Universidad de las Ciencias Informáticas

Facultad 1

Título: Módulo para registrar acciones de administradores en la Interfaz Web de Smart Keeper

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

Autoras: Yaisy María Montero Martínez
Llanela Suárez Artiles

Tutores: Ing. Dovier Antonio Ripoll Méndez
Ing. Yurisleidy Hernández Moya

La Habana, 22 de junio de 2012

"Año 54 de la Revolución"

Declaración de autoría

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

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

Autora:	Autora:
Yaisy María Montero Martínez	Llanela Suárez Artiles
Tutor:	Tutora:
Ing. Dovier Antonio Ripoll Méndez	Ing. Yurisleidy Hernández Moya

Resumen

En sistemas que cuentan con interfaces web de administración y almacenan información en bases de datos, comúnmente resulta de interés registrar las acciones que realizan los administradores sobre las entidades persistentes de dichas bases de datos, para contribuir a posteriores análisis sobre el comportamiento de tales usuarios. En este trabajo se presenta el desarrollo de un módulo para registrar acciones (adicionar, eliminar y modificar) que realizan administradores a través de la Interfaz Web (IW) y sobre entidades persistentes de la base de datos. El componente fue diseñado en forma de plugin para Smart Keeper: filtro cubano de contenido web que surgió en la Universidad de las Ciencias Informáticas (UCI). Para el desarrollo, que fue guiado por la metodología Proceso Unificado de Rational (RUP, por sus siglas en Inglés), se empleó el *framework* symfony y se utilizaron algunos patrones y estándares. A partir de su incorporación a Smart Keeper, al módulo se le realizaron diversas pruebas (de aceptación, funcionalidad e integración), las cuales permitieron corregir errores y mejorar el funcionamiento del mismo.

PALABRAS CLAVE: administradores, base de datos, entidades persistentes, Interfaz de Administración Web, registro, Smart Keeper.

Índice general

Introducción

Internet ha alcanzado un papel preponderante en la industria de la información y es actualmente una de las herramientas comunicacionales de mayor repercusión para la sociedad [1]. Su desarrollo ha devenido en la disponibilidad de recursos educativos, científicos, informativos, artísticos, recreativos; sin embargo, también ha facilitado la proliferación de materiales que distorsionan valores éticos y morales, tales como: los relacionados con el terrorismo, la violencia, la pornografía infantil, entre otros.

Muchas instituciones brindan servicio de Internet a sus trabajadores, con el objetivo de facilitar la búsqueda de información u otras actividades relacionadas con el contenido de trabajo.Sin embargo, en tales instituciones los trabajadores no siempre hacen un uso adecuado de dicho servicio pues, por ejemplo, acceden durante la jornada laboral a contenidos que carecen de interés para la institución.

Varios gobiernos han creado iniciativas de leyes, intencionadas a regular la publicación y el acceso a la información en la Gran Telaraña Mundial (WWW, por sus siglas en Inglés) [2]. De igual manera y en correspondencia con las normas gubernamentales vigentes del lugar, las instituciones u organizaciones que ofrecen servicio de Internet establecen Políticas de Uso Aceptable (AUP, por sus siglas en Inglés) [3]. El empleo de los sistemas de filtrado de contenidos es una vía práctica para poner en funcionamiento tales políticas. Estos sistemas se integran por varios elementos de software que operan juntos para regular (permitir o denegar) el acceso de los usuarios a determinados materiales que se encuentran en Internet; además, contribuye al cumplimiento de ciertas reglas definidas por la organización y configuradas previamente en el sistema por sus administradores [4].

En la UCI se desarrolló un software de filtrado de contenidos, el cual fue nombrado Smart Keeper y diseñado para su instalación en un servidor dedicado[1]. Este sistema cuenta con una Interfaz de Administración Web (IAW), que facilita a los administradores la configuración y el uso de funcionalidades.

Para el correcto funcionamiento de Smart Keeper resulta determinante la acción de sus administradores ya que éstos, de manera intencionada o no, podrían configurarlo y utilizarlo inadecuadamente. Lo ante-

[1] Es un servidor que ofrece sólo un servicio determinado, configurando el hardware y software para un mayor rendimiento en el servicio en cuestión.

rior cobra mayor importancia si se considera que pueden existir varios administradores, lo cual dificulta conocer las acciones específicas que realizan cada uno de ellos. A continuación se explicitan algunas de las acciones que, realizadas por un administrador, provocan un funcionamiento inadecuado del filtro:

- Cambiar la categoría asociada a un Localizador Uniforme de Recursos (URL, por sus siglas en Inglés). Esta situación puede provocar una incorrecta navegación de todos los usuarios con relación a la URL modificada. Si a una página con material inadecuado (pornografía infantil, violencia, terrorismo, etc.) se le asigna incorrectamente una categoría adecuada, provocará que todos los usuarios tengan acceso a la misma.
- Modificar los permisos de navegación relacionados a grupos de usuarios. Esta situación implicará que determinados usuarios tengan acceso a sitios que normalmente no pueden ver o en caso contrario, denegarle el acceso a contenidos que sí pueden ver.
- Asignar cuotas de navegación sin previa autorización de los directivos de la entidad. Esta situación conducirá a una distribución inadecuada de dicho recurso.

Cualquiera de las acciones anteriores podría implicar un impacto negativo en las instituciones que empleen el filtro Smart Keeper, tales como: incumplimiento de las AUP establecidas en la institución, uso incorrecto de las Tecnologías de la Información y las Comunicaciones (TIC) e incluso un aumento de costos por concepto del uso de Internet. Además, no contar con un registro de las acciones de los administradores, entre otras desventajas, provoca: (a) que los directivos de la institución no puedan tomar las medidas adecuadas en caso de ajustes/usos incorrectos y (b) que los propios administradores sientan ”libertad”para realizar determinadas acciones negativas.

Actualmente, en la IAW de Smart Keeper no es posible obtener toda la información de las actividades que realizan los administradores, porque el sistema no posee un mecanismo para ello. Además, para analizar los datos que sí se registran, se necesitan conocimientos avanzados del funcionamiento interno del sistema.

De la problemática anterior se deriva el siguiente **problema a resolver:** ¿Cómo contribuir al registro de las acciones de los administradores en la IAW de Smart Keeper?

Una vez definido el problema a cuya solución contribuye este Trabajo de Diploma, se identificó como

objeto de estudio: Proceso de registro de acciones de usuarios en aplicaciones web. El **campo de acción** está enfocado en el registro de acciones de administradores en interfaces web de sistemas de filtrado.

El **objetivo general** del presente Trabajo de Diploma es: desarrollar un módulo para el registro de acciones de los administradores en la IW de Smart Keeper. Este objetivo general se articula, a su vez, en los siguientes **objetivos específicos:**

- Sistematizar sobre el marco teórico conceptual de las principales herramientas desarrolladas para el registro de las acciones de los administradores en aplicaciones web.
- Seleccionar la metodología y las tecnologías a emplear en el desarrollo del módulo para el registro de las acciones de los administradores en la IW de Smart Keeper.
- Diseñar el módulo para el registro de las acciones de los administradores en la IW de Smart Keeper.
- Implementar el módulo para el registro de las acciones de los administradores en la IW de Smart Keeper.
- Probar el correcto funcionamiento del módulo desarrollado, así como su integración a Smart Keeper.

Para abordar el problema, en correspondencia con el objetivo trazado, se aplicaron los siguientes métodos de la investigación científica:

Métodos teóricos:

El Analítico-Sintético se aplicó para entender el proceso de registro de las modificaciones sobre las entidades persistentes en interfaces web de administración, a partir del análisis de sus características y objetivos. Además, para formular conclusiones a través de la síntesis de los conocimientos y los resultados obtenidos.

Métodos empíricos:

El Análisis de Documentos se aplicó para revisar la literatura especializada y extraer la información necesaria que luego permitió realizar el proceso de investigación.

Estructura del documento

El presente documento está organizado en cuatro capítulos, tal y como se describen a continuación:

En el **primer capítulo "Fundamentación teórica"**: se abordan temas relacionados con la actividad de los administradores en aplicaciones web. En él se analizan elementos útiles para diseñar el módulo para el registro de las acciones de los administradores en la IW de Smart Keeper. Además, se identifican las tecnologías y herramientas a utilizar, así como la metodología para guiar el desarrollo del módulo.

En el **segundo capítulo "Características del sistema"**: se profundiza en la propuesta de solución, a partir del Modelo del Dominio y la descripción de las clases de dicho modelo. Más adelante se explicita la modelación del módulo a través de: requisitos funcionales, requisitos no funcionales y Modelo de Casos de Uso.

En el **tercer capítulo "Análisis y diseño del sistema"**: se define la arquitectura del módulo con la elección adecuada de los patrones de diseño. Además se exponen las ideas relacionadas con la validación de los datos, el tratamiento de errores y la seguridad, basándose en los mecanismos que provee el *framework* symfony. También se describen algunos de los artefactos que son generados durante el diseño de la solución.

En el **cuarto capítulo "Implementación y prueba"**: se aborda la implementación y prueba de la solución propuesta, tomando como base los resultados que se presentan en capítulos anteriores. En él se muestra el Diagrama de Componentes del módulo y las vistas principales de la aplicación, así como los resultados de las principales pruebas.

Capítulo 1

Fundamentación teórica

Introducción

En el presente capítulo se abordan las acciones que frecuentemente desempeñan los administradores de una aplicación web. Se estudian aplicaciones web que proporcionan registros de modificaciones sobre entidades persistentes. Además, se analizan y proponen las tecnologías, las herramientas y la metodología de desarrollo que luego se utilizarán en el diseño y la implementación de la solución.

1.1. Acciones de los administradores

Una aplicación web es un software basado en tecnologías y estándares del Consorcio de la Gran Telaraña Mundial (W3C, por sus siglas en Inglés) que provee recursos específicos, tales como: contenidos y servicios a través de una Interfaz de Usuario (IU) a la que puede accederse mediante un navegador web [5]. Según las funcionalidades que brinda, una aplicación web puede clasificarse como orientada a transacciones si permite a los usuarios cierta interactividad. Dichas aplicaciones tienen como requisito el uso de bases de datos para el almacenamiento de información y la posibilidad de realizar consultas sobre las mismas de forma eficiente y consistente [6].

Entre las aplicaciones web que se orientan a transacciones, se encuentran las especializadas en administración. En dichas aplicaciones se utiliza el rol de administrador para nombrar a la(s) persona(s) encargada(s) de administrar el sistema. Las labores típicas de un administrador de una aplicación web suelen ser diversas y variar según el propósito de la aplicación web. Generalmente, algunas de las funciones que desempeñan son: administrar los usuarios y las contraseñas, realizar configuraciones y administrar permisos.

Mantener un registro de las acciones de los administradores puede ayudar a satisfacer requisitos legales

que se le impongan a las organizaciones y demostrar que estas mantienen control estricto sobre la labor de los administradores. El control de las acciones de un administrador sobre una aplicación permite verificar el cumplimiento de las políticas institucionales establecidas, lo que constituye un elemento de gran importancia en la toma de decisiones administrativas.

1.2. Filtros de contenidos

Varios software emplean una IW para facilitar las tareas de administración. En determinados sistemas (como los filtros de contenidos) resulta interesante llevar un registro de las acciones de los administradores. Tales filtros controlan el acceso de usuarios a recursos que se ubican en Internet, a partir de ciertos parámetros que se configuran con anterioridad.

1.2.1. Smart Keeper

La versión 2.2 de Smart Keeper posee una IAW que facilita las tareas de administración. Dicho sistema se basa en el uso e integración del servidor proxy Squid-Cache, el servidor web Apache2 y el servidor de base de datos PostgreSQL. Además, fue desarrollado usando la versión 1.2 del *framework* symfony y sus funcionalidades se distribuyeron en tres subsistemas: Almacenamiento, IAW y Filtrado. Dentro de sus características fundamentales se destacan [7]:

- Desarrollado con tecnologías libres.
- Efectividad en el filtrado: ofrece una aceptable efectividad en el proceso de filtrado.
- Reporte de actividades: ofrece informes acerca del intento de acceso a contenidos inadecuados, los cuales son valiosos para conocer el uso de Internet que hacen los usuarios y para asistir a la toma de decisiones.
- Basado en servidores: la instalación del sistema se produce en un servidor, lo cual es ventajoso para grandes redes de usuarios con acceso a Internet.
- Listas personalizables de contenidos categorizados: de manera fácil e intuitiva el administrador puede adicionar, eliminar, actualizar y verificar la existencia de los contenidos almacenados en la

base de datos. Funcionalidad significativa teniendo en cuenta que, en la mayoría de los casos, estas listas representan secretos comerciales para los proveedores de filtros de contenidos.

