Universidad de las Ciencias Informáticas

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

Título: Sistema de generación y publicación de ficheros de actualización de Smart Keeper.

Autores: Enier Gutierrez Piñeiro.
Leandro Castellanos Breña.

Tutor: Ing. Yurisleidy Hernández Moya.

La Habana
"Año 54 del triunfo de la Revolución"

Declaración de autoría

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

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____

Leandro Castellanos Breña

_____

Firma del autor

Enier Gutierrez Piñeiro

_____

Firma del autor

Ing. Yurisleidy Hernández Moya

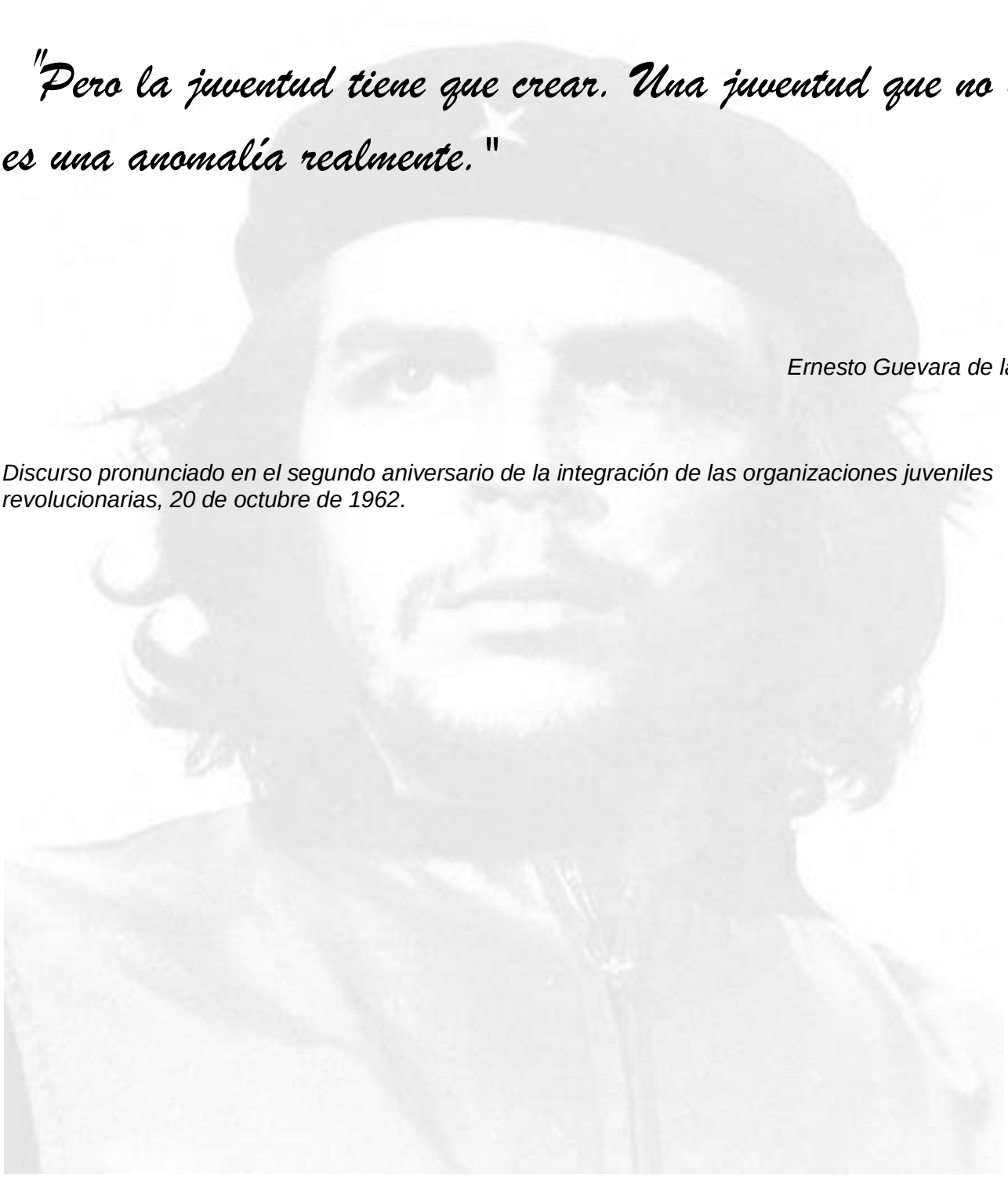
_____

Firma del Tutor

"Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente."

Ernesto Guevara de la Serna.

Discurso pronunciado en el segundo aniversario de la integración de las organizaciones juveniles revolucionarias, 20 de octubre de 1962.



Agradecimientos

A Dios y a todos mis santos por permitirme cumplir este sueño.
A mi madre por su amor y apoyo incondicional en todo momento.
A mi padre que aunque no pueda estar aquí físicamente sé que siempre me acompaña.
Al hombre que ha sido un padre para mí, Demetrio.
Al resto de mi familia por su preocupación y cariño.
A todos los que de una forma u otra me han brindado su ayuda.

Enier Gutierrez Piñeiro

A Dios por permitirme vivir este momento.
A mi novia por haber estado conmigo en todo momento, en las buenas y en las malas, siempre apoyándome.
A mis padres y familia por darme fuerzas cuando la necesité.
A la familia que me ha acogido y me ha hecho parte de ella aquí en La Habana.
A mis amigos por ayudarme cuando se los pedí.

Leandro Castellanos Breña

Agradecemos además a:

Nuestra tutora por toda la ayuda que nos ha brindado, por sus orientaciones, la confianza que depositó en nosotros y por enfrentar junto a nosotros momentos difíciles durante todo este tiempo. A ti Yuri un inmenso agradecimiento.

Dedicatoria

Enier Gutierrez Piñeiro:

Dedico este trabajo a mi madre (Lourdes) por su apoyo incondicional y por estar ahí siempre que la necesito. A mi padre (Renié) aunque no esté entre nosotros, a mi hermano, mis abuelas, al calvo (Demetrio) por apoyarme de diversas maneras a realizar este sueño. A mis amistades de la Universidad; Evelio, Marlon, Arderi y a todos aquellos que me han ayudado, aunque no los mencione los tengo presente. A mi compañero de tesis por permitir la realización de este trabajo de diploma.

… a todos muchas gracias.

Leandro Castellanos Breña:

Dedico este trabajo a mi madre y a mi padre especialmente porque sé que se sienten realizados al igual que yo por este momento, por saber que su fruto ha llegado hasta aquí y porque todo en la vida lo he hecho para hacerlos sentir orgullosos de mí. Dedico además a mi familia en general por ser parte de mi vida, por saber que cuento con ellos para todo y que siempre me estarán apoyando. Dedico grandemente este trabajo a una persona muy especial, mi novia, que apareció en mi vida en los últimos años de mi carrera para hacerme feliz y compartir conmigo todo los momentos difíciles por los que pasé con la realización de este trabajo, fue mi apoyo en todo momento. A todas las personas que de una forma u otra hicieron posible este trabajo…

Muchas gracias.

Resumen

Los filtros de contenidos son herramientas de software para regular el acceso a contenidos disponible en Internet. Smart Keeper es un filtro desarrollado en la Universidad de las Ciencias Informáticas. Para su funcionamiento utiliza una base de datos que contiene direcciones de páginas web clasificadas según su contenido. Dado el dinamismo de Internet, Smart Keeper necesita actualizar la información de su base de datos pues se encuentra desactualizada hace varios años. El filtro no posee una forma automatizada de generar sus actualizaciones ni cuenta con mecanismo que brinde dichas actualizaciones a los usuarios que utilicen el filtro. En aras de solucionar esta deficiencia este trabajo tiene como propósito el desarrollo de un sistema que permita la generación y publicación de las actualizaciones para su base de datos. El desarrollo del sistema se dividió en dos pasos fundamentales (generación y publicación). La generación se basó en la implementación de *scripts* encargados de procesar la información de las listas de URLs Isak y Toulouse obtenidas libremente de Internet para crear los ficheros de actualización. La publicación se logró con la creación de un sitio web donde se encuentran las actualizaciones que están a disposición de los clientes. El sistema desarrollado contribuyó a aumentar la cantidad de URLs almacenada en la base de datos de Smart Keeper y posibilitó que los clientes puedan conocer y tener acceso a las actualizaciones evitando así la obsolescencia de la información utilizada por el filtro de contenidos.

Palabras clave: listas de URLs, generación, publicación, filtro de contenidos, *script*, actualizaciones, mecanismos.

Índice de Contenidos

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC), se han insertado vertiginosamente en diversos sectores de la vida, provocando importantes transformaciones en instituciones tanto públicas como privadas. Su aplicación va desde el correo y la mensajería instantánea, hasta cumplir papeles de gran importancia a nivel laboral. Entre sus aportes significativos se encuentran la digitalización de la información y la globalización de la misma a través del surgimiento de Internet.

Internet ha abierto una puerta al conocimiento y a la información, pero también hay que destacar que este fenómeno social se ha convertido en una adicción para muchas personas. La comunidad de usuarios en esta red se incrementa notablemente [1] y en la actualidad es posible realizar una publicación y ponerla a disposición de internautas de cualquier rincón del planeta, lo que trae consigo que existan materiales adecuados para una correcta formación personal, pero a su vez contenidos nocivos e incluso ilegales según las leyes en algunos países. Para evitar que el acceso a una determinada información se convierta en un elemento que lacera los intereses de un individuo o institución, se ha hecho necesario identificar soluciones que permitan controlar el acceso de los usuarios a dicha información.

Entre las medidas que permiten regular el acceso a materiales disponibles en Internet se encuentra el uso de los filtros de contenidos: herramientas de software diseñadas para controlar el contenido que se visualiza desde un equipo [2]. Su uso se evidencia en hogares, escuelas e instituciones con el objetivo de seleccionar los contenidos que se estimen adecuados según sus características, previniendo al usuario de ciertas informaciones que no les son deseadas.

En la Universidad de las Ciencias Informáticas (UCI), Cuba, se desarrolla un software de filtrado de contenidos de Internet denominado Smart Keeper, este permite regular la navegación por Internet, de los usuarios en la institución o lugar donde se emplee.

Actualmente Smart Keeper cuenta con una base de datos que contiene direcciones de páginas web clasificadas por categorías según su contenido. Diariamente varias páginas web cambian, desaparecen y otras son creadas. En noviembre del 2011 se habían registrado 525 998 433 sitios en Internet mostrando un aumento de 22 millones, equivalente a un 4.3 %, con respecto al mes anterior [3]. Como se puede apreciar, el aumento de la cantidad de sitios de un mes a otro es considerable, provocando que la información almacenada en la base de datos del filtro de contenidos Smart Keeper necesite actualizarse, acción que no se realiza hace varios años y hasta el momento no existe un mecanismo automatizado que permita generar las actualizaciones, ocasionando que dicha información caduque.

Cuando un usuario accede a una URL que no existe en la base de datos, en dependencia de la política definida para dicho usuario, se le deniega o permite el acceso a este material sin importar la información que posee. Esta situación tiene como consecuencia que pueda permitirse el acceso a una información que el usuario no debe poder ver o se deniega un material cuyo contenido es permisible al usuario.

Aun existiendo la posibilidad de una vía para generar actualizaciones para la base de datos, tampoco se cuenta con un mecanismo, que de manera centralizada, le permita a los usuarios del sistema acceder a dichas actualizaciones e incluso tener el conocimiento de que existen. Esta situación ocasiona que las personas o instituciones que utilicen el filtro no puedan actualizarlo, lo que implica que se presenten varias deficiencias a la hora de realizar el filtrado, como brechas en el control de acceso a determinados contenidos y la posibilidad de que el personal incurra de forma intencional o fortuita en delitos prevenibles.

Los elementos expuestos hasta el momento, revelan la necesidad del filtro de contenidos Smart Keeper de generar las actualizaciones para su base de datos y ponerlas a disposición de sus clientes.

A partir de este análisis se identificó el siguiente **problema a resolver:** ¿Cómo generar y publicar actualizaciones de la información almacenada en la base de datos del filtro de contenidos Smart Keeper?

Se determinó como **objeto de estudio** del presente Trabajo de Diploma, los Filtros de contenidos de Internet y **campo de acción** los Mecanismos de Actualización en los Filtros de contenidos de Internet.

Objetivo General: Desarrollar un sistema que permita la generación y publicación de actualizaciones de la información de la base de datos del filtro de contenidos Smart Keeper.

Del objetivo general enunciado se desglosan los siguientes **objetivos específicos**:

- Sistematizar acerca de los mecanismos de actualización en los filtros de contenidos.
- Diseñar un sistema que permita generar y publicar actualizaciones para la base de datos del software Smart Keeper.
- Implementar las funcionalidades diseñadas.
- Probar las diferentes funcionalidades del sistema desarrollado.
- Integrar el sistema desarrollado al software Smart Keeper.

En el cumplimiento de las tareas propuestas se utilizarán los siguientes **Métodos Científicos**:

- **Analítico-sintético:** Permitirá realizar el análisis a diversos mecanismos de actualización utilizados por algunos filtros de contenidos para luego identificar si alguno de estos u otros elementos pueden ser reutilizados o servir de guía para el desarrollo de la solución.
- **Histórico-lógico:** Permitirá estudiar la evolución de los mecanismos de actualización en los filtros de contenidos para comprender cuáles son las tendencias actuales y las soluciones más potentes.

Teniendo como **justificación de la investigación** que el desarrollo de un sistema de generación y publicación de actualizaciones para Smart Keeper permitirá aumentar la cantidad de URLs registradas en la base de datos y contribuirá a un mejor control del acceso de los usuarios a los contenidos de Internet.

El documento está estructurado en cuatro capítulos:

Capítulo 1 "Generación y publicación de actualizaciones en filtros de contenidos": Abarca los aspectos teóricos en los que se basa la investigación. Incluye un estudio del estado del arte referente a los distintos mecanismos de actualización existentes a nivel internacional y nacional, utilizados por los filtros de contenidos de Internet. También se analizan algunas herramientas, tecnologías, lenguajes de

programación y metodologías de desarrollo de software existentes en la actualidad para determinar las apropiadas a emplear en el desarrollo de la solución al problema planteado.

Capítulo 2 "Sistema de generación y publicación de actualizaciones": Se describe el flujo actual de los procesos involucrados, así como un análisis de los mismos y las causas que originan el problema a resolver. Se realiza una descripción general de la propuesta del sistema y cómo debe funcionar. Se presenta el modelo de dominio, la especificación de los principales requisitos de software y la descripción de los casos de uso.

Capítulo 3 "Diseño de la solución propuesta": Se describe el patrón arquitectónico y los patrones de diseño que se evidencian el desarrollo de la solución así como los diagramas de clases, de interacción, el modelo de despliegue del sistema y una breve descripción de cada uno de ellos.

Capítulo 4 "Implementación y validación de la solución propuesta": Se exponen los elementos relacionados con la implementación y validación del sistema como son el diagrama de componentes, las principales vistas del sistema y los resultados obtenidos al aplicar pruebas al sistema.

Capítulo 1. Generación y publicación de actualizaciones en filtros de contenidos.

En este capítulo se abordan los elementos teóricos que constituyen la base de la presente investigación, se exponen los resultados sobre estudios realizados a los mecanismos de actualización empleados por algunos filtros de contenidos de Internet, tanto en el ámbito nacional como internacional. Además, se justifica la selección de la metodología, herramientas y lenguajes de programación que formarán parte de la propuesta de solución, teniendo en cuenta las necesidades existentes.

1.1 Filtros de contenidos

El filtrado de contenido web es una tarea de suma importancia en la actualidad dada a la diversidad de información que viaja a través de Internet. Debido a esto se hace necesario contar con soluciones que contribuyan a un mayor control y monitoreo al acceso de los usuarios a Internet.

Actualmente existen diversas herramientas de control de accesos y monitorización. Una de estas son los filtros de contenidos, programas informáticos destinados a regular el acceso de los usuarios a determinados contenidos de Internet y permiten el cumplimiento de un conjunto de reglas o normas de filtrado establecidas por determinada institución o persona. Estas herramientas están registradas bajo licencias libres o privativas y varían en cuanto a las funcionalidades que ofrecen, entre las que se pueden mencionar el control de los horarios de acceso a Internet, establecer perfiles de usuarios y controlar los servicios que pueden ser consultados por un usuario o en un momento determinado. Poseen diferentes mecanismos para filtrar la información de Internet a la que acceden los usuarios, que se pueden basar en [2, 4]:

Listas de URLs: son un listado de direcciones web clasificadas según su contenido. Pueden ser generadas de forma automática por programas informáticos o manualmente por los seres humanos. Independientemente de los métodos utilizados para la confección de tales listas, es prácticamente

imposible que estas posean las direcciones de todas las páginas web debido al dinamismo de Internet. Se pueden encontrar bajo licencias libres o comerciales. Los filtros hacen uso de estas para bloquear o permitir el acceso a páginas web según la categoría que poseen en la lista y la configuración del filtro.

Palabras clave: consiste en definir una serie de palabras para identificar los contenidos que se consideren inapropiados. Los filtros utilizan este método para bloquear todas las páginas web que contengan dichas palabras.

Clasificación inteligente de contenido: es utilizado para clasificar las páginas web de Internet según su contenido utilizando técnicas de Inteligencia Artificial. Los filtros utilizan este mecanismo para dar respuesta a las solicitudes de los usuarios.

Etiquetas o la clasificación moral de los contenidos: se basa en que las personas que publiquen páginas web en Internet pongan en estas etiquetas, especificando la categoría de la información que poseen. Estas etiquetas se basan en el estándar que ofrece PICS (siglas en inglés de *Platform for Internet Content Selection*). Los filtros analizan dichas etiquetas y según la configuración establecida, permiten o deniegan el acceso a las páginas.

