Universidad de las Ciencias Informáticas

Facultad 10

Sistema de Categorización Manual de URLs

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

Autores: Rolando González Herryman

Raúl Amambay Tarajano Pérez

Tutores: Ing. José Ramón Hermosilla Moreno

Lic. Leannys De La Caridad Labrada García

Ciudad de la Habana, 3 de junio del 2010

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes ___ del año_____.

_____ _____

Rolando González Herryman

Autor

Raúl Amambay Tarajano Pérez

Autor

_____ _____

Ing. José Ramón Hermosilla Moreno

Tutor

Lic. Leannys de la Caridad Labrada

Tutor

Agradecimientos

A mi mamá por estar siempre conmigo, a Escandell por enseñarme casi todo lo que sé, a mi hermano Gabriel por darme alegrías todos los fines de semana y a mi hermanita por comerse todo lo que me tocaba a mi antes.

A mi papá que a pesar de no estar siempre conmigo, en todo momento estuvo preocupado y ayudando mucho en mi educación.

A mi tío Pelayo y mi tía Leonor por nunca decir que no, a mis primos Fernando y Miguel que me enseñaron el valor del trabajo.

A María por ser mi cielito, por meterme en el proyecto hombre y lograr que me convirtiera en ingeniero.

A todos mis amigos al Zurdo a Jose y el Curbe por ser mis hermanos de la semana y estar siempre conmigo, por no dejarme caer en el error de estudiar todos los días.

A mis amigos del aula Enoel, Frank, el Danis que gracias a ellos pude sacar 5 en muchos seminarios y proyectos finales.

A mi compañero de tesis Amambay por dejarme hacer este trabajo con él.

A todas las personas que contribuyeron en la realización de este trabajo muchas gracias

Roly

A mi mamá y mi papá que los quiero tanto, ellos son los responsables de todo lo que soy, a mi hermana porque se que me quiere como yo la quiero a ella, a Daileny por estar siempre a mi lado y convertirse en alguien tan importante en mi vida, a mis primos y mis tíos por siempre ser tan atentos conmigo, a Mari Isel y Daily porque me acogieron como uno más de su familia , A mi gente de Ciego que nunca me han dejado solo y siempre están ahí cuando los necesito Ermis, Ruben, Villy, al Flaco (te dije que no estaba trabajando en los comedores), a mis otros hermanos Pomy, Juank, Diony y Dainery. A los amigos que he hecho en estos años y que se quedan para toda la vida: Lisi, Ale, Yadira, Katy que pasó de no rodarme a no poder vivir sin mi, por suerte para ella el sentimiento es mutuo. A mi compañero de tesis y amigo (El habanerito del Vedado) por todo el ánimo y apoyo que me dio en todo momento. A Yaisy, mi compañera inseparable de cuanto evento científico apareció. A Jorgito por toda la ayuda que nos brindó y nunca decir que no. Al piquete del 503 por aceptarme como soy y aguantarme aún cuando en ocasiones sentían un leve deseo de extrangularme: Yordan, Abel, Danis, Frank el breve, Makano, Jansel, Adrian, Evian, Denia, Yarelis, en fin todos ellos. A Mabel por ser una de las personas más nobles que he conocido y mi amiga. A todos mis amigos del proyecto, con los que he compartido tanto y que tanto me han ayudado, Jorge y Dayana, Yasmany, Adrian, Yuri con sus críticas exhaustivas y constructivas, Kiuver que después de casi 6 años me aceptó un café de hora y media. Koty que aportó la idea que salvó el tren militar, a Indira por tantas risas y sustos. A mis tutores: Leannys, Luis y José. A mi oponente Dovier. A mis profesores. A los miembros del tribunal por sus críticas y sugerencias. También a todos aquellos que se preocuparon por como iba la tesis y finalmente a la UCI por darme la oportunidad compartir con todas estas personas los mejores años de mi vida.

Amambay

Dedicatoria

A mi mamá para que se sienta orgullosa de mi, a mis hermanos para que sigan mi ejemplo y estudien mucho mas.

A María para que me quiera más.

A mi papá para ver si viene.

Roly

En especial a mis padres, para ellos y de ellos son todos mis logros.

A mi hermanita, que se gradúa junto conmigo porque mi título también es de ella.

A mi sobrinita Yuliet, con la esperanza de ser su ejemplo para el futuro.

Amambay

Resumen

El Filtrado de Contenidos es una solución técnica al problema que plantea la presencia de contenidos inadecuados en Internet y es utilizado para regular el acceso de los usuarios en empresas, centros de enseñanza y hogares. Filpacon es un software de este tipo que para realizar el filtrado se basa en una base de datos con URLs categorizadas según su contenido. Actualmente esta base de datos se actualiza por listas de URLs categorizadas publicadas por entidades externas. Para contribuir en la búsqueda de la independencia tecnológica se realizó este trabajo que tiene como objetivo desarrollar un sistema de categorización manual de URLs que permita a los usuarios contribuir al proceso de actualización de la base de datos de URLs categorizadas de Filpacon. Para lograr dicho objetivo se utilizó el framework Symfony y el desarrollo del software estuvo guiado por la metodología XP. Los diferentes artefactos generados junto a las facilidades que brindó el framework seleccionado, hicieron posible obtener una aplicación amigable, funcional y eficiente.

Índice general

Introducción

El software Filtrado de Paquetes por Contenido (Filpacon) permite regular el acceso de los usuarios a determinados contenidos de internet, basándose fundamentalmente en una base de datos con URLs categorizadas según su contenido. Siempre que el usuario hace la petición de una URL al sistema, este consulta su base de datos y teniendo en cuenta las categorías asociadas con dicha URL y los permisos que posea el usuario, el sistema tomará la decisión de permitir o denegar el acceso a la misma. Actualmente para actualizar la base de datos de Filpacon se usan listas de URLs categorizadas generadas por La Universidad de Toulouse, cuyo propósito es ayudar a los administradores a regular el uso de internet, e Internet Secure Access Kit (iSAK), plataforma para la administración del tráfico en la web desarrollada sobre software libre bajo la licencia General Public License (GPL). Un cambio en la política con que son publicadas estas listas o la desactualización de las mismas constituyen problemas que están fuera del alcance del grupo de desarrollo de Filpacon, razón por la que se hace necesario independizarse de las mismas. Existen otras maneras de generar listas de URLs categorizadas, una de ellas es el empleo de la inteligencia artificial para categorizar automáticamente las URLs, en este sentido se está desarrollando el Motor de Clasificación Inteligente de Contenidos (MOCIC). Otro mecanismo para la generación de estas listas es la categorización manual de URLs, este mecanismo puede presentar debilidades tales como la inexperiencia de los usuarios a la hora de categorizar la URL, inexistencia de un proceso centralizado para la categorización de las URLs, además la cantidad de URLs categorizadas por esta vía está dada por el interés que tengan los usuarios en colaborar con la categorización.

A partir de lo planteado anteriormente se identificó el siguiente **problema científico**: ¿Cómo contribuir a perfeccionar el proceso de actualización de la base de datos de URLs categorizadas de Filpacon?

Teniendo en cuenta los planteamientos anteriores, se define como **objeto de estudio** las técnicas de categorización de URLs y el **campo de acción** será los Sistemas de Categorización Manual de URLs.

Se define como **objetivo general** de la investigación: Desarrollar un Sistema de Categorización Manual de URLs que permita a los usuarios contribuir al proceso de actualización de la base de datos de URLs categorizadas de Filpacon.

Para orientar la investigación se realizaron las siguientes **preguntas científicas**:

1. ¿Cuáles son los fundamentos teóricos y metodológicos que justifican el desarrollo de un Sistema de Categorización Manual de URLs que permita a los usuarios contribuir al proceso actualización de la base de datos de URLs categorizadas de Filpacon?

2. ¿Cuál es el estado actual del empleo de los Sistemas de Categorización Manual de URLs para la actualización de bases de datos de URLs categorizadas?

3. ¿Cómo diseñar un Sistema de Categorización Manual de URLs que propicie que los usuarios contribuyan al proceso de actualización de la base de datos de URLs categorizadas de Filpacon?

4. ¿Cómo implementar un Sistema de Categorización Manual de URLs que propicie que los usuarios contribuyan al proceso de actualización de la base de datos de URLs categorizadas de Filpacon?

Para dar respuesta a las preguntas científicas se identificaron las siguientes **tareas**:

1. Determinación de los fundamentos teóricos y metodológicos que justifican el desarrollo de un Sistema de Categorización Manual de URLs que permita a los usuarios contribuir al proceso de actualización de la base de datos de URLs categorizadas de Filpacon.

2. Determinación del estado actual del empleo de los Sistemas de Categorización Manual de URLs para la actualización de bases de datos de URLs categorizadas.

3. Diseño de un Sistema de Categorización Manual de URLs que propicie que los usuarios contribuyan al proceso de actualización de la base de datos de URLs categorizadas de Filpacon.

4. Implementación de un Sistema de Categorización Manual de URLs que propicie que los usuarios contribuyan al proceso de actualización de la base de datos de URLs categorizadas de Filpacon.

En el cumplimiento de las tareas se usarán los siguientes métodos teóricos:

El **Analítico-Sintético** se aplicará para entender las técnicas de categorización de URLs, a partir del análisis de sus características y para formular conclusiones a través de la síntesis de los conocimientos y resultados obtenidos.

La **Modelación** mediante el lenguaje de modelado Unified Modeling Language (UML), permitirá reflejar la estructura, relaciones internas y características de la solución a través de diagramas.

El **Inductivo-Deductivo** permitirá conocer cómo funcionan los Sistemas de Categorización Manual de URLs a partir del análisis de las técnicas de categorización de URLs.

El presente documento está estructurado en cuatro capítulos descritos a continuación.

El primer capítulo se centra en el estado del arte de los Sistemas de Categorización Manual de URLs. En él se analiza la existencia de soluciones de este tipo a nivel nacional e internacional que aportan al desarrollo de la solución propuesta. Abordándose aspectos tales como sus características. De igual manera se identifican las tecnologías y herramientas así como la metodología para la creación de este sistema.

En el segundo capítulo se profundiza en el problema científico a través de su descripción. Se realiza una propuesta de solución y se identifican los requisitos funcionales y no funcionales que se deben tener en cuenta. Por último, se llevan a cabo las dos primeras fases de la metodología de desarrollo utilizada.

En el tercer capítulo se presentan aspectos del diseño, así como se exponen ideas relacionadas con la validación de los datos, el tratamiento de errores y la seguridad, basándose en los mecanismos que el framework Symfony provee para dichas cuestiones y se describen una serie de aspectos relativos a la base de datos y el diseño gráfico de la presentación.

El cuarto y último capítulo se enfoca en la implementación y prueba de la solución propuesta tomando como base los resultados que se presentan en los capítulos anteriores. En este capítulo se describen las tareas de implementación y las pruebas de aceptación realizadas en las diferentes iteraciones del proceso de desarrollo de la solución propuesta.

Capítulo 1 Fundamento teórico

1.1 Introducción

Las técnicas de categorización permiten asignar a una URL categorías asociadas a ella. La categorización manual de URLs ayuda en la creación de listas de URLs categorizadas confiables, pues estas son confeccionadas bajo la supervisión humana. Crear un sistema que permita llevar a cabo el proceso de categorización manual de URLs implica analizar soluciones similares y seleccionar las tecnologías y herramientas adecuadas para su desarrollo.

1.2 ¿Por qué categorizar los URLs?

“Internet es el medio de difusión más potente de la actualidad. Investigadores, científicos, universitarios, profesionales y ciudadanos en general con el solo hecho de tener un ordenador y estar conectados a la red tienen la posibilidad de acceder a gran cantidad de información. Esto que en principio es una de las grandes virtudes de Internet, también es un problema, ya que tanta información desestructurada puede llegar a ser incontrolable por el usuario” (1). Además “si bien es cierto que Internet contiene una gran variedad de contenidos de incuestionable valor. También son abundantes los contenidos de dudoso valor, en algunos casos degradantes e incluso ilegales en muchos países. Cada día el número de sitios web que albergan contenidos: pornográficos, pedofílicos, terroristas, xenofóbicos, racistas, extremistas, violentos y muchos más, va en aumento. La complejidad del problema se ve empeorada por la no jurisdicción de las leyes, así por ejemplo de nada sirve que en Alemania, Francia, Austria e Italia tengan leyes que prohíban la existencia de sitios web con contenidos xenofóbicos o nazis si en los Estados Unidos esto es perfectamente legal” (2).

1.3 Técnicas de categorización de URLs

Para conformar una lista de URLs categorizadas son necesarios dos pasos esenciales: primeramente es imprescindible generar una lista de dominios o páginas web para su posible categorización; esta lista puede obtenerse mediante la utilización de softwares recuperadores de información, a través del procesamiento de logs de navegación de un proxy o mediante sugerencias hechas por contribuyentes, y en segundo lugar es necesario examinar cada dominio o página web con el propósito de asignarle una o varias categorías, es precisamente aquí donde se ponen en práctica las diferentes técnicas de

categorización de URLs, surgidas como resultado de numerosas investigaciones realizadas a partir de la necesidad de organizar la información en Internet y de controlar el acceso de los usuarios a la misma.

Después de realizar un estudio de las diferentes técnicas utilizadas para categorizar URLs se puede afirmar que estas se dividen en dos grupos: técnicas de categorización automatizada y técnicas de categorización manual.

Categorización automatizada de URLs

La categorización automatizada de URLs consiste en la utilización de softwares que se apoyan en el análisis de diferentes patrones para realizar el proceso de asignar a una página web una o más categorías definidas. A continuación se hace referencia a algunas de las técnicas más utilizadas.

Categorización automatizada de textos: Consiste en descubrir variables que sean útiles para discriminar textos que pertenecen a categorías existentes. Por tanto, el objetivo de un categorizador (programa que ejecuta algoritmos de categorización) es indicar la categoría que debe asociarse a cada texto, partiendo de un esquema o modelo de categorización previamente creado.

Categorización basada en etiquetas META: Esta técnica de clasificación consiste en confiar únicamente en los atributos de las etiquetas meta <META name="keywords"> y <META name="description">. Sin embargo, hay una posibilidad de que el autor de la página web incluya palabras clave, que no reflejan el contenido de la página, sólo para aumentar la tasa de éxito de su página en los resultados de los motores de búsqueda.

Categorización basada en URL: A menudo, la URL en sí es bastante informativa, se puede recoger una gran cantidad de información de ella sin necesidad de examinar el contenido real de los recursos. Este mecanismo funciona dividiendo la URL para obtener una segmentación de base de referencia tal como figura en el protocolo URL (por ejemplo, esquema://host/ruta_elementos/documento.extensión), cada segmento es analizado por separado siendo comparado con palabras clave existentes en una base de datos logrando así la categorización de la URL. (3)

Los sistemas capaces de realizar la categorización automatizada de URL generalmente utilizan técnicas de inteligencia artificial para vincular y llevar a cabo los diferentes tipos de clasificación antes

mencionados, algunos de los sistemas que se destacan en este proceso son:

PureSight:

Filtro de Contenidos que se adapta a la naturaleza dinámica de Internet. Su tecnología de filtrado dinámico es una tecnología sofisticada de Reconocimiento Artificial de Contenidos, que puede identificar el contenido de un sitio web y entonces decidir si permitirlo o no. Cada sitio web solicitado es inspeccionado por su algoritmo inteligente para asegurar el cumplimiento de las políticas de uso corporativas, institucionales o de grupos familiares. Provee una cobertura completa y confiable de la Web con una exactitud de reconocimiento incomparable. El Reconocimiento Artificial de Contenidos contiene un poderoso conjunto de algoritmos de Inteligencia Artificial que analizan y categorizan datos en tiempo real. (4)

Optimal Internet (OPTENET):

Filtro de Contenidos que, basado en técnicas de Inteligencia Artificial, reduce la dependencia de listas de URLs categorizadas y se adapta a la naturaleza dinámica de Internet. Su Analizador Inteligente de Contenidos proporciona un rápido análisis, con un tiempo de procesamiento de 0,1 milisegundos por página web solicitada.

Categorización manual de URLs

La categorización manual de URLs es el proceso donde revisores evalúan y organizan las URLs en categorías previamente definidas a partir de un análisis del contenido de dichas URLs. Con la supervisión humana se logra un elevado nivel de efectividad en la categorización de URLs, razón por la cual este proceso es ampliamente utilizado para la creación de directorios temáticos y de listas de URLs categorizadas, que pueden ser empleadas por los filtros web para actualizar sus bases de datos y además constituyen colecciones de entrenamiento para sistemas de categorización automatizada basados en el aprendizaje.