- Administración web: a partir de una amena e intuitiva Interfaz de Administración, los administradores del sistema pueden configurarlo/adaptarlo.
- Seguridad: la IAW (principal punto de interacción con los usuarios) está protegida contra disímiles ataques (prevención de XSS, SQL injection, Session hijacking, Session fixation, etc.).
- Uso y control de cuotas de Internet: la utilización de un sistema de cuotas (con diferenciación por usuarios), proporciona a la institución un mecanismo para mejorar el aprovechamiento del ancho de banda de Internet.

Smart Keeper utiliza los mecanismos del *framework* symfony para registrar la actividad de los usuarios; estos registros pueden efectuarse con las peticiones, las respuestas y las consultas a la base de datos. En cambio, las acciones de los administradores en la IAW no son almacenadas. En dicho filtro se considera como administrador a un usuario que, a través de la IAW, tenga permisos para consultar o modificar alguna entidad persistente en la base de datos. Con el objetivo de conocer las acciones específicas que realizan cada uno de los administradores, resulta importante registrar estas acciones que se realizan sobre las entidades persistentes en la base de datos. Esta característica debe añadirse a Smart Keeper, sin realizar modificaciones significativas en su código fuente original.

En correspondencia con lo anterior, las autoras consideran conveniente la realización de un estudio que permita identificar técnicas, algoritmos y procedimientos que puedan aplicarse a un módulo para el registro de las acciones de los administradores en la IW de Smart Keeper.

1.3. Registro de modificaciones sobre entidades persistentes

Para registrar las acciones de los administradores (a través de una IAW) sobre las entidades persistentes en una base de datos, resulta de interés estudiar sistemas que almacenan este tipo de información; así como los mecanismos utilizados por los gestores de base de datos, el sistema de eventos del *framework* symfony y sistemas de filtrado.

1.3.1. Aplicaciones web

Entre los sistemas web que registran las modificaciones en las entidades persistentes por los administradores a través de una interfaz, se encuentran el *plugin* ncPropelChangeLogBehaviorPlugin y el sistema Administrador de Concurso en Línea Bombonera (BOCA, por sus siglas en Inglés).

El *plugin* **ncPropelChangeLogBehaviorPlugin**, desarrollado para la versión 1.1.12 de symfony, posee licencia Instituto Tecnológico de Massachusetts (MIT, por sus siglas en Inglés) [8]. Dicho *plugin* registra: el identificador de los objetos que se modifican, el usuario que realiza la modificación; además, en codificación Base 64[1], almacena la información del objeto sobre el que se realizó la operación. A pesar de las características que posee, no cuenta con otros elementos que son de interés para Smart Keeper:

- No almacena la dirección IP desde donde se hizo la modificación, por lo que en sistemas con varios clientes no es posible conocer cuál de ellos realizó la modificación.
- Su arquitectura actual no es compatible para integrarse a Smart Keeper, pues los *plugins* desarrollados para este sistema, poseen una estructura muy específica.
- Cuando se edita una entidad no se guarda la información del objeto original; es decir, solo se almacena la información del objeto nuevo. Por otra parte, las acciones realizadas por los administradores se identifican con números, de modo que se dificulta saber su tipo (adicionar, editar o eliminar).

Pese a que el *plugin* ncPropelChangeLogBehaviorPlugin tiene algunas limitantes, podría ser empleado como parte de la solución; no obstante, será requerido modificar el código fuente para incorporar elementos que son necesarios en Smart Keeper.

El sistema **BOCA**, desarrollado para ser usado en el Maratón de Programación de la Sociedad de Computación de Brasil, está desarrollado con PHP Pre-procesador de Hypertexto (PHP, por sus siglas en Inglés) y utiliza a PostgreSQL como sistema gestor de bases de datos. Se caracteriza por contar con una IW sencilla, posibilitando que las acciones sobre la misma se realicen de manera rápida y eficiente. Dicho software de gestión de concursos cuenta con varios módulos que facilitan el trabajo de los usuarios

[1] La codificación Base 64 se basa en el uso de los caracteres US-ASCII (no acentuados) para codificar información mediante un código de 8 bits.

(concursantes, administradores y jueces), entre los que destacan: Runs, Score, Clarifications, Users, Problems, Languages y Logs.

Específicamente, el módulo Logs es el encargado de llevar el control de todas las acciones que realizan los distintos usuarios en el sistema; de cada acción se registra: la hora en que se realizó, el identificador del usuario que la efectúo, la dirección IP y una breve descripción de la misma. Para registrar una acción el software utiliza una función que se invoca cada vez que una acción se realiza, obligando a incluir esta función en el código fuente del programa.

El sistema BOCA no se adecua a los intereses del grupo de desarrollo de Smart Keeper porque, al utilizar una técnica similar, se incumpliría el principio de no modificar el código fuente original.

1.3.2. *Trigger* en gestores de base de datos

Existen investigaciones que expresan la posibilidad de registrar las acciones de los administradores a partir del propio gestor de base de datos; es decir, empleando características y funcionalidades que poseen los gestores para registrar una modificación en algún campo determinado. Entre tales trabajos se encuentran:

- En la referencia [9] se informa cómo habilitar la opción de revisar todos los cambios ocurridos en una tabla de una base de datos SQL Server. Este gestor de base de datos posibilita habilitar la opción *traceo*, la cual permite registrar de forma fácil las operaciones de inserción, eliminación y modificación en las tablas seleccionadas. Además, se explica que los sistemas gestores de bases de datos, como SQL Server o PostgreSQL, que permiten la utilización de disparadores (*trigger*) pueden almacenar registros de cambios realizados. Estos disparadores se ejecutan automáticamente luego de una operación de insertar, editar o eliminar. De esta forma se puede almacenar un registro de cambios de una base de datos.
- En la referencia [10] se evidencian los principales resultados de LogManager, desarrollado del lado del servidor; dicho producto emplea un mecanismo para auditar los sistemas de base de datos más conocidos, tales como: InterBase, Firebird, Microsoft SQL Server 2000 o superiores (Advantage Database Server y NexusDB v2). Este producto funciona con bases de datos complejas con

un gran número de tablas; además, posee una interfaz gráfica de usuario que permite, de forma sencilla, revisar todas las operaciones de las tablas y campos. En su funcionamiento emplea los disparadores de la base de datos.

Estos estudios están enfocados fundamentalmente a gestores que usan disparadores, lo cual puede constituir una desventaja para Smart Keeper (pues lo hace dependiente de este tipo de software). Si se crea esta dependencia se rompería con el principio de poseer una capa de abstracción del gestor de base de datos.

1.3.3. Symfony, sistema de eventos

Partiendo del hecho que Smart Keeper utiliza symfony, resulta oportuno realizar un estudio acerca del sistema de eventos que maneja dicho *framework*. Los comportamientos y métodos que posee el Mapeo Objeto-Relacional (ORM, por sus siglas en Inglés) Propel, utilizado por symfony para el trabajo con eventos, pueden aportar elementos para un rápido desarrollo del módulo.

Para el *framework* symfony, un evento es una acción que se ejecuta y captura por un componente. Algunas de las clases de symfony notifican los eventos en varios momentos de su ejecución. Cuando se produce un evento una aplicación puede realizar cierto proceso, como por ejemplo: cuando un usuario modifica alguno de sus atributos se notifica el evento correspondiente a ese atributo, guardando el mismo en una base de datos. Dentro de los elementos que almacenan se encuentran: el identificador (que es una cadena de texto) y su nombre. El registro de los eventos permite saber cuáles se deben ejecutar cuando se notifique un evento; es decir, se puede identificar el tipo de acción que se ejecutó (ya sea de eliminación o modificación) [11].

Los comportamientos o *Behaviors* son clases extras que proporcionan métodos a las clases del modelo. Symfony no incluye por defecto ningún comportamiento; estos se introducen a las clases mediante una sola línea de código. Entre los comportamientos que utiliza Propel se encuentran [12]:

preInsert: se ejecuta antes de la inserción de un nuevo objeto.

postInsert: se ejecuta después de la inserción de un nuevo objeto.

preUpdate: se ejecuta antes de actualizar (editar) un objeto existente.

postUpdate: se ejecuta después de actualizar (editar) un objeto existente.

preSave: se ejecuta antes de salvar un objeto existente o nuevo.

postSave: se ejecuta después de salvar un objeto existente o nuevo.

preDelete: se ejecuta antes de eliminar un objeto.

postDelete: se ejecuta después de eliminar un objeto.

Luego de un análisis de los comportamientos de Propel y las facilidades que ofrece, se concluye que será utilizado para el desarrollo de la solución.

Considerando que Smart Keeper carece de un mecanismo para registrar las modificaciones (en las entidades persistentes) de los administradores, resulta conveniente estudiar otros sistemas de filtrado para conocer cómo implementan dicha funcionalidad.

1.3.4. Sistemas de filtrado

Según la referencia [13], los siguientes sistemas de filtrado han logrado un espacio importante si se consideran los aspectos funcional y comercial: SentryPC, Net Nanny Parental Controls y PureSight PC.

Los sistemas antes mencionados se enfocan en almacenar un registro de la actividad de los usuarios con el sistema de filtrado. Fundamentalmente se almacena información sobre los sitios que fueron bloqueados e información adicional sobre este evento (usuario, fecha, hora, ect.); además, mediante el empleo de parámetros predefinidos algunos permiten refinar la búsqueda de tales reportes.

La mayoría de estos sistemas de filtrado son privativos, por lo que generalmente la forma en que están implementados constituye un secreto comercial y no se publica mucha información acerca del funcionamiento interno de los mismos. Aunque típicamente estos sistemas brindan la posibilidad de registrar cada petición realizada por los usuarios, las autoras de este Trabajo de Diploma no encontraron referencias que permita afirmar que registran las modificaciones en las entidades persistentes por parte de los administradores.

En el ámbito nacional no se tiene referencia de otro filtro de contenidos además de Filpacon [7]. Este es una versión inferior de Smart Keeper, por lo cual no tiene la funcionalidad de registrar las modificaciones efectuadas por los administradores, sobre las entidades persistentes en la base de datos.

1.4. Tecnologías para desarrollar el sistema

La selección adecuada de las tecnologías para desarrollar el software, sin dudas, contribuye al aumento de la calidad de la solución final. Dado que Smart Keeper utiliza como base el sistema operativo Debian GNU/Linux v6.4, se seleccionan tecnologías que actualmente son compatibles con dicho sistema operativo. El análisis de las características particulares del sistema y las ventajas que lograría el uso de cada herramienta son elementos considerados en la selección.

1.4.1. *Framework*

Los *frameworks* facilitan el desarrollo de software, proporcionando una estructura definida que ayuda a crear aplicaciones con mayor rapidez. Brindan a los programadores y diseñadores una mejor organización y estructura de sus proyectos. Permiten definir la estructura global de un sistema teniendo en cuenta las clases y los objetos, sus responsabilidades clave y como colaboran entre sí, de modo que los diseñadores e implementadores puedan concentrarse en los elementos específicos del mismo [14].

En la referencia [6] se presenta un estudio encaminado a la selección adecuada de las tecnologías para el desarrollo de la IAW de Smart Keeper, incluyendo la propuesta de usar a symfony como *framework* para el desarrollo. El estudio antes mencionado se tuvo en cuenta para la elección, pues el módulo para el registro de acciones de los administradores en la IW de Smart Keeper será integrado en dicho sistema.

En definitiva, considerando fundamentalmente las razones contenidas en la referencia [6], se utilizará symfony V.1.2 como *framework* para desarrollar el módulo propuesto.

1.4.2. Herramienta de ORM

El ORM es una técnica de programación que permite convertir datos entre el lenguaje de programación orientado a objetos y el sistema de base de datos relacional [15]. Propel es un ORM para PHP v.5, que permite acceder a una base de datos utilizando un conjunto de objetos, proporcionado a su vez una interfaz para almacenar y recuperar datos. Además, provee las herramientas necesarias para trabajar con la base de datos, de la misma manera que se trabaja con otras clases y objetos en PHP [16].