Los métodos mencionados anteriormente no son suficientes para que los filtros de contenidos realicen el filtrado satisfactoriamente. Existen filtros que utilizan una base de datos que contienen direcciones de páginas web de Internet y la clasificación de cada una de estas según su contenido. Dicha información debe ser actualizada con regularidad, debido a que en Internet constantemente se eliminan, modifican o surgen nuevas páginas. Entonces puede existir un mecanismo para realizar el filtrado, pero si no se actualiza la información empleada ocurre que esta se convierte en obsoleta y afecta el correcto funcionamiento del servicio que se ofrece. Un ejemplo de filtro de contenidos que emplea base de datos con URLs categorizadas en su funcionamiento es Smart Keeper.

1.2 Smart Keeper

Es un filtro de contenidos de Internet desarrollado en la UCI, Cuba. Algunas de las características que posee es que permite realizar reportes de las actividades de los usuarios, personalizar listas de contenidos categorizados y realizar modificaciones a la información de la base de datos. Además, ofrece

información sobre el acceso de los usuarios a los contenidos de Internet y posee una Interfaz de Administración Web que posibilita a los administradores del sistema configurarlo con mayor facilidad.

Actualmente la información de su base de datos de URLs está desactualizada pues no cuenta con un mecanismo para actualizarla. Con el objetivo de cambiar esta situación se concibe el Motor de Clasificación Inteligente de Contenido (MOCIC).

1.2.1 Motor de Clasificación Inteligente de Contenido

MOCIC tiene como principal objetivo, clasificar diferentes tipos de contenido según categorías preestablecidas, de forma automatizada e inteligente. Los contenidos a analizar podrán encontrarse en distintos formatos, como pueden ser PDF y HTML.

Actualmente este software está en desarrollo, sólo ha sido completamente aprobado y aceptado el subsistema de procesamiento de imágenes, no obstante, se planea que una vez obtenido el motor completamente funcional se integre a Smart Keeper y le permita a este su funcionamiento en varios modos:

- *Online:* permitirá analizar la información de las páginas web en el momento que estas son solicitadas por el usuario, para determinar el contenido que posee y permitir o denegar el acceso a esta.
- *Semi-online:* posibilitará añadir a la base de datos del filtro las URLs y las clasificaciones, según el contenido y las categorías preestablecidas, de todas las páginas web detectadas por el motor o las que sean consultadas por el usuario.
- *Offline*: será utilizado para trabajar basándose en listas de URLs que serán provistas por el motor.

MOCIC al estar actualmente en desarrollo, no puede brindarle a Smart Keeper un mecanismo de actualización. Ante esta situación es necesario idear otra vía que permita generar y publicar las actualizaciones para este filtro.

1.3 Sistema para la generación y publicación de actualizaciones

Teniendo en cuenta que MOCIC no está en funcionamiento y en busca de un mecanismo de actualización de la información que es almacenada en la base de datos de Smart Keeper, se realiza un estudio de

diversos filtros de contenidos sobre cómo estos se actualizan; con el objetivo de identificar si algún mecanismo empleado o determinados aspectos de ellos pueden ser reutilizados.

1.3.1 Filtros de contenidos

Barracuda Web Filter: está conformado por un equipo de hardware con un software instalado y puede ser conectado en cualquier punto de la red. Sus actualizaciones se generan de forma automática por la organización encargada de la administración del filtro y son suministradas cada una hora mediante del servicio de suscripción que es accesible vía web y se tiene que comprar con cualquier producto de Barracuda [5].

DansGuardian: filtro de contenidos de código abierto licenciado bajo la licencia GPL (siglas en inglés de General Public License) versión 2, se puede descargar desde su sitio oficial libremente pero posee ciertas restricciones y un costo adicional para su uso comercial. Se ejecuta en sistemas operativos GNU/Linux, FreeBSD, OpenBSD, NetBSD, Mac OS X, HP-UX y Solaris. Se filtra el contenido de las páginas basándose en métodos como la búsqueda de una frase, filtrado por etiquetas y filtrado por URL. Para el filtrado por URL utiliza listas de URL, las cuales no son únicamente para este filtro, sino que son listas libres y desde el sitio oficial del filtro se hace referencia a estas, entre las que se encuentran URLBlacklist, Shalla y MESD. Una vez obtenida la lista a utilizar, es ubicada en un directorio que se crea por defecto con la instalación de DansGuardian y mediante la configuración del filtro se tiene acceso a ella para procesar su información [6].

OPTENET: filtro de contenidos privativo que para utilizarlo debe comprarse y obtener un código de licencia que será requerido en el momento de instalación del software, este proceso se realiza mediante el envío de datos personales. Funciona en las siguientes plataformas: Windows, GNU/Linux y Solaris.

Posee un motor de clasificación inteligente de contenido que le permite clasificar una página en el instante de ser consultada y también se basa en listas de URL. Se puede encontrar en dos versiones, la de computadoras personales y la de servidor. En ambas la actualización de las listas se realiza de forma automática, conectándose de forma continua a diferentes servidores de actualizaciones para ir completando su base de datos de URLs. La diferencia que existe entre estas versiones radica en que la versión para computadora personal se actualiza sin ofrecer ningún tipo de administración y en la versión

de servidor se pueden configurar elementos como el proxy, puerto, frecuencia de actualización (diaria, semanal, mensual), la hora de inicio y fin de la actualización. Las listas del filtro son confidenciales y propiedad de Optenet. Brinda la posibilidad de crear listas de páginas prohibidas y permitidas, incluso para cada perfil de usuario [7, 8].

SquidGuard: se trata de una herramienta nativa sólo UNIX y se puede instalar en sistemas operativos como GNU/Linux y FreeBSD. El uso de este filtro es gratis, se publica bajo licencia GPL y su desarrollo se basa en las contribuciones de todas partes del mundo. Utiliza listas de URL para realizar el filtrado y desde su sitio oficial se ofrece un servicio de descarga gratis del filtro y también se hace referencia a las listas que les son compatibles como: Shalla, URLBlacklist, Toulouse y MESD [9].

En Cuba solo se tiene referencia del filtro de contenidos Filpacon [10], desarrollado en la UCI, el cual es una versión anterior de Smart Keeper y por tanto tampoco cuenta con un mecanismo de actualización.

Concluido el estudio de determinados filtros de contenidos de Internet, no se obtuvo información específica de cómo generan sus actualizaciones, aunque sí hacen referencia al uso de clasificación inteligente de contenidos y listas de URLs clasificadas. En algunos filtros se ofrecen determinados aspectos del procedimiento a seguir una vez obtenidas tales actualizaciones, así como la vía empleada para ponerla a disposición de sus clientes.

El estudio realizado permitió a los autores del presente trabajo, determinar que los filtros de contenidos varían en cuanto a los métodos de publicación de sus actualizaciones, la diferencia radica en la decisión de los fabricantes. En el caso de los filtros de contenidos de código abierto basados en listas de URLs, hacen públicas las listas a utilizar permitiendo su obtención desde un sitio web. En algunos casos se permite realizar la descarga de manera gratis, en otros hay que obtener una suscripción para acceder al servicio de descarga. Los filtros de contenidos con licencias privativas poseen variados métodos para ofrecer las actualizaciones como:

- Las actualizaciones son parte del producto y deben ser compradas junto con el software.
- Debe ser pagada una suscripción para recibir las posteriores actualizaciones.
- La actualización es realizada de forma automática desde Internet y sin intervención del usuario.

Los filtros de contenidos estudiados anteriormente utilizan mecanismos de actualización diferentes y ninguno se puede reutilizar ni adaptar totalmente a las necesidades del filtro de contenidos Smart Keeper. Es por esta razón que se decide desarrollar un sistema para la generación y publicación de las actualizaciones de dicho filtro. No obstante, algunos aspectos observados, pueden ser utilizados como modelo a seguir en Smart Keeper para el desarrollo de la solución. Específicamente se propone el uso de listas de URLs para actualizarse, obtenidas de diferentes servidores de Internet y el empleo de un sitio web para publicar y descargar tales actualizaciones. Esta decisión está sustentada también por la idea de mantener una compatibilidad con MOCIC que, como fue mencionado anteriormente, será el proveedor de listas de URLs para el filtro de contenidos Smart Keeper.

1.4 Listas de URLs

Dada la propuesta de utilizar listas de URLs para generar las actualizaciones de la base de datos de Smart Keeper, es necesario realizar un estudio de listas de URLs disponibles en Internet para determinar cuál o cuáles son convenientes emplear.

URLBlacklist: contiene algunas URLs que están mal clasificadas debido fundamentalmente al uso de *scripts* automáticos que conforman la lista. Es utilizada por filtros como DansGuardian y SquidGuard. Es creada con fines comerciales, es decir, para adquirirla se debe comprar una suscripción. Comúnmente es generada automáticamente a diario y puede ser obtenida desde su página oficial[1] [11].

Shalla: utilizada por filtros de contenidos como SquidGuard y DansGuardian. Su licencia establece que es gratis para uso personal, no debe ser entregada a terceros y para el uso comercial hay que registrarse mediante la firma de un contrato de uso. Si se planea venderla ya sea pura o mezclada con las entradas de otras listas, o se incluye en un paquete de software comercial se debe obtener un contrato por escrito con los servicios de Shalla, los costos dependen del producto. Suelen actualizarse diariamente aunque hay ocasiones en que pueden tardar varios días. La lista puede ser descargada desde su página oficial[2] [12].

[1] Sitio oficial de la lista URLBlacklist: http://www.urlblacklist.com/
[2] Sitio oficial de la lista Shalla: http://www.shallalist.de/Downloads/shallalist.tar.gz

Toulouse: conformada por la Universidad de Toulouse. En caso de desacuerdo con la clasificación de alguna URL que ofrece la lista, se puede reportar mediante correo electrónico al representante de la lista o mediante una interfaz web[3] destinada para esta acción. También esta interfaz acepta adicionar alguna URL que no esté registrada en la lista. Es de libre distribución, se actualiza aproximadamente dos veces por semana y está disponible bajo licencia *Creative Commons*. Se estructura fundamentalmente de dos fuentes, mediante contribuyentes que envían las modificaciones y un sistema que navega por Internet y clasifica las páginas. Se puede descargar desde varias ubicaciones[4] [13].

MESD (siglas en inglés de *Multnomah Education Service District*): es una organización que utiliza el filtro de contenidos SquidGuard y también ofrece una lista para este. La lista es de libre distribución y se compone a partir de otras como Toulouse y BN-PAF (siglas en de alemán *Bürgernetz Pfaffenhofen*). El resultado final es publicado en un sitio web[5] [14].

ISAK: utiliza la licencia GPL y es actualizada diariamente. Se puede descargar desde un sitio web[6] y se conforma a partir de varias listas como *Open Directory Project* (ODP), listas de URLs de SquidGuard (Shalla, MESD, Toulouse, URLBlacklist.com) e Ingrid [15].

Dada las características de las listas antes mencionadas y la necesidad del sistema Smart Keeper de generar su actualización mediante listas de URLs categorizadas, se propone emplear las listas Toulouse e ISAK. Esta selección se basó en que la lista URLBlacklist es de uso comercial, por lo que hay que pagar para obtenerla. La lista Shalla, según lo que establece su licencia, tiene ciertas restricciones para el uso comercial. La lista MESD combina dos listas, una es Toulouse y la otra a la que se hace referencia, aún no se ha comprobado su existencia ni se ha encontrado información referente a esta por parte del equipo de desarrollo. Otro aspecto determinante es que Toulouse e ISAK están bajo licencia libre y en el caso de ISAK se compone a partir de una variedad de listas, algunas de las estudiadas y otras que no pudieron ser localizadas.

[3] URL de la interfaz para reportar modificaciones de la lista Toulouse: http://cri.univ-tlse1.fr/cgi-bin/squidguard_modify.cgi
[4] URLs para descargar la lista Toulouse: http://cri.univ-tlse1.fr/blacklists/download y ftp://ftp.univ-tlse1.fr/blacklist/
[5] URL de descarga de la lista MESD: http://squidguard.mesd.k12.or.us/blacklists.tgz
[6] URL de descarga de la lista Isak: http://download.isak.gplindustries.com/isakurldbtext.tar.gz

Una vez identificado el mecanismo de actualización a utilizar en el filtro de contenidos Smart Keeper se hace necesario realizar un estudio de elementos de seguridad tenidos en cuenta en la actualidad para el desarrollo de aplicaciones web. Este estudio determinará los aspectos de seguridad a seguir para publicar dichas actualizaciones en un sitio web y garantizar la integridad y seguridad de la información que se publique en él.

1.5 Elementos de seguridad

El estudio realizado a algunos filtros de contenidos de Internet, permitió a los autores del presente trabajo identificar los elementos de seguridad empleados en los sitios web que utilizan los filtros para publicar las actualizaciones. En algunos casos para acceder a determinados sitios web y obtener información disponible en estos, se cumple con determinados requisitos de seguridad como es la autenticación en el sitio web mediante el uso de contraseña, solicitar determinada información del cliente, realizar un pago por el servicio que se ofrece o solicitar una suscripción. Otros casos con menos restricciones permiten que cualquier cliente sin verificación alguna tenga acceso a la información.

Hoy en día son utilizados varios métodos para garantizar la seguridad en el desarrollo de aplicaciones web, como son la identificación y autenticación de usuarios, control de acceso y flujos de la información, algoritmos para el cifrado de datos y el intercambio de llaves. Una aplicación web para que sea segura, debe asegurar que ninguna persona ajena al sistema, pueda modificar o falsificar datos de dicha aplicación web [16].

Existen mecanismos de seguridad un poco más complejos como es la criptografía. Consiste en codificar los datos haciendo uso de algoritmos de cifrado. Es utilizado para almacenar información sensible o transmitirla a través de canales inseguros (como Internet) de tal forma que no pueda ser leída por nadie que no sea la persona a la que va dirigida o que tiene los permisos para hacerlo. Una vez en su destino, los datos pueden ser descifrados e interpretados. Existe la criptografía simétrica y la asimétrica, la primera utiliza una misma llave tanto para cifrar como descifrar los datos y la segunda se basa en dos llaves, una pública y una privada, la pública sirve para cifrar la información y la privada para descifrarla [16].

Las técnicas empleadas para garantizar la seguridad en los sitios web varían en cuanto a la complejidad de su aplicación. Entre los elementos que se sugieren emplear se encuentran que exista algún

mecanismo de seguridad para acceder al sitio, como lo puede ser el uso de contraseñas. También que la modificación de la información y administración del sitio web sea realizada por personas seleccionadas, garantizando la integridad de la información. Además debe manejarse la información de forma tal que se pueda definir quiénes tendrán acceso a esta.

Para la confección del sitio web donde se publicarán las actualizaciones de la información de la base de datos de Smart Keeper, se tendrán en cuenta algunos de los elementos anteriormente expuestos. Se establecerá la autenticación por contraseña para acceder al sitio, la descarga de las actualizaciones será controlada de forma tal que cualquier persona no pueda realizar esta acción, se establecerán niveles de acceso a la información del sitio y además la información de las actualizaciones estará cifrada.

Para el desarrollo de la solución es de gran importancia contar con los medios necesarios que hacen posible su completamiento. A continuación se realizan comparaciones entre herramientas, lenguajes, metodologías y se escoge la base tecnológica a emplear por los autores para el cumplimiento del objetivo general que estable el presente trabajo.

1.6 Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar un software [17]. En un proceso de desarrollo la elección de una correcta metodología puede ayudar al programador a entender la problemática y desarrollar una solución. La elección de la metodología debe hacerse de acuerdo al tipo de proyecto, a los recursos con los que se cuentan (tiempo, dinero y otros) y a la facilidad de interacción con el usuario real.

Para la selección de la metodología a utilizar en el desarrollo de esta aplicación, es necesario tener en cuenta que la metodología RUP (siglas en inglés de *Rational Unified Process*) fue la utilizada durante el desarrollo del software Smart Keeper y resulta importante mantener una compatibilidad entre los procesos y documentación de ambos. Por ello se realizará un estudio sobre dicha metodología para saber si debe ser la que rija el desarrollo del nuevo sistema.

1.6.1 RUP

El proceso de software propuesto por RUP tiene tres características esenciales: está dirigido por los casos de uso, centrado en la arquitectura y es iterativo e incremental. RUP ha agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales: modelamiento del negocio, requerimientos, análisis y diseño, implementación, prueba, instalación, administración del proyecto, administración de configuración y cambios y ambiente. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Los principales elementos que definen esta metodología son los trabajadores, actividades, artefactos y el flujo de actividades. El proceso también define una serie de roles que se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado que se espera de ellos [17]. Utiliza UML (siglas en inglés de *Unified Modeling Language*) para el modelado de los sistemas.

Se propone RUP dada las características antes mencionadas. Permite generar los artefactos para el desarrollo de la nueva aplicación y la documentación necesaria para garantizar que los futuros desarrolladores que vayan a trabajar con la aplicación, lo puedan hacer apoyándose en la misma. También es una metodología adaptable al contexto y necesidades de cada organización, se puede adecuar para no tener que generar necesariamente todos los artefactos y la documentación de la misma.

1.7 Herramientas para el desarrollo de software

Decidir las herramientas adecuadas a utilizar, contribuye en la calidad del producto así como una mejor facilidad a la hora de desarrollar la solución.

1.7.1 Lenguaje de representación visual

UML es un lenguaje de modelado de sistemas de software muy utilizado; permite visualizar, especificar, construir y documentar un sistema. Este lenguaje ofrece un estándar para describir un “plano” del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema. También aborda aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables [18].

Se puede aplicar en el desarrollo de software generando variedad de modelos que son utilizadas por diferentes metodologías de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso utilizar.

Dada la propuesta de utilizar UML en el modelado del sistema, se hace necesario realizar un estudio de herramientas que utilicen este lenguaje y permitan generar todos los artefactos necesarios para el desarrollo de la solución.