Aunque categorizar manualmente las URLs es un proceso efectivo, no es menos cierto que el número de URLs categorizadas por esta vía es mucho menor que la cifra que puede alcanzarse utilizando técnicas automatizadas. Por otra parte, es un proceso consumidor de tiempo, propenso a errores humanos y

altamente dependiente de sus costumbres, conocimientos y compromisos con la labor de categorización.

Debido a que la solución propuesta es un sistema que utiliza la categorización manual para generar listas de URLs en el siguiente epígrafe se realizará un análisis de las principales características y funcionalidades de los sistemas que se destacan en el empleo de la categorización manual de URLs.

1.4 Trabajos similares

Ante la necesidad de desarrollar un Sistema de Categorización Manual de URLs que permita a los usuarios contribuir al proceso de actualización de la base de datos de URLs categorizadas de Filpacon se precisa el estudio de las soluciones similares existentes a nivel mundial.

Ámbito internacional

En el ámbito internacional existen numerosos sistemas que permiten categorizar manualmente las URLs, pero la gran mayoría de ellos son directorios temáticos que utilizan esta técnica como vía para incrementar sus bases de datos, sin embargo no generan listas de URLs categorizadas, solo una pequeña minoría hace de este su objetivo principal o al menos uno de sus objetivos. A continuación se describen y analizan las características de cinco de estos sistemas que teniendo en cuenta el impacto que tienen en el marco de los filtros de web, se consideran los más importantes a nivel internacional.

Open Directory Project (ODP)

El Open Directory Project es el directorio temático[1] más extenso y completo del Web. Su construcción y mantenimiento son realizados por una gran comunidad global de editores voluntarios. Debido a que el propósito del ODP es listar y categorizar sitios web, posee un sistema que permite a los usuarios sugerir una categoría a las URLs, por esto se considera importante su estudio. Esta distribuido bajo su propia licencia que entre otras cosas, concede una licencia gratuita y no exclusiva para usar, reproducir, modificar y crear trabajos derivados, distribuir y publicar el Open Directory y sus trabajos derivados.

Shalla Secure Services KG

[1] Constituyen guías jerárquicas de temas o áreas creadas por humanos que abarcan desde lo más general a lo más particular. Agrupan las páginas web que pertenecen a un área bajo categorías para facilitar las búsquedas a los usuarios. (5)

Shalla Secure Services KG cuenta con un sistema de categorización manual de URLs a través del cual han logrado la colección de URLs categorizadas más grande de Alemania. Dicha colección está dividida actualmente en 70 categorías y su principal objetivo es servir como base de datos de URLs categorizadas para SquidGuard[2] aunque también puede ser utilizada por otros sistemas de filtrado o incluida en la instalación de Squid[3] sin la necesidad de utilizar un redirector o un filtro web.

Esta colección es distribuida bajo una licencia que permite a los usuarios domésticos utilizar las listas de forma gratuita sin necesidad de registrarse y las empresas, instituciones y organismos gubernamentales pueden utilizarla libre de pago pero deben registrarse mediante la firma de un contrato de uso, por otra parte, esta licencia define que si las listas están integradas en un producto/servicio comercial se debe pagar una cuota anual.

Funcionalidades del sistema

- Listado de categorías.
- Adicionar un dominio o URL a una categoría.
- Remover un dominio o URL de una categoría.
- Mover un dominio o URL a otra categoría.
- Proponer una nueva categoría.
- Buscar la categoría asociada a un dominio o URL.
- Descargar la colección de URL categorizadas.
- Ver estado de los envíos realizados por los usuarios.

Sistema de categorización manual de URLs de la Universidad de Toulouse.

Este sistema ofrece listas de URLs organizadas en 39 categorías y gestionada por Fabrice Prigent para ayudar a los administradores a regular el uso de Internet. Son generadas mediante la categorización híbrida de URLs, pues aunque casi la totalidad de las mismas es construida gracias a muchos contribuyentes, utilizan un spider para buscar URLs con contenidos para adultos. Estas listas están

[2] Sistema de filtrado web por listas negras, usa una base de datos con miles de URLs clasificadas en varias categorías.
[3] Programa de software libre que implementa un servidor proxy y un demonio para caché de páginas web, publicado bajo licencia GPL.

disponibles bajo la licencia Creative Commons Contract.

Funcionalidades del sistema

- Listado de categorías.
- Adicionar un dominio o URL a una categoría.
- Remover un dominio o URL a una categoría.
- Descargar la colección de URLs categorizadas.

URLBlacklist

URLBlacklist brinda como servicio comercial, una colección de más de 3 millones de URLs organizadas en 72 categorías para ser utilizadas fundamentalmente por SquidGuard y DansGuardian[4]. Para su confección se emplean las listas generadas por el ODP además de las contribuciones realizadas a través de su propia interfaz web. Para este servicio se definen diferentes tarifas en dependencia del uso que se le dé a la colección y el número de descargas en un tiempo determinado que se hagan de la misma.

Funcionalidades del sistema

- Listado de categorías.
- Adicionar un dominio o URL a una categoría.
- Remover un dominio o URL a una categoría.
- Descargar la colección de URLs categorizadas.
- Buscar la categoría asociada a un dominio o ULR.

Las listas generadas por Shalla Secure Services KG, la Universidad de Toulouse y URLBlacklist están compuestas por archivos de texto plano, los archivos para cada categoría están incluidos en una carpeta marcada con el nombre de la categoría. En cada categoría hay dos archivos de texto con la siguiente estructura:

[4] Filtro web que filtra el contenido de los sitios basándose en varios métodos de clasificación pero fundamentalmente basado en una lista de URLs categorizadas.

- domains.txt: Donde los dominios y direcciones IP son almacenados.
- urls.txt: Donde son almacenadas las URLs.

iSAKurlDB

iSAKurlDB genera listas con más de 2,100,000 dominios y URLs categorizadas que son utilizadas por Squid, SquidGuard y DansGuardian para realizar el filtrado, incluso Filpacon las usa para la actualización de su base de datos, pero es más usada por iSAK en la generación de reportes para categorizar el tráfico en la red. Para su confección se emplean las listas generadas por la universidad de Toulouse, SquidGuard y el ODP además de las contribuciones realizadas a través de su propia interfaz web. Está distribuida bajo la licencia GPL.

Funcionalidades del sistema

- Listado de categorías.
- Adicionar un dominio o URL a una categoría.
- Remover un dominio o URL a una categoría.
- Descargar la colección de URLs categorizadas.
- Buscar la categoría asociada a un dominio o ULR.

Las listas son archivos de texto plano, para cada categoría son generados tres archivos, con la siguiente estructura:

- nombrecategoría domains.txt: Donde los dominios y direcciones IP son almacenados.
- nombrecategoría squid domains.txt: Archivo de Access Control List (ACL) para Squid.
- nombrecategoría urls.txt: Donde son almacenadas las URLs.

Ámbito nacional

En el ámbito nacional el uso de técnicas de categorización manual de URLs podría afirmarse que es prácticamente nulo. Exceptuando el uso de esta técnica en un pequeño directorio temático disponible en el sitio de la Oficina para la Informatización, en el que son organizadas en solo 9 categorías las URLs de los sitios de entidades individuales y colectivas con base en el territorio nacional. Este directorio brinda un

enlace en el que se puede sugerir la inclusión de un nuevo sitio o la corrección en la categorización de una URL. Aparte de esto no se encontraron referencias de ninguna solución que utilice la técnica de categorización manual de URLs.

En el ámbito internacional algunos de los sistemas de categorización manual de URLs tienen la limitante de que deben pagarse con cuotas anuales o de otro tipo para ser introducidos en un software de filtrado que posteriormente será comercializado. Por consiguiente no es conveniente el uso de estos sistemas en Filpacon. Los sistemas de categorización manual de URLs libres de pago o bajo licencias que no restringen sus usos, no responden a los intereses del grupo de desarrollo de Filpacon ya que son soluciones diseñadas a la medida de los problemas de otras empresas. Además, un cambio en la política de distribución de las listas generadas por estos sistemas de categorización podría traer consecuencias negativas para Filpacon.

Como en el ámbito nacional no se encontraron soluciones que respondan a los intereses de Filpacon se justifica la novedad de la propuesta y la necesidad de desarrollar en Cuba un sistema de categorización manual de URLs totalmente independiente para contribuir con el proceso de filtrado que realiza Filpacon.

1.5 Tecnologías a utilizar

Decidir desde el inicio del desarrollo de un software las tecnologías adecuadas a utilizar, contribuye a evitar el aumento del tiempo requerido y de los costos, así como evitar una mala calidad del producto. Dado que la solución propuesta tiene como objetivo lograr que los usuarios de cualquier parte del mundo contribuyan al proceso de categorización de URLs, la Web como medio para propiciar la interacción requerida, converge con el uso que hacen de esta los usuarios al navegar por Internet de modo que una Interfaz Web (IW) sería la solución adecuada para realizar esta tarea. Por tanto, se precisa el estudio de las tecnologías que serán utilizadas para el desarrollo de la IW. Otro elemento a tener en cuenta en la selección son las ventajas que reportaría el uso de cada herramienta.

Lenguaje de programación

Para el desarrollo de aplicaciones Web existen diferentes lenguajes de programación, tales como: Perl, Java, PHP, Ruby y Python. La elección de uno de ellos debe sustentarse principalmente en las particularidades de la solución que se proponga y los conocimientos de los desarrolladores de los mismos.

A partir de esto se eligió PHP, teniendo en cuenta sus características, ventajas y desventajas.

Características (6)

- **Simplicidad:** usuarios con experiencia en Perl, C o C++ podrán utilizarlo rápidamente.
- **Velocidad:** alta velocidad de ejecución sin introducir demoras en la máquina, bajo consumo de recursos y muy buena integración con Apache.
- **Estabilidad:** ninguna aplicación está completamente libre de errores (bugs), pero con una amplia comunidad de programadores y usuarios corregirlos es mucho más fácil. Posee un sofisticado manejo de variables que lo hacen muy robusto y estable.
- **Seguridad:** permite la protección contra diversos ataques a través de diferentes niveles de seguridad que pueden ser configurados desde el archivo .ini.

Ventajas (6)

- PHP es Open Source o´ código abierto, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no está forzado a pagar actualizaciones anuales para tener una versión que funcione.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, Oracle, Informix, PostgreSQL y otros.
- Actualmente se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD.
- PHP corre en “casi cualquier plataforma” utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en aproximadamente 25 plataformas, incluyendo diferentes versiones de Unix, Windows (95, 98, NT, ME, 2000, XP, etc.) y Macs.

Desventajas (6)

- No posee un adecuado manejo de la internacionalización y Unicode.
- Por su diseño dinámico no puede ser compilado y es muy difícil de optimizar.
- Por sus características promueve la creación de código desordenado y complejo de mantener.
- Está diseñado especialmente para un modo de hacer las aplicaciones Web que es ampliamente considerado problemático y obsoleto (mezclar el código con la creación de la página web).

Las dos últimas desventajas pueden evitarse mediante la disciplina y el diseño previo de lo que se va a desarrollar. Aunque PHP no obliga a seguir una determinada metodología a la hora de programar, existen formas de obtener código ordenado, estructurado y manejable. Una de ellas es el uso de un framework.

¿Por qué PHP5?

El éxito y aceptación del lenguaje PHP5 ha sido mayor que el de sus versiones anteriores (PHP3 y PHP4). Los cambios más significativos que introducen están relacionados con la programación orientada a objetos (POO). El uso de modificadores de acceso (private, protected, public) a atributos y métodos, el paso de parámetros por referencia de forma implícita y el soporte para interfaces son algunos de ellos. Además incluye mejoras importantes en el rendimiento y la seguridad. Otro factor clave para su uso fue el anuncio de que PHP4 no tiene soporte desde el 31 de diciembre de 2007 y sólo se corregirán los fallos más importantes relacionados con la seguridad hasta el 8 de agosto de 2008.7

Lenguaje de modelado

UML, por sus siglas en inglés, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un ”plano” del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (8)

Framework seleccionado

Un framework permite reutilizar un diseño para un dominio específico de software rigiendo la arquitectura del mismo. Además define la estructura global de un sistema teniendo en cuenta las clases y objetos, sus responsabilidades claves y como colaboran entre sí, de modo que los diseñadores e implementadores puedan concentrarse en los elementos específicos del mismo (9). Por último, hace la programación más fácil encapsulando operaciones complejas en instrucciones sencillas.

La mayoría de los frameworks para aplicaciones Web escritos en PHP, se basan en el patrón MVC como elemento principal para la arquitectura. Tomando como fuente principal el artículo “Taking a look at ten

different PHP frameworks" (10) se han identificado los siguientes: CodeIgniter, ZooP Framework, WACT, Seagull Framework, CakePHP, Zend Framework, Symfony, Prado, PHP on TRAX y eZ Components. De todos ellos se seleccionó Symfony teniendo en cuenta las características que se mencionan a continuación:

- Está escrito en PHP5 basándose en el patrón MVC.
- Cuenta con la versión estable 1.0.
- El 28 de Octubre de 2006 se anunciaba en el sitio de Symfony que Yahoo Bookmarks había lanzado una nueva versión beta de su portal, desarrollado con Symfony, el cual contaba con 12 millones de usuarios y estaba disponible en 12 idiomas.
- Pueden usarse fácilmente componentes de eZ Components y Zend Framework para proyectos con requisitos cuya solución no está presente en Symfony, lo que brinda junto a otros mecanismos de extensión una gran flexibilidad.
- Cuenta con una excelente documentación en español que incluye la traducción del libro "The Definitive Guide to Symfony" que posee 19 capítulos en los que se explica todo lo relativo al trabajo con la versión 1.2.
- Numerosos plugins son creados y actualizados constantemente ante las necesidades de los desarrolladores.

Herramienta de programación

Las ventajas principales de un Integrated Development Enviroment (IDE) radican en la automatización de algunas tareas asociadas al desarrollo de aplicaciones y al soporte para algunas funcionalidades como el autocompletamiento de código y la depuración, contribuyendo a evitar pérdidas de tiempo. La selección de un IDE para el desarrollo de aplicaciones Web con soporte para PHP se centró en herramientas libres. Aunque existen varios candidatos entre los que están gPHPEdit, Komodo y EclipsePDT, se escogió este último por la experiencia en el trabajo con Eclipse y a partir de las siguientes características (11):

- Fácil de usar e intuitivo.
- Se adhiere a los estándares de Eclipse.
- Extensibilidad y soporte continuo para el desarrollo de PHP.
- Integración con subversión como herramienta para el control de versiones.

Herramienta de modelado

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta un entorno todo-en--uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de casos de usos.

1.6 Metodología de desarrollo

Todo desarrollo de software es riesgoso y difícil de controlar y, si no se emplea una metodología que guie este proceso, los resultados obtenidos son clientes insatisfechos y desarrolladores aún más descontentos. Actualmente a nivel internacional, en dependencia del tiempo de vida y la complejidad del proyecto que se vaya a desarrollar, se proponen diferentes metodologías. Una metodología es el conjunto de técnicas y procedimientos que permiten conocer los elementos necesarios para definir un proyecto de software, es la base para la edificación de un producto de este tipo. Esencialmente, sirve para aumentar la “calidad” del software y controlar de manera transparente todo el proceso de desarrollo. Si se quiere que un proyecto sea escalable y flexible a los cambios, es lógico pensar que para lograr ese propósito se necesite tomar en cuenta una de las muchas metodologías para el proceso de desarrollo que existen. Las metodologías, dadas sus características, se enmarcan en dos grandes grupos: las llamadas “metodologías pesadas” y las “metodologías ágiles”. La diferencia más notable entre estos dos grupos es que mientras los métodos pesados intentan obtener los resultados apoyándose principalmente en la documentación ordenada, los métodos ágiles tienen como base de sus resultados la comunicación e interacción directa con todos los usuarios involucrados en el proceso.

En el presente trabajo se investigó sobre dos metodologías, una de tipo pesada y otra de tipo ágil, la metodología pesada o tradicional que se estudió fue RUP y la ágil fue Extreme Programming (XP).