Se empleará el ORM Propel, teniendo en cuenta los siguientes motivos fundamentales: (a) es el utilizado por el equipo de desarrollo de Smart Keeper; (b) está disponible para la versión 1.2 del *framework* symfony; (c) utiliza los estándares de PHP v.5; (d) es una herramienta de código abierto; (d) provee soporte para varios gestores de bases de datos (MySQL, PostgreSQL, entre otros); y (f) posee una comunidad amplia y activa que proporciona un apoyo rápido.

1.4.3. Lenguaje de desarrollo

La elección del lenguaje de programación se sustentó en las particularidades de la solución a desarrollar. Como el *framework* de desarrollo seleccionado es symfony y está desarrollado con PHP, implica que el desarrollo de la solución se realice utilizando dicho lenguaje. Además, el equipo de desarrollo posee experiencia en el empleo de ese lenguaje de programación.

Algunas de las ventajas de usar PHP son [17]:

- Es software libre.
- Puede interactuar con varios motores de bases de datos, tales como: MySQL, Oracle, Informix, Postgre y otros.
- Actualmente se puede ejecutar con Apache, IIS, AOLServer, Roxen y THTTPD.
- Funciona en varias plataformas utilizando el mismo código fuente, permitiendo ser compilado y ejecutado en aproximadamente 25 plataformas, incluyendo diferentes versiones de Unix, Windows (95, 98, NT, ME, 2000, XP) y Macs.

Como herramienta de modelado se seleccionó Visual Paradigm, por ser una alternativa factible en entornos libres.

1.4.4. Herramienta de desarrollo

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en Inglés) permite la automatización de tareas asociadas al desarrollo y posee funcionalidades como el autocompletamiento de código y la depuración [18]. En la selección de un IDE de desarrollo para PHP se consideró que fuera una herramienta libre. Aunque existen varias alternativas (gPHPEdit, Komodol, EclipsePDT, NetBeans, entre otras), se escogió NetBeans por la experiencia de trabajo con la misma, además de poseer características que lo enaltecen [18]:

- Es un producto libre y gratuito (sin restricciones de uso).
- Es multilenguaje; está conformado por módulos independientes los cuales, mediante la adición o eliminación de funcionalidades permiten que se modifique fácilmente.
- Incluye un aceptable completamiento de código, además de resultar sencillo y rápido el desarrollo con el mismo. Cuenta con una comunidad activa y es un producto estable.
- Permite la integración con subversión como herramienta para el control de versiones.

1.5. Metodología de desarrollo de software

Las metodologías de desarrollo se pueden definir como un conjunto de pasos y procedimientos para estructurar, planear y controlar el proceso de desarrollo del software. Su utilización permite identificar qué hacer, cómo y cuándo durante el planeamiento, desarrollo y mantenimiento de un proyecto [19].

RUP fue la metodología seleccionada, sobre todo, teniendo en cuenta los siguientes elementos:

- El estudio realizado en la referencia [6] demuestra que es la metodología adecuada para el desarrollo del sistema Smart Keeper. La documentación generada en el proceso de desarrollo del módulo

será compatible con los artefactos que documentan el desarrollo de la IAW del filtro.

- El equipo de desarrollo posee experiencia en el uso de dicha metodología.
- Utiliza el Lenguaje Unificado de Modelado (UML, por sus siglas en Inglés), lo cual favorece y se encuentra en concordancia con el desarrollo que se pretende realizar.

Conclusiones

A partir del estudio de la actividad de los administradores en aplicaciones web, se identificaron conceptos y funciones de los administradores que servirán de guía para la propuesta de solución. Se empleará el *plugin* ncPropelChangeLogBehaviorPlugin como parte de la solución, modificando elementos de su código para la creación del módulo para el registro de acciones de los administradores en IW de Smart Keeper. Debido a que el sistema Smart Keeper está completamente desarrollado usando el *framework* symfony, con PHP y el ORM Propel, además de las ventajas que estos poseen para el desarrollo de aplicaciones web, fueron seleccionados para el desarrollo de la solución, contribuyendo a garantizar un desarrollo homogéneo con el sistema Smart Kepeer.

Capítulo 2

Características del sistema

Introducción

Durante la fase de elaboración y con el objetivo de analizar las especificidades del problema antes de implementar la solución, la metodología RUP establece el levantamiento de requisitos y la realización del modelo de negocio o dominio. En el presente capítulo se aborda un poco más la problemática en torno a Smart Keeper, se realiza el modelado del dominio, se identifican los requisitos funcionales y los no funcionales, se describe la propuesta de solución y se detalla el diagrama de casos de uso del sistema.

2.1. Descripción del problema

Actualmente el sistema Smart Keeper registra la actividad de los usuarios, sobre todo la relacionada con las peticiones y las respuestas del software Squid, así como las consultas a la base de datos. Sin embargo, no es posible obtener un registro de las acciones que realizan los administradores desde la IAW. En el capítulo anterior se evidenció la necesidad de crear un módulo que se integre a la IAW y permita registrar las modificaciones efectuadas por los administradores sobre las entidades persistentes de la base de datos.

Las acciones que realizan los administradores pueden ser de tipos adicionar, editar y eliminar. Tales acciones pueden ser efectuadas en un momento dado y manipulan determinados datos relacionados con los siguientes módulos de Smart Keeper:

- Usuarios
- Grupos de administración
- Políticas de navegación

- Grupos de navegación
- Categorías de contenido
- URLs
- Programaciones de tiempo
- Excepciones
- Expresiones regulares
- Subredes
- Reglas de tipo MIME
- Reglas de chequeo de virus
- MIME y extensiones

2.2. Modelado del dominio

El modelo de dominio captura en el contexto del sistema los tipos más importantes de objetos. Estos objetos representan los eventos que suceden en el entorno donde trabaja el sistema. El modelo de dominio se describe mediante diagramas del UML (especialmente a través de diagramas de clases). Dichos modelos muestran las clases del dominio (incluyendo sus relaciones mediante asociaciones) a los clientes, usuarios, revisores y otros desarrolladores [20]. Para explicar gráficamente un modelo de dominio a través de diagramas del UML (paso esencial en un análisis orientado a objetos), el problema debe descomponerse en conceptos individuales. En estos diagramas se muestran:

- conceptos
- asociaciones de conceptos
- atributos de conceptos

Debido a la relativa simplicidad del entorno en el cual se desarrolla el sistema y el conocimiento que se posee acerca de su funcionamiento, no es necesario realizar un modelo de negocio para entender la problemática a resolver, siendo suficiente un modelo de dominio. La Figura 2.1 muestra el modelo de dominio, teniendo en cuenta lo planteado en la descripción del problema.

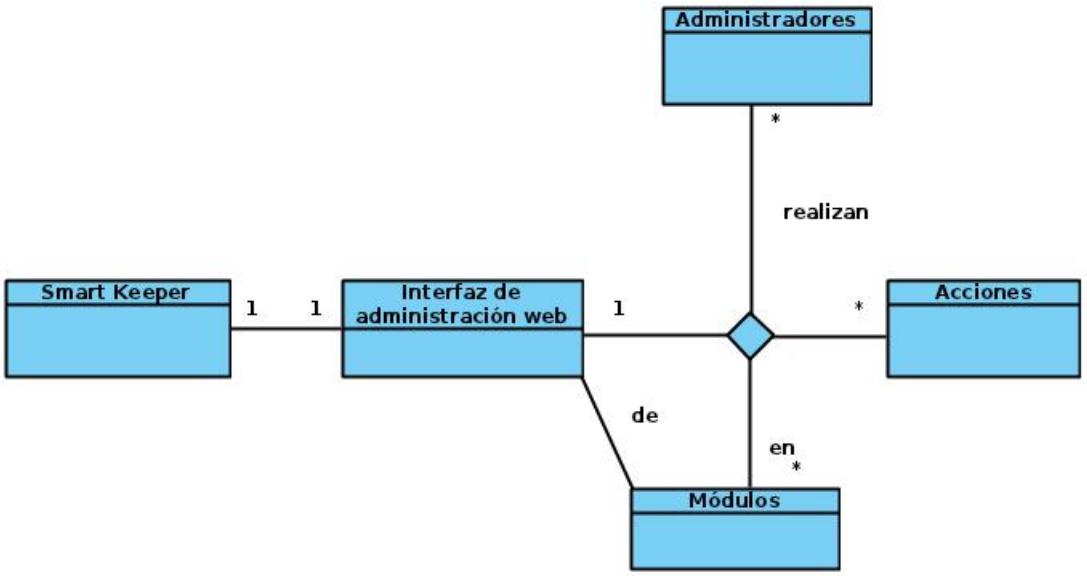

Figura 2.1: Modelo del dominio.

En los módulos de la IAW de Smart Keeper, los administradores realizan acciones, para configurar el comportamiento del sistema. A continuación se específica lo que representa cada una de las clases que componen el modelo de dominio:

Administradores: los usuarios administradores que interactúan con la IAW de Smart Kepeer.

Acciones: las acciones que los administradores ejecutan en la IAW.

Módulos: los módulos que componen la IAW del sistema Smart Keeper.

IAW: la interfaz con la que interactúa el administrador y mediante la cual se realizan las configuraciones para la administración del sistema Smart Keeper.

Smart Keeper: el filtro de contenidos web.

2.3. Solución propuesta

Se propone como solución: el desarrollo de un módulo que brindará la posibilidad de registrar las modificaciones efectuadas por los administradores, las cuales se efectuarán sobre las entidades persistentes de la base de datos y a través de la IAW de Smart Keeper. De los módulos que componen dicha IW de Smart Keeper y se agrupan en tres pestañas (Filtrado Básico, Filtrado Extendido y Reportes), se registrará la información de las modificaciones en los dos primeros grupos. No se almacenará ninguna información de las acciones realizadas en los módulos de Reportes del sistema, pues estos no alteran la información contenida en la base de datos.

La solución propuesta permitirá registrar los datos que se manipulan de las acciones, así como la información del administrador que la llevó a cabo y la hora en que se realizó la acción. Se desarrollará como un *plugin*, característica importante que presenta algunas ventajas significativas [21]:

Reusabilidad: posibilitará ser reutilizado por otro sistema para el registro de las modificaciones sobre las entidades persistentes a través de la IAW.

Encapsulación: permitirá ser utilizado sin conocimientos de su funcionamiento interno, abstrayendo a los desarrolladores de la especificación interna del mismo. Además, se evitará que componentes y piezas de software dentro de otro sistema se vean afectados por cambios en su diseño.

Unidad independiente de desarrollo: permitirá que el componente pueda ser desarrollado de manera independiente, cambiando el diseño o agregando nuevas funcionalidades, sin afectar a otro sistema que lo utilice.

Para la implementación de la arquitectura del módulo como un *plugin* se tendrá en cuenta que Smart Keeper define una forma específica de desarrollar los *plugins* que emplea, por lo que el módulo deberá adecuarse a esta estructura. La Figura 2.2 muestra la estructura de directorios a seguir; la misma es similar a la propuesta por el *framework* symfony con algunas especificaciones, las cuales se mencionan a continuación:

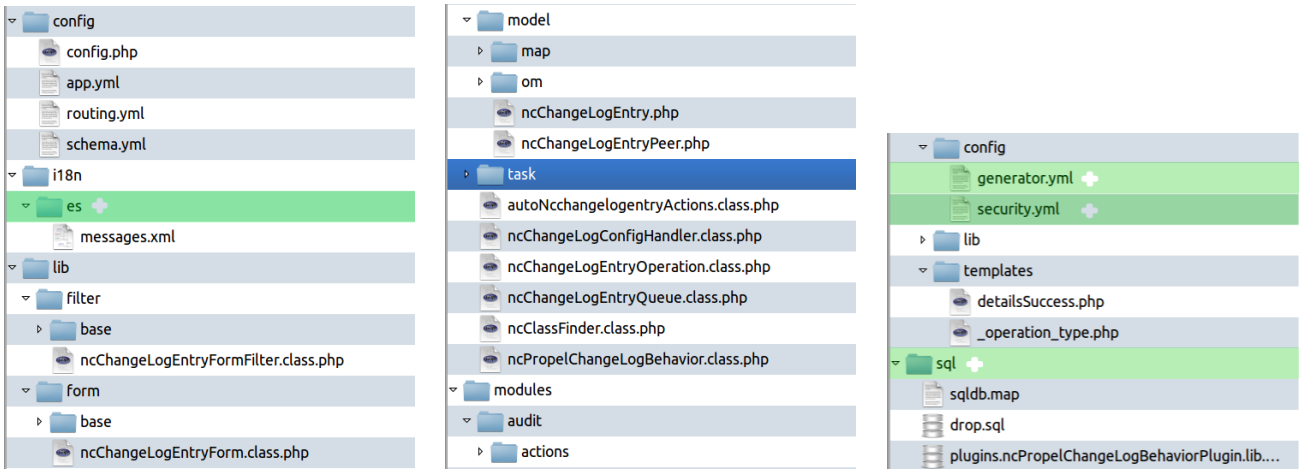

Figura 2.2: Estructura de directorio para desarrollar un *plugin* de Smart Keeper.

