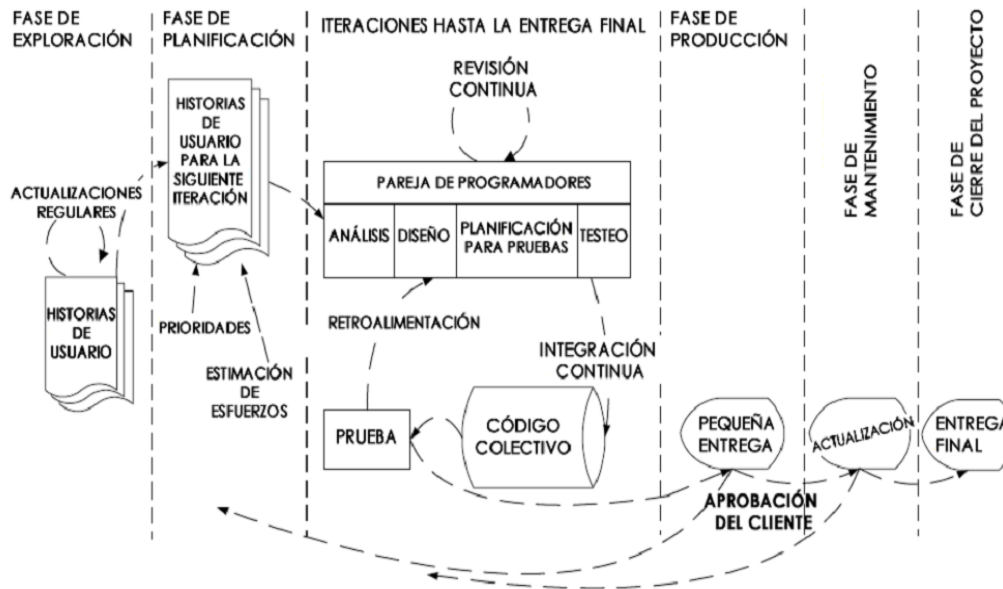


Figura 1.2 Fases XP

Cosas que propone esta metodología:

- Hay una comunicación frecuente entre el cliente y el equipo de desarrollo. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más de tres meses.
- Integración continua. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

- Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- Disponibilidad del cliente. El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita (12).

### 1.3.3 Fundamentación de la metodología a utilizar

A partir del estudio realizado se decidió que la metodología XP es la óptima a utilizar, ya que está empleada para proyectos de corto plazo, con un equipo de trabajo pequeño, propone una realimentación continua entre el cliente y el equipo de desarrollo, el modelo de 40 horas semanales que define para no trabajar horas extras y la propiedad colectiva del código.

Dada las condiciones, facilidades que brinda y la idea de desarrollo que se tiene de la solución, se acordó que sus características son las más asociadas al proyecto que se lleva a cabo durante el desarrollo de la presente investigación. Hace énfasis que la comunicación y satisfacción del cliente es lo principal.

Para una mejor estructuración del documento y un mejor entendimiento, se decide agrupar las seis fases en cuatro, sin violar el ciclo de vida de la metodología, estas fases son:

Fase I: Exploración

Fase II: Planificación

Fase III: Implementación

Fase IV: Pruebas

Con la aplicación de las características antes mencionadas, se espera llevar a cabo un proceso de desarrollo que guíe los pasos hacia la construcción de un producto con calidad y que cumpla con los plazos de tiempo y alcance previstos.

### 1.4 Herramientas y tecnologías

La solución propuesta será integrada al módulo ZERA Support que está actualmente en desarrollo en La Plataforma Educativa ZERA, por lo que se consideró trabajar con las versiones actuales del ambiente de desarrollo existente.

Para dar inicio al desarrollo de la herramienta para el monitoreo de errores de las aplicaciones web en el Centro FORTES, la metodología seleccionada permitió realizar previamente un estudio riguroso y detallado de tecnologías, herramientas, lenguaje de modelado, *framework*, servidores web, gestores de bases de datos, lenguajes de desarrollo y servicios web definidas por el equipo

de desarrollo del módulo “ZERA Support”. Se priorizaron las herramientas y tecnologías de código abierto o pertenecientes al *software* libre, en las cuales el desarrollador tenga experiencias en su uso y que cuenten con una amplia documentación y comunidad de usuarios, aspectos importantes en la retroalimentación y rectificación de errores.

Para el desarrollo de la solución propuesta se utilizará:

#### **1.4.1 Lenguaje de modelado**

La bibliografía constata que el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad; entre sus funciones se define que es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir el modelo de un sistema. UML cuenta con una gran variedad de propiedades entre las que se destacan:

- Ampliamente utilizado por la industria.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas.

UML se ha convertido ya en una de las mejores herramientas para el diseño y desarrollo de *software* fiable, eficiente y de calidad. Permite modelar sistemas de información, y su objetivo es lograr modelos que, además de describir con cierto grado de formalismo tales sistemas, puedan ser entendidos por los clientes o usuarios de aquello que se modela (18).

#### **1.4.2 Herramienta CASE**

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el costo de las mismas en términos de tiempo y de dinero. La evolución de las herramientas CASE está ligada a la evolución de la Ingeniería de *Software* como disciplina. El propósito de una herramienta CASE es dar soporte automatizado para la aplicación de todas o algunas técnicas usadas por una o varias metodologías (19). Algunas de estas herramientas son: Umbrello, Rational Rose, Visual Paradigm, Software Modeling entre otras.

Para modelar la propuesta de solución se hará uso de la herramienta Visual Paradigm en su versión 8.0. Esta herramienta CASE soporta los principales estándares de la industria tales como UML, la Notación para el Modelado de Procesos de Negocio (BPMN), entre otras especificaciones

definidas por el grupo de estandarización OMG (*Object Management Group*). Además ofrece un completo conjunto de herramientas que facilitan a los desarrolladores la captura de requisitos, planificación de *software*, planificación de controles, el modelado de clases, modelado de datos, entre otros (19).

Está concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software*. Dentro de sus características se destacan las siguientes.

- Disponible para múltiples plataformas.
- Posee capacidades de ingeniería directa e inversa.
- Soporta aplicaciones web.
- Las imágenes y reportes que genera son de buena calidad.
- Su modelo y código permanece sincronizado en todo el ciclo de desarrollo.
- Está disponible en múltiples versiones, para cada necesidad.
- Es fácil de instalar y actualizar.
- Transforma los diagramas de Entidad- Relación en tablas de base de datos.
- Importación y exportación de ficheros XML (Lenguaje de Marcas Extensible, de sus siglas en inglés de *eXtensible Markup Language*).
- Permite la reorganización de las figuras y conectores de los diagramas UML (20).

#### 1.4.4 Lenguajes de desarrollo del lado del cliente

##### HTML 5

HTML5 (*HyperText Markup Language*, versión 5). HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Entre los nuevos atributos que se destacan se encuentran:

Tabla 1.1 Nuevos atributos del HTML5 (21)

Etiquetas	Atributos
<article>	Atributos globales
<audio>	autobuffer   autoplay   controls   loop   src
<canvas>	height   width
<command>	checked   default   disabled   hidden   icon   label   radiogroup   type
<dialog>	Atributos globales
<embed>	height   src   type   width
<figure>	Atributos globales
<footer>	Atributos globales
<hgroup>	Atributos globales
<meter>	high   low   max   min   optimum   value
<output>	form
<progress>	max   value



Etiquetas	Atributos
<ruby>	cite
<section>	cite
<source>	media   src   type
<time>	datetime   pubdate
<video>	src   poster   autobuffer   autoplay   loop   controls   width   height

Entre sus nuevas características se tiene que:

- Incorpora etiquetas (*canvas*<sup>6</sup> 2D y 3D, audio, video) con *codecs* (codificador - decodificador) para mostrar los contenidos multimedia.
- Etiquetas para manejar grandes conjuntos de datos. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos y facilidades para validar el contenido sin Javascript.
- Añade etiquetas para manejar la Web Semántica (Web 3.0) (21).

### CSS3

CSS del inglés *Cascading Style Sheets* (hoja de estilo en cascada). CSS3 es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. El uso del CSS es para 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.

Para la propuesta de solución se usará el CSS3 ya que en diferencia a versiones anteriores, está dividida en varios documentos separados, llamados "módulos". Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad (22).

### Java Script v9.7

Para el desarrollo de la propuesta de solución se utilizará JavaScript en su versión 9.7. Según Suárez "...es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar determinadas acciones dinámicas dentro del ámbito de una página web. Técnicamente es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos" (23). En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Por otra parte pone a disposición del programador todos los elementos que forman parte de la página web, para que el mismo pueda acceder a ellos y modificarlos dinámicamente.

<sup>6</sup> **Canvas:** un canvas es un lienzo de mapa de bits dependiente de la resolución de pantalla. <http://www.desarrolloweb.com/articulos/guia-canvas-html5-desarrolladores.html>.

Java Script 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.

### 1.4.5 Lenguaje de desarrollo del lado del servidor

#### PHP v5.4.24

PHP, es un lenguaje de código abierto muy popular, especialmente adecuado para el desarrollo web. Es un lenguaje multiplataforma, y tiene soporte para cualquiera de los principales sistemas operativos del mercado y para una gran variedad de gestores de bases de datos, dentro de los que incluye PostgreSQL. Además soporta la mayoría de los servidores web como el Apache (24). Está formado de un conjunto de símbolos y reglas tanto sintácticas como semánticas que especifican su estructura y el significado de sus elementos y expresiones. Cada lenguaje de programación tiene sus propias características, las cuales lo hacen más potente o más débil para determinada finalidad. El estudio de los mismos garantiza contar con los recursos apropiados para codificar la solución a desarrollar.

PHP es un lenguaje interpretado de propósito general ampliamente utilizado, diseñado especialmente para aumentar e incrementar el dinamismo de las páginas web y puede ser incrustado dentro de código HTML (*HyperText Markup Language*, Lenguaje de Marcado de Hipertexto por su traducción al español). Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida al usuario final, por lo que PHP convierte una página estática en una dinámica. Puede ser desplegado en la mayoría de los servidores web y plataformas sin costo alguno. La nueva versión de PHP posee una serie de nuevas características como son:

- Se ha añadido la sintaxis corta de *array*.
- Se ha añadido la posibilidad de referenciar la función de *array*.
- Los cierres ahora soportan `$this`.
- `<?=>` ahora está siempre disponible, sin tener en cuenta la opción de `php.ini short_open_tag`.
- Se ha añadido el acceso a miembro de clase en la instanciación.
- Ahora está soportada la sintaxis `Clase::{expr}()`.
- Se han mejorado los mensajes de error de análisis y las advertencias de argumentos incompatibles.
- La extensión de sesiones ahora puede rastrear el progreso de subida de ficheros (25).

#### 1.4.6 Tecnología del lado del cliente

Para la propuesta de solución se usará como tecnología del lado del cliente: **Ajax**, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Es una tecnología asíncrona, ya que los datos

adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

Ajax es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y Hojas de Estilos en Cascada (CSS) para el diseño que acompaña a la información.
- *Document Object Model* (DOM) accedido con un lenguaje de guion por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos *frameworks* y en algunas situaciones concretas, se usa un objeto *iframe* en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre-formateado, texto plano, JSON y hasta EBML.
- Como el DHTML, LAMP o SPA, Ajax no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente (26).

### 1.4.7 *Framework* de desarrollo del lado del cliente

#### JQuery v1.9.1

El *framework* de desarrollo del lado del cliente que se utilizará en la propuesta de solución es el JQuery en su versión 1.9.1. JQuery es un *framework* de Javascript, es decir, sirve como base para la programación avanzada de aplicaciones con código JavaScript aportando una serie de funciones o códigos para realizar tareas habituales. Es una librería que se encarga de acceder a los objetos del DOM (Modelo en Objetos para la Representación del Documento) de un modo simplificado. Ofrece una infraestructura con la que se tiene mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Permite agregar efectos a las páginas haciéndolas más interactivas y con una considerable disminución de código. Tiene una estructura que le da organización al proyecto y evita la implementación de funcionalidades comunes (27).

#### Twitter Bootstrap v2.2.2

Como *framework* de desarrollo CSS se utilizará Twitter Bootstrap en su versión 2.2.2. Twitter Bootstrap simplifica el proceso de creación de diseños web combinando CSS y JavaScript. La mayor ventaja es que posibilita crear interfaces que se adapten a los distintos navegadores, apoyándose en un *framework* potente con numerosos componentes web que ahorrarán mucho esfuerzo y tiempo. Bootstrap ofrece una serie de plantillas CSS y ficheros JavaScript que permiten integrar el *framework* de forma sencilla y potente en las aplicaciones web. Ofrece un diseño sólido

usando estándares como CSS3/HTML5 (28).

#### **1.4.8 Framework de desarrollo del lado del servidor**

Symfony2, es un marco de desarrollo para PHP, desarrollado completamente con PHP5, diseñado para optimizar el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web (Modelo - Vista - Controlador). Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Una aplicación desarrollada con el marco de trabajo Symfony2, contará con código claro y organizado consistentemente. Symfony2 promueve la reutilización y permite a los nuevos desarrolladores ser productivos en el proyecto con mayor rapidez. El 100% del código que es escrito, es para la aplicación, pues no se necesita desarrollar o mantener servicios públicos de bajo nivel, tales como la carga automática de clases, el enrutado o la reproducción de controladores.

Proporciona acceso a librerías como Doctrine, además a plantillas, seguridad, formularios, validación y traducción. Permite que las URL sean totalmente flexibles gracias al componente Routing (enrutamiento). La arquitectura centrada en HTTP de Symfony2 le da acceso a herramientas, tal como la memoria caché HTTP (29).