RUP

Es una de las metodologías más usadas y mayormente probadas a nivel internacional para el desarrollo de software.

Características de RUP

- **Dirigido por los casos de uso:** Tiene a los casos de uso como el hilo conductor que orienta las actividades de desarrollo. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él.
- **Centrado en la arquitectura:** Propone la arquitectura de forma similar a la de un edificio. Es necesario tener varios planos con diferentes aspectos para tener una imagen completa del edificio antes que comience su construcción. Aquí entra en juego el término Arquitectura de Software, que abarca las diferentes vistas que se mencionaron anteriormente.
- **Iterativo e incremental:** Propone la descomposición de proyectos grandes en proyectos más pequeños o subproyectos, cada uno de los cuales es una iteración, y cada iteración debe estar controlada y tratar un determinado grupo de casos de uso.

RUP provee un acercamiento a disciplinas para asignar tareas y responsabilidades en un desarrollo organizado. Su objetivo es asegurar la producción de software de alta calidad que conozca la necesidad de sus clientes dentro de un predecible espacio de tiempo y presupuesto. La figura 1.1 ilustra la arquitectura de RUP.

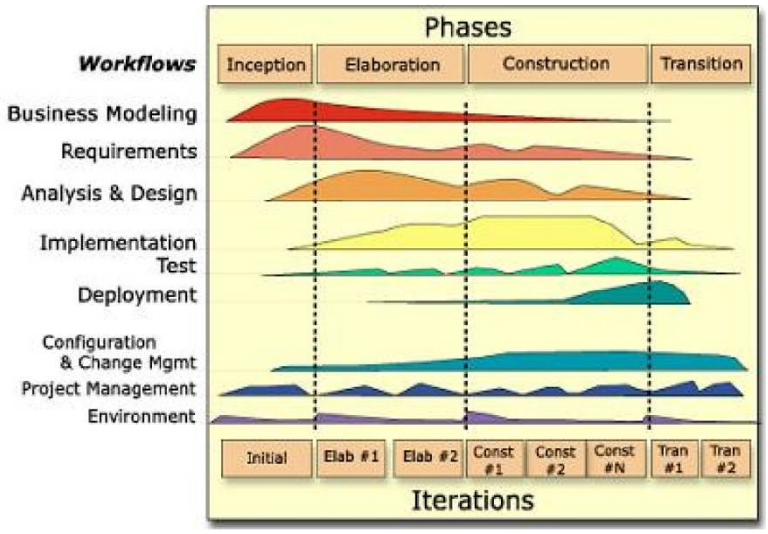

Figura 1.1: Arquitectura de RUP.

Fases del desarrollo

- **Inicio:** Determinar la visión del proyecto.
- **Elaboración:** Determinar la arquitectura óptima.
- **Construcción:** Llegar a obtener la capacidad operacional inicial.
- **Transición:** Llegar a obtener una primera versión del proyecto (release).

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

Disciplina de Desarrollo Ingeniería de Negocios: Entiendo las necesidades del negocio.

- **Requerimientos:** Traslado las necesidades del negocio a un sistema automatizado.
- **Análisis y Diseño:** Trasladando los requerimientos dentro de la arquitectura de software.
- **Implementación:** Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- **Pruebas:** Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado

está presente.

Disciplina de Admitidas configuración y administración del cambio: Guardando todas las versiones del proyecto.

- **Administración del proyecto:** Administrando horarios y recursos.
- **Ambiente:** Administrando el ambiente de desarrollo.
- **Distribución:** Hacer todo lo necesario para la salida del proyecto.
- **Elementos del RUP Actividades:** Procesos que se llegan a determinar en cada iteración.
- **Trabajadores:** Las personas o entes involucrados en cada proceso.
- **Artefactos:** Un artefacto puede ser un documento, un modelo, o un elemento de modelo. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

XP

Las metodologías ágiles forman parte del movimiento de desarrollo ágil de software, que se basan en la adaptabilidad de cualquier cambio, como medio para aumentar las posibilidades de éxito de un proyecto. De forma que una metodología ágil es la que tiene como principios que:

- Los individuos y sus interacciones, son más importantes que los procesos y las herramientas.
- El software que funciona, es más importante que la documentación exhaustiva.
- La colaboración con el cliente, en lugar de la negociación de contratos.
- La respuesta delante del cambio, en lugar de seguir un plan cerrado.

La programación extrema es una metodología de desarrollo ligera (o ágil), basada en una serie de valores y de prácticas de buenas maneras, que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software, en la que se da prioridad a los trabajos que dan un resultado directo, y que reducen la burocracia que hay alrededor de la programación.

Una de las características principales de este método de programación, es que sus ingredientes son

conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores, y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, si que el resultado es una nueva manera de ver el desarrollo de software.

El objetivo que se perseguía, en el momento de crear esta metodología, era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos.

Valores de XP

- **Comunicación:** Crear software requiere de Sistemas comunicados.
- **Simplicidad:** Empezar con lo necesario y requerido y trabajar desde ahí.
- **Retroalimentación:** Del Sistema, del cliente, y del equipo.
- **Valentía:** Programa para hoy y no para mañana.
- **Respeto:** El equipo debe trabajar como uno, sin hacer decisiones repentinas.

Actividades

- **Codificación:** La parte más importante de XP.
- **Pruebas:** Nunca se puede estar seguro de algo hasta haberlo probado.
- **Escuchar:** Escuchar los requisitos del cliente acerca del sistema a crear.
- **Diseño:** Crear una estructura del diseño para evitar problemas.

Ventajas

- Programación organizada.
- Menor taza de errores.
- Satisfacción del cliente.
- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el desarrollo del proyecto.
- XP es mejor utilizada en la implementación de nuevas tecnologías donde los requerimientos cambian rápidamente.

Desventajas

- Es recomendable emplearla solo en proyectos de corto plazo.
- Altas comisiones en caso de fallar.

Por tanto, se selecciona XP como metodología de desarrollo de software ya que se adapta en gran medida, tanto al tipo de proyecto a desarrollar como a las condiciones de trabajo.

- El proyecto es pequeño: XP está concebida para ser utilizada dentro de proyectos pequeños.
- El cliente forma parte del equipo de desarrollo: Mediante la aplicación de XP se puede lograr una retroalimentación mayor y lograr un producto que satisfaga sus necesidades.
- Poca disponibilidad de personal: El sistema debe ser realizado por dos personas solamente, no siendo posible la existencia de muchos roles ni la especialización en un rol específico por parte de los miembros. Uno de los principios básicos de XP es la programación en equipos pequeños (2 a 12 personas) con pocos roles, pudiendo los miembros del equipo intercambiar responsabilidades en un momento determinado.
- Propiedad colectiva del código: XP plantea que todos los programadores pueden realizar cambios en cualquier parte del código en cualquier momento. Enfatiza la comunicación de los programadores a través del código, utilizando líneas directivas para la codificación que están bien establecidas.

1.7 Conclusiones

El estudio de las diferentes técnicas de categorización y de los sistemas que permiten llevar a cabo una categorización de URLs de forma manual, ha sentado las bases que justifican el desarrollo de un sistema propio que responda a los intereses de Filpacon. Mediante el análisis de las diferentes herramientas, tecnologías y metodologías existentes se ha realizado una selección de aquellas que permitirán la confección de la solución propuesta.

Capítulo 2 Características del sistema

2.1 Introducción

El objetivo primario de este capítulo es ofrecer una valoración general de las características que tendrá el sistema a desarrollar, describiendo las personas que se relacionan con el sistema y cómo interactúan con la aplicación. También se abordarán detalladamente los requerimientos funcionales y no funcionales, con los que debe cumplir el sistema una vez implementado.

2.2 Descripción del sistema

El sistema que se propone permitirá a los usuarios contribuir al proceso de actualización de la base de datos de URLs categorizadas de Filpacon. En aras del cumplimiento de este objetivo, será estructurado en seis módulos en los que se agruparán las tareas de gestión de usuarios, categorías, sugerencias, URLs, solicitudes de revisión y las listas de URLs categorizadas.

La aplicación debe permitir que cualquier usuario sin necesidad de autenticarse consulte las categorías asociadas a una URL y sugiera la clasificación de una URL en una o varias categorías. Posteriormente un revisor de cada una de estas categorías analizará el contenido de la URL con el objetivo de determinar si pertenece o no a la categoría que él revisa y modificará el estado de la misma en el sistema (El estado de las sugerencias puede ser: no revisada, aceptada, denegada o pendiente). Los usuarios pueden registrarse en el sistema, al hacerlo podrán llevar un registro de sus sugerencias, el estado de las mismas y su nivel de eficacia, además tendrán la posibilidad de solicitar convertirse en revisores de una o varias categorías. Cuando un usuario se convierte en revisor asume la responsabilidad de revisar las sugerencias que les sean asignadas. Los administradores son los encargados de gestionar todas las tareas que realiza el sistema. También tendrán la posibilidad de generar un fichero con las listas de URLs categorizadas.

2.3 Personas que interactúan con el sistema

Invitado: Es el usuario que puede consultar las categorías asociadas a una URL y sugerir la clasificación de una URL en una o varias categorías, además tiene la posibilidad de convertirse en usuario registrado del sistema.

Registrado: Es el usuario registrado en el sistema que posee los mismos privilegios de un usuario invitado pero puede llevar un registro de sus sugerencias y modificar los datos de su perfil.

Revisor: Posee todos los privilegios de un usuario Miembro, además puede cambiar el estado de las sugerencias que le sean asignadas para su revisión.

Administrador: Es el usuario que posee todos los privilegios del sistema y es el encargado de gestionar todo lo relacionado con el mismo.

2.4 Requerimientos del sistema

Durante muchos años muchas aplicaciones han fallado (no se culminaron o no se usaron) porque existieron incongruencias entre lo que el cliente quería, lo que realmente necesitaba, lo que interpretaba cada miembro del equipo de proyecto y lo que realmente se obtiene. Lograr una comunicación efectiva entre los usuarios y el equipo de desarrollo con el objetivo de llegar a un entendimiento de lo que hay que hacer, es la clave del éxito en la producción de un software. Aquí radica la importancia que se le ha dado a la identificación de los requerimientos como parte del proceso de desarrollo del software.

Requisitos funcionales

A partir del análisis de la situación problemática planteada al inicio de esta investigación, se identifican las posibles prestaciones que el sistema deber tener para cumplir los objetivos propuestos, las cuales se definirán como requisitos funcionales de la aplicación a implementar. En ellos están incluidas las acciones ejecutadas por los usuarios y las propias del sistema. El sistema debe ser capaz de:

1. Listar usuarios.
2. Adicionar usuario.
3. Eliminar usuario.
4. Modificar datos de un usuario.
5. Buscar usuarios.
6. Listar categorías.
7. Adicionar categoría.
8. Eliminar categoría.

9. Modificar datos de una categoría.
10. Buscar categorías.
11. Listar sugerencias.
12. Buscar sugerencias.
13. Eliminar sugerencia.
14. Sugerir clasificación de URL.
15. Revisar sugerencia.
16. Consultar categorías asociadas a una URL.
17. Generar listas de URLs categorizadas.
18. Solicitar convertirse revisor.
19. Asignar sugerencias a revisor.
20. Convertirse en usuario registrado.
21. Iniciar sesión de usuario.
22. Cerrar sesión de usuario.

Requisitos no funcionales

Las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es un sistema, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Por ello una vez determinado las funcionalidades del sistema, se deben analizar las propiedades o cualidades que la aplicación debe tener, permitiendo así, sea un software de calidad.

Relacionados con la calidad

El estándar ISO/IEC 9126 define un modelo de calidad de software en el que se presentan seis características de calidad. Como esta normativa define los elementos de calidad que debe tener un software, estos se toman como requisitos no funcionales que debe cumplir el sistema, siempre y cuando se mantengan las condiciones especificadas para su uso, ellos son:

- Debe ser un sistema funcional, o sea, satisfacer los requisitos funcionales declarados e implícitos.
- Debe ser un sistema confiable, de modo que mantenga el mismo nivel de ejecución a lo largo del tiempo.

- Debe ser un sistema usable, el sistema debe permitir a usuarios con un nivel mínimo o básico en conocimientos de informática, acceder a él.
- Debe ser un sistema que funcione eficientemente consumiendo el mínimo de recursos.
- Debe ser un sistema donde las modificaciones puedan hacerse fácilmente.
- Debe ser un sistema portable de forma tal que pueda ser transferido de un ambiente a otro, reemplazado por nuevas versiones y además fácil de instalar y adaptar.

Relacionados con la seguridad

La seguridad es un asunto de interés en el desarrollo de cualquier tipo de aplicación ya que tienen un impacto directo en la funcionabilidad y por tanto en la calidad final. Para las aplicaciones Web este aspecto adquiere dimensiones propias a partir de las características que poseen. Puede descomponerse en los siguientes puntos a tener en cuenta:

Confidencialidad: Significa que los datos intercambiados entre el consumidor y el proveedor no deben ser accesibles por terceros.

Integridad: Consiste en que nadie puede modificar la información intercambiada.

Disponibilidad: En este marco tiene que ver más con mantener la aplicación accesible todo el tiempo.

Autenticación: Es el proceso mediante el cual se comprueba que alguien o algo es realmente quien dice ser, puede ser persona e incluso otra aplicación.

Autorización: Es el proceso de asignar a un usuario autenticado los privilegios necesarios para acceder a la información, según las características del mismo.

Relacionados con la interfaz de usuario

En el marco de las aplicaciones Web la estructura de navegación y la presentación son aspectos importantes. Estos requisitos garantizan que el sistema cuente con la apariencia necesaria para ser de aceptación por el usuario.

- Debe poseer un diseño líquido, es decir debe adaptarse a las dimensiones de la pantalla.

- Debe utilizarse un patrón de navegación que permita estructurarla y hacerla intuitiva. La interfaz debe tener el mismo diseño para todos los usuarios.
- Deben usarse colores apropiados acorde al propósito de la interfaz.

Relacionados con el entorno o ambiente

- Describen como el sistema estará relacionado con su entorno y los requisitos mínimos para que pueda ser usado.
- El servidor debe poseer las versiones del servidor Web Apache2 y PHP5 iguales o superiores a la 2.2 y la 5.2 respectivamente. Además debe contar con el módulo libapache2-mod-php5 para permitir el soporte de PHP5 en el servidor Web y con las extensiones php5-pgsql y php5-json para este lenguaje.
- El servidor Web mencionado anteriormente debe tener habilitado el soporte para SSL.
- El navegador que se utilice debe tener activado el soporte para cookies y javascript.

2.5 Fase 1: Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. (12)

Historias de usuarios

Representan una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

Estas deben proporcionar solo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia, difiere de los casos de uso porque son escritos

por el cliente, no por los programadores, empleando terminología del cliente. ”Las historias de usuario son más amigables que los casos de uso formales”. (12)

Las Historias de Usuario tienen tres aspectos:

- **Tarjeta:** En ella se almacena suficiente información para identificar y detallar la historia.
- **Conversación:** Cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación).
- **Pruebas de Aceptación:** Permite confirmar que la historia ha sido implementada correctamente.

Historia de Usuario		
No:	**Iteración Asignada:**	**Nombre:**
Usuario:		
Prioridad del Negocio: (alta/media/baja)		**Nivel de Complejidad: (alta/media/baja)**
Puntos de Estimación:		**Riesgo en el Desarrollo:**
Descripción		
Observaciones:		

Cuadro 2.1: Descripción de la tarjeta Historia de Usuario

Campos de la Tarjeta Historia de Usuario (12)

Número: Índice de la historia de usuario. Es un número único que se le asigna a cada historia de usuario con el fin de lograr una mejor organización de estas.

Iteración Asignada: Define en que iteración del proceso de desarrollo de software será implementada la historia de usuario.

Nombre: Nombre de la Historia de Usuario. Debe ser descriptivo, en la medida de las posibilidades, de lo que se implementará y no muy extenso.

Usuario: Muestra los roles que van a interactuar con las funcionalidades relacionadas en la historia de usuario.

Prioridad en el Negocio: Define la prioridad que presenta la historia de usuario para el negocio, puede ser: Alta, Media o Baja.

Nivel de Complejidad: Define el nivel de complejidad que presenta la historia de usuario para su implementación, puede ser: Alta, Media o Baja.

Descripción: Se describe el requerimiento que da origen a la Historia de Usuario. La descripción debe ser corta, precisa y dejar claro qué es lo que se desea hacer.