- Se elimina el directorio **web**, pues todos los elementos referentes a este se tomarán del proyecto de Smart Keeper (como es el caso del tema de la IAW).
- Como Smart Keeper construye todos sus módulos mediante la herramienta Admin-Generate de symfony, en el directorio **config** se deben adicionar los ficheros:
  - **generator.yml**, en este fichero se encuentran todas las configuraciones de visibilidad del *plugin*.
  - **security.yml**, en este fichero se encuentran las credenciales de acceso al módulo, este fichero es utilizado por el *plugin* de symfony fsGuardPlugin para gestionar la seguridad de la aplicación.
- Se añade el directorio **sql** que debe de contener los ficheros:
  - **sqldb.map**, indica al ORM Propel la ruta del esquema de la base de datos que genera el *plugin*.
  - **drop.yml**, indica los elementos de la base de datos que se eliminarán cuando se desinstale el *plugin*.
  - **schema.yml**, representa el esquema de la base de datos del *plugin*

.

- Se deben configurar todos los ficheros del directorio **config**, de los cuales se puede prescindir solo del **schema.yml**.
- En el directorio **i18n** se añade el directorio **es**, el cual contendrá un fichero llamado **messages.xml** que permite la internacionalización del módulo.

El módulo se desarrollará sin realizar modificaciones en la lógica de negocio de Smart Keeper o su código fuente. Se utilizarán los comportamientos del ORM Propel que permitirán capturar y registrar los eventos en la base de datos con la información de los mismos. La conexión entre el cliente y el servidor web Apache de la IAW se establecerá utilizando el Protocolo de Transferencia de Hipertexto Seguro (HTTPS, por sus siglas en Inglés) que implementa dicho servidor. Para establecer la conexión con la base de datos se utilizarán los componentes que provee symfony (ORM Propel).

Desde la IAW, en una pestaña con el nombre “Registros” (en el menú principal), se visualizarán los registros obtenidos. Al acceder al módulo se listarán los registros con los datos principales de las acciones de los administradores y se le proporcionará al usuario la posibilidad de paginar si así se requiere. De cada registro será posible visualizar sus detalles, mostrando todos los campos de este a través de la opción ”Detalles”. También se presentará un formulario que permitirá filtrar los resultados del listado de los registros.

2.4. Requisitos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir; no alteran la funcionalidad del producto, por lo que se mantienen invariables sin importar las propiedades o cualidades con que se relacionan [20].

La nomenclatura definida para los requisitos funcionales fue la siguiente: el prefijo RF, que significa Requisito Funcional, seguido del número del requisito. A continuación, para más detalles de la solución propuesta, se especifican los requisitos funcionales:

RF1: Capturar eventos del módulo Usuarios

RF2: Capturar eventos del módulo Grupos de administración

RF3: Capturar eventos del módulo Grupos de navegación

RF4: Capturar eventos del módulo Políticas

RF5: Capturar eventos del módulo URL

RF6: Capturar eventos del módulo Categorías

RF7: Capturar eventos del módulo Programaciones de tiempo

RF8: Capturar eventos del módulo Excepciones

RF9: Capturar eventos del módulo Expresiones regulares

RF10: Capturar eventos del módulo Subredes

RF11: Capturar eventos del módulo Restricciones de ficheros

RF12: Capturar eventos del módulo Chequeos de virus

RF13: Registrar eventos capturado

RF14: Listar registro de acciones de los administradores

RF15: Filtrar resultados del listado de acciones

RF16: Mostrar detalles de acción registrada

2.5. Requisitos no funcionales

Los requisitos no funcionales no se refieren directamente a las funciones específicas, sino a propiedades del sistema [22].

Usabilidad

Los requisitos no funcionales referentes a la usabilidad versan sobre la base de lo manejable o manipulable que el sistema deberá constituir para el usuario final.

- Utilizar un patrón de navegación que permita estructurar y hacer intuitiva el acceso a las funcionalidades del sistema.

Disponibilidad

- El sistema debe estar disponible las 24 horas del día, durante cada uno de los 7 días de la semana.

Seguridad

La seguridad es un tema de suma importancia para cualquier software. En el caso que se presenta, donde se desarrollará un módulo para el registro de acciones de administradores, para un software ya existente, se trabajará con datos que representan evidencias de la información gestionada por los administradores, los cuales por tanto deben tener un mecanismo de seguridad.

- Debe ser accedido a través del protocolo HTTPS.
- Debe contar con mecanismos de protección contra ataques *Cross-site scripting* (XSS).
- Debe contar con mecanismos de protección contra ataques *Cross Site Request Forguery* (CSRF).
- Debe contar con mecanismos de protección contra ataques SQL.

Restricciones del diseño e implementación

- Se seguirán los estándares de codificación utilizados por las bibliotecas y el *frameworks* symfony.
- Se utilizará PHP v.5.2.4 o superior para la implementación.
- Deben utilizarse herramientas de desarrollo libre.
- Se utilizará symfony V.1.2.12 como versión del *framework* symfony.

Software

- Con el fin de acceder al módulo para el registro de acciones de los administradores en la IW de Smart Keeper, debe usarse una versión del navegador Mozilla Firefox igual o superior a la 1.5. No se garantiza la correcta visualización de la misma en otros navegadores.
- Con vistas a acceder al módulo para el registro de acciones de los administradores en la IW de Smart Keeper, el navegador debe tener habilitado el soporte para Cookies y Javascript.

Hardware

- El hardware donde se instale el sistema debe poseer al menos una interfaz de red, cuya velocidad de transferencia iguale o supere los 100 Mbps.

2.6. Definición de los casos de uso del sistema

Los casos de uso son la base para la implementación de las fases y disciplinas del RUP. Una de las características de esta metodología consiste en ser guiada por casos de uso, de ahí la importancia de definirlos correctamente.

2.6.1. Definición de los actores

Un actor es un conjunto coherente de roles que los usuarios de casos de uso desempeñan cuando interaccionan con los casos de uso [23]. La tabla 2.1 muestra la definición de los actores del sistema, (Administrador y ORM Propel), así como sus descripciones.

Actor	Descripción
Administrador	Puede acceder a las funcionalidades para consultar y obtener los registros de las acciones de los administradores.
ORM Propel	Captura los eventos cuando los administradores realizan alguna configuración en la IAW.

Tabla 2.1: Definición de los actores.

2.7. Listado de casos de uso del sistema

Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Especifica una secuencia de acciones, incluyendo variantes, que el sistema debe llevar a cabo, y que producen un resultado observable para un actor concreto [24]. Las tablas 2.2, 2.3, 2.4, 2.5 y 2.6 muestran el listado de los casos de uso del sistema.

CU1	Capturar evento
Actor	ORM Propel
Descripción	El caso de uso inicia cuando el ORM Propel recibe un evento de tipo: adicionar, eliminar o editar.

Tabla 2.2: Definición del caso de uso Capturar evento.

CU2	Registrar evento capturado
Actor	ORM Propel
Descripción	El caso de uso inicia cuando en el ORM Propel lanza el evento de tipo: adicionar, eliminar o editar.

Tabla 2.3: Definición del caso de uso Registrar evento capturado.

CU3	Listar registro de acciones de los administradores
Actor	Administrador
Descripción	El caso de uso inicia cuando el administrador entra al módulo de registro.

Tabla 2.4: Definición del caso de uso Listar registro de acciones de los administradores.

CU4	Filtrar resultados del listado de acciones
Actor	Administrador
Descripción	El caso de uso inicia cuando el administrador selecciona la acción filtrar.

Tabla 2.5: Definición del caso de uso Filtrar resultados del listado de acciones.

CU5	Mostrar detalles de acción registrada
Actor	Administrador
Descripción	El caso de uso inicia cuando el administrador selecciona la acción detalles.

Tabla 2.6: Definición del caso de uso Mostrar detalles de acción registrada.

2.7.1. Diagrama de casos de uso del sistema

A continuación (en la Figura 2.3) se muestran las funcionalidades del módulo a través de un diagrama de casos de uso que refleja las interacciones entre los actores y los casos de uso.

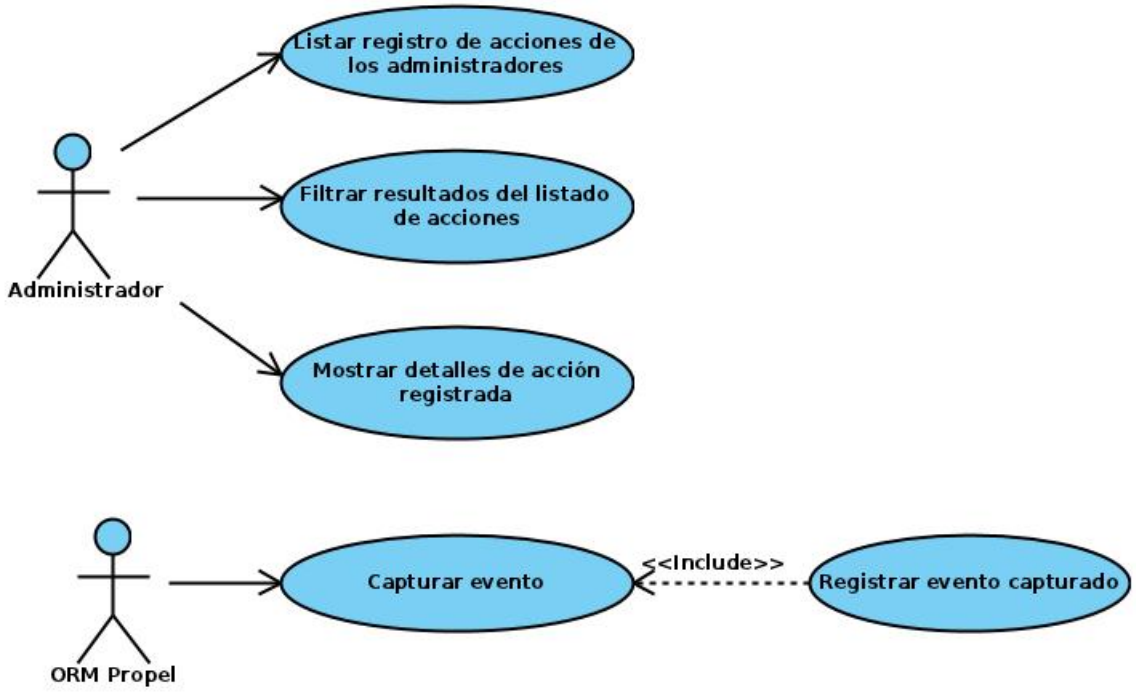

Figura 2.3: Diagrama de casos de usos del sistema.

El diagrama muestra al actor administrador inicializando los casos de uso: Listar registro de acciones de los administradores, Filtrar los resultados del listado de acciones y Mostrar detalles de acción registrada. Además, modela al actor ORM Propel inicializado el caso de uso Capturar evento; dicho caso de uso tiene una relación de inclusión con el caso de uso Registrar evento capturado.

2.8. Descripción expandida de los casos de uso del sistema

Las tablas 2.7 y 2.8 contienen la descripción expandida de los casos de uso críticos del sistema. La descripción de los restantes casos de uso se encuentran en el Anexo A.

	Capturar evento
Objetivo	El objetivo que persigue el actor es capturar los eventos o acciones de tipo: adicionar, editar o eliminar.
	Continúa en la próxima página

Actores	ORM Propel
Resumen	El caso de uso inicia cuando el ORM Propel recibe un evento de tipo: adicionar, eliminar o editar.
Complejidad	Alta
Prioridad	Crítico
Referencia	RF1, RF2, RF3, RF4, RF5, RF6, RF7, RF8, RF9, RF10, RF11, RF12
Precondiciones	Usuario autenticado en la IAW debe haber ejecutado alguna de las acciones: adicionar, editar o eliminar.
Postcondiciones	La información de los eventos es capturada por el ORM Propel.
Flujo de eventos	
Flujo básico: Capturar evento	

Num	**Actor**	**Sistema**
1.	Recibe un evento de tipo: adicionar, eliminar o editar.	1.1 Se lanza el evento del tipo recibido para ser utilizado por otro componente del sistema. 1.2 Se realiza la acción adicionar, eliminar o editar.