Para el desarrollo de la propuesta de solución de empleará el *framework* Symfony en su versión 2.4.0 que trae consigo una serie de nuevas características como son:

- Se ha añadido un nuevo componente llamado *Expression Language*.
- Se ha introducido un nuevo servicio llamado `request_stack` y que reemplaza al servicio *request*. A partir de esta versión, se desaconseja el uso del objeto *Request* en los servicios y se recomienda utilizar en su lugar el objeto Request Stack.
- Las plantillas *Twig* ahora permiten medir en detalle el tiempo que tardan en renderizar cada parte.
- Ahora es mucho más fácil definir tus propios proveedores de usuarios y autenticadores y también puedes asociar *firewalls* a *hosts* (detalles).
- Los *logs* de la aplicación ahora se pueden mostrar en la consola y también se han añadido muchas pequeñas mejoras a la consola.
- Ahora es posible detener un proceso si lleva demasiado tiempo inactivo.
- Se ha añadido un panel de depuración de formularios en la barra de depuración y se ha mejorado bastante el validador de imágenes (30).

#### **1.4.9 Gestor de Base de Datos**

Los Sistemas de Gestión de Bases de Datos (SGBD) (en inglés *Database Management System*,

abreviado DBMS) son un tipo de *software* específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Existen muchos SGBD pero en su gran mayoría son sistemas propietarios entre los que se destacan: MySQL, Advantage Database, FileMaker, etc. También se encuentran SGBD disponible como Sistemas Libres como son: PostgreSQL, FileMaker, MySQL, etc.

Como gestor de base de datos para la solución propuesta se utilizará PostgreSQL en su versión 9.2.4. PostgreSQL es un SGBD Relacional orientada a objetos y libre, publicado bajo la licencia BSD. Algunas de las características de este sistema son:

- Alta concurrencia.
- Alta variedad de tipos nativos.
- Claves Ajenas.
- Disparadores.
- Vistas.
- Tipos de datos y operaciones geométricas.
- Integridad transaccional.
- Herencia de tablas.
- Soporte para transacciones distribuidas.
- Funciones (31).

#### 1.4.10 Servidor Web

Un servidor web es un programa informático para procesar las aplicaciones del lado del servidor mediante conexiones que generan una respuesta desde las aplicaciones del lado del cliente.

El servidor web seleccionado para utilizar en la propuesta de solución es Apache en su versión 2.2.22.

Apache es un servidor web multiplataforma de código abierto, es el ejemplo de código libre de mayor éxito. Permite generar informes de errores HTTP y gestionar recursos para procesos hijos. El servidor Apache presenta entre otras características: mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache, existen gran cantidad de módulos Apache disponibles para su utilización. Tiene una buena compatibilidad con el lenguaje PHP compartiendo muchas de sus características.

Principales características de Apache.

- **Fiabilidad:** Alrededor del 90% de los servidores con más alta disponibilidad funcionan con Apache.
- **Gratuidad:** Apache es totalmente gratuito, y se distribuye bajo la licencia *Apache Software*

*License*, favoreciendo la modificación del código.

- **Extensibilidad:** Se pueden añadir módulos para ampliar las amplias capacidades de Apache. Existe una amplia variedad de módulos, que permiten generar contenido dinámico con PHP, Java, Perl, entre otros, además de monitorizar el rendimiento del servidor. Estos módulos pueden ser creados por cualquier persona con conocimientos de programación (32).

#### 1.4.11 Entorno de desarrollo integrado

##### NetBeans v7.4

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Para la propuesta de solución se seleccionó el IDE NetBeans en su versión 7.4. NetBeans es un entorno de desarrollo integrado, multiplataforma, de código abierto, permite a los desarrolladores crear rápidamente aplicaciones web, móviles y de escritorio con el uso de plataforma Java, PHP, JavaScript, Ajax, Groovy, Grails y C / C++. Es gratuito, tiene una gran comunidad de usuarios y desarrolladores en todo el mundo, proporciona herramientas de análisis estático para identificar y solucionar problemas comunes en código Java; además el IDE tiene editores y herramientas para XML, HTML, PHP, Groovy, Javadoc, JavaScript y JSP (33).

El IDE NetBeans 7.4 ofrece un rendimiento significativamente mejorado y mayor experiencia de codificación. La base en la que se sustenta su elección es que permite desarrollar aplicaciones utilizando el *framework* Symfony y ejecutar los comandos del mismo directamente desde la interfaz del IDE.

#### 1.5 Servicios web

Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posibilidad sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web (34).

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo

sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría el Protocolo Simple de Acceso a Objetos o SOAP (siglas del inglés *Simple Object Access Protocol*). Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc. SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto por un “sobre” (*envelope*), cuya estructura está formada por los siguientes elementos: cabecera (*header*) y cuerpo (*body*).

Por otro lado se tiene Transferencia de Estado Representacional o REST (por sus siglas en inglés *Representational State Transfer*), es un estilo de arquitectura de *software* para sistemas distribuidos tales como la web, a diferencia de SOAP, se centra en el uso de los estándares HTTP y XML para la transmisión de datos sin la necesidad de contar con una capa adicional. Las operaciones (o funciones) se solicitarán mediante *GET*, *POST*, *PUT* y *DELETE*, por lo que no requiere de implementaciones especiales para consumir estos servicios. Además se podrá utilizar JSON en vez de XML como contenedor de la información (35).

Se determina utilizar REST dado a que sus características son más adaptables al entorno de desarrollo, siendo estas las siguientes:

- Un bajo consumo de recursos.
- Las instancias del proceso son creadas explícitamente.
- El cliente no necesita información de enrutamiento a partir de la URI inicial.
- Los clientes pueden tener una interfaz escuchadora (*listener*) genérica para las notificaciones.
- Generalmente fácil de construir y adoptar.

## **1.6 Conclusiones del capítulo**

La metodología de investigación posibilitó realizar el estudio del arte del objeto que se estudia. Luego del estudio realizado se pudo puntualizar los conceptos más importantes para dar cumplimiento al problema de la investigación que se realiza. Se pudo determinar que las herramientas similares no cumplen con las expectativas del problema creado en el Centro FORTES de la Facultad cuatro para la monitorización de aplicaciones web. Además, no cumplen con la política de utilización de software libre implementada en el país, no se ajustan a las condiciones económicas actuales y no realizan las tareas específicas para cumplir con la necesidad de la Plataforma.

El estudio de las herramientas para el desarrollo de la propuesta de solución estuvo apoyado en el criterio de selección de herramientas libres y competentes para la obtención del producto final con la calidad requerida.



## Capítulo 2 ANÁLISIS Y DISEÑO

El presente capítulo detalla las fases de Exploración y Planificación definidas en el ciclo de vida de la metodología de desarrollo XP que posibilita describir la solución propuesta.

De igual forma se muestran los principios, prácticas y técnicas que sirven de guía para los desarrolladores, entre las que se destacan: HU, plan de iteraciones, plan de entrega y las tarjetas CRC (Clases, Responsabilidad y Colaboración).

### 2.1 Modelo de dominio

La metodología XP no utiliza el modelo de dominio. A pesar de esto, se decide utilizar dicho modelo, con el fin de describir y limitar el alcance del dominio del problema. Este diagrama no se puede confundir con un modelo de clase de *software*, aunque puede ser utilizado como base para su construcción.

Se puede definir este modelo como una visualización de los conceptos del dominio en el mundo real, para realizar dicho modelo se necesita definir las clases conceptuales (solo las más significativas), así como las relaciones existentes entre ellas para brindar un mejor entendimiento del sistema. Las clases conceptuales son para representar de la manera más fácil posible y de forma adecuada las estructuras del lenguaje natural, se encargan de representar las entidades presentes en el dominio (ideas, objetos, personas, etc.) El modelo de dominio es una representación estática del sistema (36).

Para realizar un modelo de dominio se tienen los siguientes pasos:

- Listar las clases conceptuales candidatas.
- Representarlas clases en el modelo.
- Añadir las asociaciones necesarias para registrar las relaciones que hay que mantener en memoria.

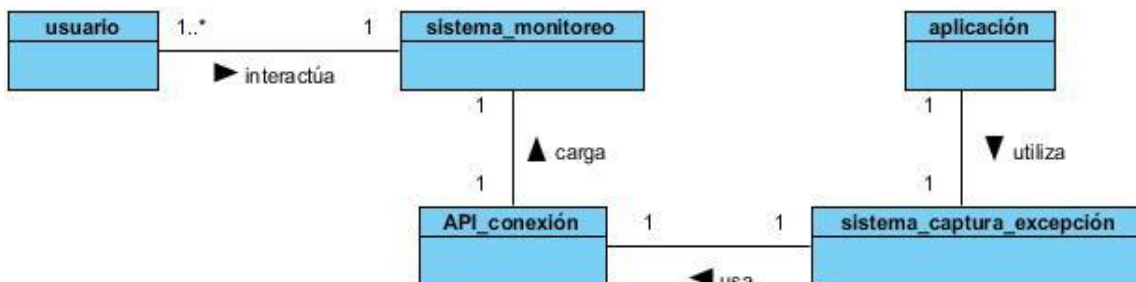


Figura 2.1 Modelo de dominio (elaboración propia)

En la Figura 2.1 se muestra el funcionamiento de un sistema de monitoreo de errores. El usuario interactúa con el sistema, el cual mediante la Interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) de conexión se encarga de guardar todas las excepciones capturadas por un sistema de captura implementado en las aplicaciones web.

**Descripción del modelo de dominio:****usuario:**

El usuario será la persona que interactúe de forma dinámica con el sistema. Este debe estar previamente registrado por un miembro del grupo de administradores, deben estar asociados a una o más aplicaciones web que se monitoree por el sistema.

**sistema\_monitoreo:**

El sistema de monitoreo se encargará de controlar los errores http de las aplicaciones que controla. Este sistema enviará a los desarrolladores un reporte completo desde la última notificación, de los errores ocurridos en las aplicaciones web que estos controlen.

**API\_conexión:**

El API de conexión es una clase que tendrá tres funcionalidades. La primera de estas, se encargará de guardar los errores en un archivo local, otra funcionalidad reportará los errores que existan en el fichero y la otra funcionalidad es encargada de cambiar la contraseña del usuario de la aplicación que está registrado en el sistema de monitoreo.

**sistema\_captura\_excepción:**

Sistema encargado de capturar los errores que ocurren en las aplicaciones web, este sistema lo implementa cada aplicación.

**aplicación:**

Son las aplicaciones web, que hacen uso del sistema de monitoreo, para llevar un control de sus errores.

**2.2 Propuesta de solución**

Se propone desarrollar una solución informática que se encargue de monitorear los errores lanzados por las aplicaciones web. La propuesta de solución será capaz de integrarse con aplicaciones desarrolladas sobre el *framework* Symfony2. Una de las herramientas que se encuentra en desarrollo sobre el *framework* mencionado es "ZERA Support", con la cual se va a integrar la solución informática posibilitando hacer uso de su gestión de usuario. Debido a esto el sistema constará con restricciones de acceso, de acuerdo al rol que posee el usuario que interactúe con él. Existirán solo dos roles, el rol administrador y el rol usuario registrado. El usuario que cuente con el rol administrador, es el encargado del correcto funcionamiento del sistema y la gestión de las aplicaciones que serán monitoreadas. El sistema admitirá que exista más de un usuario con este rol. La persona que posea el rol usuario registrado, va a estar asociado a aplicaciones que se estén monitoreando y solo podrá obtener la información de los errores generados por estas.

El monitoreo será realizado haciendo uso de dos servicios web. El servicio reportar, se encarga de guardar los errores en el sistema, el cual requiere el envío de un *token*<sup>7</sup> que se genera mediante el proceso de autenticación de las aplicaciones. Este proceso se realiza utilizando el servicio autenticación, el cual recibe un usuario y contraseña que se les crea a las aplicaciones que solicitan ser monitoreadas y solamente serán utilizados por los servicios web. La herramienta informática permitirá además la generación de reportes de los errores ocurridos, cumpliendo con determinados filtros especificados por el usuario como son: rango de fecha, tipo de error, aplicación o estado.

Uno de los requisitos para la utilización del servicio de monitoreo es que las aplicaciones web que se desea monitorear deben contar con un sistema de captura de errores previamente implementado, el cual debe ser reconfigurado para hacer uso de la API proporcionada por la propuesta de solución. La API REST como complemento de la propuesta de solución, es una clase php configurable, encargada de hacer uso de los servicios web brindados por la herramienta informática.

La interfaz gráfica será amigable y ofrecerá a los usuarios una fácil interacción con el sistema. La propuesta de solución estará dirigida a usuarios del equipo de desarrollo de la aplicación que se esté monitoreando.

A continuación se mencionan las principales funcionalidades que serán descritas posteriormente en las Historias de Usuario (HU).

- Servicio REST.
- API REST.
- Adicionar aplicación.
- Listar aplicaciones.
- Eliminar aplicaciones.
- Visualización detallada de las aplicaciones.
- Listar errores.
- Marcar errores como vistos.
- Eliminar los errores.
- Exportar reporte.
  - Generar reportes de errores en formato pdf y xls.
  - Generar reporte para notificación en formato pdf.
- Visualización detallada de errores.

---

<sup>7</sup> **Token:** Un **token** o también llamado **componente léxico** es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación. <http://www.alegsa.com.ar/Dic/token.php>.

- Administrar notificaciones.
  - Crear periodo de notificación.
  - Listar notificación.
  - Eliminar notificación.

### 2.3 Personas relacionadas con el sistema

Se define como personas relacionadas con el sistema a todas las personas que de una forma u otra van a interactuar con la aplicación, incluyendo a los que van a mantener el sistema (37).

**Tabla 2.1** Personas relacionadas con el sistema

Personas relacionadas con el sistema	Justificación
<b>Usuario registrado</b>	Es la persona que se encuentra registrada en el sistema. El usuario registrado será el personal encargado del mantenimiento o parte del equipo de desarrollo de la aplicación con la cual estará relacionado en el sistema. Un usuario puede pertenecer a varias aplicaciones. Este usuario una vez autenticado tendrá acceso a ver los errores que generados por las distintas aplicaciones a las que pertenece.
<b>Administrador</b>	Es la persona autorizada para la gestión del sistema. Tiene todos los privilegios para acceder a todo el contenido del sistema. Es la persona encargada de adicionar las aplicaciones al sistema.