Observaciones: Aspectos que se deben tener en cuenta para implementar la historia de usuario. Precondiciones o poscondiciones que se deben tener presentes al implementar la historia de usuario.

Para la construcción de la solución propuesta se han identificado 11 Historias de Usuarios que serán analizadas a continuación.

Historia de Usuario		
No: 1	**Iteración Asignada:** 1	**Nombre:** Gestionar usuarios
Usuario: Administrador		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta
Descripción: Agrupa las acciones: adicionar y eliminar usuario, modificar los datos del mismo, mostrar el listado de usuarios, buscar los usuarios que coincidan con un criterio de búsqueda especificado e invitar a uno o varios usuarios a convertirse en revisores.		
Observaciones: Los datos que se necesitan conocer de cada usuario son: nick, nombre y apellidos, dirección de correo electrónico, el rol que desempeña, su grado de fiabilidad		

determinado a partir del análisis de la eficacia de sus sugerencias y si el usuario desea o no, convertirse en revisor. La búsqueda puede realizarse con cualquier combinación de los siguientes criterios: • nick • nombre y apellidos • rol • grado de fiabilidad • quiere ser revisor • fecha de creación dentro del rango especificado • última fecha de acceso al sistema esté dentro del rango especificado. Se hace referencia a los requisitos: R1, R2, R3, R4, R5

Cuadro 2.2: HU "Gestionar Usuarios"

Historia de Usuario		
No: 2	**Iteración Asignada:** 1	**Nombre:** Gestionar categorías
Usuario: Administrador		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** baja
Descripción: Agrupa las acciones: adicionar y eliminar categoría, modificar los datos de la misma, mostrar el listado de categorías y buscar las categorías que coincidan con un criterio de búsqueda especificado.		
Observaciones: Los datos que se necesita conocer de cada categoría son: nombre y descripción. La búsqueda puede realizarse con cualquier combinación de los siguientes criterios: • nombre • descripción Se hace referencia a los requisitos: R6, R7, R8, R9, R10		

Cuadro 2.3: HU "Gestionar Categorías"

Historia de Usuario		
No: 3	**Iteración Asignada:** 1	**Nombre:** Gestionar sugerencias
Usuario: Administrador, Revisor		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta
Descripción: Agrupa las acciones: Listar, buscar y eliminar sugerencias.		
Observaciones: En el listado, de cada sugerencia se muestra la URL, el usuario que la realizó, el revisor que se le asignó y el estado de la misma. La búsqueda puede realizarse con cualquier combinación de los siguientes criterios: • URL • categoría • usuario que realizó la sugerencia • estado de la sugerencia • fecha de creación dentro del rango especificado Se hace referencia a los requisitos: R11, R12, R13		

Cuadro 2.4: HU “Gestionar Sugerencias”

Historia de Usuario		
No: 4	**Iteración Asignada:** 1	**Nombre:** Sugerir clasificación de URL
Usuario: Administrador, Revisor, Registrado, Invitado		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta
Descripción: Se brinda la opción de sugerir la clasificación de una URL en una o varias categorías previamente definidas en el sistema. Dicha sugerencia será posteriormente revisada para su posible aceptación o denegación.		
Observaciones: Los datos que se necesita conocer de cada sugerencia son: URL, categorías sugeridas para la clasificación de la URL, usuario que la realizó y la fecha en que tuvo lugar. La búsqueda puede realizarse con cualquier combinación de los siguientes criterios:		

- URL
- categoría
- usuario que hizo la sugerencia
- fecha de creación dentro del rango especificado

Se hace referencia a los requisitos: R14

Cuadro 2.5: HU "Sugerir Clasificación de URL"

Historia de Usuario		
No: 5	**Iteración Asignada:** 1	**Nombre:** Revisar sugerencia
Usuario: Administrador, Revisor		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta
Descripción: Consiste en analizar el contenido de la URL especificada en una sugerencia y luego de determinar si dicha sugerencia es acertada, se decide su aceptación o denegación.		
Observaciones: El estado de una sugerencia puede ser: • No revisada • Aceptada • denegada • pendiente (En caso de no poderse determinar si una sugerencia es acertada o no, se marca como pendiente, para su posterior análisis.) Se hace referencia a los requisitos: R15		

Cuadro 2.6: HU "Revisar Sugerencia"

Historia de Usuario		
No: 6	**Iteración Asignada:** 2	**Nombre:** Consultar categorías asociadas a una URL
Usuario: Administrador, Revisor, Miembro, Invitado		

Prioridad del Negocio: media	**Nivel de Complejidad:** media
Descripción: El sistema muestra las categorías asociadas a una URL especificada por el usuario.	
Observaciones: Se hace referencia a los requisitos: R16	

Cuadro 2.7: HU "Consultar categorías asociadas a una URL"

Historia de Usuario		
No: 7	**Iteración Asignada:** 2	**Nombre:** Generar listas de URLs categorizadas
Usuario: Administrador		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta
Descripción: El sistema brinda la opción de generar listas de URLs categorizadas que puedan ser utilizadas posteriormente para la actualización de la base de datos de Filpacon.		
Observaciones: Se hace referencia al requisito: R17		

Cuadro 2.8: HU "Generar listas de URLs categorizadas"

Historia de Usuario		
No: 8	**Iteración Asignada:** 2	**Nombre:** Solicitar convertirse en revisor
Usuario: Miembro		
Prioridad del Negocio: Media		**Nivel de Complejidad:** Alta
Descripción: El usuario puede especificar que desea unirse al equipo de revisores, además de además de esto, debe especificar las categorías que estaría dispuesto a revisar. El sistema debe mostrar una explicación de los deberes de un revisor, cuáles son sus responsabilidades y cómo debe llevar a cabo su labor.		

Cuadro 2.9: HU "Solicitar convertirse en revisor"

Historia de Usuario

No: 9	**Iteración Asignada:** 1	**Nombre:** Asignar sugerencias a revisor
Usuario: Administrador		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta
Descripción: Cada categoría sugerida a una URL, será asignada al revisor de esta categoría que menos sugerencias asignadas tenga.		
Observaciones: Se hace referencia a los requisitos: R19		

Cuadro 2.10: HU "Asignar sugerencias a revisor"

Historia de Usuario		
No: 10	**Iteración Asignada:** 2	**Nombre:** Convertirse en usuario registrado
Usuario: Invitado		
Prioridad del Negocio: Media		**Nivel de Complejidad:** Media
Descripción: El sistema brinda la posibilidad al usuario de convertirse en usuario registrado del sistema.		
Observaciones: Los datos que se necesitan para convertirse en usuario registrado son: Nick, nombre y apellidos, contraseña, dirección de correo electrónico, país, si el usuario desea o no, convertirse en revisor, en caso que desee convertirse en revisor las categorías que quiera revisar. Se hace referencia a los requisitos: R20		

Cuadro 2.11: HU "Convertirse en usuario registrado"

Historia de Usuario		
No: 11	**Iteración Asignada:** 1	**Nombre:** Gestionar sesión de usuario
Usuario: Administrador, Revisor, Miembro, Invitado		
Prioridad del Negocio: Alta		**Nivel de Complejidad:** Alta

Descripción: El usuario puede acceder al sistema a través de su nombre de usuario y contraseña, además de poder salir de su sesión luego de haber accedido a la misma.
Observaciones: En dependencia de sus privilegios, el usuario puede acceder a las distintas funcionalidades del sistema Se hace referencia a los requisitos: R21, R22

Cuadro 2.12: HU "Gestionar sesión de usuario"

2.6 Fase 2: Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación. (12)

Estimación de esfuerzos por Historias de Usuario

En la siguiente tabla se muestran los resultados de la estimación realizada con cada una de las historias de usuarios determinadas para la elaboración del sistema propuesto.

Historias de Usuario	Puntos de Estimación

Gestionar usuarios	1
Gestionar categorías	
Gestionar sesión de usuario	
Gestionar sugerencias	1
Revisar sugerencia	
Sugerir clasificación de URL	
Consultar categorías asociadas a una URL	1
Generar listas de URLs categorizadas	2
Solicitar convertirse en revisor	1
Convertirse en usuario registrado	
Asignar sugerencias a revisor	2

Cuadro 2.15: Estimación de esfuerzos por historias de usuario

Planificación de iteraciones

Una vez identificadas las historias de usuarios y estimado el esfuerzo asociado a la implementación de las mismas se procede a especificar el número de iteraciones necesario para el desarrollo del sistema. Además se definen que historias de usuario serán implementadas en cada iteración del sistema. Para el desarrollo de la presente propuesta se realizarán dos iteraciones.

Iteración 1: En esta iteración se implementaran las historias de usuarios que presentan mayor prioridad para el cliente. Luego de finalizar esta iteración se contará con una versión de prueba que agrupará las funcionalidades: Gestionar usuarios, Gestionar categorías, Gestionar sugerencias, Sugerir clasificación de URL, Asignar sugerencias a revisor, revisar sugerencia y Gestionar sesión de usuario.

Iteración 2: En esta iteración se implementaran las restantes historias de usuarios, o sea, aquellas que presentan media o baja prioridad para el cliente. Al finalizar esta iteración se debe contar con la versión inicial del sistema que agrupa todas las funcionalidades descritas anteriormente.

2.7 Plan de duración de las iteraciones

La metodología XP propone la realización de un plan de duración de las iteraciones con el fin de ilustrar la duración de cada iteración y las historias de usuarios que serán implementadas en cada una de ellas. Es importante destacar que el este plan se realiza teniendo en cuenta el equipo de desarrollo existente.

Iteraciones	Orden de las Historias de Usuario a implementar	Duración total (Semanas)
Iteración 1	1. Gestionar usuarios 2. Gestionar categorías 3. Gestionar sesión de usuario 4. Gestionar sugerencias 5. Sugerir clasificación de URL 6. Asignar sugerencias a revisor 7. Revisar sugerencia	4 (Se entregará 15/marzo/2010)
Iteración 2	1. Consultar categorías asociadas a una URL 2. Generar listas de URLs categorizadas 3. Solicitar convertirse en revisor 4. Convertirse en usuario registrado	4 (Se entregará 20/mayo/2010)

Cuadro 2.16: Plan de duración de iteraciones

Como se puede observar en la tabla anterior la duración total de las iteraciones es de ocho semanas y la fecha de entrega del sistema con todas sus historias de usuarios implementadas será el 20 de Mayo del 2010.

2.8 Conclusiones

Del presente capítulo se puede concluir que para dar solución a la situación problemática planteada se necesita una aplicación que responda a los 22 requisitos funcionales identificados, estos requisitos estarán contenidos en las 11 historias de usuario descritas anteriormente. Para la implementación de las historias de usuarios se realizarán 2 iteraciones, en la primera iteración se le dará solución a las historias de usuarios que presentan mayor prioridad para el cliente y en la segunda las restantes historias de usuarios. Se estimó que la aplicación estará culminada el 20 de mayo del 2010.

Capítulo 3 Diseño del sistema.

3.1 Introducción

La metodología de desarrollo XP establece prácticas especializadas, que inciden directamente en la realización y elaboración del diseño de un software, sin embargo no requiere que la representación del sistema sea mediante diagramas de clases basados en UML, sino que pueden emplearse indistintamente sencillos esquemas descritos en pizarras u otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración).(14)

En el presente capítulo se analizará la arquitectura de software, según el framework de desarrollo Symfony, utilizada para la confección del sistema que se presenta. Posteriormente se describirán las tarjetas CRC generadas para la confección de la solución propuesta.

3.2 Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (15)

El patrón MVC

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles:

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes al gestor de base de datos utilizado por la aplicación. En la figura 3.1 se ilustra mejor el funcionamiento de este patrón.

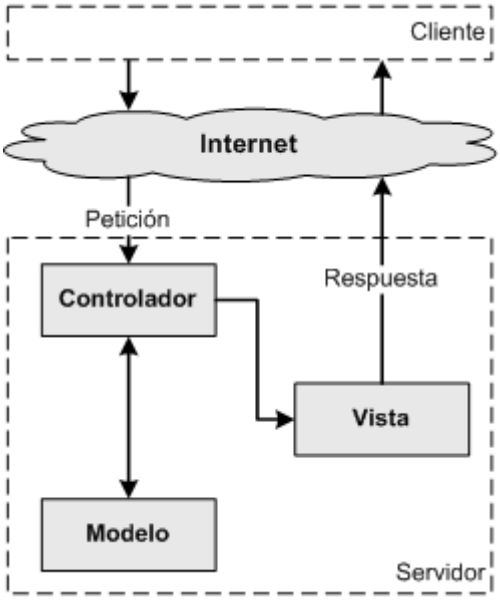

Figura 3.1: Patrón MVC

Implementación del patrón MVC por Symfony

Symfony ha sido escrito completamente en PHP5 con el objetivo de aprovechar todas las ventajas de esta versión del lenguaje y se basa en el patrón de diseño MVC, implementándolo de modo que el desarrollo de aplicaciones sea rápido y sencillo. Además se hace una separación en capas más allá del MVC tal como se muestra en la figura 3.2.

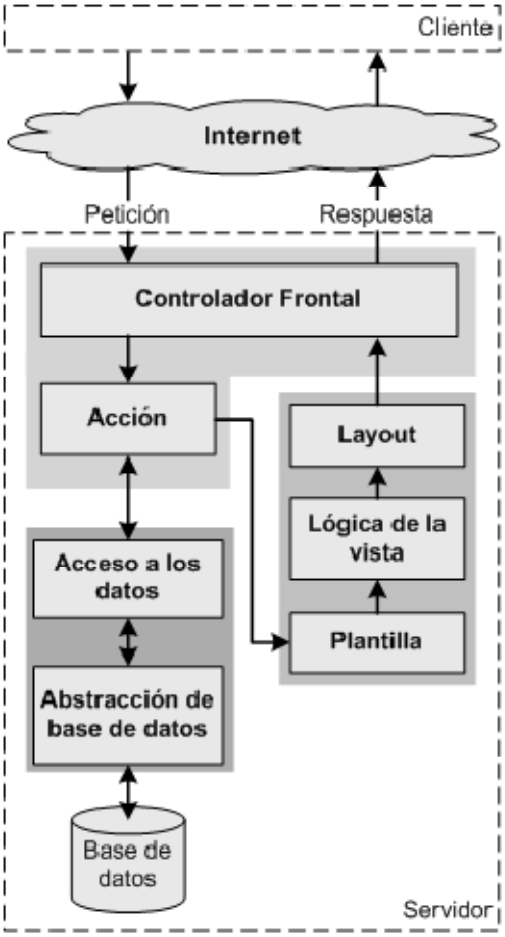

Figura3.2: Patrón MVC por Symfony

En esta variante el controlador que contiene el código que liga la lógica de negocio con la presentación, está compuesto por el controlador frontal que es el único punto de entrada a la aplicación y las acciones que se encargan de verificar la validez de las peticiones y preparar los datos para la vista. Esta última está formada por el layout que posee el código común a todas las acciones, las plantillas que contienen el código asociado a cada acción en particular y la lógica que se puede manipular a través de un fichero de configuración sencillo sin necesidad de programarla. Por último, el modelo está formado por la capa de acceso a datos cuyas clases son generadas automáticamente por Propel (ORM) en función de las estructura de datos de la aplicación y por la capa de abstracción de la base de datos Creole, que es completamente transparente al programador; así que si se cambia el sistema gestor de bases de datos en

cualquier momento, no se debe reescribir ni una línea de código, siendo necesario solamente modificar un parámetro en un archivo de configuración.

Filtros

Los filtros son un mecanismo que Symfony usa para realizar las tareas que son comunes a toda la aplicación, tales como la validación y la seguridad. De hecho cada petición se procesa como una cadena de filtros que son ejecutados de forma sucesiva. A continuación se mencionan cuales son los filtros que se utilizarán en el sistema de categorización manual de URLs, en la figura 3.3 se ilustra la idea a través de un diagrama de secuencia.

- sfRenderingFilter: encargado de renderizar la vista.
- sfBasicSecurityFilter: chequea la seguridad de cada petición.
- sfCacheFilter: controla el mecanismo de cache del framework.
- sfCommonFilter: añade los ficheros javascript y css a la respuesta.
- sfFlashFilter: elimina las variables flash de la sesión.
- sfExecutionFilter: se encarga de validar los parámetros de la petición, la ejecución de la acción y de la vista.