1.7.2 Herramientas CASE

Las herramientas CASE (siglas en inglés de *Computer Aided Software Engineering*) en el uso del desarrollo de proyectos informáticos permiten modelar los procesos de negocios de las empresas para el desarrollo de los sistemas. Hacen uso de UML para modelar la información de negocios cuando ésta se transfiere entre distintas entidades organizativas. Uno de los objetivos de estas herramientas consiste en representar objetos de datos de negocios, sus relaciones y visualizar la forma en que fluyen entre distintas zonas de negocio [19]. Algunas de las herramientas CASE que pueden emplearse son:

Visual Paradigm: esta herramienta permite una construcción rápida y eficaz de las aplicaciones, con una elevada calidad. Es fácil de utilizar, de código abierto y además facilita la interoperabilidad con otras herramientas CASE. Permite crear todos los diagramas de clase, así como generar documentación y código a partir de diagramas [20].

Rational Rose: admite de forma completa la especificación del UML, permite crear los diagramas que se generan durante el proceso de ingeniería de un sistema informático. Brinda facilidades para la generación de la documentación del software que se está desarrollando y posee un gran número de estereotipos predefinidos que agilizan el proceso de modelación [21].

Luego del estudio realizado se propone para el modelado del sistema la herramienta CASE Visual Paradigm ya que esta brinda una respuesta rápida, permite aplicar la ingeniería inversa de múltiples formas y es de código abierto.

1.7.3 Sistema Gestor de Base de Datos.

Para el desarrollo del sistema es necesario almacenar cierta información y tener un control sobre esta. Por ello es necesario realizar un estudio de herramientas que permitan la gestión y control de los datos que se almacenarán.

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad [22]. Por tanto, debe permitir realizar varias operaciones en la base de datos como: definir tipos, estructuras y restricciones de datos, guardar los datos en algún medio controlado por el mismo SGBD, realizar consultas, actualizar y generar informes. Ejemplos de SGBD son: Oracle, SQL Server de Microsoft, PostgreSQL y MySQL [22].

Algunas de las características de un SGBD puede ser el control de la redundancia, pues la redundancia posee varios efectos negativos (duplicar el trabajo al actualizar, desperdicia espacio en disco, puede provocar inconsistencia de datos) aunque a veces puede ser deseable por cuestiones de rendimiento. La restricción de los accesos no autorizados es otra característica importante pues cada usuario ha de tener permisos de acceso y autorización. También debe ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad [22].

Smart Keeper emplea en su funcionamiento una base de datos en el SGBD Postgresql. Para el sistema a desarrollar se propone utilizar este mismo SGBD para mantener una compatibilidad entre las bases de datos de ambos sistemas. Además Smart Keeper importa tres tablas del nuevo sistema hacia su base de datos.

PostgreSQL está distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Las grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema no influye en su correcto funcionamiento. Se ejecuta en sistemas operativos como, GNU/Linux, varias versiones basadas en UNIX y Windows [23].

1.7.4 Sistema para la gestión de contenidos

Un Sistema de Gestión de Contenido: (CMS siglas en inglés de *Content Management System*) permite la creación y administración de contenidos de páginas y portales web. Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido (textos, imágenes, etc.) que se visualizará en el sitio web. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible conservar el contenido y cambiar en cualquier momento el diseño del sitio sin afectar la información del sitio [24].

Un Framework: es una estructura definida, formada por bibliotecas orientadas a la reutilización de componentes para el desarrollo de aplicaciones. Sus componentes pueden ser utilizados en una aplicación y si no los tiene existen parámetros definidos para crearlos y que se integren con el resto de las funcionalidades del framework, que lo hace extensible [25].

Se propone la utilización de un CMS y no un *framework* pues este primero trae incorporado varias funcionalidades útiles en el sistema a desarrollar como manejo de permisos y sesiones de usuario. Con el empleo de un *framework* todas estas funcionalidades, necesitadas en el sistema, deberían ser construidas completamente. No obstante, un CMS trae incorporado un *framework* para permitir que el sitio web sea modular y escalable. Esto también permitirá que desarrolladores terceros puedan implementar sus necesidades.

El CMS Drupal: es un sistema de código abierto y administrador de contenido para la construcción de sitios web dinámicos. Ofrecen una amplia gama de rasgos y servicios como pueden ser: la administración de usuarios, la inclusión de noticias y creación de blogs. Drupal puede apoyar gran diversidad de sitios web que pueden ir desde sitios web personales hasta grandes manejados por comunidades. Entre sus principales características se encuentran que su código fuente está libremente disponible bajo los términos de la licencia GPL (siglas en inglés de General Public License), contiene un robusto entorno de personalización donde el contenido y la presentación pueden ser individualizados de acuerdo las preferencias definidas por el usuario. Su sistema de permiso está basado en roles lo cual permite que los administradores no tengan que establecer permisos para cada usuario, en lugar de eso pueden asignar permisos a un ‘rol’ y agrupar los usuarios por roles. También está diseñado para ser multi-plataforma por

lo que puede funcionar con Apache o Microsoft IIS como servidor web y en sistemas operativos como GNU/Linux, BSD, Solaris, Windows y Mac OS X [26].

El CMS Joomla: es gratuito y bastante utilizado en el mundo de la Web, es un administrador de contenido para la construcción de sitios web dinámicos que ofrecen una amplia gama de posibilidades y servicios. El cual permite gestionar con mucha facilidad toda la web, crear un nuevo apartado, modificar los actuales, añadir nuevas imágenes, crear nuevas opciones de menú. Dentro de sus principales características se encuentra está preparado para organizar los contenidos en secciones y categorías, creando así una estructura ordenada y sencilla para los administradores. Posee un sistema de permiso jerárquico donde los usuarios poseen diferentes niveles de facultades/permisos dentro de la gestión y administración del sitio. También utiliza un sistema de *templates* el cual permite cambiar todo el aspecto del sitio [27].

Después de analizar las características de ambos CMS, se propone para el desarrollo de la solución Drupal, pues Joomla no permite trabajar con Postgresql y éste es un requisito importante para lograr compatibilidad entre la base de datos que utiliza Smart Keeper y la del nuevo sistema. También permite una mayor personalización de sus módulos, en cambio Joomla es más rígido en cuanto a su estructura.

1.7.5 Lenguaje para la publicación de las actualizaciones

PHP: (siglas en inglés de *Hypertext Preprocessor*), es un lenguaje interpretado de alto nivel y ejecutado en el servidor. Es un lenguaje de *script* que puede ser incrustado dentro del HTML y la mayor parte de su sintaxis ha sido tomada de lenguajes como C, Java y Perl con algunas características específicas de sí mismo. Permite a los desarrolladores la generación dinámica de páginas, la técnica de programación orientada a objetos y no requiere definición de tipos de variables. Es un producto de código abierto, por lo que cuenta con la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente [28].

Una de sus características es que permite trabajar con diferentes bases de datos, entre las cuales se pueden mencionar MySQL, Oracle y PostgreSQL. También ofrece la integración con varias bibliotecas externas, que permiten expandir su potencial. Es un lenguaje usado en el desarrollo web, es libre y multiplataforma [28].

1.7.6 Lenguaje para la generación de las actualizaciones

Para la implementación de *script* existen varios lenguajes de programación como son Java, Python, Perl y Bash. De ellos se propone Perl[7] para la creación de los ficheros de actualización de la base de datos de Smart Keeper, debido a que es un lenguaje que fue desarrollado para procesar textos. Además por la experiencia que posee el equipo de desarrollo, las características y ventajas del lenguaje.

Perl es un lenguaje interpretado, es utilizado en diferentes paradigmas de programación tanto estructural como orientada a objetos. Posee una gran cantidad de módulos que constituyen un conjunto de funciones u objetos que son accedidos por los programas o *scripts* para realizar determinadas funcionalidades y está disponible para distintos sistemas operativos [29].

1.7.7 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado IDE (siglas en inglés de *Integrated Development Environment*) es un programa que permite editar, compilar, ejecutar y depurar programas de una forma cómoda y ágil. Está compuesto por un conjunto de herramientas y se puede dedicar a un solo lenguaje de programación o múltiples [30].

NetBeans: está escrito en Java, pero puede servir para muchos otros lenguajes de programación. Permite crear aplicaciones Web con PHP 5 y posee un potente *debugger* integrado. Existe además un número importante de módulos para extender este IDE; es un producto libre pues está bajo una variante de la licencia pública de Mozilla, gratuito, sin restricciones de uso y multiplataforma. Posee resaltado de sintaxis, completamiento de código, ayuda de código y lista de parámetros de funciones y métodos de clase [31].

Eclipse: fue creado originalmente por IBM, es libre pues está bajo la Licencia Pública Eclipse. Presenta un marco de trabajo modular ampliable mediante complementos (*plugins*). Existen complementos que permiten usar Eclipse para programar en PHP, Perl, Python, C/C++ y otros. Se puede encontrar una gran documentación acerca de Eclipse tanto en su sitio oficial como en la red [32].

[7]Practical Extraction and Report Language

Se propone Eclipse, por la experiencia en el equipo de desarrollo y porque cuenta con diversos módulos que permiten programar y depurar códigos en Perl, lo cual proporciona facilidad a la hora de la implementación de la solución.

1.8 Conclusiones

- A partir del estudio realizado a algunos filtros de contenidos sobre cómo generan y publican sus actualizaciones, no se identificó ninguna solución que pudiera ser aprovechada en Smart Keeper. A pesar de esto se determinaron aspectos que pueden ser utilizados en la concepción de la solución.

- Con la decisión de utilizar listas de URLs clasificadas para emplear en el desarrollo de la solución y el estudio realizado a algunas de estas se pudieron seleccionar las más factibles a utilizar, específicamente las listas Toulouse e Isak.

- El estudio realizado a los mecanismos de actualización utilizados por algunos filtros de contenidos, propició la decisión de crear un sitio web para publicar las actualizaciones generadas, teniendo en cuenta determinados elementos de seguridad.

- El estudio y comparación de herramientas, lenguajes de programación y metodologías permitió seleccionar la base tecnológica a emplear durante el desarrollo de la solución.

Capítulo 2. Sistema de generación y publicación de actualizaciones

En este capítulo se describe el problema que enfrenta Smart Keeper y que da origen a la realización del presente trabajo. Luego se establece una propuesta de solución para la problemática del filtro y se identifican los requisitos funcionales y no funcionales, los cuales influyen directamente en que la solución propuesta satisfaga las necesidades del cliente. Se muestra el modelo del dominio, para comprender el entorno relacionado con la solución. Posteriormente se muestra el Diagrama de Casos de Uso del Sistema y la descripción de cada uno de los casos de uso para una mejor comprensión del funcionamiento del sistema.

2.1 Descripción del problema

Smart Keeper es un filtro de contenidos cuyo objetivo es regular el acceso de los usuarios a las páginas de Internet, para evitar que estos interactúen con contenidos que se consideren inadecuados o no deseados. Una de sus metas es poseer un mecanismo de actualización que posibilite mantener actualizada la información de su base de datos y así ofrecer un mejor servicio.

Cuando un usuario accede a una URL determinada, el sistema se encarga de consultar la base de datos, si existe la URL se obtiene la categoría y se deniega o permite el acceso a la misma, basándose en la configuración establecida en el filtro. En caso de que no exista en la base de datos la URL solicitada, la decisión es permitir o denegar sin tener en cuenta el contenido consultado.

Dado el dinamismo de Internet, constantemente surgen o desaparecen nuevas URLs lo cual hace que la información almacenada en la base de datos deba ser actualizada. Si no se realiza esta acción pude que la decisión tomada de permitir o denegar el acceso a una dirección web no sea la correcta según las políticas establecidas. Los clientes que posean Smart Keeper también requieren actualizar la información de la base de datos, sin embargo actualmente no se cuenta con un sistema que permita a los usuarios de

manera centralizada obtener las actualizaciones. Es decir, suponiendo que exista una actualización, aun así no hay una vía para que los clientes tengan acceso a esta.

2.2 Solución propuesta

Para solucionar la problemática planteada se propone desarrollar un sistema que permita generar y publicar la información para actualizar la base de datos de Smart Keeper. Dicho sistema contará con dos pasos fundamentales: el primero será generar los ficheros con el contenido a actualizar y el segundo publicar dichos ficheros para la consulta y descarga por parte de los clientes, mediante un sitio web.

Para la generación de las actualizaciones se utilizarán listas de URLs (ISAK y Toulouse) que pueden ser obtenidas gratuitamente desde Internet y no poseen restricciones comerciales. La información de estas listas será procesada por programas de software elaborados en el lenguaje de programación Perl. Estos programas de software tomarán la información a utilizar (URLs y dominios) del listado de categorías que se encuentra dentro de cada una de las listas generando un fichero. Posteriormente se genera otro fichero para cada lista utilizada, donde se encuentra la diferencia entre la versión anterior de las listas y las nuevas listas. Con la información de esos ficheros de diferencia se realiza un proceso que consiste en saber que URLs se van a eliminar o adicionar en la base de datos. A través de este proceso se obtienen los ficheros con la información necesaria para realizar la actualización. Los ficheros serán generados con una estructura de carpetas definida consistiendo esta en: la creación de una carpeta con la fecha de la actualización la cual contiene un fichero por cada categoría que se actualizó y estará ubicada dentro de una carpeta nombrada Actualización, la cual también contiene la carpeta Histórico donde se lleva un registro de cada categoría donde se encuentran todas las modificaciones por las que han transcurrido. Un fichero contiene el listado con los datos necesarios correspondientes a su categoría para poder actualizar la información de la base de datos. Estos ficheros estarán estructurados de la siguiente forma: su primera línea contiene la fecha en que se generó y el resto están compuestas por dos parámetros, la acción a realizar (eliminar o adicionar) y la URL cifrada.

Una vez obtenidos los ficheros se procederá a publicarlos en un sitio web, que posibilitará la descarga a los clientes. El sitio contará con determinados elementos que garanticen la seguridad de la información

que se publica, como lo es el uso de la autenticación por contraseña para acceder al sitio web y descargar una actualización. Se usará el protocolo “https” (siglas en inglés de Hypertext Transfer Protocol Secure) para asegurar el envío seguro de los datos hacia el servidor. El registro de usuarios en el sitio será supervisado, garantizando que sólo tengan acceso los usuarios que sean aprobados por el administrador. Existirán diferentes roles de usuario y por cada uno diferentes permisos, definiendo los niveles de acceso a la información. Los roles con los que contará el sistema son:

- Usuario, que son todas las personas que acceden al sitio web.
- Administrador, encargados de validar y otorgar permisos a los usuarios que se registran así como realizar tareas de administración.
- Cliente, usuarios con permisos para realizar una descarga.

El sitio web no solo ofrecerá la última actualización, también contará con un histórico de anteriores actualizaciones. Además el usuario podrá realizar una selección de las categorías que desee, dada una fecha determinada y así obtener una actualización personalizada basándose en sus necesidades. El funcionamiento estará basado en conformar una actualización por cada petición de descarga, conteniendo un único fichero por cada categoría seleccionada. Si se seleccionan varios ficheros para descargar de la misma categoría pero de fechas distintas, se debe crear un único fichero con la información de todos los seleccionados y la fecha del más actual. La actualización final será ofrecida en un compactado de tipo zip que será utilizado directamente por Smart Keeper para actualizarse. El sitio web estará confeccionado en la versión más reciente del CMS Drupal, la 7.14. El diseño será provisto por un especialista perteneciente al Departamento Creativo de la Dirección de Comunicación Visual de la universidad y cuando sea aprobado podrá ser utilizado en la solución.

2.3 Modelo de Dominio

Un modelo de dominio contiene los objetos que existen relacionados con el proyecto que se va a acometer y las relaciones que hay entre ellos, pero no son clases de software, aunque algunos objetos del Modelo de Dominio pueden terminar siéndolo. Es aplicado en entornos simples y donde se tiene conocimiento y dominio de su funcionamiento del sistema. Ayuda a comprender los conceptos que utilizan

los usuarios, con los que trabajan y con los que deberá trabajar el sistema [33]. En la Figura 1 se muestra el Modelo de Dominio realizado.

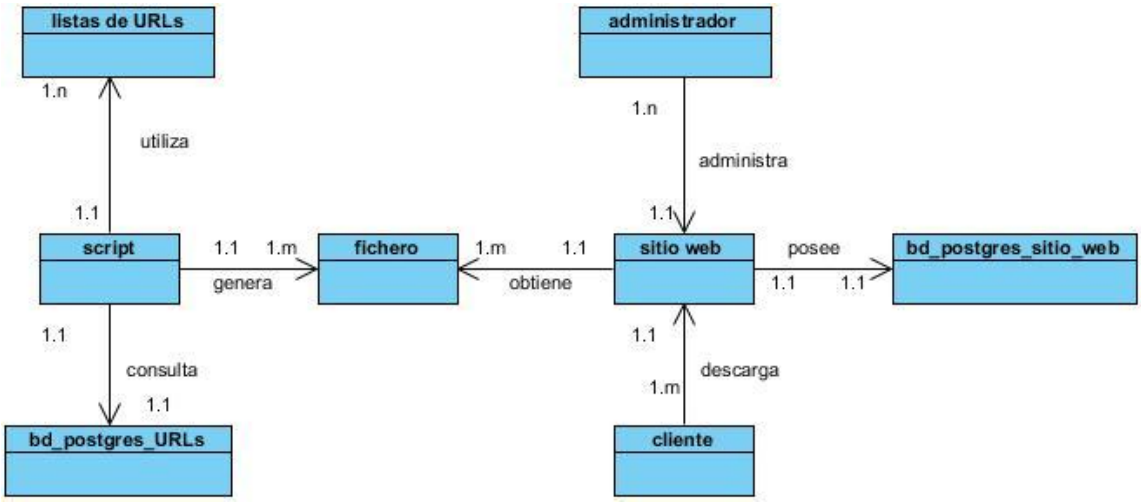


Figura 1: Modelo de dominio

listas de URLs: representa las listas de URLs que serán usadas para la confección del fichero de actualización.

script: representa los *script* que implementará para generar el fichero.

fichero: representa los ficheros que se van a generar para luego ser publicados.

sitio web: representa el sitio que se va a utilizar para publicar los ficheros y que permitirá la descarga a los clientes.

bd_postgre_sitio_web: representa la base de datos del sitio web y es donde se guarda toda la información que maneja el sitio.

bd_postgre_URLs: representa la base de datos que será consultada para obtener los cambios a realizar.

cliente: representa el usuario que accede a descargar uno o varios ficheros del sitio.

administrador: representa al usuario que genera el fichero y lo publica en el sitio web.