Tabla 2.7: Descripción del caso de uso Capturar evento

	Registrar evento capturado
Objetivo	El objetivo que persigue el actor es registrar toda la información del evento.
Actores	ORM Propel

Continúa en la próxima página

Resumen	El caso de uso inicia cuando el ORM Propel lanza el evento de tipo: adicionar, eliminar o editar.
Complejidad	Alta
Prioridad	Crítico
Referencia	RF13
Precondiciones	Usuario haya ejecutado alguna acción de tipo adicionar, eliminar o editar en la IAW.
Postcondiciones	Datos del evento registrados en la base de dato.
Flujo de eventos	
Flujo básico: Registrar evento capturado	

Num	**Actor**	**Sistema**
1.	Recibe el evento: adicionar o eliminar.	1.1 Registra el identificador, acción realizada, fecha con hora y los datos del objeto al que se le realizó la acción según su tipo.
2.	Recibe el evento: editar.	2.1 Registra identificador del objeto, identificador del usuario, tipo de acción realizada, fecha con hora y los datos del objeto antes de editarlo y del objeto editado según su tipo.

Tabla 2.8: Descripción del caso de uso Registrar evento capturado.

Conclusiones

La obtención de los requisitos funcionales y no funcionales permitió identificar las capacidades y cualidades que debe tener el módulo para el registro de acciones de los administradores en la IW de Smart Keeper, las cuales guiarán de forma correcta el proceso de desarrollo de la solución propuesta. La descripción expandida de los casos de uso posibilitó establecer los detalles relacionados con las respuestas del sistema a las acciones del actor. La definición de las características del módulo de registro de acciones de administradores en la IAW de Smart Keeper constituye el punto de partida para efectuar el Análisis y Diseño del módulo.

Capítulo 3

Análisis y diseño del sistema

Introducción

Durante la fase de elaboración, el flujo de trabajo de análisis y diseño de una aplicación alcanza mayor relevancia. En este capítulo se define la arquitectura del módulo a desarrollar, seleccionando los patrones de diseños adecuados. Además, se describen los artefactos generados en este flujo de trabajo: diagramas de clases, diagramas de interacción y diagrama de despliegue del módulo para el registro de acciones de los administradores en la IW de Smart Keeper.

3.1. Diseño

El modelo de diseño no es más que un modelo de objetos que describe cómo será la realización física de los casos de uso. Se basa en hacer notar el impacto que tendrán los requisitos funcionales y no funcionales en el sistema. El diseño cobra relevancia al final de la fase de elaboración y al comienzo de las iteraciones de construcción.

3.2. Definición de la arquitectura

La arquitectura de software es una vista del sistema que incluye los principales componentes del mismo, la conducta de esos componentes desde la visión del resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar los objetivos del mismo. La arquitectura de un sistema consiste en la vista conceptual de toda su estructura [25].

El uso del *framework* symfony que utiliza el Modelo Vista Controlador (MVC, por sus siglas en Inglés) obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el *framework* [11].

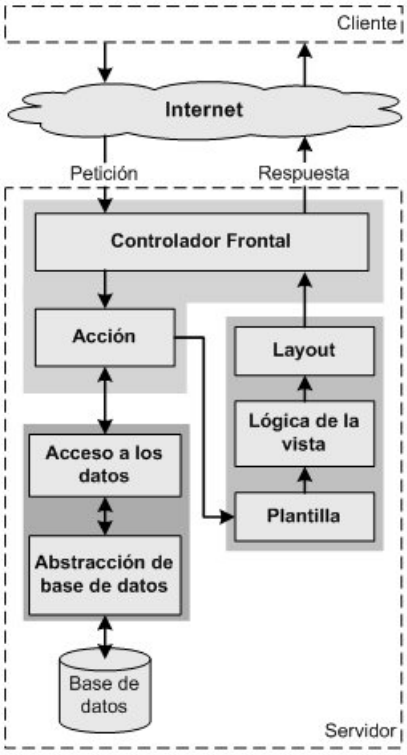

Figura 3.1: Esquema del MVC en symfony

Como se observa en la Figura 3.1 en esta variante el controlador que contiene el código que liga la lógica de negocio con la presentación, está compuesto por el controlador frontal que es el único punto de entrada a la aplicación y las acciones que se encargan de verificar la validez de las peticiones y preparar los datos para la vista. Esta última está formada por el *layout* que posee el código común a todas las acciones, las plantillas que contienen el código asociado a cada acción en particular y la lógica que se puede manipular a través de un fichero de configuración sencillo sin necesidad de programarla. Además, el modelo está formado por la capa de acceso a datos cuyas clases son generadas automáticamente por Propel en función de las estructura de datos de la aplicación y por la capa de abstracción de la base de datos *Creole*, que es completamente transparente al programador; así, si se cambia el sistema gestor de bases de datos no se debe reescribir el código, siendo necesario solamente modificar un parámetro en un archivo de configuración.

3.3. Organización de la aplicación

El código fuente en symfony es organizado en una estructura de tipo proyecto y los archivos del proyecto están estructurados en forma de árbol. Un proyecto se considera como “un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos”. Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones y cada aplicación está formada por uno o más módulos. Un módulo normalmente representa a una página web o un grupo de páginas con un propósito relacionado y almacenan las acciones que se pueden realizar en un módulo.

Para el diseño de la arquitectura del sistema se utilizó el patrón de Arquitectura Basada en Componentes. El mismo trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software.

En la Figura 3.2 se muestra el módulo para el registro de acciones de los administradores en la IW de Smart Keeper. Este se ha separado en componentes permitiendo una mejor operatividad sobre ellos para su configuración y se organizan en las capas que separa symfony. En la capa del modelo se destaca el componente ncPropelChangeLogBehaivor y el ORM Propel, este último encargado del acceso a datos.

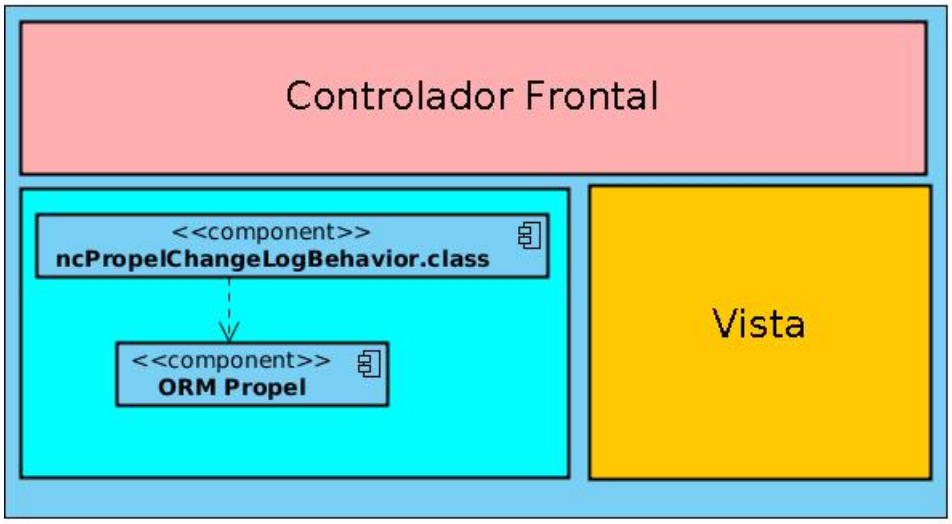

Figura 3.2: Vista de los componentes en capas.

3.4. Patrones de diseño empleados

Symfony sigue las mejores prácticas y patrones de diseño para el desarrollo de aplicaciones. Entre ellos los conocidos como Patrones de Diseño Orientado a Objetos (GoF, del Inglés Gang of Four). Algunos de los más significativos que se utilizan son [11]:

- Patrón Factory Method: las clases que conforman el núcleo están basadas en este patrón permitiendo la creación y extensión de los objetos de forma sencilla.
- Patrón Decorator: es usado para decorar el layout de la aplicación con el template asociado a cada acción, dando como resultado la vista que se muestra al usuario.
- Patrón Singleton: es usado para permitir el acceso desde cualquier lugar de la aplicación a los objetos relacionado con el núcleo del *framework*. Se evidencia en la clase sfRouting utilizada por el controlador frontal (sfWebFrontController) para enrutar todas las peticiones que se realizan a la aplicación.
- Patrón Command: se observa en la clase sfWebFrontController, en el método dispatch(). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario.
- Patrón Registry: se aplica en la clase sfConfig, que es la encargada de acumular todas las variables de uso global en el sistema.
- Front Controller: symfony posee una estructura bien organizada de controladores, que comienza desde el la clase index.php del ambiente y termina en los *Actions*. Cada clase de esta capa tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, y otros que se encargan de identificar mediante algunos datos las clases que deben realizar determinadas tareas (Patrón GoF Command, clase sfRouting) y las clases relacionadas con la configuración del sistema (sfConfig y sfConfigHandler).

También se utilizaron los Patrones de Software de Asignación de Responsabilidades Generales (GRASP, por sus siglas en Inglés) [11]:

- Experto: se utiliza cuando se trabaja con la inclusión de la biblioteca Propel para generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo). Estas clases poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla en la base de datos. En el módulo para el registro de acciones de los administradores en la IW de Smart Keeper también se evidencia en algunas clases como: RecordAction, RecordManagement, TprincipalPeer.

- Creador: se utiliza en la clase *Actions* donde se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase *Actions* es "creador" de dichas entidades. Ejemplos de algunas funciones utilizadas en la clase *Actions* son: doSelect(), retrieveByPK(), doSelectOne(). Además, se utiliza cuando el ORM Propel mapea la Base de Datos y crea las clases: BasencChangeLogEntry y BasencChangeLogEntryPeer. La clase BaseChangeLogEntryPeer es la encargada de comunicarse con al Base de Datos y devuelve un objeto de tipo BaseChangeLogEntry.

- Alta cohesión: symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase *Actions*, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las *properties*.

- Bajo Acoplamiento: la clase *Actions* hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

- Controlador: todas las peticiones web son manipuladas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Este patrón se evidencia en las clases sfFrontController, sfWebFrontController, sfContex, los *actions* y el index.php del ambiente.

Además de ellos se utilizan otros descritos por Martin Fowler en [26], entre los que están Row Data Gateway y Table Data Gateway. Son usados en las clases de acceso a datos y permiten realizar operaciones sobre un registro de una tabla y un conjunto de estos.

3.5. Validación y tratamiento de errores

La validación y el tratamiento de errores en el módulo se han diseñado en torno a los mecanismos que symfony provee. El *framework* brinda un mecanismo de validación en el servidor a través del uso de ficheros de validación y una serie de validadores que permite hacer este proceso más fácil y extensible, sin la consecuente promoción de errores que puede traer consigo la programación relacionada con este aspecto. Una característica importante de symfony es que permite el relleno automático de datos (repopulation), lo que asegura la obtención de datos correctos y mejora la experiencia de los usuarios. Para el tratamiento de excepciones se utilizará el mecanismo provisto por PHP y las clases que para este fin symfony posee.

3.5.1. Defensa contra ataques

A continuación se describen brevemente los principales ataques que podrían afectar la seguridad del módulo y por tanto la integridad de los datos que se manejan, así como los principios que regirán la prevención de los mismos en el contexto de symfony.

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

CSRF aunque parezca parecido, es contrario al XSS porque se basa en aprovechar la confianza que un sitio posee en un usuario para realizar peticiones, al mismo tiempo que puede comprometer su seguridad. Aunque en la versión que se utiliza no posee defensa contra el mismo, existe una extensión que permite añadirlo al *framework* como un filtro.

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

La mayor preocupación relacionada con el uso de sesiones está relacionada con el identificador de la sesión. Generalmente para obtener un identificador valido, un atacante puede utilizar tres variantes, la predicción, la captura y una tercera conocida en Inglés como *Session Fixation*. La primera y la segunda pueden ser mitigadas por el carácter altamente aleatorio que posee la generación de identificadores de sesión en PHP y por el uso de la Capa de *Sockets* Seguro (SSL, por sus siglas en Inglés) para la transmisión de datos entre cliente y servidor. Sin embargo, en la última el atacante puede lograr que la víctima use un identificador de sesión elegido por él, para así obtener un identificador valido. Como medio para prevenirlo no se utilizará la URL para enviar el identificador de sesión y se regenerar a uno nuevo para cada usuario una vez que se autentique en la aplicación. Tal como se mencionó anteriormente el uso de SSL para la transmisión de los datos contribuye a la integridad de los mismos y a la vez fortalece la defensa prevista contra los ataques anteriores. Por ello la aplicación utilizará el protocolo HTTPS para todas sus funcionalidades.