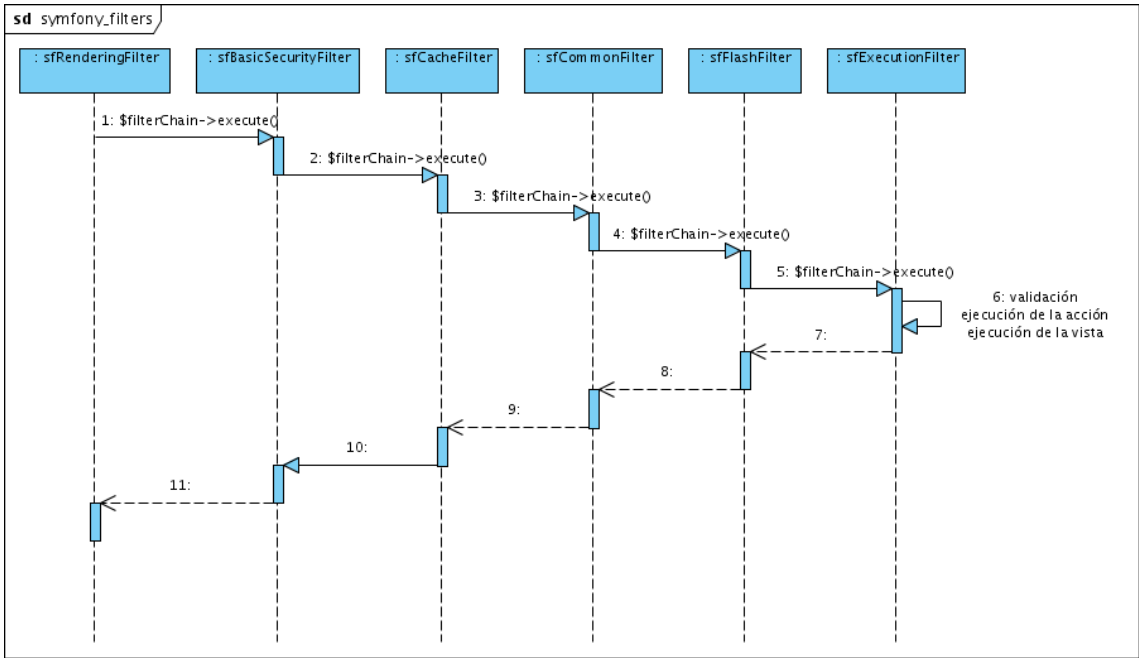

Figura 3.3: Cadena de filtros.

3.3 Organización de la aplicación

Symfony organiza el código fuente en una estructura de tipo proyecto y almacena los archivos del proyecto en una estructura estandarizada de tipo árbol.

En Symfony un proyecto se considera como “un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos”. Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones y cada aplicación está formada por uno o más módulos. Un módulo normalmente representa a una página Web o un grupo de páginas con un propósito relacionado y almacenan las acciones que representan cada una de las operaciones que se pueden realizar en el mismo.

3.4 Patrones de diseño

Symfony sigue las mejores prácticas y patrones de diseño para el desarrollo de aplicaciones Web. Seguidamente se mencionan cuales son aquellos patrones más significativos que se utilizan.

Las clases que conforman el núcleo están basadas en el patrón Factory Method permitiendo la creación fácil de los objetos y su extensión de forma sencilla.

El patrón Decorator es usado para decorar el layout de la aplicación con el template asociado a cada acción, dando como resultado la vista que es mostrada al usuario. Esto se ilustra en la figura 3.4.

Figura 3.4: Patrón Decorator entre el layout y el template

El patrón Singleton es usado en Symfony para permitir el acceso desde cualquier lugar de la aplicación a los objetos relacionado con el núcleo del framework.

La ejecución de la secuencia de filtros se basa en el patrón Chain of Responsibility que permite que más de un objeto pueda manejar una petición.

Los patrones anteriores pertenecen a los conocidos como GoF[5]. Además de ellos se utilizan otros descritos por Martin Fowler en (16), entre los que están Front controller, Row Data Gateway y Table Data Gateway. Los dos últimos son usados en las clases de acceso a datos y permiten realizar operaciones sobre un registro de una tabla y un conjunto de estos respectivamente.

3.5 Validación y tratamiento de errores

Symfony provee una serie de mecanismos para que la validación y el tratamiento de errores no sea una tarea compleja, pues posee un mecanismo de validación en el servidor a través del uso de ficheros de validación y una serie de validadores que permite hacer este proceso más fácil y extensible, sin la consecuente promoción de errores que puede traer consigo la programación relacionada con este aspecto. No obstante hay casos especiales donde es necesario hacer la validación manualmente o combinar ambos métodos. Una característica importante de Symfony en este sentido es que permite el relleno automático de datos (repopulation), lo que asegura la obtención de datos correctos y mejora la experiencia de los usuarios. La figura 3.5 muestra como se realiza este proceso.

[5] Gan of Four patrones de diseño orientado a objetos

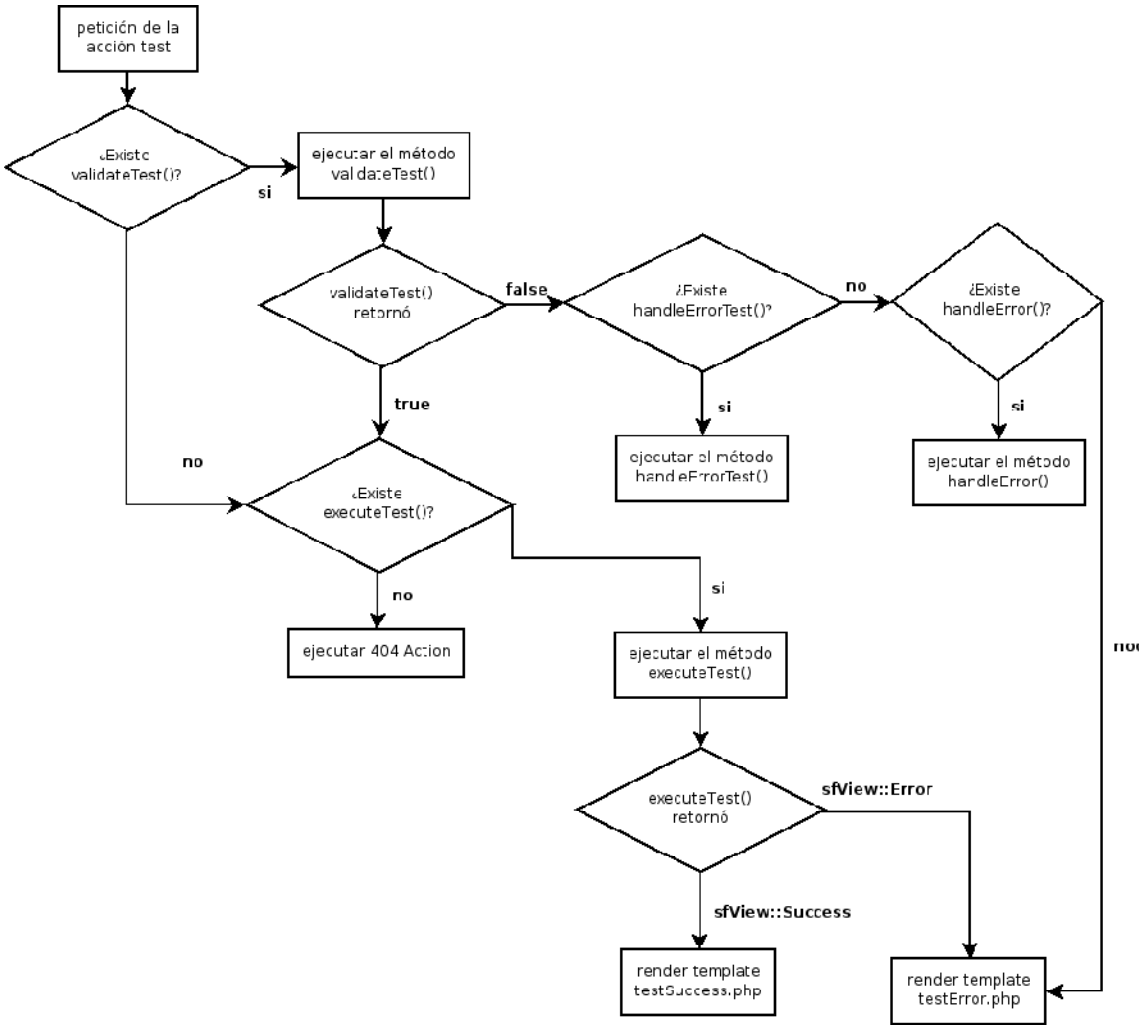

Figura 3.5: Diagrama de flujo del proceso de validación

3.6 Seguridad

La posibilidad de ejecutar una acción puede ser restringida a usuarios con ciertos privilegios. Las herramientas proporcionadas por Symfony para este propósito permiten la creación de aplicaciones seguras, en las que los usuarios necesitan estar autenticados antes de acceder a alguna característica o a partes de la aplicación.

Autenticación y autorización

Cuando se diseña una aplicación Web es importante definir los aspectos relacionados con la seguridad de la misma. En este contexto es importante prestar atención a la autenticación y autorización de los usuarios. Symfony provee todo lo necesario para que ambas cosas puedan realizarse de forma sencilla, mejorando el mecanismo de sesiones que utiliza PHP y haciéndolo más configurable y fácil de usar. Una vez que un usuario se autentica, se le asignan los privilegios necesarios (credenciales) para que pueda utilizar las funcionalidades que brinda la aplicación según el rol que desempeñe.

Teniendo en cuenta lo planteado en el capítulo anterior el sistema cuenta con cuatro tipos de usuarios: Administrador, Revisor, Registrado e Invitado por lo que solo se necesita definir las acciones que podrá realizar cada rol sobre el sistema pues Symfony se encarga de verificar si el usuario actual tiene los privilegios necesarios para acceder a una determina acción.

Defensa contra ataques

Existen numerosos ataques informáticos que pueden afectar a las aplicaciones Web. Cuando se produce alguno de ellos es porque se explota alguna vulnerabilidad existente relacionada con los principios de diseño e implementación y con los mecanismos de control o defensa con que se cuente. A continuación se describen brevemente los principales ataques que podrían afectar la seguridad de la aplicación y por tanto la integridad de los datos que se manejan, así como los principios que regirán la prevención de los mismos en el contexto de Symfony.

Cross Site Scripting (XSS) es un tipo de ataque que aprovecha la confianza que un usuario posee en un sitio determinado, se basa en la inyección de códigos hechos en javascript que pueden impedir el uso normal de una aplicación. Para su prevención Symfony posee un mecanismo de escape que puede ser activado mediante un parámetro en un fichero de configuración.

Cross Site Request Forgery (CSRF) aunque parezca parecido, es contrario al XSS ya que este se basa en aprovechar la confianza que un sitio posee en un usuario para realizar peticiones, al mismo tiempo que puede comprometer su seguridad. Aunque en la versión que se utiliza no posee defensa contra el mismo, existe una extensión que permite añadirlo al framework como un filtro, que se insertará en la secuencia de filtros después del sfBasicSecurityFilter.

SQL injection es un tipo de ataque que permite realizar acciones con el objetivo de corromper la información de un sistema o ganar acceso al mismo través de la entrada de sentencias SQL maliciosas por medio de formularios o como parámetros de la petición en la URL. EL uso de Propel y Creole ayuda a prevenirlo, ya que ambos poseen los mecanismos de escape necesarios para este tipo de ataque.

Como se pudo apreciar Symfony posee una serie de mecanismos que brinda cierta confianza a los desarrolladores de aplicaciones web, pero cabe destacar que no existen sistemas totalmente seguros y que la protección contra el acceso físico al servidor recae en la institución donde se utilice el sistema.

3.7 Diseño de la base de datos

El diseño de una base de datos es un paso fundamental en la construcción de un sistema que gestione gran cantidad de información, pues es muy importante que los datos se almacenen de forma coherente y organizada, para que dicha información no se pierda.

El principal objetivo para realizar el diseño de la base de datos en cuestión, es lograr la persistencia de los datos de los usuarios, así como la información de los recursos que se gestionan por los mismos, controlando que esta información no tenga redundancia y no existan datos innecesarios que no cumplen ningún objetivo al ser guardados. En el anexo A se puede observar el diagrama entidad-relación que describe cómo está estructurada la base de datos de la solución propuesta.

3.8 Tarjetas CRC

Ward Cunningham y Kent Beck propusieron la utilización de una herramienta de reflexión en el diseño de software orientado a objetos, así surgieron las tarjetas Contenido, Responsabilidad y Colaboración. Para la creación de estas es necesario primeramente definir todas las clases y su interacción luego de estar implementadas.

La utilización de estas tarjetas reduce al mínimo la complejidad del diseño. Esto logra que el programador se centre en el diseño de las bases fundamentales de la clase y le impide entrar en detalles de su funcionamiento interno que en un momento pueden llegar a ser contraproducentes. Estas tarjetas propician que los diseñadores no otorguen demasiadas responsabilidades a las clases, las CRC tiene la ventaja que pueden ser portátiles y pueden fácilmente ser expuestas en una mesa y reformularse el

diseño en el debate con otras personas.

Tarjeta CRC	
Clase: Nombre de la clase que se está modelando.	
Súper Clase: Nombre de la clase padre en la herencia.	
Sub Clase(s): Nombre de la(s) clase(s) hija en la herencia.	
Responsabilidades: Es una descripción de alto nivel del propósito de la clase.	**Colaboraciones:** Indica con cuáles otras clases se requiere relación para cumplir la responsabilidad.

Cuadro 3.1: Descripción tarjeta CRC

En Symfony cuando se crea una nueva clase, se generan con ella las clases bases del modelo y por supuesto las clases verdaderas del modelo que heredan de las clases bases. En este trabajo solo se describirán las clases verdaderas puesto que estas son las que los programadores modificaran.

Por cada clase verdadera del modelo se crean dos tipos de clases, las clases objetos y las clases “peer”. Las clases objeto representan un registro de la base de datos. Permiten acceder a las columnas de un registro y los registros relacionados; por otro lado las clases “peer” que son clases que poseen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven por lo general un objeto o una colección de objetos de la clase objeto relacionada.

Por esta razón se realizará solo la descripción de las clases objeto identificadas a partir del análisis de las diferentes historias de usuario.

Tarjeta CRC	
Clase: CategorizedUrl	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Trabajar con las sugerencias que han sido aprovadas	**Colaboraciones:** Url, Category

Cuadro 3.2: Tarjeta CRC "CategorizedUrl"

Tarjeta CRC	
Clase: Category	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Trabajar con las categorías que ya han sido definidas en el sistema.	**Colaboraciones:**

Cuadro 3.3: Tarjeta CRC "Category"

Tarjeta CRC	
Clase: Url	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Trabajar con las URL que se adicionan en el sistema como parte de la sugerencia.	**Colaboraciones:**

Cuadro 3.4: Tarjeta CRC "URL"

Tarjeta CRC	
Clase: Profile	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Trabajar con los usuarios que se registran en el sistema.	**Colaboraciones:**

Cuadro 3.5: Tarjeta CRC "Profile"

Tarjeta CRC
Clase: Reviewer_Category

Súper Clase:	
Sub Clase(s):	
Responsabilidades: Almacenar los usuarios revisores en cada una de las categorías.	**Colaboraciones:** Profile, Category, RequestToReview

Cuadro 3.6: Tarjeta CRC "Reviewer_Category"

Tarjeta CRC	
Clase: Suggestion	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Trabajar con las sugerencias que realizan los diferentes usuarios.	**Colaboraciones:** Url, Category, Profile, Reviewer_Category*, CategorizedUrl*

Cuadro 3.7: Tarjeta CRC "Suggestion"

Tarjeta CRC	
Clase: RequestToReview	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Trabajar con las peticiones que realizan los usuarios de convertirse en revisores	**Colaboraciones:** Url, Category, Profile, Reviewer_Category

Cuadro 3.8: Tarjeta CRC "RequestToReview"

3.9 Conclusiones

El diseño de un software es un paso decisivo en la construcción del mismo, por esta razón en el presente capítulo se crearon las bases para la implementación de un sistema de categorización manual de URLs basado en el diseño y arquitectura propuestos por el framework Symfony. Para describir el diseño la metodología XP genera un artefacto llamado tarjetas CRC, anteriormente se realizó la descripción de siete

de ellas. Otras ventajas del framework de desarrollo seleccionado para la confección de la aplicación son los mecanismos de defensa y seguridad que posee, en el presente capítulo se describieron algunos de los ataques más importantes que puede sufrir el sistema presentado, así como los mecanismos que responden a dichas amenazas.