### 2.4 Historias de usuarios planificadas

Las historias de usuario son la técnica utilizada en XP para especificar los aspectos funcionales del *software*. Se realizó una por cada funcionalidad del sistema, se emplearon para hacer estimaciones de tiempo y para el plan de iteraciones. Cada HU se definió lo suficientemente comprensible y delimitada para que el programador pueda implementar en pocas semanas. Si la estimación es superior a las tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia de usuario. Cada historia de usuario debe tener en algún momento pruebas de validación asociadas, lo que permitirá al desarrollador, y más tarde al cliente, verificar si la historia ha sido completada. Para los prototipos de interfaz de usuarios ver Anexo 1: Prototipos de interfaz de usuario.

Tabla 2.2 Historia de Usuario 1: Servicio REST

Historia de usuario	
<b>Nombre:</b> Servicio REST	<b>Número:</b> HU_1
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 1
<b>Prioridad en el negocio:</b> alta	<b>Puntos Estimados:</b> 1.5
<b>Riesgo en Desarrollo:</b> alto	<b>Puntos Reales:</b> 1.5
<b>Descripción:</b> el servicio REST se encargará de mantener un monitoreo constante de las aplicaciones que estén usando el sistema.	
<b>Observaciones:</b> el servicio sólo podrá ser utilizado por las aplicaciones que estén registradas en el sistema.	

Tabla 2.3 Historia de Usuario 2: Conexión

Historia de usuario	
<b>Nombre:</b> Api REST	<b>Número:</b> HU_2
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 1
<b>Prioridad en el negocio:</b> alta	<b>Puntos Estimados:</b> 1.5
<b>Riesgo en Desarrollo:</b> alto	<b>Puntos Reales:</b> 1.5
<b>Descripción:</b> se encarga de hacer uso del servicio brindado por el sistema para almacenar los errores que se generen en las aplicaciones.	
<b>Observaciones:</b> las aplicaciones tienen que tener un sistema de captura de excepciones donde se incluya la clase y se siga con las instrucciones descritas en esta. La clase consta de parámetros que deben ser cambiados para su utilización.	

Tabla 2.4 Historia de Usuario 3: Adicionar aplicación

Historia de usuario	
<b>Nombre:</b> Adicionar aplicación	<b>Número:</b> HU_3
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> administrador	<b>Iteración Asignada:</b> 2
<b>Prioridad en el negocio:</b> alta	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> alto	<b>Puntos Reales:</b> 1
<b>Descripción:</b> brinda la posibilidad de registrar en el sistema las aplicaciones web que se les va a realizar el monitoreo.	
<b>Observaciones:</b>	

Tabla 2.5 Historia de Usuario 4: Listar aplicación

Historia de usuario	
<b>Nombre:</b> Listar aplicación	<b>Número:</b> HU_4
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> administrador	<b>Iteración Asignada:</b> 2
<b>Prioridad en el negocio:</b> alta	<b>Puntos Estimados:</b> 0.5
<b>Riesgo en Desarrollo:</b> medio	<b>Puntos Reales:</b> 0.5
<b>Descripción:</b> lista todas las aplicaciones que se están controlando por el sistema.	
<b>Observaciones:</b> se brinda la posibilidad de eliminar las aplicaciones.	

Tabla 2.6 Historia de Usuario 5: Visualizar aplicación

Historia de usuario	
<b>Nombre:</b> Visualizar aplicación	<b>Número:</b> HU_5
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> administrador	<b>Iteración Asignada:</b> 2
<b>Prioridad en el negocio:</b> media	<b>Puntos Estimados:</b> 0.5
<b>Riesgo en Desarrollo:</b> bajo	<b>Puntos Reales:</b> 0.5
<b>Descripción:</b> se muestra un resumen detallado de cada aplicación.	
<b>Observaciones:</b>	

Tabla 2.7 Historia de Usuario 6: Listar errores

Historia de usuario	
<b>Nombre:</b> Listar errores	<b>Número:</b> HU_6
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 3
<b>Prioridad en el negocio:</b> alta	<b>Puntos Estimados:</b> 1.5
<b>Riesgo en Desarrollo:</b> medio	<b>Puntos Reales:</b> 1.5
<b>Descripción:</b> crea una vista de los errores existentes.	
<b>Observaciones:</b> posee una columna "Opciones", donde se puede visualizar el error, marcar el error como revisado o eliminarlo del sistema. Permite exportar el reporte a formato PDF. Permite una serie de filtros con el objetivo de agilizar el trabajo de los usuarios. La opción Detalles genera una vista detallada del error. En el botón Opciones, si el error no ha sido revisado muestra la opción Revisado, en otro caso muestra la opción Eliminar.	

Si el usuario es Administrador se mostrarán todos los errores que se han registrado en el sistema.  
Si el usuario es Usuario Registrado se mostrarán los errores pertenecientes a las Aplicaciones con las cuales está relacionado.

Tabla 2.8 Historia de Usuario 7: Generar reportes

Historia de usuario	
<b>Nombre:</b> Generar reporte	<b>Número:</b> HU_7
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 3
<b>Prioridad en el negocio:</b> alta	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> medio	<b>Puntos Reales:</b> 1
<b>Descripción:</b> el sistema brindará la posibilidad de exportar reportes en formatos PDF y XLS seleccionando las columnas que se quieran exportar.	
<b>Observaciones:</b> cuando se genera un reporte se exportan todos los errores sin tener en cuenta los filtros, por lo que se da la posibilidad de exportar al formato editable XLS para que se pueda realizar cualquier modificación o filtros a los campos.	

Tabla 2.9 Historia de usuario 8: Visualizar errores

Historia de usuario	
<b>Nombre:</b> Visualizar errores	<b>Número:</b> HU_8
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 3
<b>Prioridad en el negocio:</b> media	<b>Puntos Estimados:</b> 0.5
<b>Riesgo en Desarrollo:</b> bajo	<b>Puntos Reales:</b> 0.5
<b>Descripción:</b> crea una vista detallada de los errores existentes mostrándolo en un objeto modal, donde se visualiza todos los datos del error.	
<b>Observaciones:</b>	

Tabla 2.10 Historia de Usuario 9: Adicionar notificaciones

Historia de usuario	
<b>Nombre:</b> Adicionar notificaciones	<b>Número:</b> HU_9
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 4
<b>Prioridad en el negocio:</b> alto	<b>Puntos Estimados:</b> 1.5
<b>Riesgo en Desarrollo:</b> alto	<b>Puntos Reales:</b> 1.5

**Descripción:** permitirá añadir una notificación llenando los campos requeridos, además, dará la posibilidad al desarrollador registrado en el sistema, de adicionar las direcciones de correo de los desarrolladores que desea que sean notificados.

**Observaciones:** se enviará un correo de confirmación a las direcciones de correo añadidas por el usuario.

**Tabla 2.11** Historia de Usuario 10: Listar notificaciones

Historia de usuario	
<b>Nombre:</b> Listar notificaciones	<b>Número:</b> HU_10
<b>Modificaciones:</b> ninguna	
<b>Usuario:</b> todos	<b>Iteración Asignada:</b> 4
<b>Prioridad en el negocio:</b> medio	<b>Puntos Estimados:</b> 0.5
<b>Riesgo en Desarrollo:</b> bajo	<b>Puntos Reales:</b> 0.5
<b>Descripción:</b> mostrará un listado de todas las notificaciones que tiene registrado el usuario y dará la posibilidad de eliminar las mismas.	
<b>Observaciones:</b>	

## 2.5 Fase de planificación

Una vez definida cada una de las historias de usuarios se pasó a definir la estimación de esfuerzo necesario para realizar cada una de ellas.

La planificación se realizó basándose en el tiempo o el alcance. Al planificar por tiempo, se multiplicó el número de iteraciones por la velocidad del proyecto, determinándose cuantos puntos se pueden completar. Al planificar según el alcance, se dividió la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

**Tabla 2.12** Velocidad del proyecto

	Iteración 1	Iteración 2	Iteración 3	Iteración 4
<b>Horas</b>	120,00	80,00	120,00	80,00
<b>Semas</b>	3	2	3	2
<b>Horas semanales</b>	40	40	40	40
<b>HU (velocidad del proyecto)</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>2</b>

### 2.5.1 Estimación de esfuerzos por historia de usuario

Para la realización de la solución propuesta se realizó la estimación de esfuerzo la cual se presenta a continuación.



**Tabla 2.13** Estimación de esfuerzos por historia de usuario

Nro.	Historia de Usuario	Puntos de estimación (semanas)
1	Servicio REST	1.5
2	Api REST	1.5
3	Adicionar aplicación	1
4	Listar aplicación	0.5
5	Visualizar aplicación	0.5
6	Listar errores	1.5
7	Generar reporte	1
8	Visualizar errores	0.5
9	Adicionar notificaciones	1.5
10	Listar notificaciones	1.5

### 2.5.2 Plan de Iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteraciones son las historias de usuario. Una vez identificadas las historias de usuario, se establecieron cuatro iteraciones para el desarrollo de la aplicación. En la selección de las historias de usuario a implementar se tuvo en cuenta la prioridad que estas presentan y la relación en cuanto a funcionalidad entre las mismas.

**Iteración 1:** en esta iteración se realizaron las historias de usuarios que hacen referencia al trabajo con el servicio REST debido a la dependencia que existe entre las demás historias de usuarios con respecto a ellas. Las historias de usuario seleccionadas fueron 1 – 2.

**Iteración 2:** en la segunda iteración se realizaron las historias de usuario 3 – 4 – 5, estas son las encargadas de realizar todas las funcionalidades del manejar aplicación.

**Iteración 3:** en esta iteración se realizaron las historias de usuario 6 – 7 – 8, estas son las encargadas de realizar operaciones sobre los errores, para la selección de esta iteración se tuvo en cuenta la dependencia que tienen estas historias de usuario con las realizadas en la iteración 1 y 2.

**Iteración 4:** en esta iteración se realizaron las historias de usuario 9 – 10, estas son las encargadas de realizar el manejar notificación. Para la selección de estas historias de usuario se tuvo en cuenta que las mismas sólo son funcionales si están implementadas las historias de usuario de las iteraciones anteriores.

### 2.5.3 Plan de duración de las iteraciones

Cada iteración va a estar conformada por las historias de usuarios seleccionadas por el cliente a implementar. En este plan se especifica detalladamente el orden de desarrollo de las historias de usuarios dentro de cada iteración así como la estimación completa de dicha iteración.

**Tabla 2.14** Plan de duración de las iteraciones

Iteraciones	Orden de las HU a implementar	Duración (semanas)
1	<ol style="list-style-type: none"> <li>Servicio REST</li> <li>Api REST</li> </ol>	3
2	<ol style="list-style-type: none"> <li>Adicionar aplicación</li> <li>Listar aplicación</li> <li>Visualizar aplicación</li> </ol>	2
3	<ol style="list-style-type: none"> <li>Listar errores</li> <li>Visualizar errores</li> <li>Generar reporte</li> </ol>	3
4	<ol style="list-style-type: none"> <li>Adicionar notificaciones</li> <li>Listar notificaciones</li> </ol>	2

### 2.5.4 Plan de entrega

El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias de usuario que fueren la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias de usuario se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para realizarle las pruebas de *software* (38).

**Tabla 2.15** Plan de entrega

Historia de Usuario	Entrega 1	Entrega 2	Entrega 3	Entrega 4	Entrega 5
<b>Servicio REST</b>	v1.0	<i>Finalizado</i>			
<b>API_REST</b>	v1.0	<i>Finalizado</i>			
<b>Adicionar aplicación</b>		v1.5	<i>Finalizado</i>		
<b>Listar aplicación</b>		v1.5	<i>Finalizado</i>		
<b>Visualizar aplicación</b>		v2.0	<i>Finalizado</i>		
<b>Listar errores</b>			v2.3	<i>Finalizado</i>	

<b>Visualizar errores</b>			v2.5	<i>Finalizado</i>	
<b>Generar reporte</b>			v3.0	<i>Finalizado</i>	
<b>Adicionar notificaciones</b>				v4.0	<i>Finalizado</i>
<b>Listar notificaciones</b>				v4.0	<i>Finalizado</i>

## 2.6 Tarjetas CRC

Una tarjeta CRC (clase, responsabilidad y colaboración) representa una entidad del sistema, a la cual asigna responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre clientes y equipo de desarrollo, y se ejecutan escenarios a partir de especificación de aspectos funcionales o historias de usuarios. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones (39).

**Clase:** Se refiere a las clases persistentes de sistema a desarrollar.

**Responsabilidad:** Función que realiza la clase dentro del sistema.

**Colaboración:** Relación que tiene la clase con otras clases persistentes del sistema.

Una de las principales piezas de diseño empleadas fueron las tarjetas CRC que no sólo sirven como columna vertebral de este, sino que también fueron la base del modelo Entidad-Relación, elaborado para modelar la base de datos. Cada tarjeta CRC se convirtió en un objeto, sus responsabilidades en métodos públicos y sus colaboradores en llamados a otras clases.

En XP el proceso de diseño es iterativo, por lo cual las tarjetas CRC no fueron creadas todas en la primera iteración. Las responsabilidades y los llamados fueron agregados al inicio de cada iteración, al igual que nuevas tarjetas CRC que fueron creadas de modo tal que el diseño se convirtió en un proceso dinámico que se adaptaba a las necesidades planteadas para el momento.

Luego de un estudio bibliográfico se determinó que XP no propone una estrategia para afrontar la implementación de las tarjetas CRC, por lo cual se creó una con la cual se garantizó el poder hacer las pruebas desde el mismo momento de la implementación. Primero fueron implementadas las clases que no hacía llamados a ninguna otra, luego se implementaron las que realizaban llamados a las clases ya implementadas. Aunque XP no tiene descrita una metodología para implementar un modelo de CRC, fue importante adoptar la estrategia antes expuesta, debido a que era la forma más cómoda para poder aplicar las pruebas en todo momento.