2.4 Requisitos funcionales

Los requisitos funcionales son características, capacidades o condiciones que debe cumplir un sistema. Están enfocados hacia todo lo que debe hacer el sistema, el usuario y los miembros del equipo del proyecto [33]. Se tuvieron en cuenta los siguientes requerimientos funcionales para identificar lo que el sistema debe hacer.

La representación utilizada por cada requisito es **RF#** donde RF significa Requisito Funcional y # es el número que le corresponde.

RF1: Autenticar usuario.

RF2: Gestionar usuarios.

- **RF2.1:** Adicionar usuario.
- **RF2.2:** Modificar usuario.
- **RF2.3:** Eliminar usuario.

RF3: Gestionar roles de usuario.

- **RF3.1:** Adicionar roles de usuario.
- **RF3.2:** Modificar roles de usuario.
- **RF3.3:** Eliminar Roles de usuario.

RF4: Gestionar permisos por cada rol.

- **RF4.1:** Adicionar permisos a un rol.
- **RF4.2:** Modificar permisos de un rol.
- **RF4.3:** Eliminar permisos de un rol.

RF5: Publicar actualizaciones.

RF6: Descargar última actualización, por usuarios autorizados.

RF7: Descargar última actualización de categorías, por usuarios autorizados.

RF8: Descargar actualización de histórico de categorías, por usuarios autorizados.

RF9: Descargar todas las actualizaciones, por usuarios autorizados.

RF10: Descargar actualización por fecha, por usuarios autorizados.

RF11: Descargar actualización de categorías por fecha, por usuarios autorizados.

2.5 Requisitos no funcionales

Una requisito no funcional es describe como el software logrará su función, no lo que hará. Los requisitos no funcionales son difíciles de verificar y comprobar, por ello son evaluados subjetivamente [34]. A continuación se muestran los requisitos no funcionales identificados.

Usabilidad: el sistema debe permitir que el administrador cumpla con todas sus funciones. La interfaz debe ser agradable e intuitiva para los usuarios.

Rendimiento: el sistema deberá responder en el mínimo de tiempo posible ante las solicitudes de información. El rendimiento también estará determinado por la cantidad de información que el usuario solicite descargar y el tiempo que demora el sistema en procesarlas según las propiedades del servidor.

Fiabilidad: aunque la generación de las actualizaciones presente problemas, el sistema de publicación debe seguir funcionando y viceversa. Las actualizaciones deben hacerse de forma periódica y verificando su integridad.

Disponibilidad: el sistema debe estar disponible las 24 horas del día. Para el correcto funcionamiento del sistema se debe contar con los módulos y dependencias requeridas.

Seguridad: la información estará protegida contra accesos no autorizados utilizando técnicas de validación como usuario, contraseña y nivel de acceso, garantizando así la confidencialidad. Se usarán mecanismos de encriptación en el caso de las URLs.

Eficiencia: el sistema deberá aumentar y actualizar el número de URLs en la base de datos y mantener a los usuarios o clientes actualizados [35].

2.6 Diagrama de Casos de Uso del Sistema

El diagrama de Casos de Uso del Sistema permite a los desarrolladores de software y a los clientes llegar a un acuerdo sobre las condiciones y posibilidades que debe cumplir el sistema [33]. En la Figura 2 se muestra el Diagrama de Casos de Uso del sistema de generación y publicación de actualizaciones.

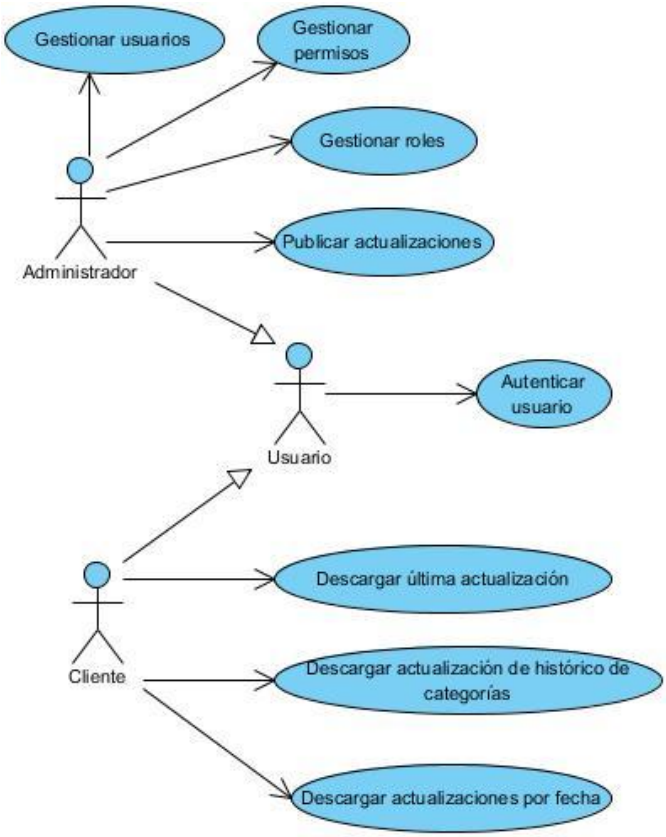


Figura 2: Diagrama de casos de uso del sistema

Este diagrama representa los diferentes roles con los que cuenta el sistema identificando las funcionalidades que son realizados por cada uno.

2.7 Definición de actores

Un actor es una persona externa o un proceso que interactúa con un sistema, un subsistema o una clase [33]. La Tabla 1 muestra la descripción de los actores que intervienen en el sistema de generación y publicación de actualizaciones.

Tabla 1: Definición de actores

Actor	Descripción
Administrador	Es quien realiza todas las acciones relacionadas con la administración del sitio y es el encargado de validar los usuarios que son clientes del sistema.
Usuario	Es toda persona que accede al sitio.
Cliente	Es el usuario con permisos para realizar una descarga.

2.8 Listado de los casos de uso expandidos

Un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia [33]. Las tablas 2 y 3 muestran las descripciones de los casos de uso Publicar actualizaciones y Descargar actualización por fecha, el resto de las descripciones se muestran en el Anexo A.

Tabla 2: Descripción del caso de uso Publicar actualizaciones

Caso de Uso:	Publicar actualizaciones
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador accede a publicar en el sitio web una nueva actualización y finaliza cuando la actualización puede ser visualizada y descargada por el cliente.
Precondiciones:	Deben existir actualizaciones para publicar.
Referencias	RF5
Prioridad	Alta
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El actor selecciona la opción de Agregar contenido. 3. El actor selecciona el contenido que representa a la actualización a publicar. 5. El actor introduce los valores.	2. El sistema muestra el listado con los contenidos existentes. 4. El sistema muestra el formulario para añadir la fecha de la actualización y las categorías que posee. 6. El sistema añade la nueva actualización.
Poscondiciones	Es publicada una nueva actualización.

Tabla 3: Descripción del caso de uso Descargar actualización por fecha

Caso de Uso:	Descargar actualización por fecha
Actores:	Cliente
Resumen:	El caso de uso se inicia cuando el cliente accede a descargar una actualización dada la fecha de publicación y finaliza cuando es completada la descarga.
Precondiciones:	Deben existir el o los ficheros de actualización y el usuario debe tener permiso de descarga.
Referencias	RF10, RF11
Prioridad	Alta
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El actor accede a la página de las actualizaciones por fecha. 3. Selecciona las categorías que desea descargar o la actualización completa dada una fecha determinada. 4. Selecciona el botón descargar. 7. Selecciona el lugar y acepta la descarga.	2. El sistema muestra la página de descarga por fecha. 5. El sistema busca los ficheros que corresponden con la selección y crea una actualización con estos. 6. Muestra una ventana para seleccionar el lugar donde se desea guardar la actualización.
Poscondiciones	Es descargado el compactado con las actualizaciones.

2.9 Conclusiones

- La descripción del problema permitió evidenciar cuál es la situación actual del software Smart Keeper y la necesidad que exige el mismo de un sistema que genere y publique sus actualizaciones.

- Bastó con un modelo de dominio para representar el funcionamiento del sistema, lo que pudo ser posible gracias a la familiarización con los conceptos y procesos por parte de las personas que intervienen en el desarrollo de la solución.

- Con la identificación de los requisitos funcionales y no funcionales se fomentó una base para las posteriores fases que establece la metodología seleccionada.

- La descripción de los casos de uso permitió que el equipo de desarrollo obtuviera una mejor visión de las funcionalidades del sistema y de cómo debe ser el resultado final de cada una.

Capítulo 3. Diseño de la solución propuesta

El flujo de trabajo diseño en el ciclo de vida de RUP, alcanza una mayor dimensión en la fase de elaboración. Asociado a este flujo de trabajo se generan varios artefactos que por su importancia constituyen la base para las fases posteriores. En este capítulo se realizan las actividades correspondientes al flujo de trabajo diseño teniendo en cuenta la arquitectura y las características principales del CMS Drupal. Se identifica el patrón arquitectónico y los patrones de diseño empleados en la solución. Haciendo uso de UML, como lenguaje de modelado, se representan los diagramas de clases del diseño, de secuencia y de despliegue que modelan los elementos y relaciones que conforman el sistema.

3.1 Patrones arquitectónicos

Un patrón arquitectónico es el nivel en el cual la arquitectura de software define la estructura básica de un sistema, pudiendo estar relacionado con otros patrones. Representa una plantilla de construcción que provee un conjunto de subsistemas aportando las normas para su organización [36].

La confección del sitio web de publicación y descarga de las actualizaciones del filtro de contenidos Smart Keeper será realizada utilizando Drupal, por tanto se ajusta a la arquitectura y patrones que este establece. Dicho CMS no se ajusta exactamente a ningún patrón arquitectónico, sin embargo posee una estructura similar al MVC (Modelo Vista Controlador). En la utilización que Drupal hace del patrón arquitectónico MVC no existe interacción entre la vista y el modelo, esto se debe a que dichas interacciones siempre se realizan a través del controlador [36].

Modelo Vista Controlador

El MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos [36].

- **Modelo**: es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista**: presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

3.2 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas específicos y comunes del desarrollo de software y para mejorar el diseño del mismo. Algunos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas [37]. Los patrones de diseño tenidos en cuenta para la implementación de la solución fueron:

Bajo acoplamiento: se basa en tener las clases lo menos relacionadas entre sí que se pueda. De forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto, potenciando la reutilización y disminuyendo las dependencias [37]. Este patrón se evidencia en la solución, en las clases que generan las actualizaciones ya sea por fecha, última actualización o el histórico de actualizaciones. Estas no poseen dependencias entre ellas ni con otras clases, un ejemplo es la clase “histórico”, que es la encargada de conformar la actualización correspondiente al histórico de actualizaciones y permitir la descarga.

Alta cohesión: define que la información que almacena una clase debe ser coherente y estar en la mayor medida de lo posible relacionada con la clase. Además, que cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos [37]. Este patrón se evidencia en la solución en las clases empleadas para crear las actualizaciones que serán descargadas desde el sitio web. Cada clase realiza únicamente la tarea que tiene asignada que es crear la

actualización que le corresponde, por ejemplo la clase “histórico” conforma la actualización perteneciente al histórico de categorías.

3.3 Diagrama de Clases del Diseño

Un diagrama de clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Está compuesto por los siguientes elementos: la clase con sus atributos, métodos y visibilidad, además las relaciones que pueden ser de herencia, composición, agregación, asociación y uso [33]. En las figuras 3 y 4 se muestran los diagramas de clases del diseño de los casos de uso Publicar actualizaciones y Descargar actualización por fecha, el resto de los diagramas se encuentran en el Anexo B.

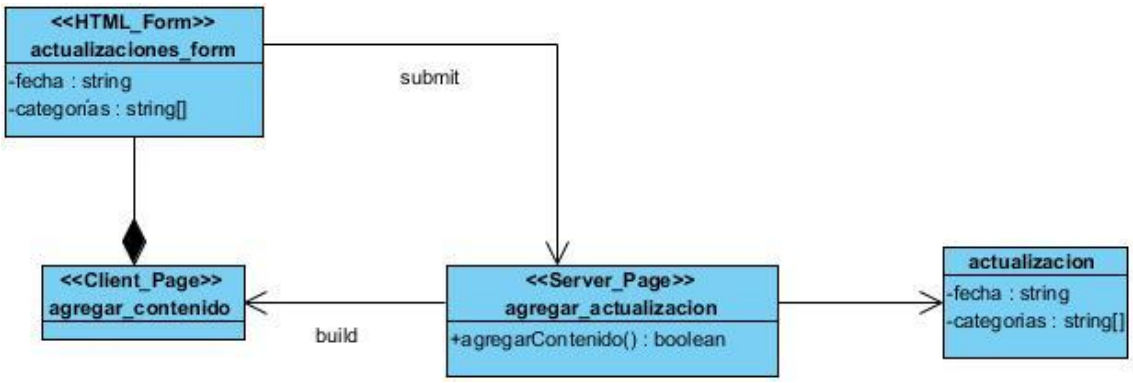


Figura 3: Diagrama de clases del diseño correspondiente al caso de uso Publicar actualizaciones

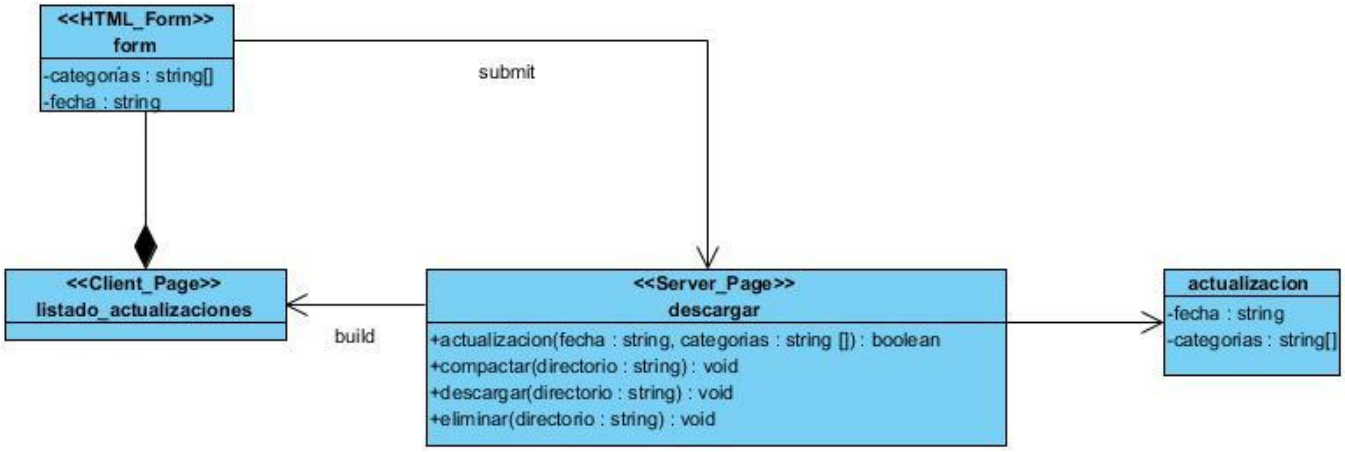


Figura 4: Diagrama de clases del diseño correspondiente al caso de uso Descargar actualización por fecha

Los diagramas mostrados anteriormente representan la distribución de las clases con sus atributos y métodos. En el diagrama mostrado en la figura 3 se encuentra la clase “agregar_contenido” que hace referencia a la interfaz del sistema con la que interactúa el Administrador para añadir una actualización, la clase “agregar_actualizacion” representa a la clase controladora que maneja todo el proceso. En la figura 4 se encuentra la clase “listado_actualizaciones” que representa la interfaz que muestra las actualizaciones, la clase “descargar” representa a la clase controladora, que procesa los datos que le envía la vista para luego confeccionar la actualización y la clase “actualización”, común para ambos diagramas, representa la información que es consultada durante el proceso.

3.4 Diagramas de Interacción

Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento. Un diagrama de interacción capta el comportamiento de un solo caso de uso y muestra determinado número de objetos y los mensajes que se pasan entre ellos dentro del caso de uso. Esta interacción se puede expresar en diagramas de colaboración y de secuencia.

En el diseño es preferible usar los diagramas de secuencia, estos muestran las secuencias de interacción detalladas y ordenadas en el tiempo. Mediante de una línea de vida del objeto se representa su existencia dentro de un periodo de tiempo [33].

En las figuras 5 y 6 se muestran los diagramas de secuencias para los casos de uso Publicar actualizaciones y Descargar actualización por fecha, el resto de los diagramas se encuentran en el Anexo C.