Capítulo 4 Implementación y Pruebas

4.1 Introducción

En el comienzo de cada iteración se realiza una revisión del plan de iteraciones y se seleccionan las historias de usuario que serán desarrolladas durante el transcurso de la iteración. Las HU se descomponen en tareas de desarrollo, asignando a un grupo de desarrollo (o una persona), la responsabilidad de su implementación. En el capítulo se describen las Pruebas de Aceptación creadas para ayudarnos a saber qué es exactamente lo que tiene que hacer el código a implementar y así una vez implementado lograr que pase dichas pruebas sin problemas. Teniendo en cuenta la planificación realizada anteriormente, se llevaron a cabo dos iteraciones de desarrollo sobre el sistema, obteniéndose como finalidad un producto funcional y con las características deseadas. A continuación se detallan cada una de las iteraciones.

4.2 Tareas de programación

Las tareas de programación son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, y poder ser implementadas entre uno y tres días ideales. La mayoría de las tareas de programación se derivan directamente de las historias de usuario. (17)

Cada historia de usuario puede ser dividida en varias tareas de programación sencillas. Para determinar las tareas de programación que componen a una historia de usuario se propone hacer una reunión con todos los miembros del equipo de desarrollo y obtener una buena lista con todas. (18)

Las tareas de programación técnicas son aquellas que no son resultado del análisis de ninguna historia de usuario pero deben ser realizadas para que el sistema funcione. (18)

Cada tarea de programación será comprobada a través de los casos de prueba. Las tareas de programación no tienen por qué ser comprendidas por el cliente. (17)

Tarea de programación	
Número tarea:	**Número historia:**
Nombre tarea:	

Tipo de tarea : Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable:	
Descripción:	

Cuadro 4.1: Descripción tarjeta Tarea de programación

Campos de la tarjeta tarea de programación

Número historia de usuario: Número de la historia de usuario a la que corresponde. Índice de la historia de usuario a la que se corresponde esta tarea de programación.

Número tarea: Índice de la tarea de programación. Es un número único que se le asigna a cada tarea de programación que pertenece a una historia de usuario determinada con el fin de lograr una mejor organización de éstas.

Nombre tarea: Nombre de la tarea de programación. Debe ser descriptivo, en la medida de las posibilidades, de lo que se realizará y no muy extenso.

Tipo de tarea: Informa el tipo de tarea a realizar. Las tareas pueden ser:

- **Desarrollo:** Tarea que se realizará por primera vez.
- **Corrección:** Tarea que se realiza a partir de una anterior que no se realizó correctamente, es decir no pasó todos los casos de prueba que le corresponden correctamente.
- **Mejora:** Tarea que se realiza a partir de una anterior que se realizó correctamente pero se incorporan nuevos requerimientos para la misma, ahora tendrá que ser modificada para pasar satisfactoriamente los nuevos casos de prueba adicionados.
- **Otra:** Tarea que no corresponde con ninguna de las anteriormente descritas.

Fecha inicio: Fecha en que se inicia la implementación de la tarea de programación.

Fecha fin: Fecha en que concluye la implementación de la tarea de programación.

Programador responsable: Programador responsable de implementar la tarea de programación.

Descripción: Se describe qué es lo que se desea realizar. La descripción debe ser corta y precisa.

Puntos estimados: Tiempo estimado que demorará la implementación de la tarea de programación. Tiempo ideal en que se estima se implementará la tarea de programación.

4.3 Pruebas

Se les llama casos de prueba a las pruebas funcionales o unitarias que se realizan al sistema para comprobar su funcionamiento.

Las pruebas funcionales no son más que pruebas escritas desde la perspectiva del cliente, y las pruebas unitarias son pruebas escritas desde la perspectiva del programador. Mientras un código no haya sido probado no existe. Estos pueden ser adicionados o eliminados en cualquier momento. El objetivo es tener una forma que permita que el cliente conozca cuando una historia de usuario está lista. (17)

Los casos de prueba se deben escribir antes de comenzar la implementación, pero siempre que sea necesario se puede incluir uno nuevo. No existe restricción de cantidad para estos, se deben escribir casos de prueba hasta que quede claro el objetivo de la historia de usuario y se verifique que cumpla con todos los requerimientos. (19)

XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

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

Las pruebas del sistema tienen como objetivo verificar la funcionalidad del sistema a través de sus interfaces externas comprobando que dicha funcionalidad sea la esperada en función de los requisitos del sistema. Generalmente las pruebas del sistema son desarrolladas por los programadores para verificar que su sistema se comporta de la manera esperada, por lo que podrían encajar dentro de la definición de pruebas unitarias que propone XP. Sin embargo, las pruebas del sistema tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden encajar dentro de la categoría de pruebas de aceptación.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente.

El cliente es la persona más indicada para escribir las pruebas funcionales porque no existe nadie que tenga una visión de lo que desea mejor que él. (19)

Una historia de usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable. (17)

Prueba de Aceptación	
Código:	**Historia de Usuario (Número):**
Nombre:	
Descripción:	
Condiciones de Ejecución:	
Entrada / Pasos de ejecución:	
Resultado Esperado:	
Evaluación de la Prueba:	

Cuadro 4.2: Descripción tarjeta Prueba de Aceptación

Campos de la tarjeta Prueba de Aceptación

Número historia de usuario: Número de la historia de usuario a la que corresponde. Índice de la historia de usuario a la que se le desea comprobar este aspecto.

Código: Índice del caso de prueba. Es un número único que se le asigna a cada caso de prueba que pertenece a una historia de usuario determinada con el fin de lograr una mejor organización de estos.

Nombre: Nombre del caso de prueba. Debe ser descriptivo, en la medida de las posibilidades, de lo que se comprobará y no muy extenso.

Descripción: Se describe qué es lo que se desea probar. La descripción debe ser corta y precisa.

Condiciones de ejecución: Condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba.

Entradas: Entradas al caso de prueba en caso de necesitarlas.

Resultado esperado: Resultado que se desea tenga el caso de prueba. Descripción breve de lo que debe suceder.

Evaluación: Se evalúa si el caso de prueba tuvo éxito o no. En caso de ser exitoso se asigna un resultado de satisfactorio, en caso contrario insatisfactorio.

4.4 Iteración 1

En el presente epígrafe se realizará la descripción de las tarjetas tarea de programación y pruebas de aceptación correspondientes a la primera iteración del ciclo de desarrollo de la solución propuesta. Como se había anunciado anteriormente, al culminar esta iteración el software contará con las funcionalidades más importantes para el cliente, o sea serán implementadas las historias de usuario Gestionar usuarios, Gestionar Categorías, Gestionar sesión de usuario, Gestionar sugerencias, Sugerir clasificación de URL, Revisar sugerencia, Asignar sugerencias a revisor, estas son las historias de usuario que más peso poseen dentro de la aplicación que se desea desarrollar.

Tarea de Programación	
Número tarea: HU1_1	**Número historia:** 1

Nombre tarea: Gestionar Usuario	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 25/2/2010	**Fecha fin:** 26/2/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la posibilidad de adicionar o eliminar usuarios, modificar los datos, mostrar el listado de los usuarios, buscar los usuarios que coincidan con un criterio de búsqueda especificado e invitar a uno o varios usuarios a convertirse en revisores.	

Cuadro 4.3: TP "Gestionar Usuario"

Tarea de Programación	
Número tarea: HU2_1	**Número historia:** 2
Nombre tarea: Gestionar Categorías	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 26/2/2010	**Fecha fin:** 27/2/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinde la posibilidad de adicionar y eliminar categoría, modificar los datos de la misma, mostrar el listado de categorías y buscar las categorías que coincidan con un criterio de búsqueda especificado.	

Cuadro 4.4: TP "Gestionar Categoría"

Tarea de Programación	
Número tarea: HU4_1	**Número historia:** 4
Nombre tarea: Gestionar Sugerencias	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27/2/2010	**Fecha fin:** 28/2/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la posibilidad de listar, buscar y eliminar sugerencias.	

Cuadro 4.5: TP "Gestionar Sugerencias"

Tarea de Programación	
Número tarea: HU3_1	**Número historia:** 3
Nombre tarea: Sugerir clasificación de URL	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 28/2/2010	**Fecha fin:** 2/3/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la opción de sugerir la clasificación de una URL en una o varias categorías previamente definidas en el sistema.	

Cuadro 4.6: TP "Sugerir clasificación de URL"

Tarea de Programación	
Número tarea: HU6_1	**Número historia:** 6
Nombre tarea: Revisar sugerencia	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 2/3/2010	**Fecha fin:** 3/3/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la opción de aceptar o denegar una sugerencia de una URL, luego se guardará en la base de datos si es aceptada.	

Cuadro 4.7: TP "Revisar sugerencia"

Tarea de Programación	
Número tarea: HU5_1	**Número historia:** 5
Nombre tarea: Asignar sugerencias a revisor	
Tipo de tarea : Desarrollo	**Puntos estimados:** 2
Fecha inicio: 3/3/2010	**Fecha fin:**9/3/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la opción de asignar sugerencias a los diferentes revisores de cada una de las categorías. La asignación se hará al revisor que menos sugerencias asignadas tenga.	

Cuadro 4.8: TP "Asignar sugerencias a revisor"

Tarea de Programación	
Número tarea: HU11_1	**Número historia:** 11
Nombre tarea: Gestionar sesión usuario	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 9/3/2010	**Fecha fin:**10/3/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se debe mostrar las opciones o acciones que cada usuario puede realizar sobre el sistema, en dependencia de los permisos que posea el mismo.	

Cuadro 4.9: TP "Gestionar sesión usuario"

Pruebas de aceptación

A continuación se presentaran las pruebas de aceptación realizadas a las diferentes tareas de programación, estas se realizan como se mencionaba anteriormente para comprobar si cada tarea funciona correctamente y en concordancia con el propósito que se desarrolló.

Prueba de Aceptación	
Código: 1.1	**Historia de Usuario:** 1
Nombre: Comprobar gestión de usuarios (Adicionar usuario)	
Descripción: Prueba para verificar si adicionar un usuario funciona correctamente.	
Condiciones de Ejecución: Tratar de adicionar un usuario.	
Entrada / Pasos de ejecución: Se adiciona un usuario.	
Resultado Esperado: El usuario se adiciona en la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.10: PA "Comprobar gestión de usuarios"

Prueba de Aceptación	
Código: 1.2	**Historia de Usuario:** 1
Nombre: Comprobar gestión de usuarios (Eliminar usuario)	
Descripción: Prueba para verificar si eliminar un usuario funciona correctamente.	
Condiciones de Ejecución: Tratar de eliminar un usuario.	

Entrada / Pasos de ejecución: Se elimina un usuario.
Resultado Esperado: El usuario se elimina de la base de datos.
Evaluación de la Prueba: Evaluación satisfactoria

Cuadro 4.11: PA "Comprobar gestión de usuarios"

Prueba de Aceptación	
Código: 1.3	**Historia de Usuario:** 1
Nombre: Comprobar gestión de usuarios (Modificar datos de un usuario)	
Descripción: Prueba para verificar si modificar un usuario funciona correctamente.	
Condiciones de Ejecución: Tratar de modificar un usuario.	
Entrada / Pasos de ejecución: Se modifican los datos de un usuario.	
Resultado Esperado: El usuario se modifica en la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.12: PA "Comprobar gestión de usuarios"

Prueba de Aceptación	
Código: 1.4	**Historia de Usuario:** 1
Nombre: Comprobar gestión de usuarios (Buscar usuario por criterio de búsqueda)	
Descripción: Prueba para verificar si la búsqueda de un usuario por criterio de búsqueda especificado funciona correctamente.	
Condiciones de Ejecución: Tratar de buscar un usuario.	
Entrada / Pasos de ejecución: Se introduce el criterio de búsqueda.	
Resultado Esperado: El sistema muestra el usuario que responde al criterio de búsqueda introducido.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.13: PA "Comprobar gestión de usuarios"

Prueba de Aceptación	
Código: 1.5	**Historia de Usuario:** 1
Nombre: Comprobar gestión de usuarios (Invitar usuario a convertirse en revisor)	

Descripción: Prueba para verificar si invitar a un usuario funciona correctamente.
Condiciones de Ejecución: Tratar de invitar a un usuario a convertirse en revisor.
Entrada / Pasos de ejecución: Se invita a un usuario específico a convertirse en revisor.
Resultado Esperado: El usuario recibe la invitación de convertirse en revisor.
Evaluación de la Prueba: Evaluación satisfactoria

Cuadro 4.14: PA "Comprobar gestión de usuarios"

Prueba de Aceptación	
Código: 2.1	**Historia de Usuario:** 2
Nombre: Comprobar gestión de categorías (Adicionar categoría)	
Descripción: Prueba para verificar si adicionar una categoría funciona correctamente.	
Condiciones de Ejecución: Tratar de adicionar una categoría.	
Entrada / Pasos de ejecución: Se adiciona una categoría.	
Resultado Esperado: La categoría se adiciona en la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.15: PA "Comprobar gestión de categorías"

Prueba de Aceptación	
Código: 2.2	**Historia de Usuario:** 2
Nombre: Comprobar gestión de categorías (Eliminar categoría)	
Descripción: Prueba para verificar si eliminar una categoría funciona correctamente.	
Condiciones de Ejecución: Tratar de eliminar una categoría.	
Entrada / Pasos de ejecución: Se elimina una categoría.	
Resultado Esperado: La categoría se elimina de la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.16: PA "Comprobar gestión de categorías"

Prueba de Aceptación	
Código: 2.3	**Historia de Usuario:** 2

Nombre: Comprobar gestión de categoría (Modificar datos de una categoría)
Descripción: Prueba para verificar si modificar una categoría funciona correctamente.
Condiciones de Ejecución: Tratar de modificar una categoría.
Entrada / Pasos de ejecución: Se modifican los datos de una categoría.
Resultado Esperado: La categoría se modifica en la base de datos.
Evaluación de la Prueba: Evaluación satisfactoria

Cuadro 4.17: PA "Comprobar gestión de categorías"

Prueba de Aceptación	
Código: 2.4	**Historia de Usuario:** 2
Nombre: Comprobar gestión de categoría (Buscar categoría por criterio de búsqueda)	
Descripción: Prueba para verificar si la búsqueda de una categoría por criterio de búsqueda especificado funciona correctamente.	
Condiciones de Ejecución: Tratar de buscar una categoría.	
Entrada / Pasos de ejecución: Se introduce el criterio de búsqueda.	
Resultado Esperado: El sistema muestra la categoría que responde al criterio de búsqueda introducido.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.18: PA "Comprobar gestión de categorías"

Prueba de Aceptación	
Código: 3	**Historia de Usuario:** 3
Nombre: Comprobar realizar sugerencia	
Descripción: Prueba para verificar si realizar una sugerencia funciona correctamente.	
Condiciones de Ejecución: Tratar de realizar una sugerencia.	
Entrada / Pasos de ejecución: Se realiza una sugerencia.	
Resultado Esperado: La sugerencia se adiciona en la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.19: PA "Comprobar realizar sugerencia"

Prueba de Aceptación	
Código: 4.1	**Historia de Usuario:** 4
Nombre: Comprobar gestión de sugerencias (Buscar sugerencia por criterio de búsqueda)	
Descripción: Prueba para verificar si la búsqueda de una sugerencia por criterio de búsqueda especificado funciona correctamente.	
Condiciones de Ejecución: Tratar de buscar una sugerencia.	
Entrada / Pasos de ejecución: Se introduce el criterio de búsqueda.	
Resultado Esperado: El sistema muestra la sugerencia que responde al criterio de búsqueda introducido.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.20: PA "Comprobar gestión de sugerencias"

Prueba de Aceptación	
Código: 4.2	**Historia de Usuario:** 4
Nombre: Comprobar gestión de sugerencia (Eliminar sugerencia)	
Descripción: Prueba para verificar si eliminar una sugerencia funciona correctamente.	
Condiciones de Ejecución: Tratar de eliminar una sugerencia.	
Entrada / Pasos de ejecución: Se elimina una sugerencia.	
Resultado Esperado: La sugerencia se elimina de la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.21: PA "Comprobar gestión de sugerencias"

Prueba de Aceptación	
Código: 5	**Historia de Usuario:** 5
Nombre: Comprobar asignar sugerencia	
Descripción: Prueba para verificar si asignar una sugerencia funciona correctamente.	
Condiciones de Ejecución: Tratar de asignar una sugerencia a un revisor.	
Entrada / Pasos de ejecución: Se asigna una sugerencia.	
Resultado Esperado: El revisor con menos sugerencias asignadas recibe la sugerencia	

para ser revisada.
Evaluación de la Prueba: Evaluación satisfactoria

Cuadro 4.22: PA "Comprobar asignar sugerencias"

Prueba de Aceptación	
Código: 6	**Historia de Usuario:** 6
Nombre: Comprobar revisar sugerencia	
Descripción: Prueba para verificar si revisar una sugerencia funciona correctamente.	
Condiciones de Ejecución: Tratar de revisar una sugerencia.	
Entrada / Pasos de ejecución: Se revisa una sugerencia, se le asigna aceptada a la sugerencia.	
Resultado Esperado: La sugerencia se adiciona en la base de datos.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.23: PA "Comprobar revisar sugerencias"

Prueba de Aceptación	
Código: 11	**Historia de Usuario:** 11
Nombre: Comprobar gestión de sesión de usuario	
Descripción: Prueba para verificar si la gestión de sesión de usuario funciona correctamente.	
Condiciones de Ejecución: Entrar al sistema con los diferentes roles.	
Entrada / Pasos de ejecución: Se autentica en el sistema con los diferentes roles.	
Resultado Esperado: El sistema muestra las acciones que puede realizar un usuario en dependencia de los permisos del mismo.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.24: PA "Comprobar gestión de sesión de usuario"

4.5 Iteración 2

En el presente epígrafe se realizará la descripción de las tarjetas tarea de programación y pruebas de aceptación correspondientes a la segunda iteración del ciclo de desarrollo de la solución propuesta. Como

se había anunciado anteriormente, al culminar esta iteración el software contará con todas las funcionalidades, o sea serán implementadas todas las historias de usuario. Al finalizar esta iteración se contará con una versión inicial del sistema.