Tabla 2.16 Tarjeta CRC: API

Clase: ApiController	
Responsabilidades	Colaboradores
postAccesoAction (Request \$request)	TbApiController
postChangeAction (Request \$request)	

Tabla 2.17 Tarjeta CRC: WebServiceController

Clase: WebServiceController	
Responsabilidades	Colaboradores
postSaveReportAction (Request \$request)	TbExceptionHandler

Tabla 2.18 Tarjeta CRC: ApiREST

Clase: ApiREST	
Responsabilidades	Colaboradores
actionSave (\$type, \$description, \$url, \$ipClient, \$ipServer, \$user, \$username, \$browser, \$os, \$message)	
actionReport (\$user, \$pass)	
actionChangePassword (\$user, \$oldPass, \$newPass)	

Tabla 2.19 Tarjeta CRC: Controlador de aplicaciones

Clase: TbApiController	
Responsabilidades	Colaboradores
indexAction ()	
createAction (Request \$request)	
newAction ()	
detalles_appAction (\$id)	
exportApiRESTAction ()	
deleteAction (Request \$request, \$id)	

Tabla 2.20 Tarjeta CRC: Controlador de errores

Clase: TbExceptionHandler	
Responsabilidades	Colaboradores
indexAction ()	TbApiController
index1Action ()	TbApiController UserController
deleteAction (\$id)	
listhttpOpcionesAction (\$id)	
detalleErrorAction (\$id)	
generateReportAction(\$id)	

Tabla 2.21 Tarjeta CRC: Controlador de mensajes

Clase: MessagesController	
Responsabilidades	Colaboradores
messagesAction ()	TbNotifyController UserController

Tabla 2.22 Tarjeta CRC: Controlador de notificaciones

Clase: TbNotifyController	
Responsabilidades	Colaboradores
indexAction ()	
createAction (Request \$request)	
confirmAction(\$token)	
newAction ()	
deleteAction (\$id)	
pdfNotifyAction ()	

## 2.7 Modelo de datos

Un modelo de datos es una parte esencial en el proceso de modelado de una base de datos. Este es una estructura abstracta que documenta y organiza la información de los datos y las relaciones entre ellos. El propósito de un modelo de datos es, por una parte, representar los datos y por otra, ser comprensible. En él se describen los datos, las relaciones de datos y la semántica de los datos. A continuación en la Figura 2.2 se muestra el diagrama Entidad-Relación de la solución propuesta.

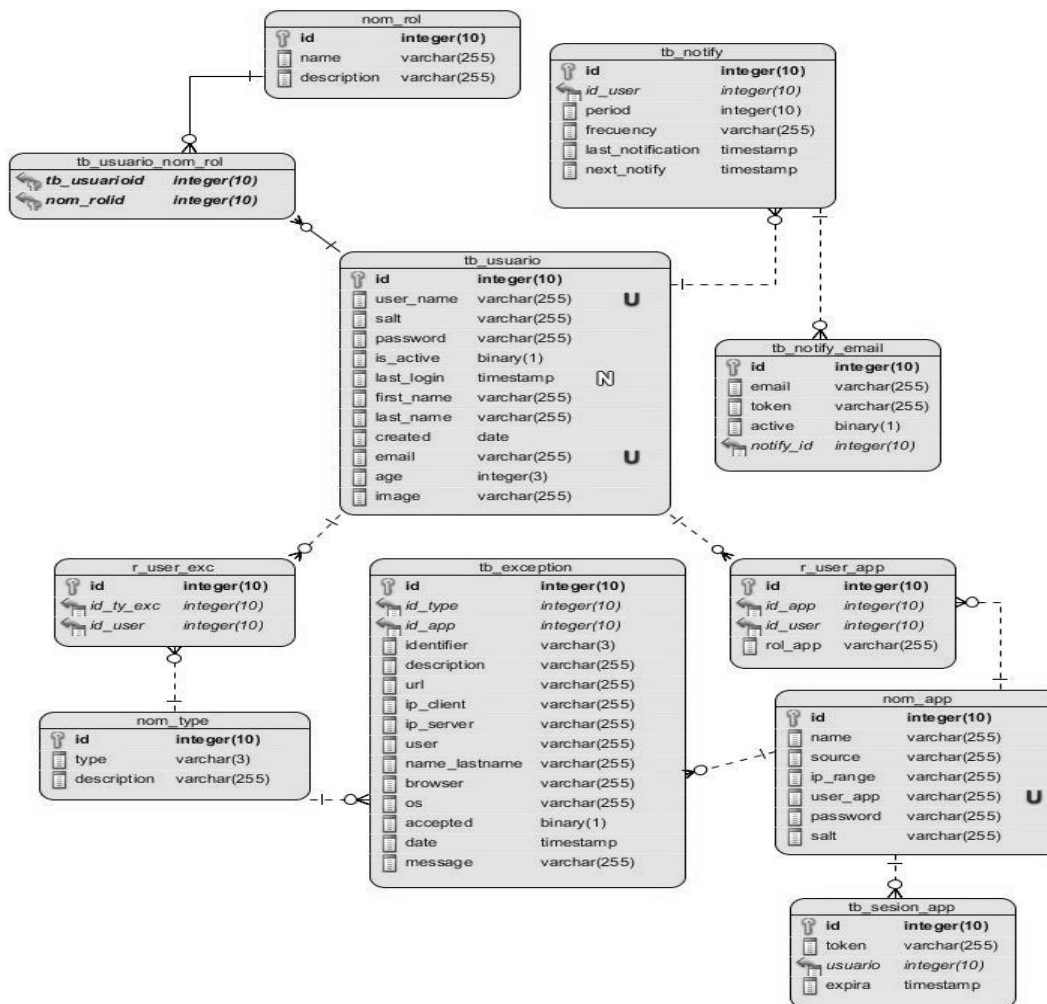


Figura 2.2 Diagrama Entidad-Relación

## 2.8 Conclusiones del capítulo

Las herramientas seleccionadas en el capítulo anterior demostraron ser consecuentes pues posibilitaron realizar el diseño de la aplicación. El estudio de la audiencia a la que va dirigida la solución delimitó que existen dos tipos de usuarios que interactúan con el sistema con diferentes niveles de acceso. Se definieron cuatro iteraciones que abarcan un total de diez historias de usuarios, que describen los aspectos principales a tener en cuenta para el desarrollo de la solución, brindando una visión futura de las funcionalidades a implementar.

Asociado a estas historias de usuarios se construyó el plan de entregas y siete tarjetas CRC que traducen los requerimientos a entidades a implementar para, de esta manera, pasar a la siguiente fase. Los principios, prácticas y técnicas propuestos por la metodología XP aportan los artefactos necesarios para la implementación de la solución.

## Capítulo 3 IMPLEMENTACIÓN Y PRUEBAS

En el proceso de desarrollo del *software* la fase de implementación es importante debido a que se desarrolla cada funcionalidad del sistema y se comprueba que se realizó exactamente lo que estaba propuesto.

La metodología utilizada en este trabajo tiene como fundamento del desarrollo de *software*, exigiendo que cada programador escriba las pruebas realizadas a su código de producción. Las pruebas se integran en el proceso de desarrollo y construcción del sistema. En este capítulo se describen las pruebas realizadas a la aplicación con el objetivo de verificar si se cumplieron las tareas de ingeniería planificadas para el desarrollo de la aplicación.

### 3.1 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí y se adaptan para resolver un problema de diseño general en un contexto particular. Los patrones de diseño pretenden proporcionar catálogos de elementos reusables en el diseño de sistemas de *software*; evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente; formalizar un vocabulario común entre los diseñadores y estandarizar el modo en que se realiza el diseño.

Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de forma tal que esa solución puede ser usada un millón de veces, sin hacerlo de la misma manera dos veces<sup>8</sup>.

#### 3.1.1 GRASP

GRASP es un acrónimo por sus siglas en inglés *General Responsibility Assignment Software Patterns*, la asignación eficiente de responsabilidades a los componentes del *software* es la actividad más importante en el análisis y diseño orientado a objetos. De los patrones GRASP existentes se utilizaron cinco que se refieren a cuestiones y aspectos fundamentales del diseño: Experto, Creador, Controlador, Bajo Acoplamiento y Alta Cohesión (40).

##### Experto

El patrón Experto resuelve el siguiente problema: ¿Cuál es el principio general para asignar responsabilidades a los objetos? La solución a dicho problema se resuelve asignando una responsabilidad al experto en información, la clase que tiene la información necesaria para realizar la responsabilidad.

Un Experto es una clase que tiene toda la información necesaria para implementar una responsabilidad.

---

<sup>8</sup> C. Alexander, "The Timeless Way of Building", 1979

Experto es uno de los patrones más utilizados al asignar responsabilidades, ya que expresa simplemente la intuición de que los objetos hacen cosas relacionadas con la información que poseen (41).

Entre los beneficios que provee el uso adecuado de este patrón, se encuentra el encapsulamiento de la información y la distribución del comportamiento del manejo de la información.

En la implementación de la propuesta de solución el uso de este patrón se evidencia por ejemplo, cuando es necesario renderizar las vistas para mostrarlas al usuario. En este caso las clases controladoras son las expertas en información y crean los formularios que permiten mostrar los campos requeridos en la vista.

```
public function add_appAction() {
    $em = $this->getDoctrine()->getManager();
    $peticion = $this->getRequest();
    $app = new \Support\ReportBundle\Entity\Entity\TbApp;
    $formulario = $this->createForm(new Form\AppType(), $app);
    $formulario->handleRequest($peticion);
    if ($peticion->isMethod("POST")) {
        if ($formulario->isValid()) {
            $em->persist($app);
            $em->flush();
        }
    }
    $appt = $em->getRepository('ReportBundle:Entity\TbApp')->findAll();
    return $this->render('ReportBundle:Default:add_app.html.twig', array(
        'formulario' => $formulario->createView(), 'aplicacion' => $appt));
}
```

**Figura 3.1** Uso del patrón Experto y Creador en el controlador de Reportes

### Creador

El patrón Creador resuelve el siguiente problema: ¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase? La solución a dicho problema es cuando B es un Creador de A si se asigna a la clase B la responsabilidad de crear una instancia de la clase A y 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 creado (por lo tanto, B es un Experto con respecto a la creación de A).



El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea común). La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación (41).

En el desarrollo de la propuesta de solución, la implementación de este patrón se evidencia cuando las clases controladoras crean los formularios ya que son las responsables de crear instancias de la clase Tipo (*Type*), también cuando la clase controladora crea las entidades que se van a asociar a un determinado formulario, ver Figura 3.1.

### **Bajo acoplamiento**

El patrón Bajo acoplamiento resuelve el siguiente problema: ¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización? La solución a dicho problema es: Asignar 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 (o débil) acoplamiento no depende demasiado de otros elementos.

El patrón Bajo Acoplamiento es un principio a tener en mente en todas las decisiones de diseño.

Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. El Bajo Acoplamiento soporta clases más independientes.

Entre sus beneficios se puede encontrar que no afectan los cambios en otros componentes; fácil de entender de manera aislada y conveniente para reutilizar (41).

### **Alta cohesión**

El patrón alta cohesión resuelve el siguiente problema: ¿Cómo mantener la complejidad manejable? su solución se da en asignar una responsabilidad de manera que la cohesión permanezca alta. En cuanto al diseño de objetos, la cohesión (cohesión funcional) es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Una clase con baja cohesión hace muchas cosas no relacionadas, o hace demasiado trabajo:

Clases difíciles de entender;

Difíciles de reutilizar;

Difíciles de mantener;

Delicadas, constantemente afectadas por los cambios.

Como regla empírica, una clase con alta cohesión tiene:

Un número relativamente pequeño de métodos, con funcionalidad altamente relacionada,

No realiza mucho trabajo.

Colabora con otros objetos para compartir el esfuerzo si la tarea es extensa. No es conveniente recargar el trabajo o incluir funcionalidad en la clase que responde a los eventos del sistema (41). Este patrón tiene como beneficio que incrementa la claridad y facilita la comprensión del diseño además de que simplifica el mantenimiento y las mejoras; soporta a menudo bajo acoplamiento e incrementa la reutilización.

### Controlador

El patrón Controlador resuelve el siguiente problema: ¿Quién debe ser el responsable de gestionar un evento de entrada del sistema? La solución a este problema es asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las siguientes operaciones:

Representa el sistema global, dispositivo o subsistema (Controlador de Fachada);

Representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de Sesión de Caso de Uso).

Utilizar la misma clase controlador para todos los eventos del sistema en el mismo escenario de caso de uso;

Una sesión es una instancia de una conversación con un actor (41).

En la estructura que propone Symfony2 este patrón se evidencia dentro de la carpeta *Controller* que se encuentra en los paquetes de cada *Bundle*<sup>9</sup>. En esta carpeta se localizan las clases controladoras de la Herramienta para el monitoreo de las aplicaciones web, que son las responsables de implementar todas las funcionalidades pertenecientes a una interfaz de un componente determinado, reciben los datos del usuario y los utilizan de acuerdo a la acción solicitada.

```
class DefaultController extends Controller {  
    public function indexAction() { ...5 lines }  
    public function serversAction($status) { ...7 lines }  
    public function listhttpadminAction() { ...7 lines }
```

**Figura 3.2** Ejemplo de uso del patrón Controlador

<sup>9</sup> **Bundle:** Un bundle no es más que un directorio que aloja todo aquello relativo a una funcionalidad determinada. Puede incluir clases PHP, plantillas, configuraciones, CSS's y Javascript. <http://juandarodriguez.es/tutoriales/tutorial-de-introduccion-a-symfony2/#bundles>.