Otros ataques que pueden afectar la aplicación es la denegación de servicios (DoS) a partir del envió masivo de peticiones al servidor, pueden ser prevenidos a través de reglas en el cortafuegos que utiliza el sistema Smart Keeper y con las configuraciones del servidor web relacionadas con la cantidad de usuarios permitidos de forma concurrente, en relación con el consumo de la aplicación y los recursos que este posea.

La protección contra el acceso físico al servidor recae en la institución donde se utilice el módulo para el registro de acciones de los administradores en la IW de Smart Keeper y en los controles de seguridad que esta posea.

3.6. Diagrama de clases del diseño

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software. Presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones [27]. En la Figura 3.3 se modela el diagrama de clases del diseño con estereotipos web para los casos de uso: Capturar evento y Registrar evento capturado.

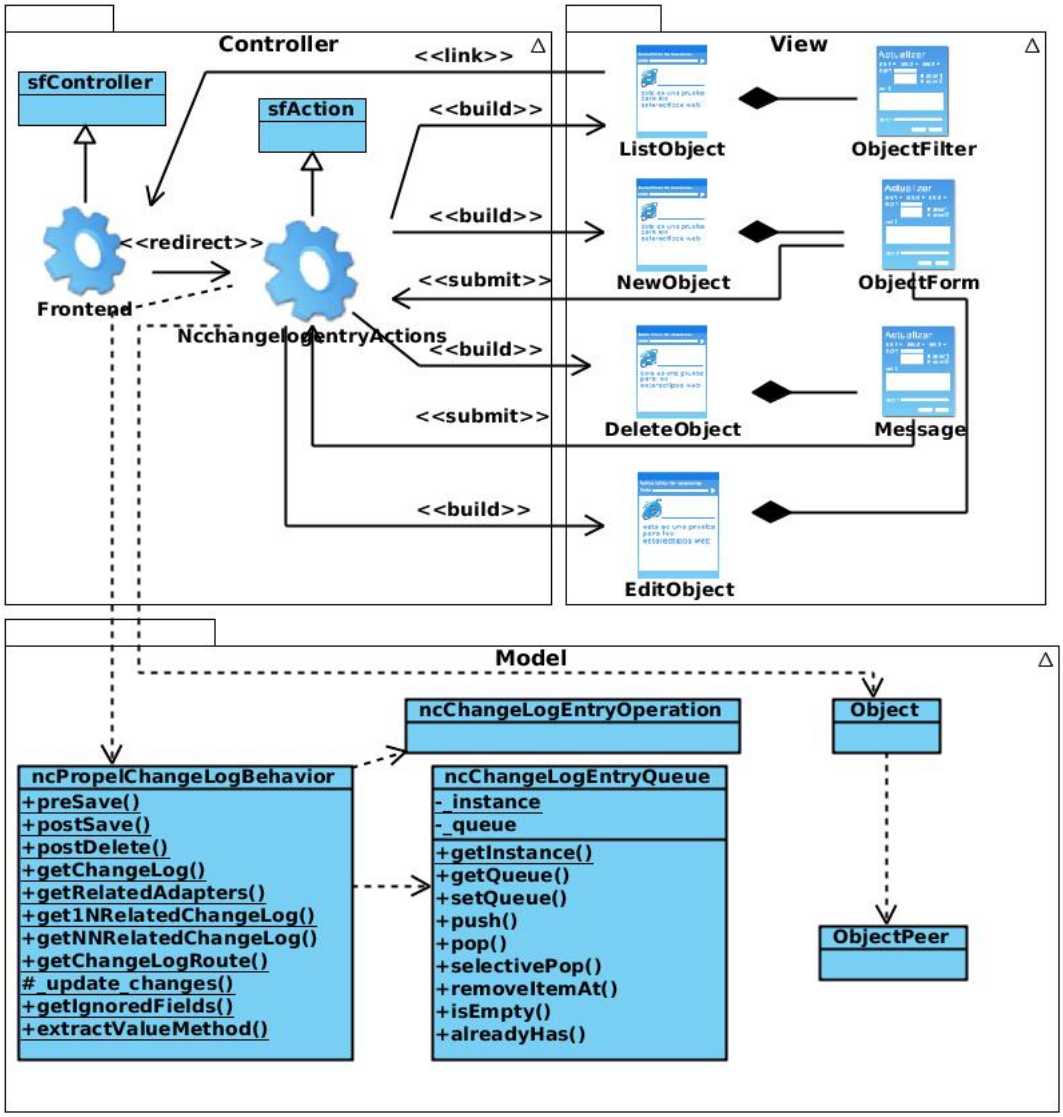

Figura 3.3: Diagrama de clases del diseño CU Capturar evento y CU Registrar evento capturado.

La clase **ListObject**, es una página cliente que muestra el listado de los objetos persistentes y tiene una relación de agregación por composición con la clase **ObjectFilter**. La clase **ObjectFilter** contiene la implementación del formulario para realizar el proceso de filtrado. Los criterios de filtrado son enviados a la clase controladora **NcchangelogentryActions**.

A través de un enlace en la clase **ListObject** se pueden realizar las acciones de adicionar, editar y eliminar. Este enlace ejecuta la clase controladora **Frontend**, la misma redirecciona para la clase **NcchangelogentryActions**. Si la opción seleccionada fue:

Adicionar, la clase **NcchangelogentryActions** construye una página cliente **(NewObject**). Ésta, se relaciona por agregación-composición con un formulario (**ObjectForm**) para adicionar los atributos del nuevo objeto. Los valores introducidos por los usuarios en el formulario, serán enviados a la clase **NcchangelogentryActions**. Dicha clase recibe la acción salvar. El flujo descrito anteriormente modela el proceso de capturar.En la clase **ncPropelChangeLogBehavior**, se ejecuta el método **preSave()** y **postSave()**. Dichos métodos representan el proceso de registrar.

Editar, se repite el mismo proceso que ocurre cuando se adiciona un objeto, descrito anteriormente, con la diferencia que el formulario se mostrará con los datos del objeto que en ese momento están almacenados en la base de datos.

Eliminar, la clase **NcchangelogentryActions** construye una página cliente (**DeleteObject**). Ésta, se relaciona por agregación-composición con un formulario (**Message**) para solicitar la confirmación de la acción. La respuesta determinada por los usuarios, será enviada a la clase **NcchangelogentryActions** mediante submit. Si se acepta la acción eliminar (esta respuesta es enviada por submit a la clase controladora **NcchangelogentryActions**). Esta clase se comunica con el ORM Propel, que lanza un evento una vez eliminado el objeto. Se ejecuta el método **postDelete()**. En caso de cancelar **NcchangelogentryActions** se contruye la página **ListObject**.

3.7. Diagramas de interacción

Los diagramas de secuencia y de colaboración (ambos llamados de interacción) se utilizan para modelar los aspectos dinámicos de un sistema. Muestran la interacción de un conjunto de objetos y sus relaciones, teniendo como valor añadido el hecho de mostrar los mensajes que se envían entre dichos objetos [27]. En el caso de los diagramas de secuencia se destaca la ordenación temporal de los mensajes; en el diagrama de colaboración se resalta la organización estructural de los objetos que envían y reciben mensajes.

A continuación en la Figura 3.4 se muestra el diagrama de interacción realizado para el caso de uso Capturar evento. En este diagrama se distingue una línea de vida que pertenece a cada objeto y que representa su existencia a lo largo de un período de tiempo. La mayoría de estos objetos perdurarán en

la medida que dure la interacción.

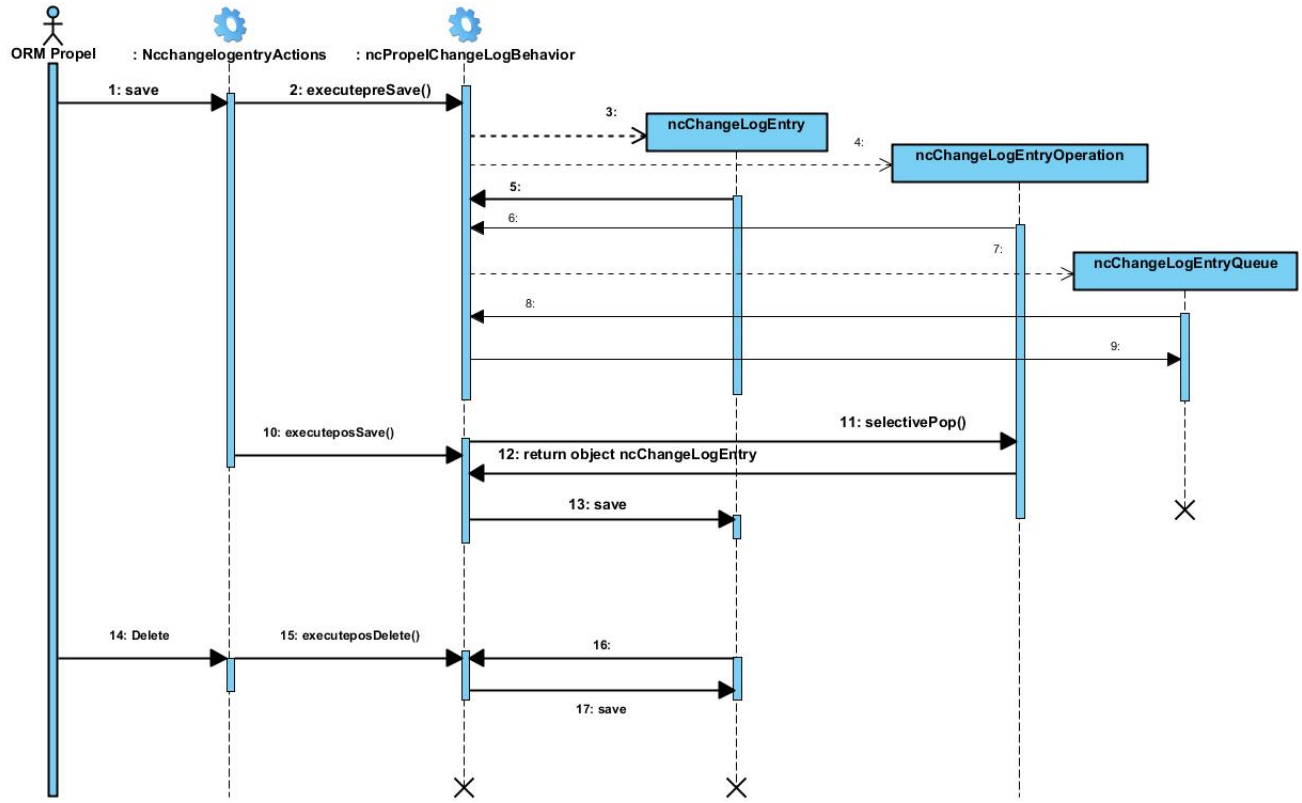

Figura 3.4: Diagrama de secuencia CU Capturar evento.

El ORM Propel recibe la acción salvar si la acción ejecutada fue adicionar o editar y la envía a la clase NcchangelogentryActions, la cual ejecuta el método preSave(). La clase ncPropelChangeLogBehavior crea un objeto de tipo ncChangeLogEntry con la fecha actual y el usuario que está ejecutando la acción. Luego comprueba si el objeto es nuevo, de ser así se actualiza el atributo tipo de acción, con el valor adicionar y el atributo descripción, con una codificación Base 64 que contiene la información del objeto a salvar. Si el objeto no es nuevo, se actualiza el atributo tipo de acción con el valor editar. Además, se identifican los valores modificados para actualizar el atributo descripción con una nueva codificación Base 64 y se adiciona el objeto de tipo ncChangeLogEntry en una cola. La clase NcchangelogentryActions ejecuta el método postSave() para extraer de la cola el objeto de tipo ncChangeLogEntry.

Los restantes diagramas de interacción se encuentran en el Anexo C.

3.8. Modelo de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación donde cada nodo puede contener instancias de componentes. En general un nodo puede ser una unidad de computación de algún tipo, desde un sensor hasta un *mainframe* y las instancias de componentes de software pueden estar unidas por relaciones de dependencia [28].