Tarea de Programación	
Número tarea: HU7_1	**Número historia:** 7
Nombre tarea: Consultar categorías asociadas	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11/3/2010	**Fecha fin:** 25/3/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la opción de buscar las categorías asociadas a una URL, estas se buscan en la tabla donde se adicionan las sugerencias aceptadas.	

Cuadro 4.25: TP "Consultar categorías asociadas"

Tarea de Programación	
Número tarea: HU8_1	**Número historia:** 8
Nombre tarea: Generar lista de URL categorizadas	
Tipo de tarea : Desarrollo	**Puntos estimados:** 2
Fecha inicio: 25/3/2010	**Fecha fin:** 6/5/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la opción de generar una lista de URLs categorizadas a partir de la tabla de sugerencias aceptadas.	

Cuadro 4.26: TP "Generar lista de URL categorizadas"

Tarea de Programación	
Número tarea: HU9_1	**Número historia:** 9
Nombre tarea: Convertirse en usuario miembro	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 7/5/2010	**Fecha fin:** 9/5/2010
Programador responsable: Raúl Amambay, Rolando González	

Descripción: Se brinda la opción de registrarse en el sistema, automáticamente se crea el perfil del usuario y se agrega el usuario a la base de datos de usuarios.

Cuadro 4.27: TP "Convertirse en usuario miembro"

Tarea de Programación	
Número tarea: HU10_1	**Número historia:** 10
Nombre tarea: Solicitar convertirse en revisor	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/5/2010	**Fecha fin:** 12/5/2010
Programador responsable: Raúl Amambay, Rolando González	
Descripción: Se brinda la opción al usuario miembro de solicitar convertirse en revisor, se marcará un campo en la tabla como que quiere ser revisor y se le notificará al administrador.	

Cuadro 4.28: TP "Solicitar convertirse en revisor"

Pruebas de aceptación.

A continuación se presentaran las pruebas de aceptación realizadas a las diferentes tareas de programación implementadas en la segunda iteración del ciclo de desarrollo del software.

Prueba de Aceptación	
Código: 7	**Historia de Usuario:** 7
Nombre: Comprobar consultar categorías asociadas a una URL	
Descripción: Prueba para verificar si consultar categorías asociadas a una URL funciona correctamente.	
Condiciones de Ejecución: Introducir una URL.	
Entrada / Pasos de ejecución: Se introduce una URL.	
Resultado Esperado: El sistema muestra las categorías asociadas a una URL.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.31: PA "Comprobar consultar categorías asociadas a una URL"

Prueba de Aceptación	
Código: 8	**Historia de Usuario:** 8
Nombre: Comprobar generar lista de URLs categorizadas	
Descripción: Prueba para verificar si se genera una lista de URLs categorizadas.	
Condiciones de Ejecución:	
Entrada / Pasos de ejecución: Se selecciona la opción generar lista de URLs.	
Resultado Esperado: El sistema genera las listas de URLs categorizadas.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.32: PA "Comprobar generar lista de URLs categorizadas"

Prueba de Aceptación	
Código: 9	**Historia de Usuario:** 9
Nombre: Comprobar convertirse en usuario miembro	
Descripción: Prueba para verificar si registrarse funciona correctamente.	
Condiciones de Ejecución: Introducir datos de usuario.	
Entrada / Pasos de ejecución: Se introducen los datos del usuario.	
Resultado Esperado: El sistema muestra la interfaz de usuario miembro del sistema.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.33: PA "Comprobar convertirse en usuario miembro"

Prueba de Aceptación	
Código: 10	**Historia de Usuario:** 10
Nombre: Comprobar solicitar ser revisor	
Descripción: Prueba para verificar si solicitar convertirse en revisor funciona correctamente.	
Condiciones de Ejecución: El usuario debe solicitar convertirse en revisor.	
Entrada / Pasos de ejecución: Se solicita ser revisor.	
Resultado Esperado: Se activa el campo quiero ser revisor en la tabla del usuario en la base de datos y se le notifica al administrador.	
Evaluación de la Prueba: Evaluación satisfactoria	

Cuadro 4.34: PA “Comprobar solicitar ser revisor”

4.6 Conclusiones

En el presente capítulo se realizó la descripción de los diferentes artefactos generados por las fases de implementación y pruebas propuestas por la metodología XP. Con el desarrollo de las diferentes historias de usuarios y con la implantación de las pruebas de aceptación se ha dado solución a la situación problemática planteada y se ha desarrollado un sistema capaz de categorizar de manera manual las URLs.

Conclusiones

Después de realizar un análisis de los diferentes sistemas que permiten categorizar URLs de manera manual y demostrar la necesidad de desarrollar la solución propuesta se puede concluir con el desarrollo de la misma que:

- El estudio del estado del arte relacionado con los sistemas de categorización manual de URLs proporcionó los elementos teóricos necesarios para guiar el proceso de desarrollo.
- La identificación de los requisitos funcionales y no funcionales sirvió como guía para que la aplicación respondiera a los intereses del cliente.
- Los artefactos generados en las fases de implementación y pruebas teniendo en cuenta las características y arquitectura del framework Symfony permitieron implementar la aplicación de forma correcta.
- Con la culminación de las diferentes fases de la metodología de desarrollo seleccionada se desarrolló un sistema de categorización manual de URLs.

Los elementos planteados anteriormente permiten afirmar que se cumplió el objetivo de desarrollar un sistema de categorización manual de URLs que permita a los usuarios contribuir al proceso de actualización de la base de datos de URLs categorizadas de Filpacon.

Recomendaciones

En versiones futuras del sistema se debe incorporar el motor de clasificación inteligente de contenidos MOCIC, vinculándose la categorización manual con la categorización automática. Esto permitirá vincular la inteligencia artificial con el factor humano logrando así crear un sistema híbrido de categorización de URLs que aporte un mayor número de categorizaciones más seguras.

Desarrollo de un plugin para el sistema Filpacon en el que mediante la utilización del protocolo ICAP, se brinde al usuario la posibilidad de sugerir la clasificación de una URL mientras navega por la misma.

Emplear mecanismos de internacionalización de Symfony para aumentar las posibilidades de uso del sistema por parte de los usuarios no hispanohablantes, logrando que un mayor número de personas de diferentes culturas colaboren con la solución.

Incorporar un módulo que genere y muestre estadísticas de la actividad del sistema lo cual podría contribuir a la toma de decisiones del administrador y estimular la participación de los usuarios en general.

Adicionar un módulo de configuración al sistema que permita a los administradores modificar determinados parámetros desde la interfaz.

Confeccionar el manual de usuario de la aplicación que posibilite que personas sin ningún conocimiento informático puedan interactuar con el sistema.

Referencias bibliográficas

[1] RODRÍGUEZ, L.; MALDONADO, A. Directorios temáticos en internet como herramienta de difusión de la ciencia: análisis comparativo. 2007. [citado el: 16 febrero 2010].

[2] MONTERO, Y. M.; TARAJANO, R. A. Sistema adaptativo de filtrado de contenidos. pages 4–7, 2008. [citado el: 20 febrero 2010].

[3] MIN-YEN, K. Web page categorization without the web page. 2007. [En línea] [citado el: 20 febrero 2010]. Disponible en: http://bibliotecas.uchile.cl/navegando/directorios.htm

[4] PÉREZ, Y.; RIPOLL, D. Asignación automatizada de categorías temáticas al contenido textual de documentos html. Technical report, UCI, 2008. Trabajo de Diploma. Ciudad de la Habana, Cuba: Universidad de las Ciencias Informáticas, Facultad 10, 2008. 94 p. [En línea] [citado el: 23 febrero 2010] Disponible en: http://bibliotecas.uci.cu

[5] Universidad de Chile. Sistema de Servicios de Información y Bibliotecas. 2004. [En línea] [citado el: 24 febrero 2010].Disponible en: http://bibliotecas.uchile.cl/navegando/directorios.htm

[6] Por qué elegir php? [En línea] [citado el: 25 febrero 2010]. Disponible en: http://www.programacion.com/php/articulo/porquephp/

[7] Php. [En línea] [citado el: 16 febrero 2010]. Disponible en: http://es.wikipedia.org/wiki/.php#Desventajas

[8] Uml resource page [En línea] [citado el: 16 febrero 2010]. Disponible en: http://www.uml.org/

[9] ERICH, G. Design Patterns: Elements of Reusable Object-Oriented Software. 1994. [En línea] [citado el: 21 Febrero 2010]. Disponible en: http://www.exciton.cs.rice.edu/JavaResources/DesignPatterns/book/hires/chap1fso.htm

[10] DENNIS, P. Taking a look at ten different php frameworks. 2006. [En línea] [citado el: 16 mayo 2008]. Disponible en: http://www.phpit.net/article/ten-different-php-frameworks

[11] Pdt Project. [En línea] [citado el: 18 febrero 2010]. Disponible en: http://www.eclipse.org/pdt/.

[12] PENADÉS, M. C.; CANÓS, J. H.; LETELIER, P. Metodologías Ágiles en el desarrollo de software. 2003. [En línea] [citado el: 10 marzo 2010]. Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

[13] MASINTER, L.; BERNERS-LEE, T.; MCCAHILL, M. Uniform resource locators (URL). request for comments1738, internet engineering task force . 1994. [En línea] [citado el: 11 febrero 2010]. Disponible en: http://www.ietf.org/rfc/rfc1738.txt

[14] DON, W. A gentle introduction. Extreme Programming. 17 de 02 de 2006. [En línea] [citado el: 9 febrero de 2009.] Disponible en: http://www.extremeprogramming.org/.

[15] POTENCIER, F.; ZANINOTTO, F. Symfony la guía definitiva. 2008. [En línea] [citado: 10 abril de 2010] Disponible en: http://www.librosweb.es/symfony_1_2

[16] MARTIN, F. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. [citado el: 20 abril de 2010]

[17] BECK, K. *Extreme Programming Explained. Embrace Change.* s.l. Addison-Wesley Professional, 1999. [citado el: 20 abril de 2010]

[18] BECK, K.; *MARTIN, F. Planning Extreme Programming.* First Edition. 2000. [citado el: 21 abril de 2010]

[19] COHN, M. *User Stories Applied: For Agile Software Development.* Addison- Wesley Professional. 2004. [citado el: 21 abril de 2010]

Bibliografía

BECK, K. *Extreme Programming Explained. Embrace Change.* s.l. Addison-Wesley Professional, 1999. [Consultado el: 20 abril de 2010]

BECK, K.; *MARTIN, F. Planning Extreme Programming.* First Edition. 2000. [Consultado el: 21 abril de 2010]

COHN, M. *User Stories Applied: For Agile Software Development.* Addison- Wesley Professional. 2004. [Consultado el: 21 abril de 2010]

DENNIS, P. Taking a look at ten different php frameworks. 2006. [En línea] [Consultado el: 16 mayo 2008]. Disponible en: http://www.phpit.net/article/ten-different-php-frameworks

DON, W. A gentle introduction. Extreme Programming. 17 de 02 de 2006. [En línea] [Consultado el: 9 febrero de 2009.] Disponible en: http://www.extremeprogramming.org

ERICH, G. Design Patterns: Elements of Reusable Object-Oriented Software. 1994. [En línea] [Consultado el: 21 Febrero 2010]. Disponible en: http://www.exciton.cs.rice.edu/JavaResources/DesignPatterns/book/hires/chap1fso.html

MARTIN, F. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. [Consultado el: 20 abril de 2010]

MASINTER, L.; BERNERS-LEE, T.; MCCAHILL, M. Uniform resource locators (url). request for comments1738, internet engineering task force . 1994. [En línea] [Consultado el: 11 febrero 2010]. Disponible en: http://www.ietf.org/rfc/rfc1738.txt

MIN-YEN, K. Web page categorization without the web page. 2007. [En línea] [Consultado el: 20 febrero 2010]. Disponible en: http://bibliotecas.uchile.cl/navegando/directorios.htm

MONTERO, Y. M.; TARAJANO, R. A. Sistema adaptativo de filtrado de contenidos. pages 4–7, 2008. [Consultado el: 20 febrero 2010]

Pdt Project. [En línea] [Consultado el: 18 febrero 2010]. Disponible en: http://www.eclipse.org/pdt/

PENADÉS, M. C.; CANÓS, J. H.; LETELIER, P. Metodologías Ágiles en el desarrollo de software. 2003. [En línea] [Consultado el: 10 marzo 2010]. Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

PÉREZ, Y.; RIPOLL, D. Asignación automatizada de categorías temáticas al contenido textual de documentos html. Technical report, UCI, 2008. Trabajo de Diploma. Ciudad de la Habana, Cuba: Universidad de las Ciencias Informáticas, Facultad 10, 2008. 94 p. [En línea] [Consultado el: 23 febrero 2010] Disponible en: http://bibliotecas.uci.cu

Php. [En línea] [Consultado el: 16 febrero 2010]. Disponible en: http://es.wikipedia.org/wiki/.php#Desventajas

Por qué elegir php? [En línea] [Consultado el: 25 febrero 2010]. Disponible en: http://www.programacion.com/php/articulo/porquephp/

RODRÍGUEZ, L.; MALDONADO, A. Directorios temáticos en internet como herramienta de difusión de la ciencia: análisis comparativo. 2007. [Consultado el: 16 febrero 2010].

Uml resource page [En línea] [Consultado el: 16 febrero 2010]. Disponible en: http://www.uml.org/

Universidad de Chile. Sistema de Servicios de Información y Bibliotecas. 2004. [En línea] [Consultado el: 24 febrero 2010]. Disponible en: http://bibliotecas.uchile.cl/navegando/directorios.htm

Glosario de términos

UCI Universidad de las Ciencias Informáticas

Filpacon Filtrado de Paquetes por Contenido: Software que permite regular el acceso de los usuarios a determinados contenidos de Internet.

URL Uniform Resource Locator: Un Localizador Uniforme de Recursos representa de un modo compacto la localización y el método de acceso de cualquier recurso de la red. (13)

Isak Internet Secure Access Kit

MOCIC Motor de Clasificación Inteligente de Contenidos

GPL General Public License: Licencia Pública General. Es una licencia creada por la Free Software Foundation (FSF) y orientada principalmente a los términos de distribución, modificación y uso del software.