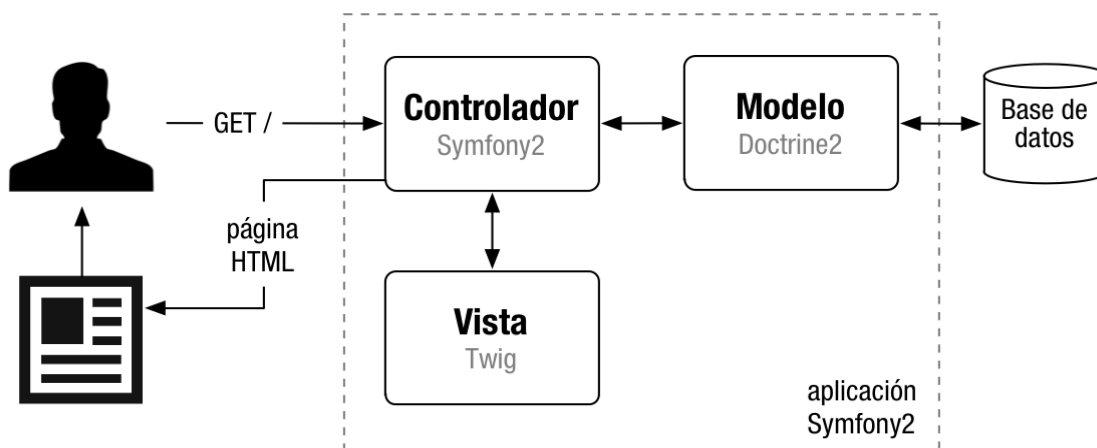
## 3.2 Patrones arquitectónicos

### 3.2.1 Modelo-Vista-Controlador (MVC)

Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. La propuesta de solución para el sistema hará uso del patrón Modelo – Vista – Controlador.

Modelo – Vista – Controlado (MVC)

Symfony basa su funcionamiento interno en la arquitectura MVC, utilizada por la mayoría de *frameworks* web. La utilización de dicho marco de trabajo para el desarrollo del producto conlleva a la implementación de este patrón. El siguiente esquema simplificado muestra el funcionamiento interno de Symfony (ver Figura 3.3) (42).



**Figura 3.3** Esquema simplificado de la arquitectura interna de Symfony (42)

Cuando el usuario realiza una solicitud, internamente sucede lo siguiente:

- 1) El sistema de enrutamiento determina que Controlador está asociado con la página de que responde a la solicitud.
- 2) Symfony ejecuta el Controlador asociado a la solicitud.
- 3) El Controlador solicita al Modelo los datos de la petición.
- 4) Con los datos devueltos por el Modelo, el Controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo.
- 5) El Controlador entrega al servidor la página creada por la Vista.

Con Symfony se pueden llegar a hacer cosas muy complejas, pero el funcionamiento interno siempre es el mismo:

- 1) El Controlador manda y ordena.
- 2) El Modelo busca la información que se le pide.

3) La Vista crea páginas con plantillas y datos.

La arquitectura MVC proporciona ventajas, como la organización del código, la reutilización y la flexibilidad. Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo.

### 3.3 Uso de componentes de terceros

La complejidad de los sistemas computacionales actuales, ha llevado a buscar la reutilización del *software* existente. El desarrollo de *software* basado en componentes permite reutilizar piezas de código pre-elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión. Para el desarrollo del Sistema de Monitorización de errores se utilizaron los siguientes componentes.

#### 3.3.1 Plug-in DataTable.js

DataTable.js es un plug-in para la biblioteca jQuery Javascript. Es una herramienta flexible, en base a los fundamentos de la mejora progresiva, que se sumarán los controles avanzados de interacción a cualquier tabla HTML. Está desarrollada bajo una licencia totalmente gratuita<sup>10</sup>.

#### 3.3.2 Librería AlivePDF.swc

AlivePDF.swc es una librería de código abierto, desarrollada con el fin de convertir a formato PDF, esta librería se emplea en el sistema para exportar los reportes que generan los usuarios<sup>11</sup>.

#### 3.3.3 Bundle SpraedPDFGeneratorBundle

SpraedPDFGeneratorBundle genera documentos HTML a PDF. El paquete te da la oportunidad de agregar un encabezado y pie de página con mucha facilidad. Funciona con una pequeña biblioteca jar. Por lo que necesita para ejecutar Java en el servidor (Java 6 o posterior). El sistema lo utiliza para generar los reportes que son enviados por correo en las notificaciones a los diferentes usuarios<sup>12</sup>.

#### 3.3.4 Bundle FosRestBundle

Este paquete proporciona varias herramientas para desarrollar rápidamente API RESTfull y aplicaciones con Symfony2. Las características incluyen<sup>13</sup>:

- Una capa de la Vista para permitir la salida y formatear Controladores agnósticos.
- Un cargador de ruta personalizada para generar siguientes convenciones REST de url.

<sup>10</sup> Para más información consulte <http://DataTables.net>

<sup>11</sup> Para más información consulte <http://www.alivepdf.org/>

<sup>12</sup> Para obtener información visite <http://www.spraed.com>

<sup>13</sup> Para más información visite <http://symfonhub.com/repo/FOSRestBundle/documentation>

- Acepta negociación de formato de cabecera incluyendo el manejo de tipos MIME personalizados.
- Decodificación REST de solicitud HTTP cuerpo y aceptar encabezados.
- Controlador de excepciones para el envío de códigos de estado HTTP apropiadas.

### 3.4 Desarrollo de las iteraciones

#### Tareas de ingeniería

Las tareas de ingeniería se usan para describir las tareas que se realizan sobre el proyecto, estas pueden ser: desarrollo, corrección, mejora, etc. Estas tareas tienen relación con una historia de usuario; se especifica la fecha de inicio y fin de la tarea, se nombra al programador responsable de cumplirla y se describe que se tratara de hacer en la tarea (15).

#### 3.4.1 Iteración 1

**Tabla 3.1** Tareas de Ingeniería 1: Servicio REST

Tareas de Ingeniería	
<b>Número tarea:</b> 1	<b>Número historia:</b> HU_1
<b>Nombre tarea:</b> Implementar el servicio REST.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 8 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar un servicio REST para la comunicación de las aplicaciones que se van a controlar con el sistema de monitoreo.	

**Tabla 3.2** Tareas de Ingeniería 2: API de Conexión

Tareas de Ingeniería	
<b>Número tarea:</b> 2	<b>Número historia:</b> HU_2
<b>Nombre tarea:</b> Implementar clase ApiREST para enviar el reporte.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 7 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar la clase de conexión encargada de utilizar el servicio para enviar los datos al sistema.	

### 3.4.2 Iteración 2

**Tabla 3.3** Tareas de Ingeniería 3: Registrar aplicación

Tareas de Ingeniería	
<b>Número tarea:</b> 3	<b>Número historia:</b> HU_3
<b>Nombre tarea:</b> Implementar la funcionalidad de registrar aplicación.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 5 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar la funcionalidad de registrar las aplicaciones que van a ser uso del sistema de monitoreo de la solución propuesta.	

**Tabla 3.4** Tareas de Ingeniería 4: Listar aplicación

Tareas de Ingeniería	
<b>Número tarea:</b> 4	<b>Número historia:</b> HU_4
<b>Nombre tarea:</b> Implementar la funcionalidad de listar aplicación.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 2 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar la funcionalidad de listar las aplicaciones que están siendo controladas por el sistema de monitoreo.	

**Tabla 3.5** Tareas de Ingeniería 5: Eliminar aplicación

Tareas de Ingeniería	
<b>Número tarea:</b> 5	<b>Número historia:</b> HU_4
<b>Nombre tarea:</b> Implementar la funcionalidad de eliminar aplicación.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 1 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar la funcionalidad de eliminar las aplicaciones que existen en el sistema.	

Tabla 3.6 Tareas de Ingeniería 6: Visualizar aplicación

Tareas de Ingeniería	
<b>Número tarea:</b> 6	<b>Número historia:</b> HU_5
<b>Nombre tarea:</b> Implementar la funcionalidad de visualizar aplicación.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 2 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar la funcionalidad de realizar una vista detallada de las aplicaciones que existen en el sistema.	

### 3.4.3 Iteración 3

Tabla 3.7 Tareas de Ingeniería 7: Listar errores

Tareas de Ingeniería	
<b>Número tarea:</b> 7	<b>Número historia:</b> HU_6
<b>Nombre tarea:</b> Implementar la funcionalidad de listar errores.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 3 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> listar errores generará una vista con todos los errores guardados en el sistema que estén relacionados con el usuario activo.	

Tabla 3.8 Tareas de Ingeniería 8: Filtrar errores

Tareas de Ingeniería	
<b>Número tarea:</b> 8	<b>Número historia:</b> HU_6
<b>Nombre tarea:</b> Implementar la funcionalidad de filtrar.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 3 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> selección de los campos para realizar los filtros. Implementación de filtros sin que se recargue la página en el navegador.	

Tabla 3.9 Tareas de Ingeniería 9: Eliminar error

Tareas de Ingeniería	
<b>Número tarea:</b> 9	<b>Número historia:</b> HU_6
<b>Nombre tarea:</b> Implementar la funcionalidad de eliminar un error.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 1 día
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> todos los usuarios deben tener acceso a esta funcionalidad.	

Tabla 3.10 Tareas de Ingeniería 10: Editar error

Tareas de Ingeniería	
<b>Número tarea:</b> 10	<b>Número historia:</b> HU_6
<b>Nombre tarea:</b> Implementar la funcionalidad de marcar error como revisado.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 1 día
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> esta funcionalidad será la encargada de actualizar un error marcando como TRUE el capo aceptado de la tabla de error.	

Tabla 3.11 Tareas de Ingeniería 11: Exportar reporte

Tareas de Ingeniería	
<b>Número tarea:</b> 11	<b>Número historia:</b> HU_7
<b>Nombre tarea:</b> Implementar la funcionalidad de exportar un reporte.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 5 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> se encargará de exportar reportes a formatos PDF y XLS o crear la vista de impresión del navegador.	

Tabla 3.12 Tareas de Ingeniería 12: Visualizar errores

Tareas de Ingeniería	
<b>Número tarea:</b> 12	<b>Número historia:</b> HU_8



<b>Nombre tarea:</b> Implementar la funcionalidad de visualizar errores	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 2 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> implementar una vista detallada de los errores.	

### 3.4.4 Iteración 4

**Tabla 3.13** Tareas de Ingeniería 13: Adicionar notificaciones

Tareas de Ingeniería	
<b>Número tarea:</b> 13	<b>Número historia:</b> HU_9
<b>Nombre tarea:</b> Implementar la funcionalidad adicionar notificaciones.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 7 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> cada usuario podrá crear sus notificaciones y especificar a los desarrolladores que desea enviar la misma notificación.	

**Tabla 3.14** Tareas de Ingeniería 14: Listar notificaciones

Tareas de Ingeniería	
<b>Número tarea:</b> 14	<b>Número historia:</b> HU_10
<b>Nombre tarea:</b> Implementar la funcionalidad listar notificaciones.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 2 días
<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez	
<b>Descripción:</b> los usuarios podrán listar sus notificaciones.	

**Tabla 3.15** Tareas de Ingeniería 15: Eliminar notificaciones

Tareas de Ingeniería	
<b>Número tarea:</b> 15	<b>Número historia:</b> HU_10
<b>Nombre tarea:</b> Implementar la funcionalidad eliminar notificaciones.	
<b>Tipo tarea:</b> desarrollo	<b>Puntos estimados:</b> 1 días

<b>Programador responsable:</b> Danis Carlos Chaviano Jiménez
<b>Descripción:</b> cada usuario podrá eliminar sus notificaciones del listado.

### 3.5 Pruebas de software

Las pruebas de *software*, en inglés *testing* son los procesos que permiten verificar y revelar la calidad de un producto *software*. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de *software* consistente en probar las aplicaciones construidas.

#### 3.5.1 Diseño de las pruebas

Después de un estudio bibliográfico se arribó a la conclusión de que la metodología XP propone un modelo inverso al propuesto por las metodologías tradicionales en cuanto a la fase de prueba se refiere, las últimas proponen la descripción y realización de las mismas al final del proyecto o finalización de cada módulo, la primera propone que lo primero a realizar es la descripción de las pruebas a pasar por el sistema (37).

- **Pruebas de integración**

*“La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar error es asociados con la interacción. El objetivo es tomar los módulos probados mediante la prueba de aceptación y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño”* (43).

Luego de realizar las pruebas de integración del *bundle Report* en la aplicación “ZERA Support”, se obtuvo como resultado el correcto funcionamiento de las HU.

- **Pruebas de aceptación**

Luego del análisis bibliográfico se arribó a la conclusión que el objetivo de estas pruebas es verificar los aspectos funcionales, por este motivo, las propias funcionalidades del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación.

Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. La garantía de calidad es una parte esencial en el proceso de XP (43).

A continuación se muestran las pruebas de aceptación realizadas al sistema.

Tabla 3.16 Prueba de aceptación 1: Servicio REST

Caso de Prueba Aceptación	
<b>Código:</b> HU1_P01	<b>Historia de Usuario:</b> HU_1
<b>Nombre:</b> Implementación del Servicio REST	
<b>Descripción:</b> prueba para la implementación del servicio REST.	
<b>Condiciones de Ejecución:</b> para la utilización de este servicio, previamente la aplicación que va a ser uso del mismo debe estar registrada en el sistema de monitoreo.	
<b>Entrada/Pasos de Ejecución:</b> mediante la clase de conexión se hace uso del servicio enviando un reporte en formato JSON con los datos requeridos de los errores ocurridos desde el último envío satisfactorio.	
<b>Resultado Esperado:</b> se actualiza la base de datos.	
<b>Evaluación de la Prueba:</b> prueba satisfactoria.	