A continuación en la Figura 3.5 se muestra la vista de despliegue del módulo para el registro de acciones de los administradores en la IW de Smart Keeper.

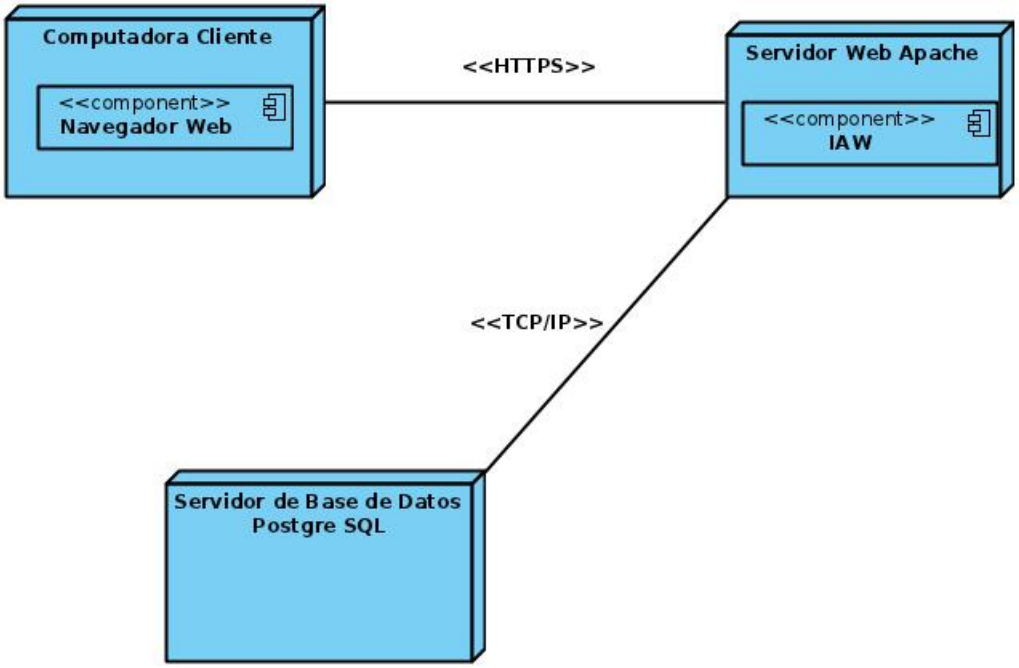

Figura 3.5: Diagrama de despliegue.

La vista de despliegue está conformada por un nodo **Computadora Cliente** que representa las estaciones de trabajo desde donde se conectan los clientes o usuarios a la IAW usando el protocolo HTTPS al nodo **Servidor Web Apache**. En dicho servidor se encuentra desplegada la aplicación web y se conecta usando el protocolo TCP/IP al **Servidor de Base de Datos**. En este nodo se encuentran almacenados todos los datos persistentes que necesita la aplicación para su funcionamiento.

El módulo puede ser instalado distribuido en la red. Interactúan 2 servicios y cada uno podrá estar en un nodo independiente o todo en un solo. Se usará la combinación que el usuario estime pertinente atendiendo principalmente a las prestaciones de hardware con que se cuente.

3.9. Modelo de datos

La base de datos que se utiliza en la IAW es un componente de Smart Keeper y en ella se almacenan los datos de los elementos manipulados por los administradores. Sólo se añadirá una nueva tabla para almacenar: el identificador, fecha y hora, módulo, objeto.

A continuación en la Figura 3.6 se muestra el diagrama entidad relación para una mejor comprensión del diseño de la base de datos.

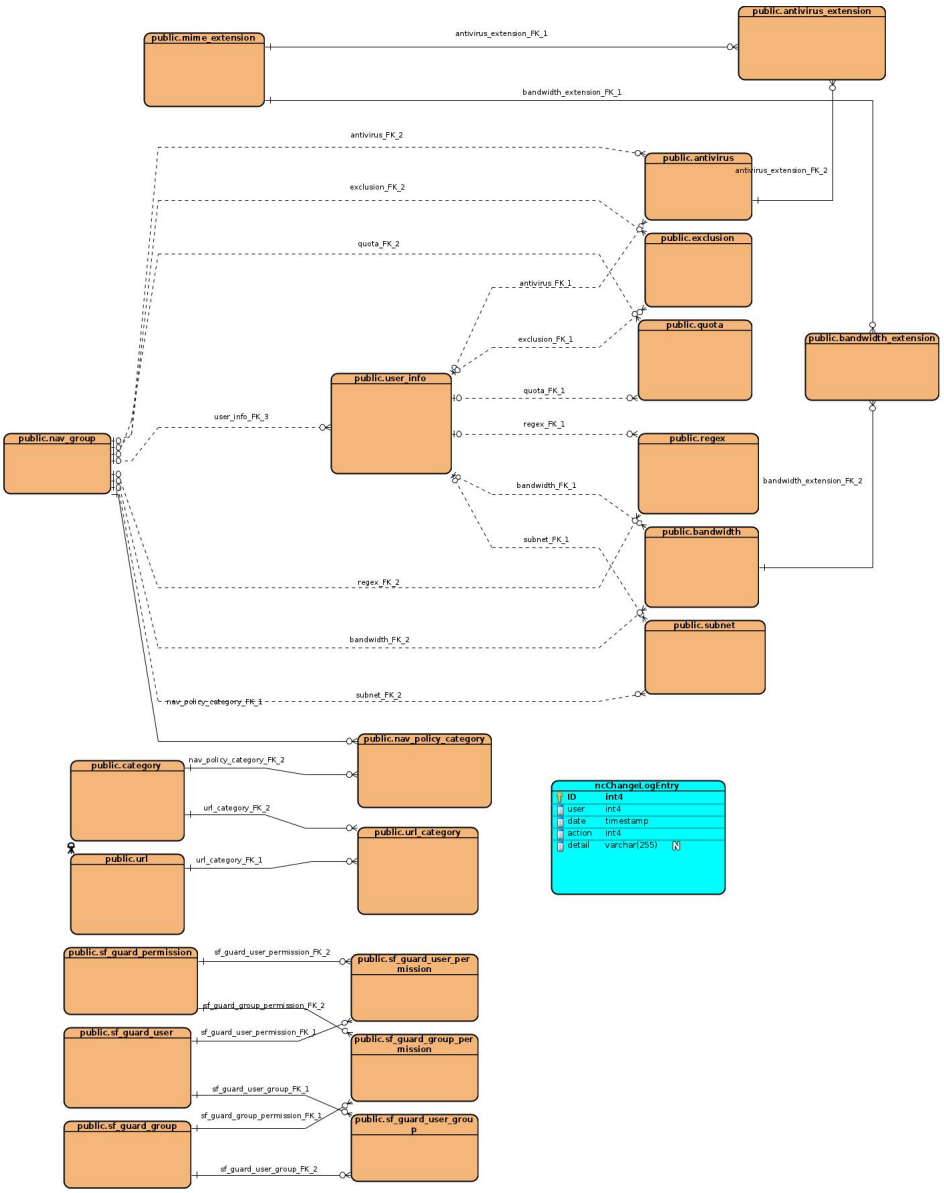

Figura 3.6: Diagrama entidad relación.

3.9.1. Descripción de la tabla ncChangeLogEntry

En la tabla 3.6 se describen los atributos y la tabla ncChangeLogEntry utilizada para .

Nombre: ncChangeLogEntry
Continúa en la próxima página

Descripción:		
Atributo:	**Tipo:**	**Descripción:**
User	varchar(24)	Es el identificador del usuario que realizó la acción.
Date	timestamp	Es la fecha y hora en que se realizó la modificación.
Action	integer	Es el tipo de acción que se realizó: adicionar, editar o eliminar.
Detail	varchar(200)	Son los datos de la entidad modificada

Tabla 3.1: Descripción de la tabla ncChangeLogEntry.

Conclusiones

La generación de los artefactos relacionados con el flujo de análisis y diseño, teniendo en cuenta la arquitectura MVC que propone symfony, permitió obtener una mayor comprensión de la aplicación y definir los principios que guiarán la implementación y organización de la misma. Mediante el modelado del diseño se obtuvo una abstracción de la implementación del sistema. Se concluye además, que todos los artefactos generados por la metodología en esta etapa guiarán de forma efectiva el desarrollo del módulo para el registro de acciones de los administradores en la Interfaz Web de Smart Keeper.

Capítulo 4

Implementación y prueba

Introducción

El flujo de trabajo de implementación y pruebas es el más trascendente en la fase de construcción del RUP. En él se construye el modelo necesario para desarrollar el proceso de implementación y se realizan las pruebas para detectar la existencia de errores en la implementación realizada. En este capítulo se muestran los diagramas de componentes del módulo y las pantallas principales de la aplicación. Además, se abordarán los temas relacionados con las pruebas aplicadas al módulo.

4.1. Diagrama de componentes

Los diagramas de componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes software que se encuentran unidos por relaciones de dependencia (compilación, ejecución). Los elementos de modelado que lo conforman son los componentes y paquetes que muestran la estructura del sistema en términos de implementación a un alto nivel [28].

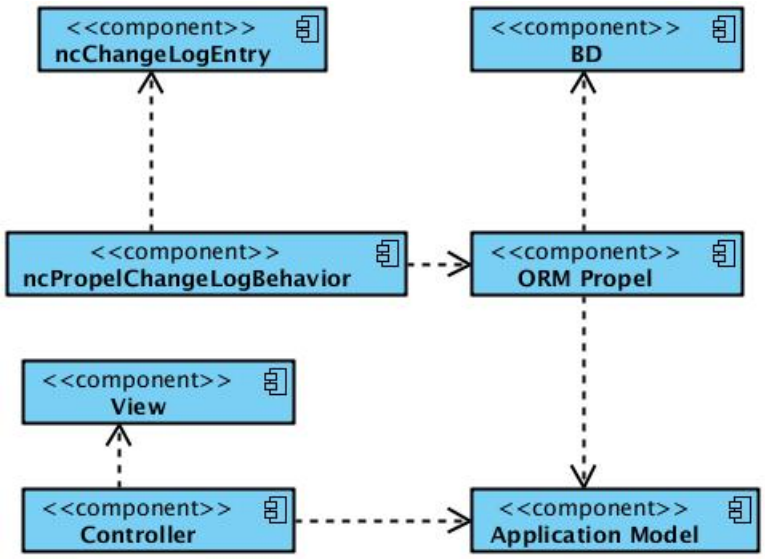

Figura 4.1: Diagrama de componentes.

En la figura 4.1 se muestra el diagrama de componentes del módulo para el registro de acciones de los administradores en la IW de Smart Keeper. El componente **Application Model**, representa la capa del modelo en la que está distribuida symfony. La capa controladora está modelada a través del componente **Controller** y la vista en el componente **View**. El **ORM Propel** representa la capa de abstracción de la base de datos lo cual permite un acceso a la base de datos independientemente del gestor que se utilice. Además, se observa el componente **BD**, encargado de almacenar la información que utiliza el sistema. A dicha base de datos se le añadirá una nueva tabla la cual almacenará los registros de las entidades modificadas. Los componente **ncChangeLogBehavior** y **ncChangeLogEntry** representan las clases principales del módulo que intervienen en el registro de acciones de los administradores en la IW de Smart Keeper.

4.2. Vistas de la aplicación

En las Figuras 4.2 se muestra una vista del módulo con el listado de los registros. En la columna **Nombre del módulo** se observa el nombre del módulo en el que fue realizada la acción. El campo **Usuario** representa el identificador del usuario que realizó la acción en la IAW, el **Tipo de operación** es el tipo de acción realizada en la IAW y la **Creado** representa la fecha y hora en que el usuario realizó la acción. En

la columna **Acciones**, a través de la opción **Detalle** se pueden acceder a los detalles de los objetos.

Además, en la esquina superior derecha se incluye un formulario con los campos para aplicar restricciones a los valores de las columnas del listado de los registros: **Nombre del módulo**, **Usuario**, **Tipo de operación** y **Creado**. Dicho formulario también ofrece las opciones **Filtrar** y **Restablecer**.