ODP Open Directory Project: Es el directorio editado por personas más extenso y completo del Web. Su construcción y mantenimiento son realizados por una gran comunidad global de editores voluntarios.

OPTENET Optimal Internet: Sistema de filtrado con más fuerza en Europa

ACL Access Control List

RUP Rational Unified Process: Metodología de desarrollo de software creada por la Corporación Rational.

XP Extreme Programming: Metodología ágil que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

OMG Object Management Group

IW Interfaz Web

Web Red: La traducción literal de esta palabra inglesa es tela de araña, pero en términos informáticos significa mucho más que eso.

UML Unified Modeling Language: Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

IDE Integrate Development Enviroment: Entorno de desarrollo integrado. Herramienta que se usa para facilitar el desarrollo de software.

Anexo A

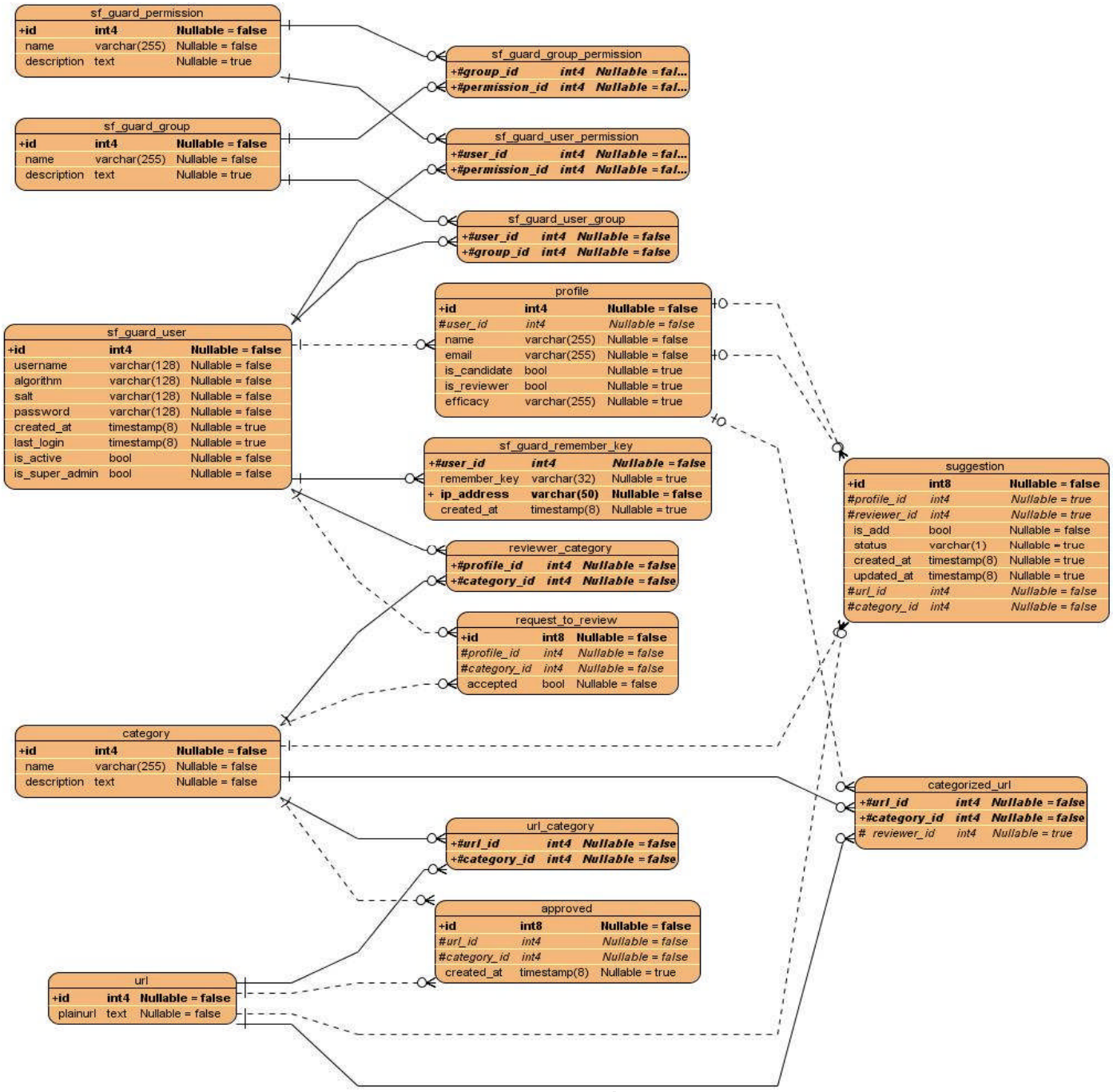

Figura A.1 Diagrama entidad relación del sistema propuesto

Anexo B

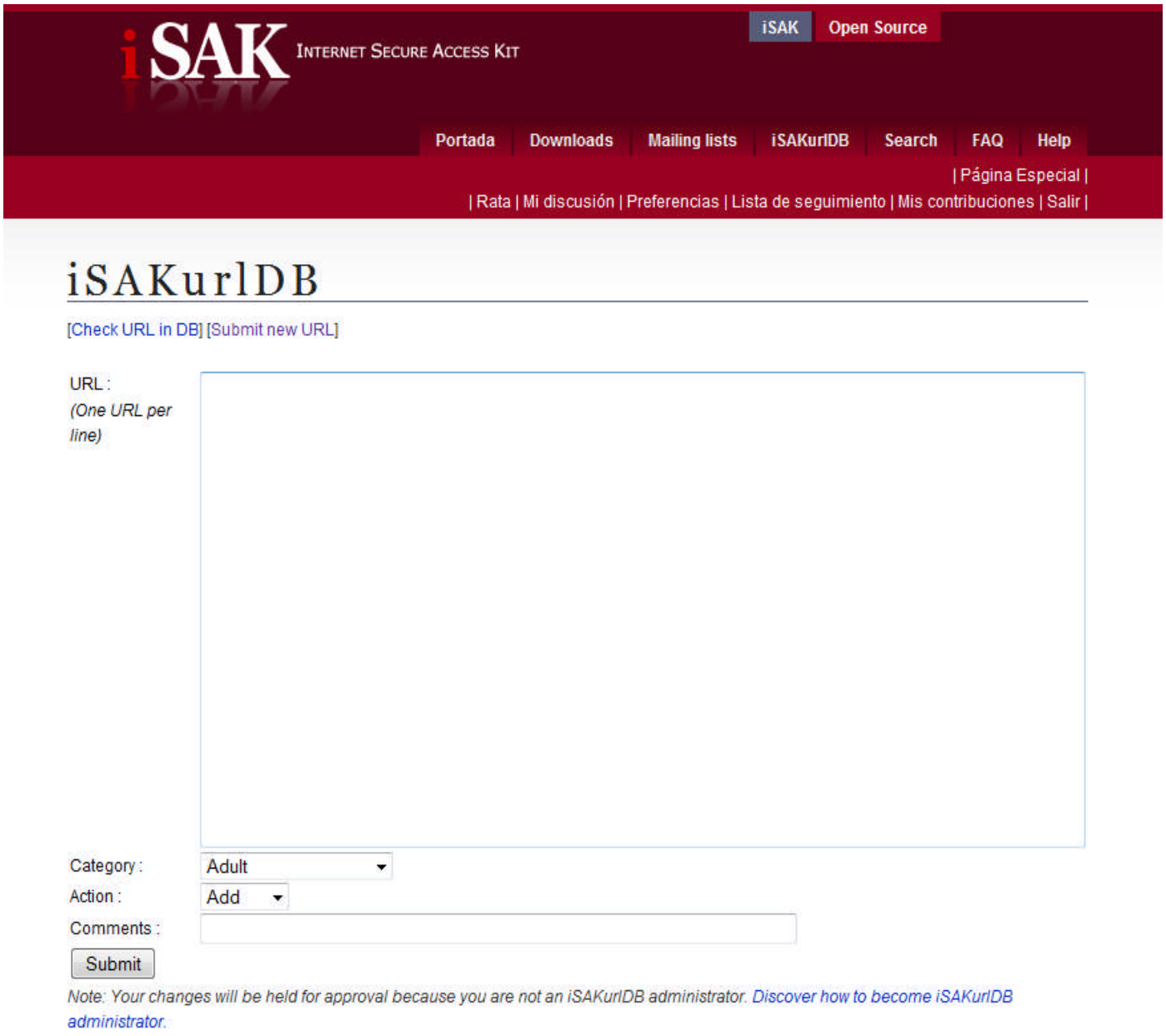

Figura B.1 Interfaz de iSAK para sugerir una o varias URLs

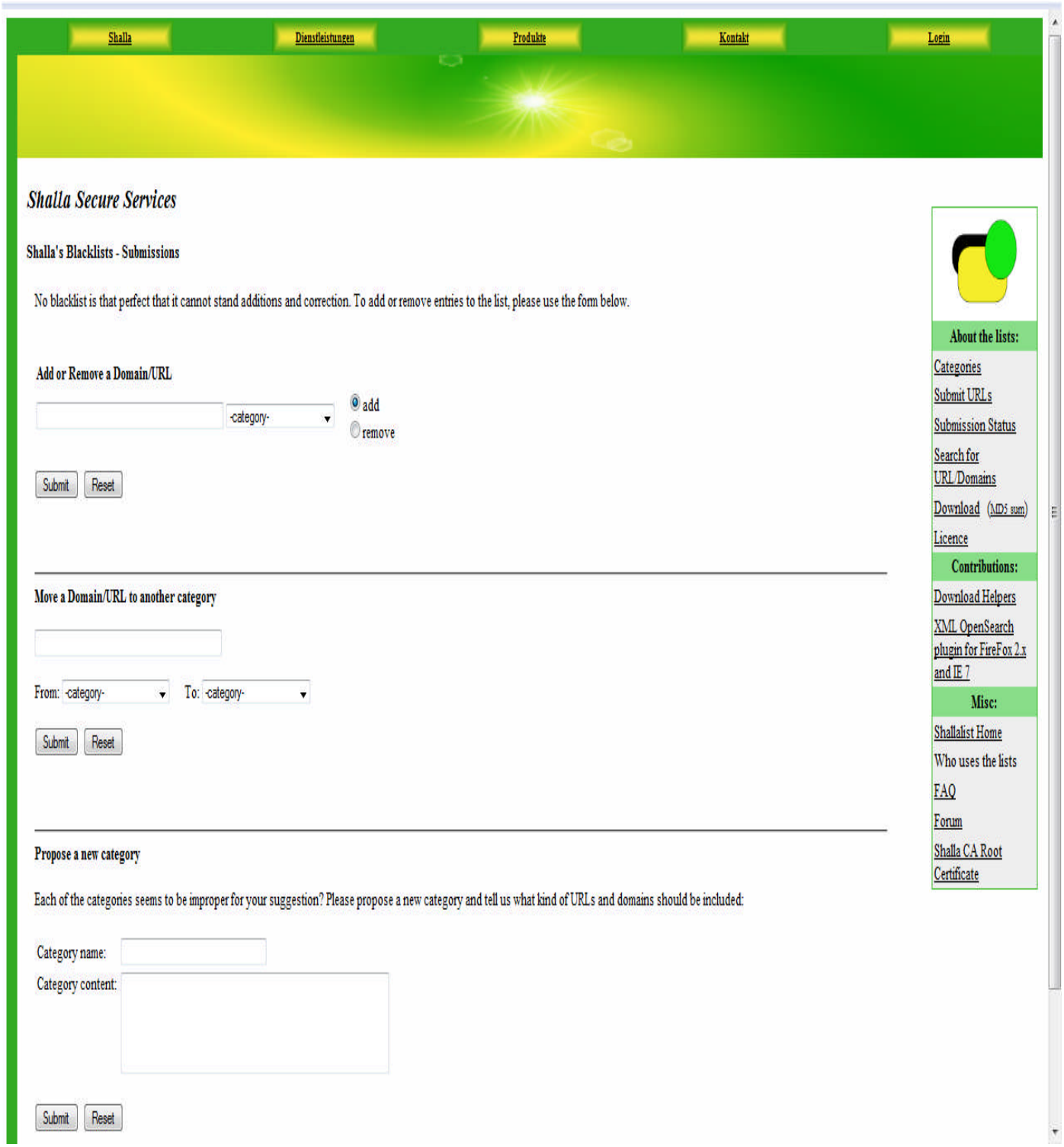

Figura B.2 Interfaz de Shalla Secure Servises para sugerir una URL

Figura B.3 Interfaz de URLBlackList para sugerir una o varias URLs

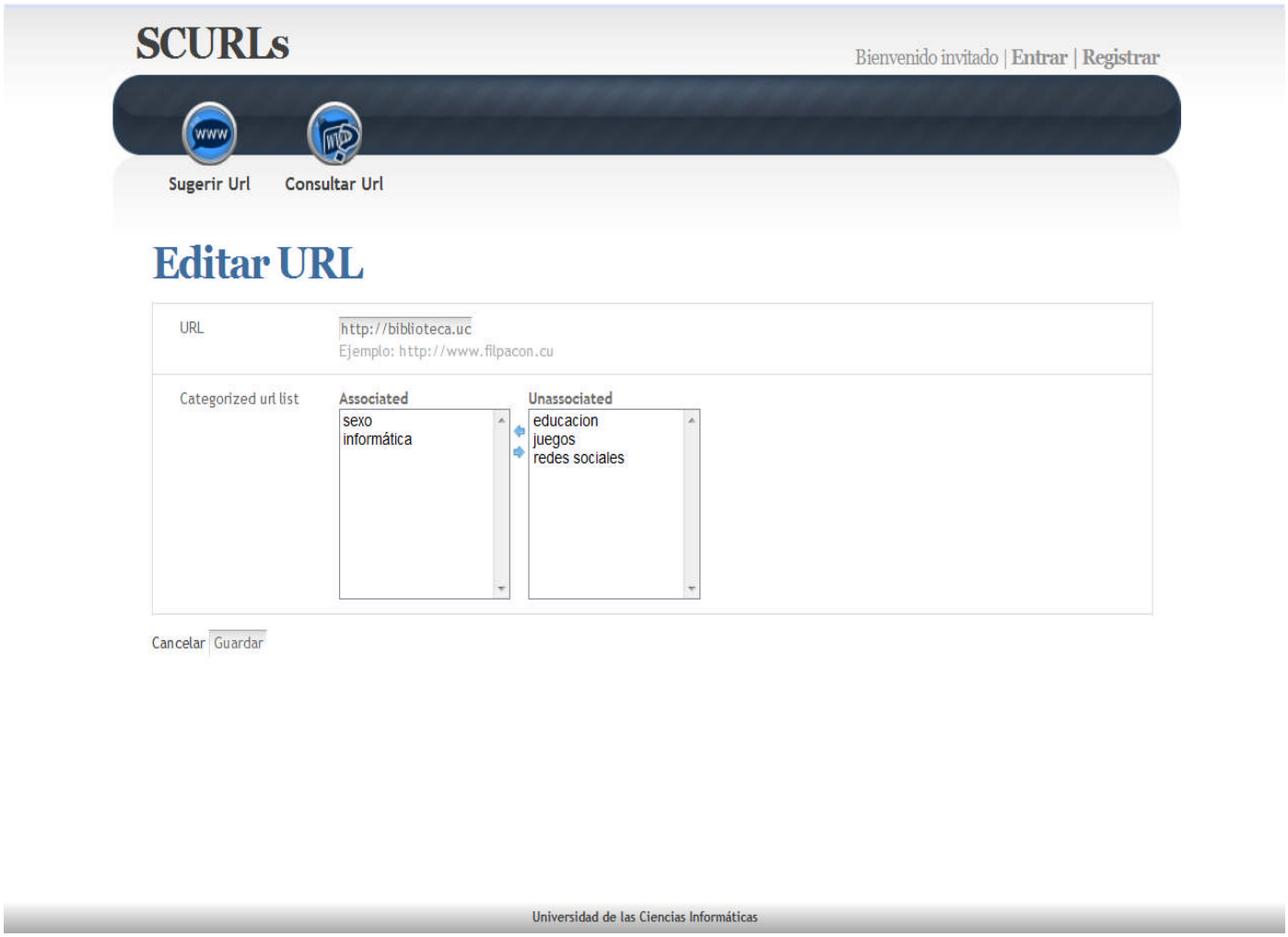

Figura B.4 Interfaz de la solución propuesta para sugerir una URL