Tabla 3.17 Prueba de aceptación 2: Clase de conexión con las aplicaciones

Caso de Prueba Aceptación	
<b>Código:</b> HU2_P02	<b>Historia de Usuario:</b> HU_2
<b>Nombre:</b> Clase de conexión con las aplicaciones.	
<b>Descripción:</b> prueba para la funcionalidad de guardar los errores en la utilización del servicio REST para consumir las opciones del sistema.	
<b>Condiciones de Ejecución:</b> las aplicaciones deben tener un sistema de captura de errores implementado.	
<b>Entrada/Pasos de Ejecución:</b> el sistema de captura de errores de la aplicación debe utilizar la clase de conexión donde se muestra la forma de consumir los servicios del sistema.	
<b>Resultado Esperado:</b> si los campos son correctos se actualiza la base de datos y se devuelve una respuesta para informar que no existieron problemas en la conexión.	
<b>Evaluación de la Prueba:</b> prueba satisfactoria.	

**Tabla 3.18** Prueba de aceptación 3: Gestionar aplicaciones

Caso de Prueba Aceptación	
<b>Código:</b> HU3_HU4_P03	<b>Historia de Usuario:</b> HU_3, HU_4
<b>Nombre:</b> Gestionar aplicaciones.	
<b>Descripción:</b> prueba para la funcionalidad de adicionar, listar y eliminar las aplicaciones que se van a monitorear por el sistema.	
<b>Condiciones de Ejecución:</b> el usuario tiene que estar autenticado y ser administrador del sistema.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el administrador en el sistema debe ir al submenú “Gestionar Aplicaciones” del menú izquierdo “Administración”, se mostrará una vista con los campos requeridos para adicionar una aplicación y una tabla con la lista de aplicaciones que existen en el sistema. Para adicionar una aplicación se deben llenar los campos correctamente teniendo en cuenta que la aplicación no exista en el sistema. Para eliminar una aplicación se debe seleccionar “Eliminar” del submenú opciones en la tabla que lista las aplicaciones existentes.	
<b>Resultado Esperado:</b> si se adiciona una aplicación y los campos son correctos actualiza la base de datos y se muestra la nueva aplicación en la lista de aplicaciones, en caso de que ocurra un error en la validación de los campos se muestra un mensaje indicando el campo incorrecto. Si se desea eliminar una aplicación, aparecerá un mensaje para confirmar la acción, se actualiza la base de datos, los cambios se mostrarán automáticamente en la lista de aplicaciones.	
<b>Evaluación de la Prueba:</b> prueba satisfactoria.	

**Tabla 3.19** Prueba de aceptación 4: Visualizar detalles de las aplicaciones

Caso de Prueba Aceptación	
<b>Código:</b> HU5_P04	<b>Historia de Usuario:</b> HU_5
<b>Nombre:</b> Visualizar detalles de las aplicaciones.	
<b>Descripción:</b> prueba para la funcionalidad de visualizar los detalles de las aplicaciones.	
<b>Condiciones de Ejecución:</b> el usuario tiene que estar autenticado y ser administrador del sistema.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el administrador en el sistema debe ir al submenú “Gestionar Aplicaciones” del menú izquierdo “Administración”, se mostrará una vista con la lista de aplicaciones que existen en el sistema. Para mostrar los detalles de una aplicación se debe seleccionar el botón “Detalles” identificado con un icono de color azul, en la parte derecha de la tabla que lista las aplicaciones.	

**Resultado Esperado:** se mostrará una ventana con los detalles de la aplicación que quedará por delante del sistema oscureciendo a este hasta que se dé *click* en un área externa a la ventana de los detalles o en el icono cerrar identificado por una X en la parte superior derecha de la ventana.

**Evaluación de la Prueba:** prueba satisfactoria.

**Tabla 3.20** Prueba de aceptación 5: Listar errores

Caso de Prueba Aceptación	
<b>Código:</b> HU6_P05	<b>Historia de Usuario:</b> HU_6
<b>Nombre:</b> Listar errores.	
<b>Descripción:</b> prueba para la funcionalidad listar errores.	
<b>Condiciones de Ejecución:</b> el usuario tiene que estar autenticado, en caso de no ser administrador debe haber especificado en el perfil los tipos de errores que desea monitorizar.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el usuario en el sistema debe ir al submenú "Errores" del menú izquierdo "Reporte".	
<b>Resultado Esperado:</b> se mostrará una vista con el listado de todos los errores que existen en el sistema, que pertenezcan al tipo de errores y aplicación que el usuario monitorea, en caso de ser administrador se mostrarán todos los errores que se han insertado en el sistema.	
<b>Evaluación de la Prueba:</b> prueba satisfactoria.	

**Tabla 3.21** Prueba de aceptación 6: Opciones de los errores

Caso de Prueba Aceptación	
<b>Código:</b> HU6_HU8_P06	<b>Historia de Usuario:</b> HU_6, HU_8
<b>Nombre:</b> Opciones de los errores.	
<b>Descripción:</b> prueba para la funcionalidad editar, visualizar y eliminar errores.	
<b>Condiciones de Ejecución:</b> el usuario tiene que estar autenticado, en caso de no ser administrador debe haber especificado en el perfil los tipos de errores que desea monitorizar.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el usuario en el sistema debe acceder al listado de errores dando <i>click</i> en el submenú "Errores" del menú izquierdo "Reporte". En la parte derecha de cada error aparecerán las opciones de "Eliminar" (icono rojo), "Detalles" (icono azul) y "Editar" (icono verde).	
<b>Resultado Esperado:</b> si el error aparece de color rojo, solo se mostrará las opciones "Detalles" y "Editar", esto significa que el error no se ha revisado, para marcar un error como revisado se debe dar <i>click</i> en el botón "Editar", esto cambiaría el error a color verde. Si el error se muestra de color verde,	

significa que ya ha sido revisado y aparecerá en las opciones “Detalles” y “Eliminar”. Si se selecciona la opción “Detalles” se mostrará una ventana emergente con los detalles del error, oscureciendo el sistema, esta ventana se puede cerrar dando *click* en un área externa a la misma o dando *click* en el icono “Cerrar” identificado por una X en la parte superior derecha de la ventana. En caso que se desee eliminar un error se debe acceder a la opción del submenú “Eliminar”, se mostrará un mensaje de confirmación, al dar *click* en eliminar se actualiza la base de datos y el listado de errores.

**Evaluación de la Prueba:** prueba satisfactoria.

**Tabla 3.22** Prueba de aceptación 7: Filtrar errores

Caso de Prueba Aceptación	
<b>Código:</b> HU6_P07	<b>Historia de Usuario:</b> HU_6
<b>Nombre:</b> Filtrar errores.	
<b>Descripción:</b> prueba para la funcionalidad de filtrar los errores por distintos campos.	
<b>Condiciones de Ejecución:</b> el usuario tiene que estar autenticado, en caso de no ser administrador debe haber especificado en el perfil los tipos de errores que desea monitorizar.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el usuario en el sistema debe ir al submenú “Errores” del menú izquierdo “Reporte”. Si se desea especificar los campos que se quieren mostrar de los errores, se debe dar <i>click</i> en el botón Columnas identificado por un icono de una tabla, que aparece en la parte superior derecha de la tabla con el listado de errores.	
<b>Resultado Esperado:</b> se mostrará una vista con el listado de los errores, en la parte superior aparecerán distintos campos por los cuáles se puedan realizar diferentes filtros, filtrar los datos se deben mostrar los cambiar automáticamente en el listado sin que la página sea recargada.	
<b>Evaluación de la Prueba:</b> prueba satisfactoria.	

**Tabla 3.23** Prueba de aceptación 8: Generar reportes

Caso de Prueba Aceptación	
<b>Código:</b> HU7_P08	<b>Historia de Usuario:</b> HU_7
<b>Nombre:</b> Generar Reportes.	
<b>Descripción:</b> prueba para la funcionalidad de generar reportes de errores.	
<b>Condiciones de Ejecución:</b> El usuario tiene que estar autenticado, en caso de no ser administrador debe haber especificado en el perfil los tipos de errores que desea monitorizar.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el usuario en el sistema, se debe acceder al listado de errores dando <i>click</i> en el submenú “Errores” del menú izquierdo “Reporte”. En la parte	

superior derecha de la tabla aparecerán los botones para generar un reporte a diferentes formatos identificados por el icono de la extensión a la cual se desea exportar el reporte.

**Resultado Esperado:** se creará un archivo con la extensión seleccionada una vez se especifique la ubicación, este archivo contendrá todos los errores que son monitorizados por el usuario.

**Evaluación de la Prueba:** prueba satisfactoria.

**Tabla 3.24** Prueba de aceptación 9: Adicionar notificaciones

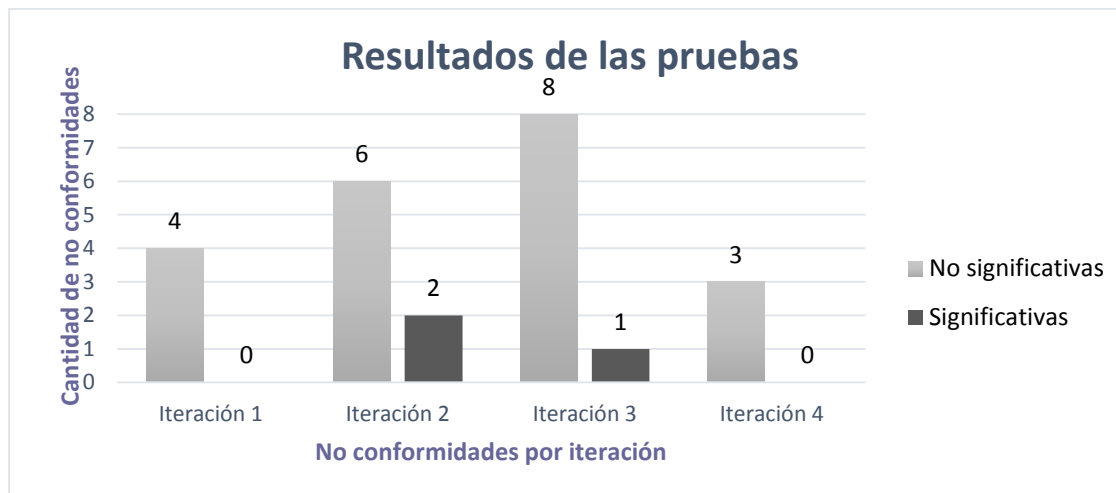
Caso de Prueba Aceptación	
<b>Código:</b> HU9_P09	<b>Historia de Usuario:</b> HU_9
<b>Nombre:</b> Administrar las notificaciones.	
<b>Descripción:</b> prueba para la funcionalidad de adicionar las notificaciones.	
<b>Condiciones de Ejecución:</b> el usuario tiene que estar autenticado.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el usuario en el sistema, se debe acceder al submenú “Adicionar” del menú izquierdo “Notificaciones”.	
<b>Resultado Esperado:</b> se creará una vista con los campos necesarios para añadir un nuevo periodo de notificación. El usuario podrá adicionar tantas direcciones de correo desee enviar esa notificación. Se enviará un correo de confirmación a las direcciones adicionadas por el usuario.	
<b>Evaluación de la Prueba:</b> prueba satisfactoria.	

**Tabla 3.25** Prueba de aceptación 10: Listar y eliminar notificaciones

Caso de Prueba Aceptación	
<b>Código:</b> HU9_P10	<b>Historia de Usuario:</b> HU_9
<b>Nombre:</b> Listar y eliminar las notificaciones.	
<b>Descripción:</b> prueba para la funcionalidad de Listar las notificaciones.	
<b>Condiciones de Ejecución:</b> El usuario tiene que estar autenticado.	
<b>Entrada/Pasos de Ejecución:</b> una vez autenticado el usuario en el sistema, se debe acceder al submenú “Listar” del menú izquierdo “Notificaciones”.	
<b>Resultado Esperado:</b> se creará una vista con el listado de todas las notificaciones que el usuario tiene registrada. Cada notificación dará la opción de ser eliminada. Si se da <i>click</i> en el botón eliminar, aparecerá un mensaje para confirmar la acción.	

**Evaluación de la Prueba:** prueba satisfactoria.

### 3.6 Resultados de las pruebas



**Gráfica 3.1** Resultados de las pruebas

En el transcurso de las iteraciones se le realizaron las pruebas de aceptación al sistema, se encontraron un total de veinticuatro no conformidades, de ellas tres fueron consideradas significativas según su importancia en el negocio. Al concluir las iteraciones, el equipo de desarrollo dio solución a la totalidad de las no conformidades detectadas, priorizando las clasificadas en estado significativo.

### 3.7 Resultados obtenidos

Se ha creado un sistema que brinda a los desarrolladores la posibilidad de realizar monitoreo de los errores generados en las aplicaciones web. El sistema se encarga de notificar los reportes, garantizando así un constante control del estado de las aplicaciones. Con el uso del servicio “reportar” se establece la conexión entre el Sistema de monitoreo y las aplicaciones a controlar.

### 3.8 Conclusiones del capítulo

- Se facilitó la implementación de la propuesta de solución como resultado del estudio bibliográfico y de las soluciones similares, que permitió la selección de las herramientas, tecnologías y lenguajes de programación adecuados.
- Se constató que cada funcionalidad se corresponde con lo definido inicialmente por el cliente.
- El análisis de las pruebas realizadas validó que las herramientas y metodologías seleccionadas para solucionar el Problema de la investigación es la correcta, pues la cantidad de no conformidades que prevalecen es las no significativas que son mínimas.



## CONCLUSIONES GENERALES

La evolución de la presente investigación permitió la obtención de una herramienta para el monitoreo de errores en la aplicación web. Esta cumple con todos los aspectos funcionales básicos para su desarrollo, cumpliendo así con los objetivos trazados para alcanzar los resultados esperados.

Estos resultados permitieron concluir los siguientes enunciados:

- La fundamentación teórica realizada en la investigación posibilitó justificar la selección de la metodología de desarrollo, los patrones de diseño, de arquitectura y las herramientas a utilizar para el desarrollo del sistema.
- La herramienta para el monitoreo de aplicaciones web proporcionó la posibilidad de generar reportes de los errores, administrar las notificaciones y ofrecer un alto nivel de seguridad en la transferencia de los datos mediante el servicio REST.
- La propuesta de solución al problema identificado contribuyó al logro del control constante de la calidad de las aplicaciones web.
- Se comprobó la efectividad de la solución propuesta a partir de los resultados satisfactorios obtenidos en las evaluaciones internas realizadas por el proyecto, pues la totalidad de las no conformidades detectadas durante las iteraciones de pruebas definidas, fueron resueltas por el equipo de desarrollo.