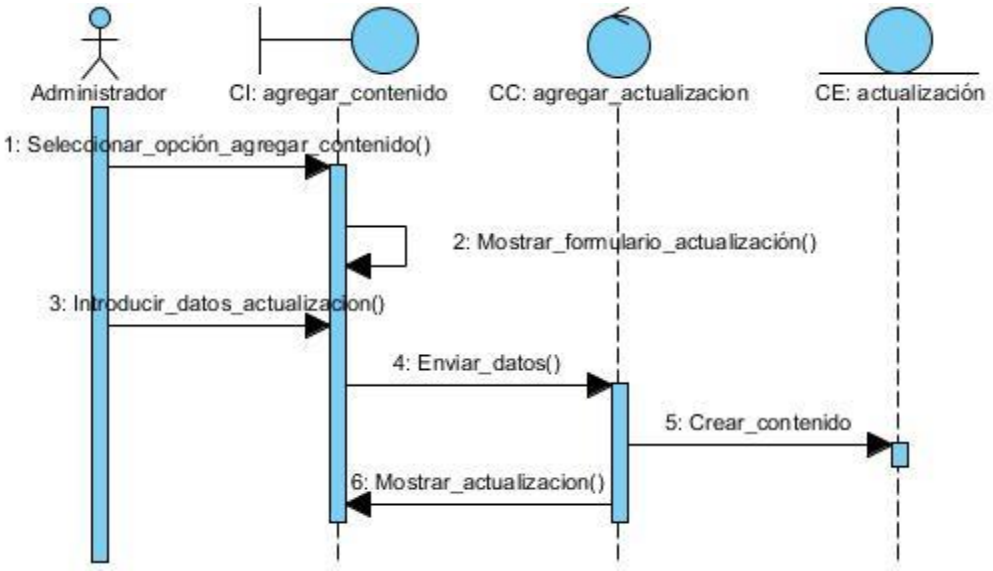


Figura 5: Diagrama de secuencia correspondiente al caso de uso Publicar actualizaciones

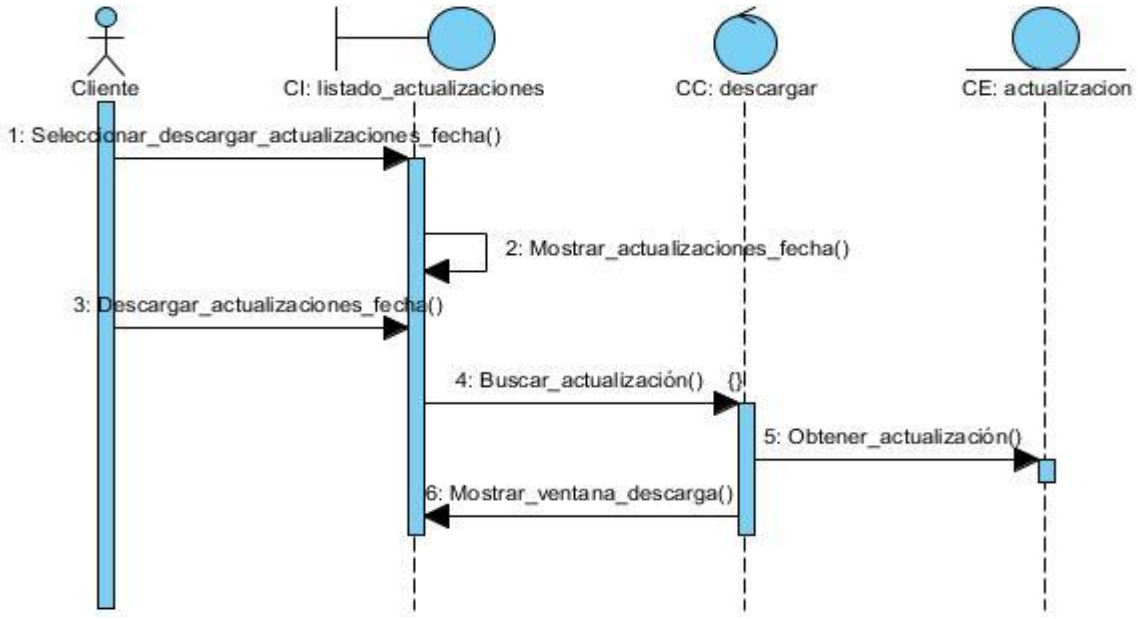


Figura 6: Diagrama de secuencia correspondiente al caso de uso Descargar actualización por fecha

Los diagramas mostrados anteriormente representan la secuencia de acciones que se realizan, derivadas de la interacción del Administrador con el sistema en el caso de publicar actualizaciones y del Cliente con el sistema en el caso de realizar la descarga de las actualizaciones por fecha.

3.5 Modelo de despliegue

Describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo sobre los que se va a instalar el sistema. El mismo incluye un artefacto fundamental, el diagrama de despliegue, el cual es usado para visualizar la distribución de los componentes de software en los nodos físicos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware. Los nodos tienen relaciones entre ellos que representan los medios de comunicación que hay entre ellos como una Intranet o Internet [33]. En la Figura 7 se muestra el Modelo de despliegue del sistema de generación y publicación de actualizaciones.

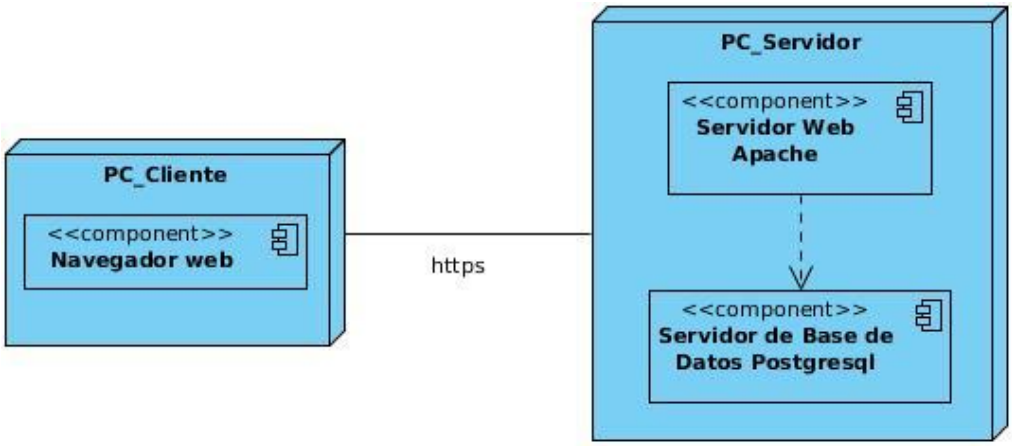


Figura 7: Modelo de despliegue correspondiente al sistema de generación y publicación de actualizaciones

PC_Cliente: en este nodo físico estará alojado el navegador web (Firefox) para acceder al sitio web, mediante el protocolo HTTPS.

PC_Servidor: en este nodo físico estará representado el servidor Web Apache donde estará montado el sitio web, el servidor de base de datos Postgresql para guardar toda la información que es manejada por el sitio y la que es consultada por el *script*.

3.6 Conclusiones

- La identificación de patrones arquitectónicos y de diseño, posibilitaron establecer una guía para el desarrollo de la estructura final del sistema.

- Con la generación de algunos de los diagramas correspondientes al flujo de análisis y diseño se logró una mayor comprensión de la estructura del sistema sirviendo como base en el desarrollo de la solución.

Capítulo 4. Implementación y validación de la solución propuesta

Una vez terminado el diseño del sistema ya se ha capturado la mayor parte de la arquitectura y se ha creado un plano del modelo de implementación. Un producto listo para ser entregado, requiere el completo desarrollo y validación de las funcionalidades previamente definidas. En el presente capítulo los resultados obtenidos de fases anteriores se modelan mediante el diagrama de componentes. Se muestran las interfaces principales del sistema y se realizan pruebas para medir la calidad del software y detectar la existencia de errores en la implementación realizada para posteriormente analizar los resultados obtenidos y comprobar si la solución cumple con las necesidades del cliente y con los niveles de calidad.

4.1 Diagrama de Componentes

Un diagrama de Componentes representa las dependencias entre los componentes que conforman un sistema de software. Los componentes pueden incluir archivos, librerías, módulos, ejecutables y otros. Son utilizados para modelar la vista estática y dinámica de un sistema. No es necesario que el diagrama incluya todos los componentes del sistema ya que se puede realizar por partes. Uno de sus usos principales es que puede servir para ver qué componentes puedan compartirse entre sistemas o diferentes partes del sistema [38].

La figura 8 muestra el Diagrama de componentes correspondiente a la solución.

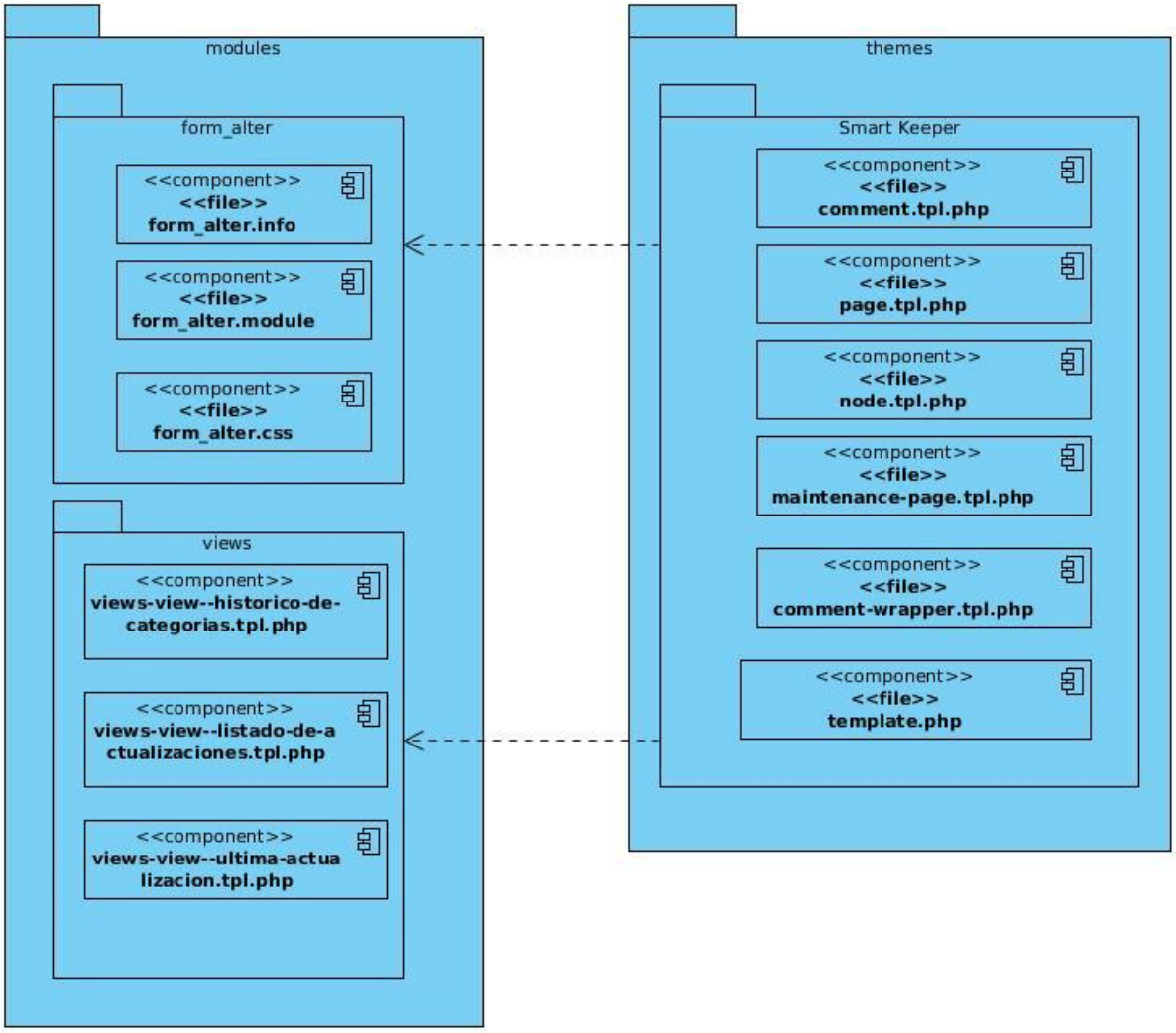


Figura 8: Diagrama de componentes del sistema.

El diagrama mostrado anteriormente representa los componentes que conforman la solución propuesta. Dentro de estos se encuentran los pertenecientes a “themes”, que intervienen en cómo será mostrada la información al cliente. Además el paquete “modules”, representa los módulos utilizados en Drupal para la confección de la solución y que son necesarios para cumplir con todas los requerimientos especificados.

4.2 Principales interfaces del sistema.

El sistema de publicación de actualizaciones de la información de la base de datos del filtro de contenidos Smart Keeper, consta de diferentes interfaces que son mostradas en dependencia de la acción que se está realizando en el sitio web. La figura 9 muestra la interfaz relacionada con la última actualización y que será consultada por las personas que no son Clientes del sistema. La interfaz permite únicamente conocer la fecha de la actualización y las categorías que posee, sin posibilidad de descarga.

Figura 9: Interfaz externa correspondiente a la última actualización

La figura 10 muestra la interfaz relacionada con la última actualización y que será consultada por los Clientes del sistema. Dicha interfaz permite consultar la fecha y las categorías de la actualización, además permitirá tener acceso a la opción de descarga ya sea de la actualización completa o de las categorías que desee.

Figura 10: Interfaz del cliente correspondiente a la última actualización

4.3 Validación del sistema

Completada la implementación del sistema se procedió a aplicar un conjunto de pruebas. Estas son una actividad en la cual un sistema o uno de sus componentes son ejecutados bajo ciertas condiciones o requerimientos específicos que permiten comprobar y revelar su calidad. Verifican que el producto funcione como se diseñó y que los requerimientos establecidos sean cumplidos teniendo como primera intención descubrir una falla. Se utilizan para identificar fallos en la implementación y evaluar la calidad del sistema. Un buen caso de prueba es aquel que tiene la probabilidad de mostrar un error no descubierto hasta entonces [17]. A continuación se muestran las pruebas realizadas al sistema y los resultados obtenidos por cada una.

Pruebas funcionales o de caja negra

Tienen como objetivo principal verificar que se cumplan los requisitos funcionales del software. Consiste en estudiar las entradas que recibe y las respuestas que produce determinado sistema, sin tener en cuenta la lógica del programa. Para definir correctamente las entradas y salidas del software hay que encontrar una serie de datos de entrada que puedan causar un comportamiento erróneo en el sistema y como consecuencia producen una serie de salidas que revelan la presencia de defectos [17].

Las pruebas funcionales que se realizarán a la solución, estarán enfocadas o dirigidas a los casos de uso del sistema para verificar su correcto funcionamiento. En este tipo de pruebas se ejecutarán los distintos servicios prestados con datos correctos e incorrectos. En caso de que los datos sean incorrectos se verificará que los mensajes de error sean los deseados y en el caso opuesto que los resultados sean los esperados.

La tabla 4 muestra las pruebas realizadas al caso de uso Publicar actualizaciones.

Tabla 4: Casos de pruebas correspondientes al caso de uso Publicar actualizaciones

CU Publicar actualizaciones		
Entrada	**Resultados**	**Condiciones**
Publicar una actualización satisfactoriamente.	El sistema muestra la actualización en una tabla con la fecha y las categorías de esa actualización.	Debe existir la actualización.
Publicar una actualización con fecha incorrecta.	El sistema muestra en el momento de realizar la descarga de la actualización una página de error indicando que las actualizaciones seleccionadas no han podido ser localizadas.	

La tabla 5 muestra las pruebas realizadas al caso de uso Descargar actualizaciones por fecha, el resto de las pruebas se encuentra en el Anexo D.

Tabla 5: Casos de pruebas correspondientes al caso de uso Descargar actualización por fecha

CU: Descargar actualizaciones por fecha		
Entrada	**Resultados**	**Condiciones**
Descargar actualizaciones por fecha satisfactoriamente	El sistema crea una actualización según las opciones seleccionadas, con un único fichero por cada categoría seleccionada y permite al usuario descargar dicha actualización.	Deben ser seleccionadas las actualizaciones a descargar y estas deben existir.
Descargar actualizaciones que no puedan ser localizadas.	El sistema muestra una página de error en caso de que no se localice ninguna actualización seleccionada y en caso de ser sólo algunas las no localizadas confecciona la actualización con las	

	restantes y permite la descarga.	
Solicitar descargar sin seleccionar una actualización.	El sistema muestra una página de error indicando que no se ha seleccionado ninguna actualización.	

La realización de las pruebas funcionales correspondientes a los casos de uso del sistema, permitió evaluar la calidad y eficiencia del software. En las primeras iteraciones realizadas se obtuvieron un conjunto de no conformidades como falta de validaciones y problemas en la configuración del servidor. Las no conformidades detectadas fueron corregidas, obteniendo resultados satisfactorios en posteriores iteraciones de pruebas.

Pruebas de rendimiento

Son realizadas para determinar que tan rápido un sistema realiza una tarea. También son utilizadas para validar y verificar diferentes aspectos de la calidad de software, como por ejemplo el buen uso de los recursos. Para su diagnóstico, se utilizan herramientas que permiten monitorear y obtener información sobre el rendimiento del sistema según la ejecución de determinadas funcionalidades [17].

La realización de estas pruebas está enfocada en determinar la capacidad del sistema de recibir múltiples peticiones sin que se vea afectado, así como la velocidad de respuesta del mismo. Se decidió utilizar el programa Jmeter, que es una herramienta de software libre que ofrece la posibilidad de medir la capacidad de carga de una aplicación y permite conocer los tiempos de respuesta dado un número de usuarios determinados y un número real de transacciones procesadas por unidad de tiempo [39].

La realización de las pruebas de rendimiento de software, con del uso de la herramienta Jmeter, permitió obtener resultados acerca la capacidad de carga del sistema. Las pruebas fueron aplicadas en dos servidores distintos con propiedades diferentes. A continuación se muestran las propiedades de cada servidor.

Servidor 1:

- Memoria RAM: 4GB.
- Procesador: Intel® Core™2 Duo CPU E4600 @ 2.40GHz × 2.
- Gráficos: Gallium 0.4 on ATI RV370.