## **RECOMENDACIONES**

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

1. Se recomienda al cliente para posteriores versiones, adicionar retroalimentación a los errores con el fin de proveer a los desarrolladores de herramientas para una rápida solución de errores similares.
2. A la dirección del centro FORTES, generalizar el presente trabajo a los demás Centros de desarrollo de la UCI.

**REFERENCIAS BIBLIOGRÁFICAS**

1. **Cabrera, A. M. C.** 2006. *Impacto de las TIC en la educación: un acercamiento desde el punto de vista de las funciones de la educación*. ISSN 1575-9393.
2. **Labrada, Sonia Morejón.** 2011. El software educativo un medio de enseñanza eficiente. [En línea] <http://www.eumed.net/rev/ced/29/sml.pdf>.
3. **Internet Assigned Numbers Authority.** Hypertext Transfer Protocol (HTTP) Status Code Registry. [En línea] Internet Assigned Numbers Authority, 17 de febrero de 2014. [www.iana.org/assignments/http-status-codes/http-status-codes.xhtml](http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml).
4. **AltoNivel.com.mx.** 3 herramientas para medir el desempeño de tu sitio. [En línea] <http://www.altonivel.com.mx/33627-3-herramientas-para-medir-el-desempeno-de-tu-sitio.html>.
5. **ProgramaSoftware.com.** Herramientas Alexa para sitios web. [En línea] <http://www.programasoftware.com/herramientas-alexa-para-sitios-web/>.
6. **Baluart.net.** Site24x7, para Monitorear Sitios Web. [En línea] 6 de marzo de 2007. <http://www.baluart.net/articulo/site24x7-para-monitorear-sitios-web>.
7. **Herramientas 2.0.** Herramientas Web 2.0 para el trabajo en competencias. [En línea] <http://herramientas20.wordpress.com/category/herramientas-20/pagina-de-inicio/netvibes/>.
8. **ManageEngine.** Funciones de monitoreo de redes. [En línea] 2013. <http://www.manageengine.com.mx/network-monitoring/url-monitoring.html>.
9. **Dos ideas.** Ágil. [En línea] [http://www.dosideas.com/wiki/Desarrollo\\_Agil\\_De\\_Software](http://www.dosideas.com/wiki/Desarrollo_Agil_De_Software).
10. **Ecured.** Metodologías de desarrollo de software. [En línea] [http://www.ecured.cu/index.php/Metodologías\\_de\\_desarrollo\\_de\\_software](http://www.ecured.cu/index.php/Metodologías_de_desarrollo_de_software).
11. **Cohen, Mike.** Una Introducción a Scrum. [En línea] Noviembre de 2013. <http://www.dc.fi.udc.es/~cabalar/md/scrum.pdf>.
12. **Carnós, José H. (et. al.).** 2003. *Metodologías Ágiles en el Desarrollo de Software*. Alicante - España : Grupo ISSI.
13. **Palacio, J.** 2007. *Flexibilidad con Scrum, principios de diseño e implantación en campos Scrum*. s.l. : SafeCreative.
14. **Faculta de Informática, Cs. De la Comunicación y Téc. Especiales.** Morón, Universidad. Metodologías Ágiles. [En línea] [http://noqualityinside.com/nqi/nqifiles/Metodologias\\_Agiles.pdf](http://noqualityinside.com/nqi/nqifiles/Metodologias_Agiles.pdf).
15. **Beck, Kent.** 1999. *Extreme Programming Explained: Embrace Change*. s.l. : Addison-Wesley Pub Co.
16. **Wikiudo.** Metodologías SCRUM y XP. [En línea] <http://wiki.monagas.udo.edu.ve/>.
17. **Wesley, Addison.** *Una explicación de la programación extrema. Aceptar el cambio*.

18. **Rumbaugh, James, J, I y Booch, Gragy.** 2000. *El Lenguaje Unificado de Modelado*. Madrid : s.n. pág. 528, Manual de Referencia. ISBN 84-7829-037-0.
19. **Dpto. de Sistemas de Informáticos y Computación.** *Introducción a Herramientas CASE y System Architect*. UNIVERSIDAD POLITÉCNICA DE VALENCIA.
20. **UML CASE.** UML CASE tool for software development. [En línea] <http://www.visual-paradigm.com/product/vpuml/>.
21. **Franganillo, Jorge.** HTML5: el nuevo estándar básico del web. [En línea] [Citado el: 29 de Enero de 2014.] <http://thinkepi.net/html5-nuevo-estandar-basico-del-web>.
22. **W3C.** HTML & CSS - W3C. [En línea] <http://www.w3.org/standards/webdesign/htmlcss#whatcss>.
23. **Suárez Santos, Humberto y Verdecia Alá, Pedro Manuel.** 2008. *Arquitectura de Software General de Auditoría*. La Habana : s.n. SN.
24. **PHP.net.** PHP Nuevas Características-Manual. *PHP.net*. [En línea] [Citado el: 30 de Enero de 2014.] <http://www.php.net/manual/es/migration54.new-features.php>.
25. **Lic. Pacheco Aguila, Yoandry.** *AJAX un nuevo acercamiento a las aplicaciones Web (página 2) - Monografias.com*.
26. **PHP.net.** ¿Qué se puede hacer con PHP? *PHP.net*. [En línea] [Citado el: 20 de Enero de 2014.] <http://www.php.net/manual/es/intro-whatcando.php>.
27. **Alvarez, Miguel Angel.** Manual de JQuery. [En línea] 2010. [Citado el: 30 de Enero de 2014.] <http://www.biblioteca-digital.net.ve/wordpress/wp-content/uploads/2010/07/manualjquery.pdf>.
28. **twitterBootstrap.** Bootstrap. *Bootstrap*. [En línea] [Citado el: 15 de Noviembre de 2013.] <http://twitter.github.io/bootstrap/index.html>.
29. **Potencier, Fabien.** Symfony 2.4.0 released - Symfony. [En línea] Diciembre de 2013. <http://symfony.com/blog/symfony-2-4-0-released>.
30. **Symfony.es.** Se publica Symfony 2.4.0. [En línea] <http://symfony.es/noticias/2013/12/04/se-publica-symfony-240/>.
31. **Ecured.** Ecured. *Sistema de Gestión de Base de Datos*. [En línea] [http://www.ecured.cu/index.php/Sistema\\_Gestor\\_de\\_Base\\_de\\_Datos](http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos).
32. **Apache.** Welcome! - The Apache HTTP Server Project. [En línea] <http://httpd.apache.org/>.
33. **NetBeans.** NetBeans IDE 7.4 Release Information. [En línea] <https://netbeans.org/community/releases/74/index.html>.
34. **w3c.** Guía Breve de Servicios Web. [En línea] <http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb>.

35. **Amodeo, Enrique.** Servicios web. [En línea]  
<http://eamodeorubio.wordpress.com/2010/07/26/servicios-web-2-%C2%BFque-es-rest/>.
36. **González Palacio, Liliana y Urrego Giraldo, Germán.** 2010. Modelo de contexto y de dominio para la ingeniería de requisitos. [En línea]  
<http://webapps.udem.edu.co/RevistaIngenierias/pdf/V9n17/P%E1ginas%20desdeRevista%20INGENIERIAS%20vo.%209%20No.%2017%20ARTICULO%2012.pdf>. 151-64 - ISSN - 1692-3324.
37. **Ing. Pérez Ramírez, Danay.** 2008. CUJAE. Metodologías Ágiles. ¿Cómo desarrollar utilizando XP?
38. **Joskowicz, José.** 2008. *Reglas y Prácticas en eXtreme Programming*. España : s.n.
39. **Carvajal Riola, Jose Carlos.** 2008. *Metodologías Ágiles*. UPC - Barcelona. Barcelona : s.n. pág. 215, Tesis Final de Máster.
40. **Beck, K.** *Extreme Programming Explained. Embrace Change*. Pearson Education.
41. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingeniería de Software*. Departamento de Informática, Universidad Técnica Federico Santa María.
42. **Eguiluz, Javier.** 2013. Desarrollo web con Symfony2.
43. **Ruiz Tenorio, Roberto.** Las Pruebas de Software y su Importancia en las Organizaciones. [En línea] Agosto de 2010.  
<http://cdigital.uv.mx/bitstream/123456789/28540/1/Ruiz%20Tenorio.pdf>.

**BIBLIOGRAFÍA**

1. **Cabrera, A. M. C.** 2006. *Impacto de las TIC en la educación: un acercamiento desde el punto de vista de las funciones de la educación*. ISSN 1575-9393.
2. **Labrada, Sonia Morejón.** 2011. El software educativo un medio de enseñanza eficiente. [En línea] <http://www.eumed.net/rev/ced/29/sml.pdf>.
3. **Internet Assigned Numbers Authority.** Hypertext Transfer Protocol (HTTP) Status Code Registry. [En línea] Internet Assigned Numbers Authority, 17 de febrero de 2014. [www.iana.org/assignments/http-status-codes/http-status-codes.xhtml](http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml).
4. **AltoNivel.com.mx.** 3 herramientas para medir el desempeño de tu sitio. [En línea] <http://www.altonivel.com.mx/33627-3-herramientas-para-medir-el-desempeno-de-tu-sitio.html>.
5. **ProgramaSoftware.com.** Herramientas Alexa para sitios web. [En línea] <http://www.programasoftware.com/herramientas-alexa-para-sitios-web/>.
6. **Baluart.net.** Site24x7, para Monitorear Sitios Web. [En línea] 6 de marzo de 2007. <http://www.baluart.net/articulo/site24x7-para-monitorear-sitios-web>.
7. **Herramientas 2.0.** Herramientas Web 2.0 para el trabajo en competencias. [En línea] <http://herramientas20.wordpress.com/category/herramientas-20/pagina-de-inicio/netvibes/>.
8. **ManageEngine.** Funciones de monitoreo de redes. [En línea] 2013. <http://www.manageengine.com.mx/network-monitoring/url-monitoring.html>.
9. **Dos ideas.** Ágil. [En línea] [http://www.dosideas.com/wiki/Desarrollo\\_Agil\\_De\\_Software](http://www.dosideas.com/wiki/Desarrollo_Agil_De_Software).
10. **Ecured.** Metodologías de desarrollo de software. [En línea] [http://www.ecured.cu/index.php/Metodologías\\_de\\_desarrollo\\_de\\_software](http://www.ecured.cu/index.php/Metodologías_de_desarrollo_de_software).
11. **Cohen, Mike.** Una Introducción a Scrum. [En línea] Noviembre de 2013. <http://www.dc.fi.udc.es/~cabalar/md/scrum.pdf>.
12. **Carnós, José H. (et. al.).** 2003. *Metodologías Ágiles en el Desarrollo de Software*. Alicante - España : Grupo ISSI.
13. **Palacio, J.** 2007. *Flexibilidad con Scrum, principios de diseño e implantación en campos Scrum*. s.l. : SafeCreative.
14. **Faculta de Informática, Cs. De la Comunicación y Téc. Especiales.** Morón, Universidad. Metodologías Ágiles. [En línea] [http://noqualityinside.com/nqi/nqifiles/Metodologias\\_Agiles.pdf](http://noqualityinside.com/nqi/nqifiles/Metodologias_Agiles.pdf).
15. **Beck, Kent.** 1999. *Extreme Programming Explained: Embrace Change*. s.l. : Addison-Wesley Pub Co.
16. **Wikiudo.** Metodologías SCRUM y XP. [En línea] <http://wiki.monagas.udo.edu.ve/>.
17. **Wesley, Addison.** *Una explicación de la programación extrema. Aceptar el cambio*.

18. **Rumbaugh, James, J, I y Booch, Gragy.** 2000. *El Lenguaje Unificado de Modelado*. Madrid : s.n. pág. 528, Manual de Referencia. ISBN 84-7829-037-0.
19. **Dpto. de Sistemas de Informáticos y Computación.** *Introducción a Herramientas CASE y System Architect*. UNIVERSIDAD POLITÉCNICA DE VALENCIA.
20. **UML CASE.** UML CASE tool for software development. [En línea] <http://www.visual-paradigm.com/product/vpuml/>.
21. **Franganillo, Jorge.** HTML5: el nuevo estándar básico del web. [En línea] [Citado el: 29 de Enero de 2014.] <http://thinkepi.net/html5-nuevo-estandar-basico-del-web>.
22. **W3C.** HTML & CSS - W3C. [En línea] <http://www.w3.org/standards/webdesign/htmlcss#whatcss>.
23. **Suárez Santos, Humberto y Verdecia Alá, Pedro Manuel.** 2008. *Arquitectura de Software General de Auditoría*. La Habana : s.n. SN.
24. **PHP.net.** PHP Nuevas Características-Manual. *PHP.net*. [En línea] [Citado el: 30 de Enero de 2014.] <http://www.php.net/manual/es/migration54.new-features.php>.
25. **Lic. Pacheco Aguila, Yoandry.** *AJAX un nuevo acercamiento a las aplicaciones Web (página 2) - Monografias.com*.
26. **PHP.net.** ¿Qué se puede hacer con PHP? *PHP.net*. [En línea] [Citado el: 20 de Enero de 2014.] <http://www.php.net/manual/es/intro-whatcando.php>.
27. **Alvarez, Miguel Angel.** Manual de JQuery. [En línea] 2010. [Citado el: 30 de Enero de 2014.] <http://www.biblioteca-digital.net.ve/wordpress/wp-content/uploads/2010/07/manualjquery.pdf>.
28. **twitterBootstrap.** Bootstrap. *Bootstrap*. [En línea] [Citado el: 15 de Noviembre de 2013.] <http://twitter.github.io/bootstrap/index.html>.
29. **Potencier, Fabien.** Symfony 2.4.0 released - Symfony. [En línea] Diciembre de 2013. <http://symfony.com/blog/symfony-2-4-0-released>.
30. **Symfony.es.** Se publica Symfony 2.4.0. [En línea] <http://symfony.es/noticias/2013/12/04/se-publica-symfony-240/>.
31. **Ecured.** Ecured. *Sistema de Gestión de Base de Datos*. [En línea] [http://www.ecured.cu/index.php/Sistema\\_Gestor\\_de\\_Base\\_de\\_Datos](http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos).
32. **Apache.** Welcome! - The Apache HTTP Server Project. [En línea] <http://httpd.apache.org/>.
33. **NetBeans.** NetBeans IDE 7.4 Release Information. [En línea] <https://netbeans.org/community/releases/74/index.html>.
34. **w3c.** Guía Breve de Servicios Web. [En línea] <http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb>.