Servidor 2:

- Memoria RAM: 1GB.
- Procesador: Intel® Pentium(R) 4 CPU 3.00GHz × 2.
- Gráficos: Intel® 945G x86/MMX/SSE2

En una fase inicial se realizaron las pruebas, en ambos servidores, con un total de 40 peticiones enviadas concurrentemente en un hilo de ejecución. Los resultados fueron para 2960 muestras que se le hicieron al servidor, en el caso del servidor 1, el tiempo mínimo de respuesta es de 1 segundo y el tiempo máximo de 35 segundos, en el servidor 2 el tiempo mínimo de respuesta es de 1 segundo y el tiempo máximo de 54 segundos. En una segunda iteración se probó, en ambos servidores, con 70 peticiones en un hilo de ejecución. Los resultados fueron para 5180 muestras que se le hicieron al servidor, en el caso del servidor 1, el tiempo mínimo de respuesta es de 1 segundo y el tiempo máximo de 50 segundos, en el servidor 2 el tiempo mínimo de respuesta es de 1 segundo y el tiempo máximo de 86 segundos. Posteriormente se realizó una comparación con los resultados obtenidos, evidenciándose que la capacidad de carga del sistema no es muy elevada, afectándose la velocidad de respuesta del sistema, dado que las acciones que este permite dependen de las propiedades del hardware del servidor y de la cantidad de información que se solicite descargar.

Pruebas de integración

Una vez aplicada un conjunto de pruebas al sistema se procedió a comprobar si este se integraba correctamente con el filtro de contenidos Smart Keeper. Para lograr lo anteriormente mencionado se realizan las pruebas de integración porque aunque las funcionalidades del sistema respondan

correctamente por separado es necesario probarlos conjuntamente con el sistema al que se deben incorporar. Una funcionalidad puede tener un efecto adverso o inadvertido sobre otra funcionalidad del sistema a incorporarse [40].

El proceso de integración se efectuó comprobando si la actualización que se descarga desde el sitio web cumple con los requerimientos necesarios y puede ser utilizada para actualizar el filtro de contenidos. En la tabla 6 se muestran los resultados obtenidos al aplicar las pruebas de integración entre el sistema desarrollado y el filtro de contenidos Smart Keeper.

Tabla 6 Casos de prueba de integración.

Actualizar base de datos de Smart Keeper.		
Entrada	**Resultados**	**Condiciones**
Actualizar la base de datos satisfactoriamente.	El sistema muestra un mensaje informando los cambios realizados en la base de datos como: cantidad total de URLs que fueron adicionadas o eliminadas y la cantidad por cada categoría.	La actualización debe estar correctamente estructurada.
Solicitar actualizar la base de datos con campo vacío.	El sistema muestra el mensaje “El campo no puede estar vacío”.	
Actualizar la base de datos con ficheros que poseen errores en la fecha.	El sistema descarta completamente la información del fichero con problemas.	
Actualizar la base de datos con ficheros que poseen errores en las URLs.	El sistema descarta las URLs que poseen errores y actualiza la base de datos con las restantes.	

Los resultados obtenidos al aplicar las pruebas de integración fueron que las actualizaciones generadas cumplen con lo especificado y al efectuar una actualización, realizando todo el proceso como está

establecido, ocurren modificaciones en la base de datos de Smart Keeper, añadiendo nuevas URLs y eliminando las obsoletas.

4.4 Conclusiones

- Con la modelación de los Diagramas de Componentes se logró tener una vista general de todas las dependencias y funcionalidades necesarias para el correcto funcionamiento del sistema.

- La realización de pruebas al software permitió corregir errores detectados, posibilitando desarrollar un sistema con mayor calidad.

- Los resultados arrojados por las pruebas realizadas, permitieron la validación de la calidad y eficiencia del sistema y demostrar que la solución pudo ser integrada con el filtro de contenidos Smart Keeper.

Conclusiones

Después del estudio realizado se determinó que ninguna de las soluciones estudiadas podría ser reutilizada completamente para dar solución a la problemática planteada. Se decidió desarrollar un sistema que permita generar y publicar actualizaciones para la información de la base de datos de Smart Keeper, empleando elementos obtenidos del estudio.

- El estudio realizado a algunos filtros de contenidos permitió identificar determinados aspectos que pudieron ser aprovechados en el desarrollo de la solución. Además el estudio de diferentes herramientas, lenguajes y tecnologías posibilitó la selección de la base tecnológica a utilizar.

- La propuesta de solución y el diseño definido para dicha propuesta dio respuesta a todos los requisitos establecidos por el cliente y creó las bases para la implementación.

- La implementación del sistema permitió aumentar la cantidad de URLs con la que contaba la base de datos de Smart Keeper y una mayor flexibilidad para el cliente al obtener las actualizaciones, dado que puede obtenerlas separadas por categorías, por fecha y a la vez contar con el historial de todas las actualizaciones.

- Las pruebas realizadas al sistema permitieron corregir los errores detectados antes de declarar el sistema completamente terminado y se comprobó que puedo ser integrado con Smart Keeper y está listo para brindar un servicio adecuado.

Con de lo planteado anteriormente se puede afirmar que se cumplieron los objetivos propuestos y puede ser comprobado con la aplicación obtenida.

Recomendaciones

Con vista a poder establecer nuevas funcionalidades al sistema de generación y publicación de las actualizaciones de la información de la base de datos de Smart Keeper, se proponen las siguientes recomendaciones del presente Trabajo de Diploma:

- Desarrollar un *script* o introducir una funcionalidad en los existentes que permita realizar la descarga de las listas de URLs de Internet de manera *online* como parte del proceso.

- Valorar la creación de una interfaz visual para la generación de las actualizaciones.

- Comprobar frecuentemente la efectividad y veracidad de la información que brindan las listas seleccionadas.

- Una vez que el sitio web de publicación de las actualizaciones esté prestando servicio y cuente con el acceso de clientes de diferentes nacionalidades, se recomienda internacionalizar el sistema.

Referencias Bibliográficas

1. *Ya hay 2.000 millones de internautas.* [En línea] 2011. [Citado enero 2012]. Disponible en: http://www.elmundo.es/elmundo/2011/01/26/navegante/1296045597.html
2. **Norton.** *Protección de menores en la red: Filtros de contenidoss [En línea].* 2011. Disponible en: http://www.nortonfanclub.com/2011/06/proteccion-de-menores-en-la-red-filtros-de-contenidos-parte-i/.
3. **Netcraft LTD 2012.** *Web Server Survey*. [En línea] Noviembre de 2011. Disponible en: http://news.netcraft.com/archives/2011/11/07/november-2011-web-server-survey.html.
4. **Enamorado, Ing. Alain Guerrero.** *Concepción del Sistema de Filtrado para Internet.* La Habana : s.n., 2009.
5. **Barracuda Networks.** *Barracuda Networks*. [En línea] 2012. Disponible en: http://www.barracudanetworks.com/ns/products/web-filter-overview.php?L=es..
6. **Barron, Daniel.** DansGuardian. *DansGuardian.* [En línea] 2012. Disponible en: http://dansguardian.org/..
7. *Manual de usuario OPTENET WEB FILTER PC Versión 9.7* [En línea] 2007. Disponible en: www.optenet.com.
8. *OPTENET WEB FILTER Server 5.27*. [En línea] 2006. Disponible en: www.optenet.com.
9. **SquidGuard.** *SquidGuard Shalla Secure Services KG*. [En línea] 2007-2010. Disponible en: http://www.squidguard.org/.
10. **Luis Enrique Sánchez Arce, Jose Ramos Hermosilla Moreno.** *Sistema cubano de filtrado de contenidos web utilizando tecnologías libres.* La Habana : s.n., 2010.
11. **URLBlacklist.** *URLBlacklist*. [En línea] 2003-2012. Disponible en: urlblacklist.com.
12. **Shalla Secure Services.** *Shalla Secure Services*. [En línea] 2007-2011. Disponible en: http://www.shallalist.de/.
13. **Blacklists UT1.** *Blacklists UT1*. [En línea] 2012. Disponible en: http://cri.univ-tlse1.fr/documentations/cache/squidguard_en.html#contrib..

14. **SquidGuard.** *SquidGuard*. [En línea] Disponible en: http://squidguard.mesd.k12.or.us/.
15. **Aguilar, Dania Fernández.** *Propuesta para Actualizar la Base de Datos de URLs.* La Habana : s.n., 2009.
16. **Gómez Aguilar, Diego Alonso. García Navarro, Juan Francisco.** *Mecanismos de seguridad de la información en aplicaiones web*. [En línea] 2006. Facultad de Ciencias, Universidad de Salamanca.
17. **Pressman, Roger S**. *Ingeniería del Software. Un enfoque práctico*. Mc Graw Hill, España, 5ta edition, 2001. ISBN 8448132149. 323 pp.
18. ***UML*.** *UML*. [En línea] 2012. Disponible en: http://www.uml.org/.
19. **Périssé, Marcelo Claudio.** *Ciencia y Técnica Administrativa*. [En línea] 2001. Disponible en: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.
20. **Visual Paradigm.** *Visual Paradigm*. [En línea] Disponible en: http://www.visual-paradigm.com/.
21. **Inst, Lic Gualter Hidalgo Cordova.** *Diseño de Sistemas de Infor*. [En línea] Disponible en: http://es.scribd.com/doc/59660069/Rational-Rose-Caracteristicas.
22. **CAVSI.** *Sistemas gestores de Bases de Datos.* 2002.
23. **postgresql.** [En línea] 2012. Disponible en: http://www.postgresql.org.es/sobre_postgresql.
24. **Cuerda, Xavier García.** *Introducción a los Sistemas de Gestión de Contenido*. [En línea] 2004. Disponible en: http://mosaic.uoc.edu/2004/11/29/introduccion-a-los-sistemas-de-gestion-de-contenidos-cms-de-codigo-abierto/..
25. *CMS y framework, dos conceptos distintos*. [En línea] 25 de marzo de 2010. Disponible en: http://nohemirojas.wordpress.com/2010/03/25/cms-y-framaework-dos-conceptos-distintos/.
26. **Drupal.** [En línea] 2012. Disponible en: http://drupal.org/.
27. **joomla.** [En línea] 2012. Disponible en: http://joomla.org/.
28. *PHP: Hypertext Processor*. [En línea] 2011. Disponible en: http://www.php.net/.
29. **Perl.** [En línea] 2011. Disponible en: www.perl.org.
30. **Diz, Alberto J. Bugarín.** [En línea] Disponible en: http://www-gsi.dec.usc.es/~alberto/fdp/practicas/SunForte/ForteSun.pdf.
31. **NetBeans.** Sito oficial de netBeans. [En línea] 2011. Disponible en: http://netbeans.org/.
32. **eclipse.** Sitio oficial eclipse. [En línea] 2012. Disponible en: http://www.eclipse.org/ .

33. **Ivar Jacopson, Grady Booch, James Rumbaugh.** *El Proceso unificado de Desarrollo de Software.* España : Addison Wesley Longman Inc, 2000.
34. **Dorfman, M. and Thayer, R.***Standards, Guidelines and Examples on System and Software.* s.l. : IEEE Computer Society Press, 1990.
35. **Wesley, Addison.** *El Proceso Unificado de Desarrollo de Software.* 2000.
36. **Bahit, Eugenia.** *POO y MVC en PHP.* 2011.
37. **LARMAN, Craig**. *UML y PATRONES: Introducción al análisis y diseño orientado a objetos.* La Habana. Editorial Félix Varela, 2004. ISBN 970-17-0261-1.
38. **Schumuller, Joseph.** *Aprendiendo UML en 24 Hora.* Prentice-Hall. 2001
39. *Mejoramiento del Proceso de Pruebas y Corrección de Defectos de Software en un Ambiente Globalizado*. [En línea] 2012. Disponible en: http://chie.uniandes.edu.co/~gsd/index.php?option=com_content&task=view&id=129&Itemid=183.
40. **Natalia Juristo, Ana M. Moreno, Sira Vegas**. *Técnicas de evaluación de software*. 2005. [Consultado: Abril 2012].

Bibliografía

Aldana, Ileana Juez. [En línea] 13 de 11 de 2010. Disponible en: http://tallertematico2010.fordes.co.cu/public/site/160.pdf.

Carlos G. Vasco, Marcelo Henrique Vithoft, Paulo Roberto C. Estante. Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba, PR – Brasil. [En línea] Disponible en: http://edilms.eti.br/uploads/file/bd/RUPvsXP_draft.pdf.

Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgreSQL. Disponible en: http://www.scielo.org.ar/scielo.php?pid=S0325-00752007000400015&script=sci_arttext.

Jesus Castagnetto, Harish Rawat, Sascha Schumann, Chris Scollo, Deepak Veliath. *Professional PHP programming.* s.l. : Wrox Press, 1999.

Kit, ISAK Internet Secure Access. 2006. ISAK Internet Secure Access Kit. ISAK Internet Secure Access Kit. [En línea] 03 11, 2006. Disponible en: http://isak.gplindustries.com/wiki/Isakurldb.

Larry Wall, Tom Christiansen, Jon Orwant. *Programming Perl .* s.l. : Reilly Media, 1996.

Oralde, Juan Murua. Metodologías para desarrollo de software. [En línea] 2004.

Orcero, David Santo. *Programación en Perl. DSI: Acceso a base de datos.* s.l. : SÓLO PROGRAMADORES LINUX, 2003.

Pressman, Roger S. *Ingeniería de Software Un Enfoque Práctico.* s.l. : Mc Graw Hill (Sexta Edición), 2005. 0-07-285318-2. [En línea] 2007. Disponible en: http://www.ati.es/IMG/pdf/Tutorial2007.pdf

Professional, Addison-Wesley. *Eclipse Rich Client Platform: Designing, Coding, and Packaging.* 2005. ISBN:0321334612.

Rasmus Lerdorf, Kevin Tatroe, Peter Macintyre. *Programming PHP.* s.l. : Reilly Media, 2006.

Sánchez, José Ramón Hermosilla y Luis Enrique. 2008. Interfaz de Administración Web para el Sistema de Filtrado Filpacon. Interfaz de Administración Web para el Sistema de Filtrado Filpacon. [En línea] 2008.

Security, Panda. 2009. Panda Security for Domino Servers. Panda Security for Domino Servers. [Online] 2009. Disponible en: http://www.pandasecurity.com/NR/rdonlyres/9B8C155E-91F1-44A7-9B6D-C294A6E98145/0/01dwn_ps_PS4DOM.pdf.

Steele, Julie. Using Drupal. s.l. : Reilly Media , 2008. CA 95472.

Usaola, Dr. Macario Polo. [En línea] 2006. Disponible en: http://alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc-2b.pdf.

VanDyk, John K. *Drupal development.* s.l. : apress, 2008.

Yunior Mesa Reyes yudisney Vázquez Ortiz. Cuba : Revista Cubana de Ciencias Informáticas, 2011.

Yurisleidy Hernández Moya, Dovier Antonio Ripoll Méndez, Luis Enrique Sánchez Arce, Kiuver Kaddiel Ibañez Castro, Karel Antonio Verdecia Ortiz. *Técnicas de Inteligencia Artificial en el filtro de contenidos web Smart Keeper para la clasificación de información.* Cuba : RCCI, 2012.

Anexo A. Listado de casos de usos expandidos

Tabla 7: Descripción del caso de uso Autenticar usuario

Caso de Uso:	Autenticar usuario
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el actor introduce los datos para acceder al sistema, estos se verifican y finaliza cuando se le otorgan los permisos que tiene según el usuario y se permite la entrada al sistema.
Precondiciones:	Debe estar registrado el usuario y con permisos de acceso.
Referencias	RF1
Prioridad	Alta
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor introduce el usuario y la contraseña.	2. Se validan los datos. 3. Se consulta la base de datos para comprobar si existe ese usuario con esa contraseña. 4. Se cargan los módulos según los niveles acceso del usuario y se muestra la página del usuario.
Flujo Alterno	
Acción del Actor	**Respuesta del sistema**

	2. Si existen campos vacíos o datos incorrectos se muestra un mensaje de error y se regresa al paso 1. 3. Si no existe ese usuario se regresa al paso 1.
Poscondiciones	EL actor logra iniciar sesión con sus niveles de acceso asociados.

Tabla 8: Descripción del caso de uso Gestionar usuarios

Caso de Uso:	Gestionar usuarios	
Actores:	Administrador	
Resumen:	El caso de uso comienza cuando el administrador accede a añadir, modificar o eliminar una cuenta de usuario y finaliza cuando son realizados los cambios.	
Precondiciones:	El administrador debe haber iniciado sesión.	
Referencias	RF2	
Prioridad	Gestionar usuarios	
Flujo Normal de Eventos		
Escenario ¨Añadir usuario¨		
Acción del Actor	**Respuesta del Sistema**	
1. El actor selecciona la opción de añadir usuarios. 3. El actor introduce los datos del nuevo usuario. 4. Selecciona la opción ¨aceptar ¨.	2. El sistema muestra los campos para añadir un nuevo usuario. 5. El sistema añade el usuario. 6. Muestra usuarios registrados.	

Escenario ¨Editar usuario¨	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción ¨editar usuario¨. 3. El actor modifica los datos que el determine. 4. Selecciona la opción ¨Guardar¨.	2. El sistema muestra los datos del usuario. 5. El sistema modifica los datos del usuario. 6. Muestra el listado de usuarios actualizados.
Escenario ¨Eliminar usuario¨	
Acción del Actor	**Respuesta del sistema**
1. El actor selecciona la opción ¨editar usuario¨. 3. Selecciona la opción ¨cancelar cuenta¨. 5. El actor confirma la eliminación.	2. El sistema muestra los datos del usuario. 4. El sistema muestra un mensaje para confirmar la acción. 6. El sistema elimina el usuario.
Poscondiciones	Escenario ¨Añadir usuario¨: El usuario es añadido. Escenario ¨Editar usuario¨: El usuario es modificado. Escenario ¨Eliminar usuario¨: El usuario es eliminado.