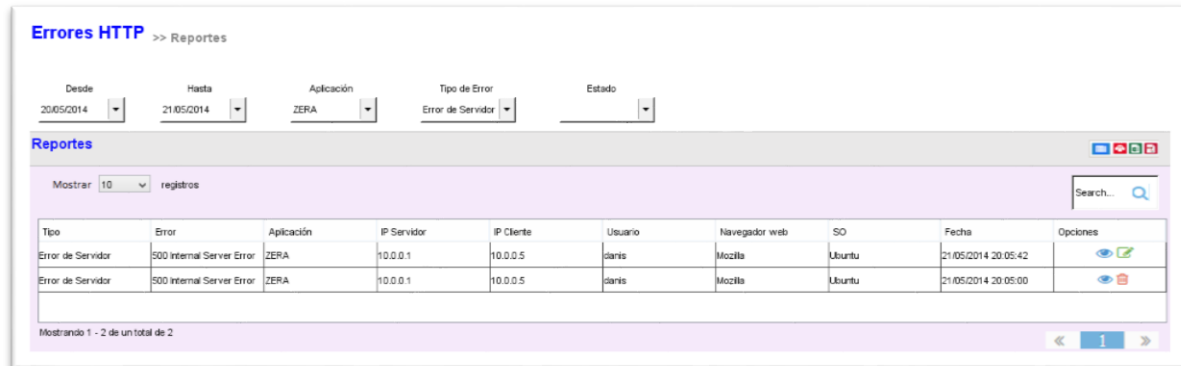
35. **Amodeo, Enrique.** Servicios web. [En línea]  
<http://eamodeorubio.wordpress.com/2010/07/26/servicios-web-2-%C2%BFque-es-rest/>.
36. **González Palacio, Liliana y Urrego Giraldo, Germán.** 2010. Modelo de contexto y de dominio para la ingeniería de requisitos. [En línea]  
<http://webapps.udem.edu.co/RevistaIngenierias/pdf/V9n17/P%E1ginas%20desdeRevista%20INGENIERIAS%20vo.%209%20No.%2017%20ARTICULO%2012.pdf>. 151-64 - ISSN - 1692-3324.
37. **Ing. Pérez Ramírez, Danay.** 2008. CUJAE. Metodologías Ágiles. ¿Cómo desarrollar utilizando XP?
38. **Joskowicz, José.** 2008. *Reglas y Prácticas en eXtreme Programming*. España : s.n.
39. **Carvajal Riola, Jose Carlos.** 2008. *Metodologías Ágiles*. UPC - Barcelona. Barcelona : s.n. pág. 215, Tesis Final de Máster.
40. **Beck, K.** *Extreme Programming Explained. Embrace Change*. Pearson Education.
41. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingeniería de Software*. Departamento de Informática, Universidad Técnica Federico Santa María.
42. **Eguiluz, Javier.** 2013. Desarrollo web con Symfony2.
43. **Ruiz Tenorio, Roberto.** Las Pruebas de Software y su Importancia en las Organizaciones. [En línea] Agosto de 2010.  
<http://cdigital.uv.mx/bitstream/123456789/28540/1/Ruiz%20Tenorio.pdf>.
44. **AWS Desarrollo web.** AlmacenPlantillasWeb. [En línea] 2013.  
<http://almacenplantillasweb.es/herramientas/herramientas-seo/herramientas-para-analisis-del-traffic-de-una-web/>.
45. **McIntyre, Angus.** Maestros del Web. *Como detectar los principales errores en aplicaciones CGI*. [En línea] 17 de Noviembre de 1999. [Citado el: 4 de Diciembre de 2013.]  
<http://www.maestrosdelweb.com/editorial/cgierr/>.
46. **R., Mauro Maulini.** Desarrollo y Seguridad de Aplicaciones Web y Móviles. *La importancia de manejar los archivos de registro de sucesos HTTP*. [En línea] 23 de Noviembre de 2010. [Citado el: 4 de Diciembre de 2013.] <http://tecnologiasweb.blogspot.com/2010/11/la-importancia-de-manejar-los-archivos.html>.
47. —. Desarrollo y Seguridad de Aplicaciones Web y Móviles. *Los 10 riesgos más comunes de las aplicaciones web según OWASP - versión 2010*. [En línea] 13 de Agosto de 2010. [Citado el: 5 de Diciembre de 2013.] <http://tecnologiasweb.blogspot.com/2010/08/los-10-riesgos-mas-comunes-de-las.html>.
48. **elrincondeajax.** El rincón de Ajax. *Manual de AJAX*. [En línea] [Citado el: 23 de Enero de 2014.] <http://www.elrincondeajax.com/manual-ajax/>.



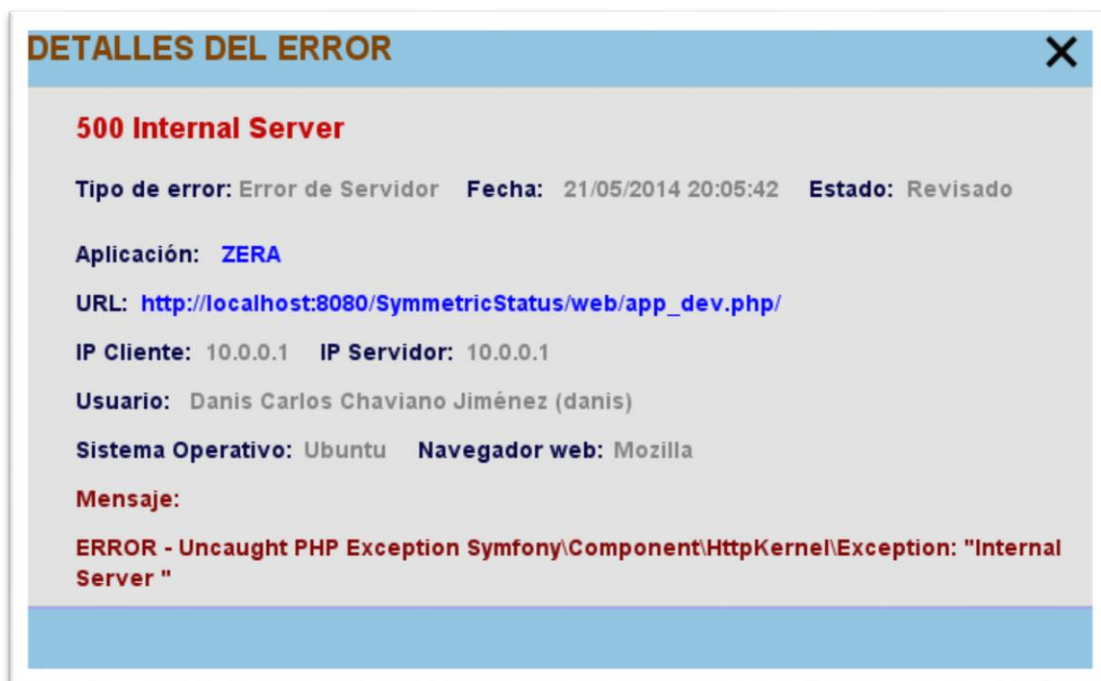
49. **apache.** <http://httpd.apache.org/>. [En línea] 5 de 12 de 2011. [Citado el: 19 de Enero de 2014.] <http://httpd.apache.org/>.
50. **Netbeans.** Netbeans. *Netbeans IDE\_Características*. [En línea] 10 de Diciembre de 2013. <http://netbeans.org/features/index.html>.
51. **visual-paradigm.** VP-UML 11.0 - Better requirements gathering, better software. [En línea] <http://www.visual-paradigm.com/product/vpuml/whats-new/>.
52. **w3.** What is CSS? *World Wide Web Consortium*. [En línea] [Citado el: 29 de Enero de 2014.] <http://www.w3.org/standards/webdesign/htmlcss#whatcss>.
53. **Angus, McIntyre.** Como detectar los principales errores en aplicaciones CGI | Maestros del WebMaestros del Web. [En línea] <http://www.maestrosdelweb.com/editorial/cgierr/>.
54. **Mulet, Manuel Angel.** Docentes Innovadores El software educativo, en el proceso enseñanza aprendizaje cubano. [En línea] <http://www.docentesinnovadores.net/Contenidos/Ver/5493>.
55. **Kniberg, Henrik.** 2007. SCRUM y XP desde las trincheras.
56. **R. C., Martin, M., Beedle y Schwaber, K.** 2001. *Agile Software Development with SCRUM*. Prentice Hall : s.n.
57. **Reynoso, Billy.** 2011. *Métodos Ágiles de desarrollo de Software*. Buenos Aires : s.n.
58. **Rodríguez Corbea, Maite y Ordóñez Pérez, Meylin.** 2007. *LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA*. Universidad de las Ciencias Informáticas. La Habana : s.n. Tesis.
59. **Selley Rojas, Héctor Julián.** 2008. *Monitoreo del comportamiento de servidores de aplicaciones*. Centro de Investigación en Computación. México : s.n. pág. 191.
60. **Mex.tl.** 2014. Programación Extrema. [En línea] [http://ingenieriadesoftware.mex.tl/52753\\_XP---Extreme-Programing.html](http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html).
61. **Hernández Sampieri, Roberto, Fernández-Collado, Carlos y Baptista Lucio, Pilar.** 2006. *Metodología de la investigación*. Mexico : Mc Graw Hill. ISBN 970-10-5753-8.

## ANEXOS

## Anexo 1: Prototipos de interfaz de usuario



Anexo 1.1 Interfaz de usuario "Listar errores"



Anexo 1.2 Interfaz de usuario "Detalles del error"


**Aplicaciones** >> Adicionar

**Aplicación:**

**Usuario:**

**Contraseña:**


**URL de la Aplicación:**

**IP del servidor:**  


[. Regresar a la lista](#)



Anexo 1.3 Interfaz de Usuario "Adicionar Aplicación"

**Aplicaciones** >> Listado

**Aplicaciones** Api REST 

Resultados por páginas

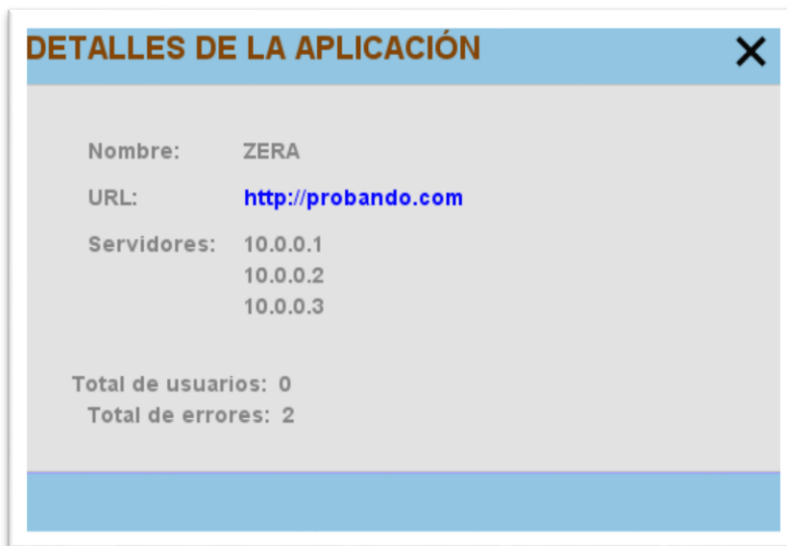
Search... 

Aplicación	Dirección url	Usuario	Opción
ZERA	http://probando.com	danis	 

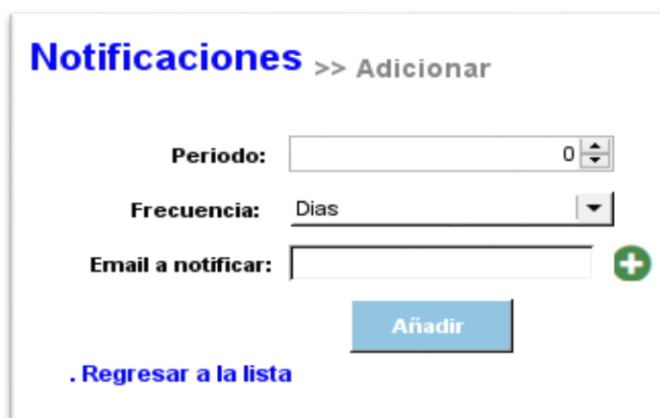
Mostrando 1 - 1 de un total de 1

<< 1 >>

Anexo 1.4 Interfaz de Usuario "Listar Aplicación"



Anexo 1.5 Interfaz de usuario "Detalles de la aplicación"



Anexo 1.6 Interfaz de usuario "Adicionar las notificaciones"



Anexo 1.7 Interfaz de usuario "Listar las notificaciones"

## **Anexo 2: Guía de observación**

Estudio de las soluciones similares:

- ✓ Para analizar los procesos que se monitorean de las aplicaciones web y como se presenta a los usuarios la información de los mismos.

Desarrollo de la investigación:

- ✓ Observar si las funcionalidades implementadas cumplen con los artefactos elaborados durante el flujo de trabajo de análisis y diseño.

Resultados de la investigación:

- ✓ Se aprecia el grado de complejidad de las no conformidades detectadas y cómo erradicarlas.