Tabla 9: Descripción del caso de uso Gestionar roles

Caso de Uso:	Gestionar roles
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador añade, modifica, o elimina determinado rol y finaliza cuando los cambios son

	realizados.
Precondiciones:	El administrador debe haber iniciado sesión.
Referencias	RF3
Prioridad	Alta

Flujo Normal de Eventos	
Escenario ¨Añadir rol¨	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción ¨roles¨. 3. El actor selecciona la opción de añadir rol. 5. El actor introduce el nombre del nuevo rol y selecciona la opción ¨añadir rol¨.	2. El sistema muestra el listado de todos los roles registrados y la opción de añadir otro. 4. El sistema muestra el formulario para añadir un nuevo rol. 6. El sistema añade el rol.
Escenario ¨Editar rol¨	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción ¨editar rol¨. 3. El actor modifica el nombre del rol y selecciona la opción ¨Guardar rol¨.	2. El sistema muestra el nombre del rol. 4. El sistema modifica el nombre del rol.
Escenario ¨Eliminar rol¨	

Acción del Actor	Respuesta del sistema
1. El actor selecciona la opción ¨editar rol¨. 3. Selecciona la opción ¨Eliminar rol¨. 5. El actor confirma la eliminación.	2. El sistema muestra el nombre del rol. 4. El sistema muestra un mensaje para confirmar la eliminación. 6. El sistema elimina el rol.
Poscondiciones	Escenario “Añadir rol”: El rol es añadido. Escenario “Editar rol”: El rol es modificado. Escenario ¨Eliminar rol¨: El rol es eliminado.

Tabla 10: Descripción del caso de uso Gestionar permisos

Caso de Uso:	Gestionar permisos
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador selecciona la opción de editar los permisos de un rol y añade o elimina alguno y finaliza cuando son guardados los cambios realizados.
Precondiciones:	Debe existir el rol.
Referencias	RF4
Prioridad	Alta
Flujo Normal de Eventos	
Escenario “Añadir permisos”	
Acción del Actor	**Respuesta del Sistema**

1. El actor selecciona la opción “Editar permisos”. 3. El actor selecciona los permisos que desea asignar a cada rol y selecciona la opción guardar.	2. El sistema muestra el listado de todos los permisos por cada rol. 4. El sistema actualiza los permisos.
Escenario “Editar permisos”	
1. El actor selecciona la opción “Editar permisos”. 3. El actor estable nuevos permisos por cada rol.	2. El sistema muestra el listado de todos los permisos por cada rol. 4. El sistema actualiza los permisos.
Escenario “Eliminar permisos”	
1. El actor selecciona la opción “Editar permisos”. 3. El actor selecciona los permisos que desea quitar a cada rol y selecciona la opción guardar.	2. El sistema muestra el listado de todos los permisos por cada rol. 4. El sistema actualiza los permisos.
Poscondiciones	Escenario “Añadir permisos”: Los permisos son añadido. Escenario “Editar permisos”: Los permisos son modificados. Escenario ¨Eliminar permisos¨: Los permisos son eliminados.

Tabla 11: Descripción del caso de uso Descargar última actualización

Caso de Uso:	Descargar última actualización
Actores:	Cliente
Resumen:	El caso de uso se inicia cuando el cliente accede a descargar la última actualización creada y finaliza cuando es completada la descarga.
Precondiciones:	Deben existir el o los ficheros de actualización y el usuario debe tener permiso de descarga.
Referencias	RF6, RF7
Prioridad	Alta

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor accede a la página de la última actualización. 2. El actor selecciona las categorías que desea descargar de la última actualización o la actualización completa. 3. Selecciona el botón descargar. 6. Selecciona el lugar y acepta la descarga.	4. El sistema busca los ficheros que corresponden con la selección y crea una actualización con estos. 5. Muestra una ventana para selecciona el lugar donde guardar la actualización.
Poscondiciones	Es descargado el compactado con las actualizaciones.

Tabla 12: Descripción del caso de uso Descargar actualización de histórico de categorías

Caso de Uso:	Descargar actualización de histórico de categorías
Actores:	Cliente
Resumen:	El caso de uso se inicia cuando el cliente accede a descargar el histórico de una, varias o todas las categorías y finaliza cuando es completada la descarga.
Precondiciones:	Deben existir el o los ficheros de actualización y el usuario debe tener permiso de descarga.
Referencias	RF8, RF9
Prioridad	Alta
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor accede a la página de la actualización de histórico de categorías. 2. El actor selecciona las categorías que desea descargar. 3. Selecciona el botón descargar. 6. Selecciona el lugar y acepta la descarga.	4. El sistema busca los ficheros que corresponden con la selección y crea una actualización con estos. 5. El sistema muestra una ventana para selecciona el lugar donde guardar la actualización.
Poscondiciones	Es descargado el compactado con las actualizaciones.

Anexo B. Diagramas de clases del diseño

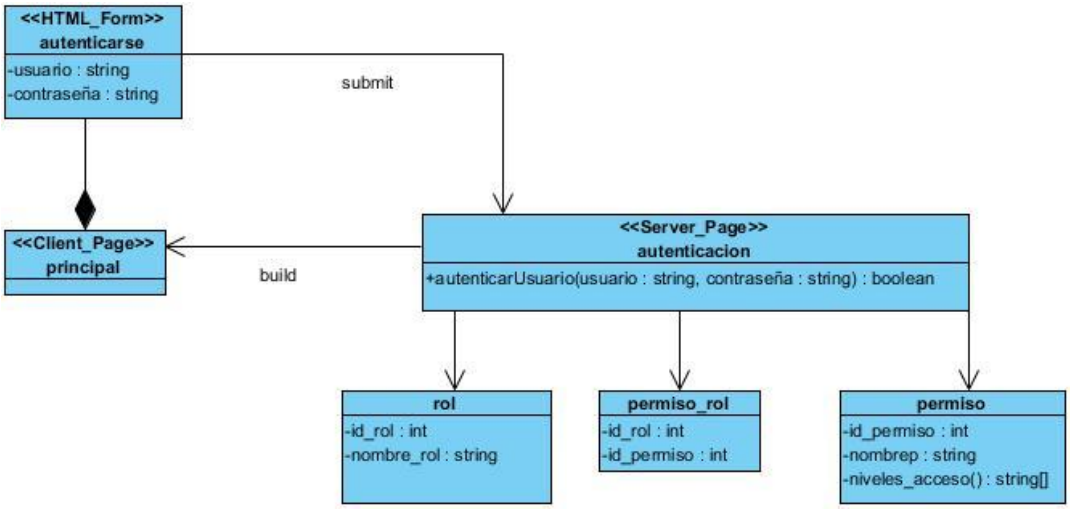


Figura 9: Diagrama de clases del diseño correspondiente al caso de uso Autenticar usuario

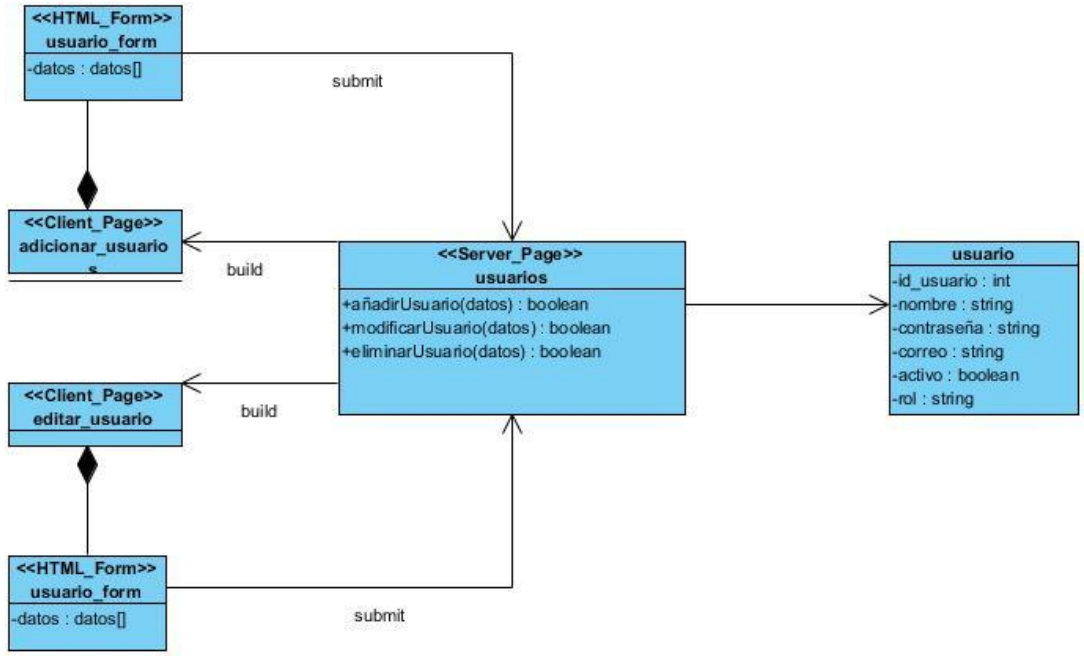


Figura 10: Diagrama de clases del diseño correspondiente al caso de uso Gestionar usuarios

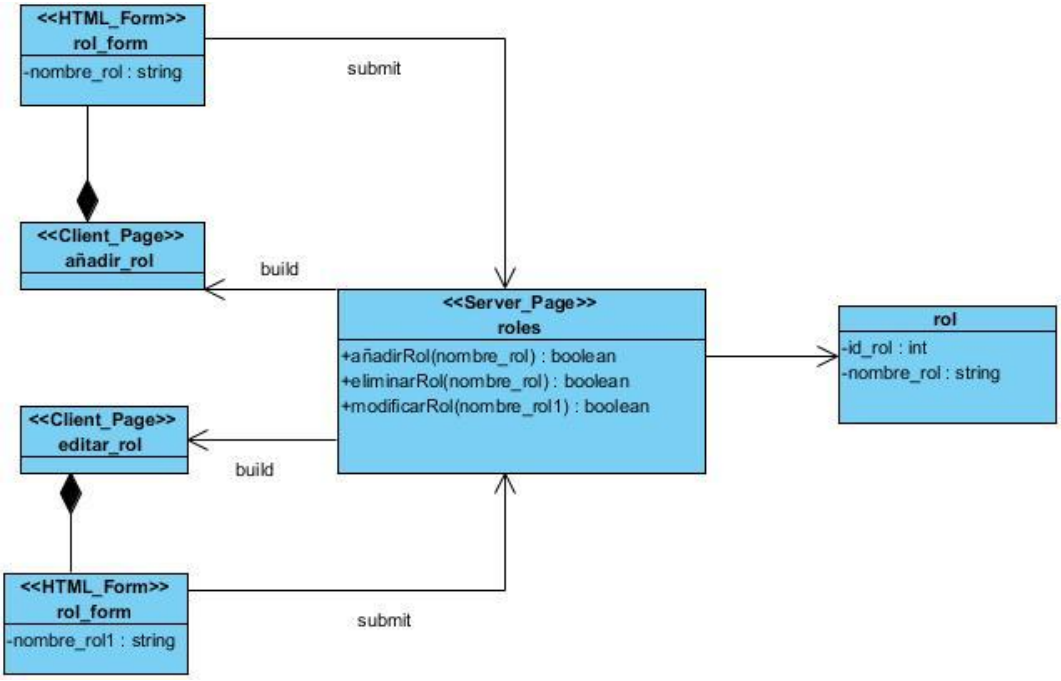


Figura 11: Diagrama de clases del diseño correspondiente al caso de uso Gestionar roles

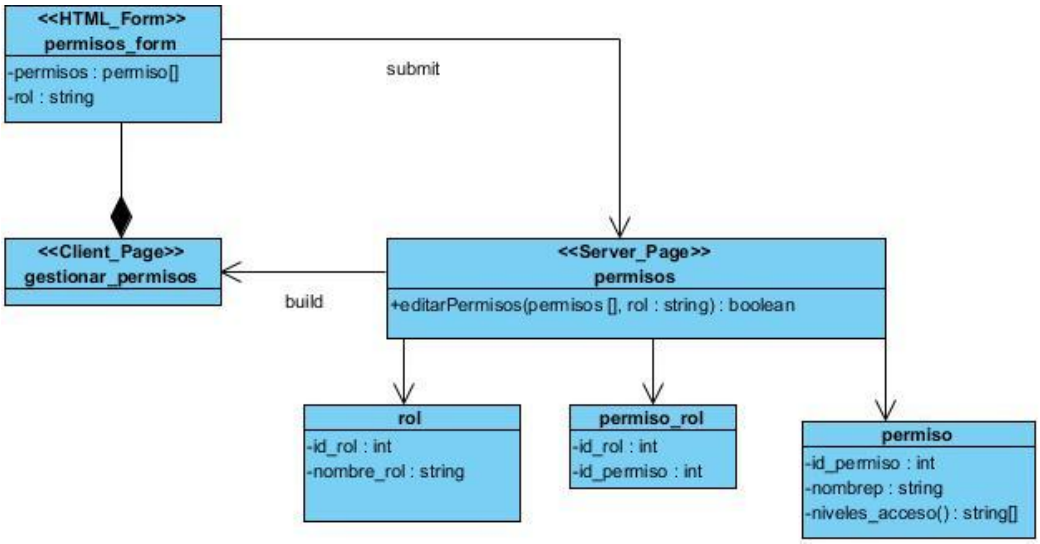


Figura 32: Diagrama de clases del diseño correspondiente al caso de uso Gestionar permisos

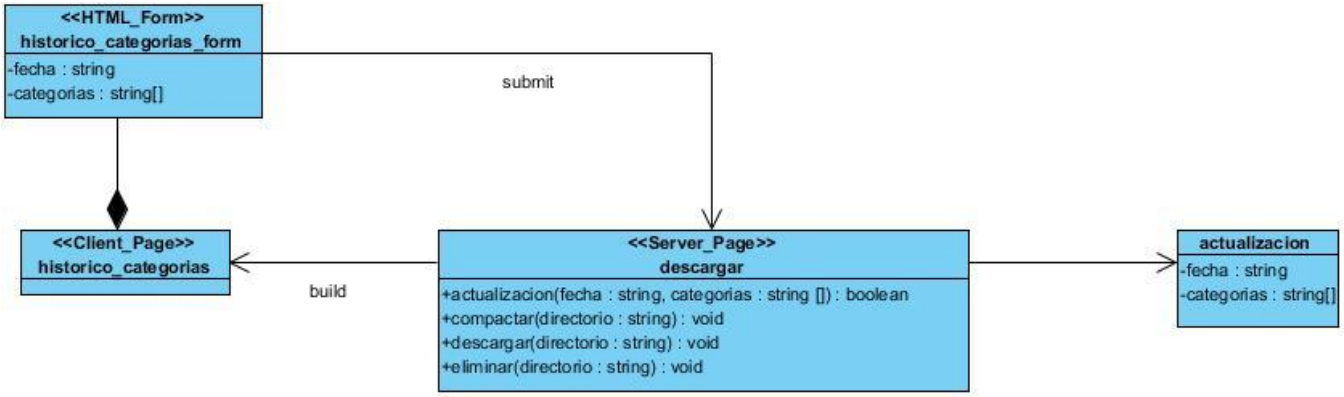


Figura 13: Diagrama de clases del diseño correspondiente al caso de uso Descargar actualización de histórico de categorías

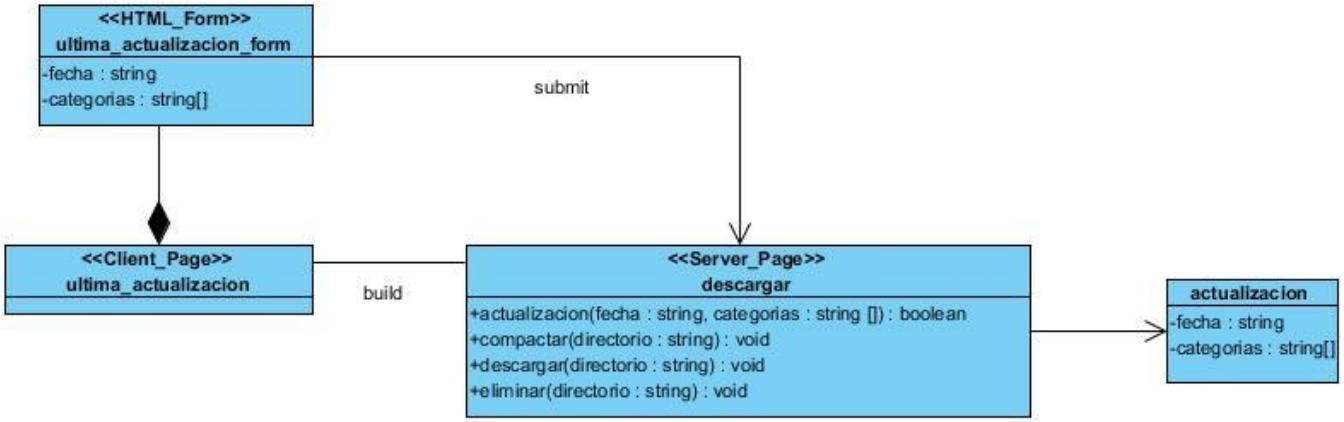


Figura 14: Diagrama de clases del diseño correspondiente al caso de uso Descargar última actualización

Anexo C. Diagramas de secuencia

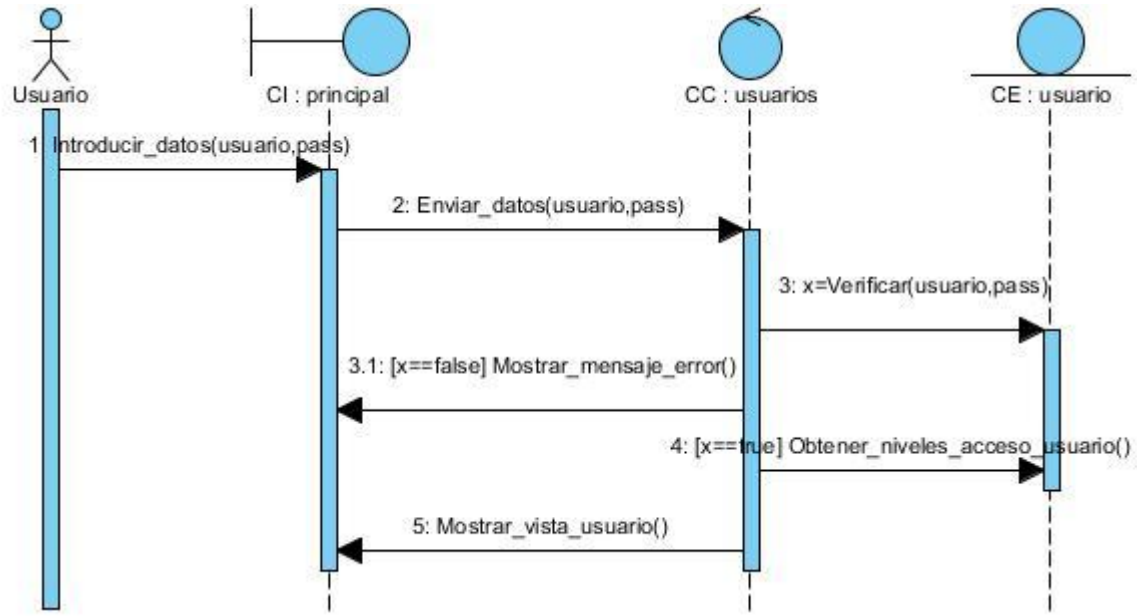


Figura 15: Diagrama de secuencia correspondiente al caso de uso Autenticar usuario

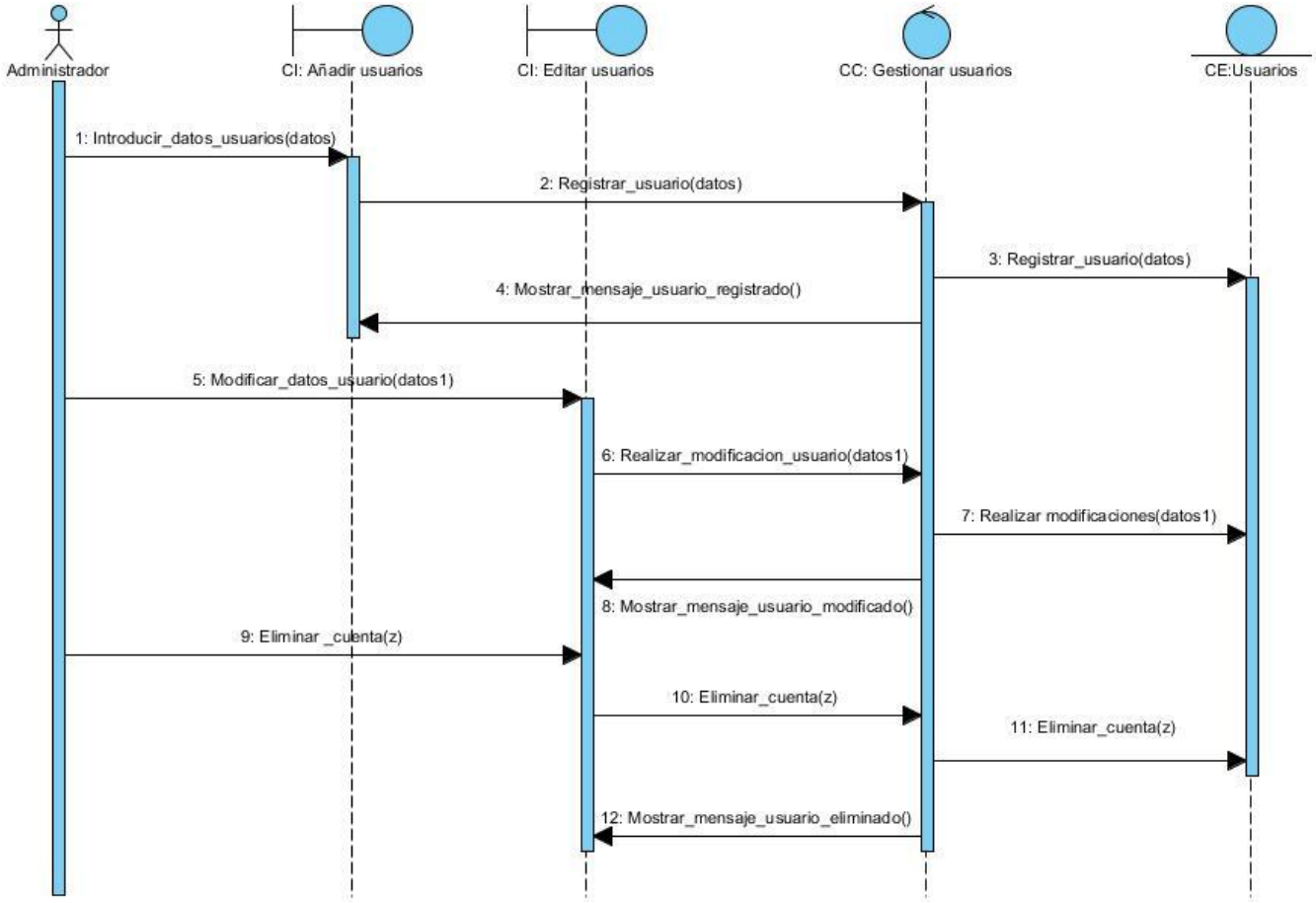


Figura 16: Diagrama de secuencia correspondiente al caso de uso Gestionar usuarios

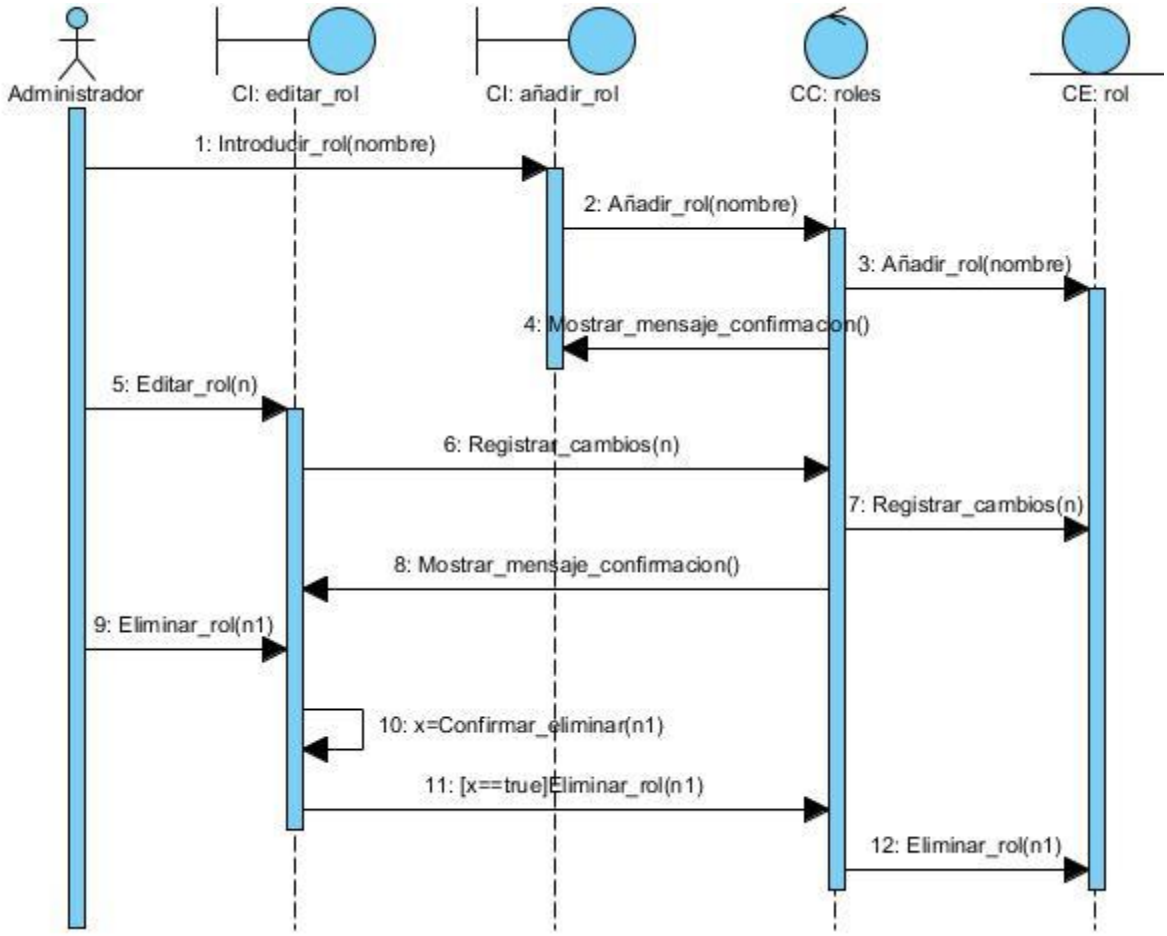


Figura 17: Diagrama de secuencia correspondiente al caso de uso Gestionar roles

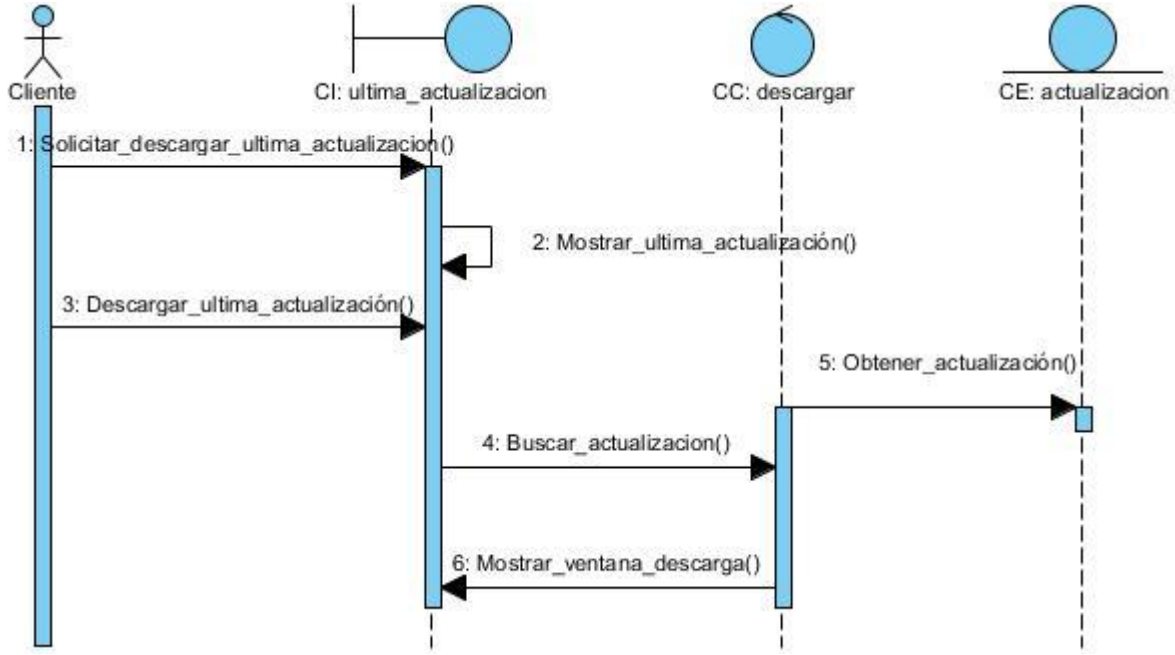


Figura 4: Diagrama de secuencia correspondiente al caso de uso Descargar última actualización

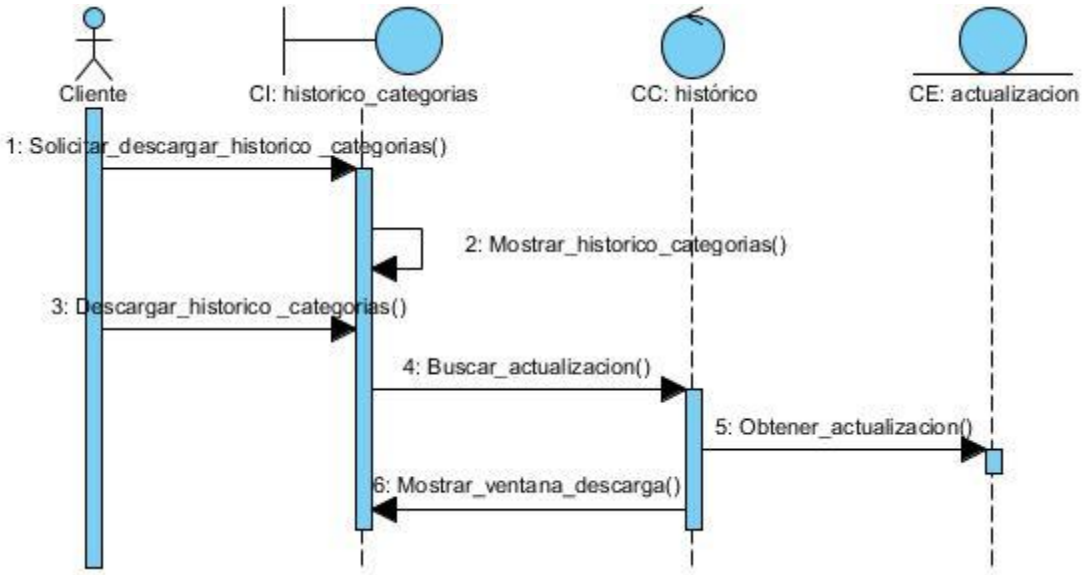


Figura 5: Diagrama de secuencia correspondiente al caso de uso Descargar histórico de categorías

Anexo D. Casos de prueba

Tabla 13: Casos de pruebas correspondientes al caso de uso Descargar última actualización

CU Descargar última actualización		
Entrada	**Resultados**	**Condiciones**
Descargar actualización satisfactoriamente.	El sistema crea una actualización según las opciones seleccionadas y permite al usuario descargar dicha actualización.	Deben ser seleccionadas las categorías o la actualización completa y estas deben existir.
Descargar actualizaciones que no puedan ser localizadas.	El sistema valida si no existen las actualizaciones y obvia la selección para crear las actualizaciones.	
Solicitar descargar sin seleccionar una actualización.	El sistema muestra una página de error indicando que no se ha seleccionado ninguna actualización.	

Tabla 14: Casos de pruebas correspondientes al caso de uso Descargar histórico categorías

CU Descargar histórico categorías		
Entrada	**Resultados**	**Condiciones**
Descargar actualización satisfactoriamente.	El sistema crea una actualización según las opciones seleccionadas, con solo un fichero por cada categoría seleccionada y permite al usuario descargar dicha actualización.	Deben ser seleccionadas las categorías o la actualización a descargar y estas deben existir.
Descargar actualizaciones que no puedan ser localizadas.	El sistema valida si no existen las actualizaciones y obvia la selección para crear las actualizaciones.	
Solicitar descargar sin seleccionar una actualización.	El sistema muestra una página de error indicando que no se ha seleccionado ninguna actualización.	

Tabla 15: Casos de pruebas correspondientes al caso de uso Gestionar roles

CU Gestionar roles		
Escenario Añadir rol		
Entrada	**Resultados**	**Condiciones**
Añadir un rol satisfactoriamente.	El sistema crea el nuevo rol.	Se debe ingresar el nombre del rol.
Añadir rol con campos vacíos.	El sistema muestra un mensaje “Debe indicar un nombre de rol válido.”.	Deben existir campos vacíos.
Escenario Editar rol		
Editar rol satisfactoriamente.	El sistema modifica el nombre del rol seleccionado.	Se deben llenar todos los campos requeridos.
Editar rol con campos vacíos.	El sistema muestra un mensaje “El campo Nombre del rol es obligatorio”.	Deben existir campos vacíos.
Escenario Eliminar rol		
Eliminar rol satisfactoriamente	El sistema elimina el rol satisfactoriamente y muestra el mensaje “Se ha eliminado el rol”.	Se debe confirmar la eliminación.
Cancelar la confirmación de eliminar rol.	El sistema cancela la eliminación del rol.	

Tabla 16: Casos de pruebas correspondientes al caso de uso Gestionar usuarios

CU Gestionar usuarios		
Escenario Añadir usuario		
Entrada	**Resultados**	**Condiciones**
Añadir un usuario satisfactoriamente.	El sistema crea el nuevo usuario.	Se deben llenar todos los campos requeridos.
Añadir usuarios con campos vacíos.	El sistema muestra un mensaje “Existen campos vacíos”.	Deben existir campos vacíos.
Escenario Editar usuario		
Editar usuario satisfactoriamente.	El sistema modifica los datos del usuario y muestra el mensaje "Se han guardado los cambios".	Se deben llenar todos los campos requeridos.
Editar usuario con campos vacíos.	El sistema muestra un mensaje “Existen campos vacíos”.	Deben existir campos vacíos.

Escenario Editar usuario		
Editar usuario satisfactoriamente.	El sistema modifica los datos del usuario seleccionado.	Se deben llenar todos los campos requeridos.
Editar usuario con campos vacíos.	El sistema muestra un mensaje “El campo Nombre del rol es obligatorio”.	Deben existir campos vacíos.
Escenario Eliminar usuario		
Eliminar usuario satisfactoriamente.	El sistema elimina el usuario satisfactoriamente y muestra el mensaje "Se ha eliminado el usuario".	Confirma la opción de cancelar cuenta.
Cancelar opción eliminar usuario.	El sistema cancela la opción de eliminar el usuario.	

Tabla 17: Casos de pruebas correspondientes al caso de uso Gestionar permisos

CU Gestionar permisos		
Entrada	**Resultados**	**Condiciones**
Asignar permiso satisfactoriamente.	El sistema asigna el permiso a un rol determinado y muestra el mensaje “Se han guardado los cambios”.	Se deben seleccionar nuevos permisos.
Eliminar permiso.	El sistema elimina el permiso a un rol determinado y muestra el mensaje “Se han guardado los cambios”.	