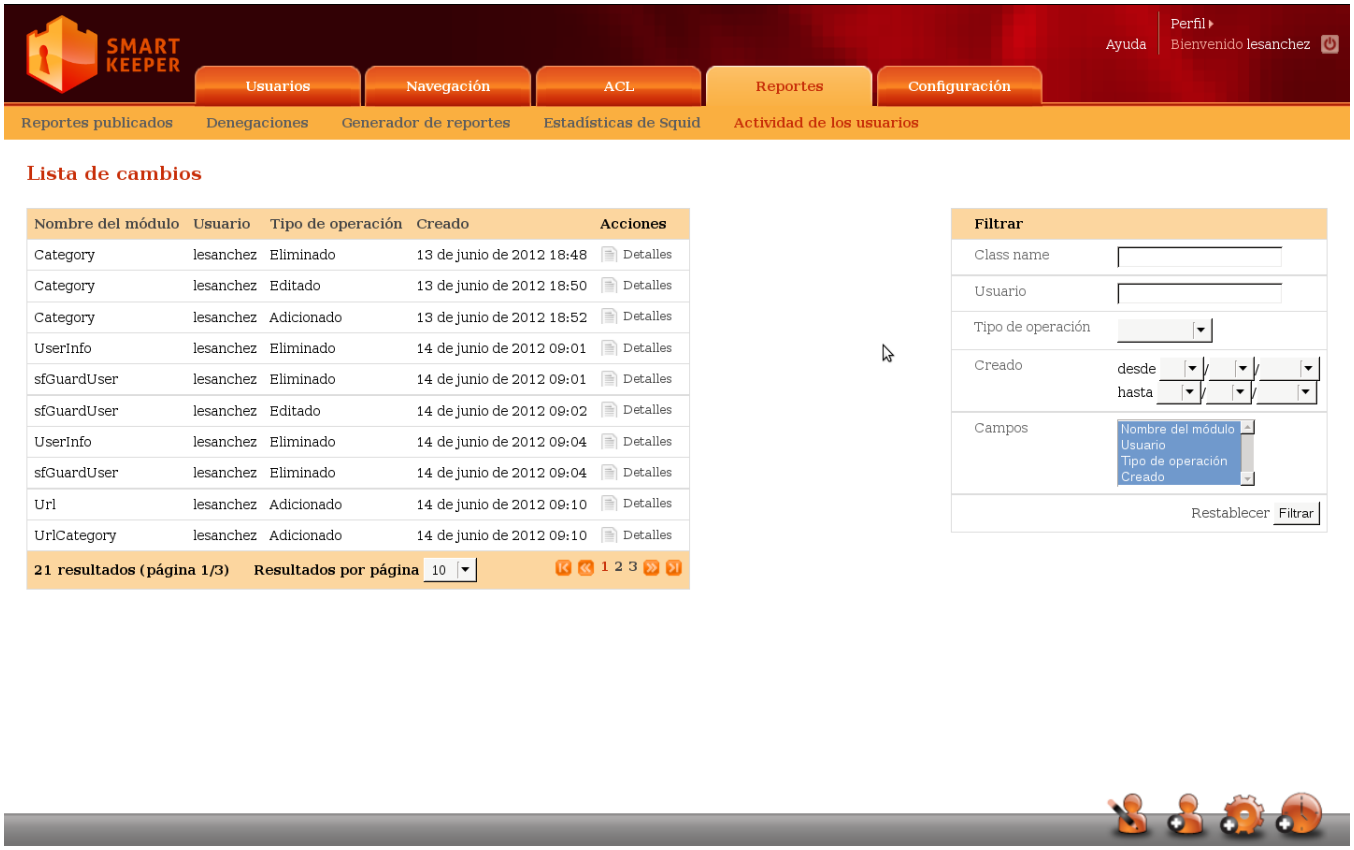

Figura 4.2: Vista1 de la aplicación listar registro de acciones de los administradores y filtrar resultados del listado de acciones.

La Figura 4.3 muestra la vista detalles del módulo, en la cual se visualizan los datos básicos y los datos del objeto. En lo sección que agrupa los **Datos básicos**, se presentan: el **Nombre del módulo**, el **Usuario**, el **Tipo de operación** y la **Fecha**. La sección **Datos del objeto**, contiene la fila **Id del objeto** que representa el identificador del objeto en la base de datos. Además muestra el campo **Detalles** que representa las modificación realizadas al objeto.

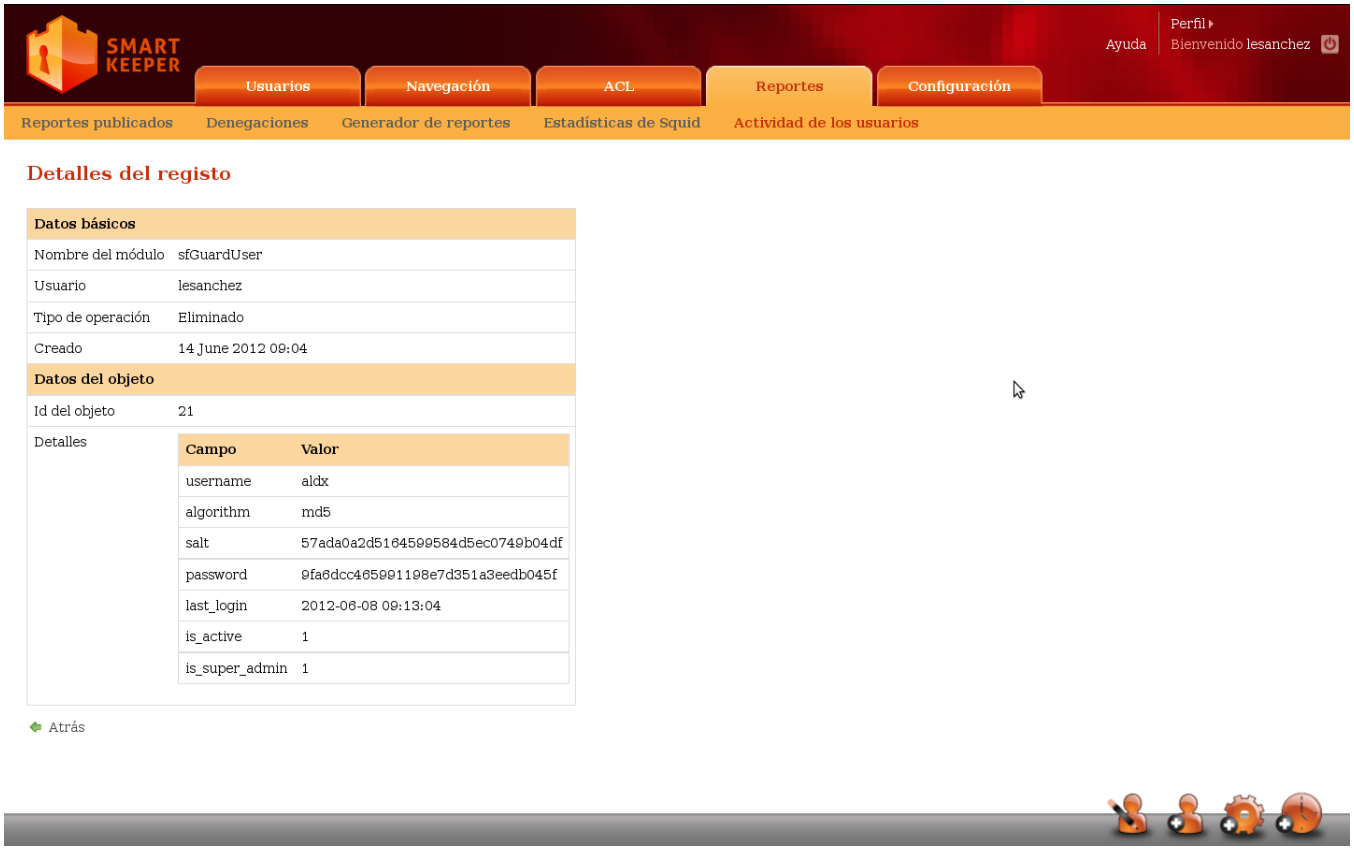

Figura 4.3: Vista de la aplicación detalles de un registro.

4.3. Pruebas

Las pruebas del software son un elemento importante para la garantía de la calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación [25]. Existen diferentes estrategias, niveles, tipos y métodos de pruebas. Para realizar estas pruebas se diseñan casos de prueba. Las estrategias que se pueden seguir son varias, entre las que están: incluir técnicas de detección de errores aplicados a los modelos de análisis y diseño, cambiar la estrategia para las pruebas de unidad e integración. Los niveles se aplican para diferentes objetivos, entre ellos están las pruebas de desarrollador, independiente, de unidad, de integración, de sistema y de aceptación. Los tipos de pruebas son varios, pueden ser de funcionalidad, usabilidad, fiabilidad, rendimiento y soportabilidad. Los métodos pueden ser pruebas de caja negra y pruebas de caja blanca.

Cualquier producto de ingeniería puede probarse de una de estas formas: (1) conociendo la función específica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa y al mismo tiempo buscando errores en cada función, (2) conociendo el funcionamiento del producto, se pueden desarrollar pruebas que aseguren que todas las piezas encajan, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han probado de forma adecuada. El primer enfoque de pruebas se denomina prueba de caja negra y el segundo, prueba de caja blanca [25].

4.3.1. Pruebas funcionales

Las pruebas funcionales de software tienen por objetivo probar que los sistemas desarrollados cumplan con los requisitos funcionales del software. A este tipo de pruebas se les denomina también pruebas de comportamiento y para realizarlas se emplea el método de caja negra, donde los probadores o analistas de pruebas no enfocan su atención en cómo se generan las respuestas del sistema, sino en el funcionamiento de la interfaz del sistema [25].

El diseño de las pruebas funcionales para el módulo de registro de acciones de los administradores en la IW de Smart Keeper, se realizó obteniendo un caso de prueba por cada requisito funcional. También fueron definidos todos los posibles resultados esperados para cada uno de los escenarios. Dichos casos de prueba poseen variables que en ocasiones pueden tomar valor válido o inválido. A continuación en la Tabla 4.3.1 se muestran los requisitos funcionales con los escenarios de los casos de prueba.

Requisito funcional	Escenarios
Registrar evento capturado	Capturar evento
	Registrar evento
Listar registro de acciones de los administradores	Listar registro de acciones
Filtrar resultados del listado de acciones	Filtrar resultados
Obtener los detalles de un registro	Mostrar detalles de acción registrada

Tabla 4.1: Diseño de las pruebas

La descripción de los casos de prueba se encuentran en el expediente de proyecto del módulo.

Con la realización de las dos iteraciones de pruebas funcionales se identificaron varias no conformidades. En la primera iteración se encontraron 8 no conformidades, relacionadas con los nombres de las columnas mostradas en las vistas y errores ortográficos. En la segunda iteración estas no conformidades fueron mitigadas satisfactoriamente excepto una de ellas que no procedió. Esta no conformidad que no procedió está relacionada con el rango límite del filtro de la fecha. Este tipo de campo fecha es proveído por el *framework* symfony por lo que su modificación implicaría cambios para toda la interfaz de Smart Keeper.

4.3.2. Pruebas integración y aceptación

Las pruebas de integración son técnicas sistemáticas para comprobar la arquitectura del software. Estas pruebas se realizan debido a la necesidad de probar conjuntamente las partes de un sistema, pues, aunque estos funcionen bien por separado, alguna parte podría tener un efecto adverso o inadvertido sobre otra. Por tanto, el objetivo de las pruebas de integración es probar el software como un todo, luego de probar sus partes por separado [29].

Las pruebas de aceptación del usuario son pruebas finales realizadas antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales fue construido. El objetivo de estas pruebas es comprobar si el cliente está satisfecho con el producto desarrollado y si este producto cumple con sus expectativas, permitiéndole al usuario que pueda determinar la aceptación del sistema [30].

Al módulo para el registro de acciones de los administradores en la IW de Smart Keeper se le realizaron pruebas de integración y aceptación. Se detectó que el mismo se integra correctamente registrando y visualizando todos los datos desde la IAW. El Ing. Luis E. Sánchez Arce líder del proyecto Smart Keeper y administrador de dicho sistema, avala la integración del módulo, así como, el cumplimiento con todos los requisitos necesarios que permiten el registro de las acciones de los administradores en la Interfaz Web de Smart Keeper. En el documento “Aval de aceptación y de integración” se pueden consultar dicho resultado.

Conclusiones

A partir del diagrama de componentes se identificó con claridad la estructura y las relaciones que existen entre los diferentes componentes empleados en la implementación del módulo. El desarrollo de las pruebas funcionales permitió comprobar el correcto funcionamiento del sistema pues, con la realización de cada caso de prueba se obtuvo un resultado favorable. El desarrollo de las pruebas de integración permitió verificar la compatibilidad de la propuesta con Smart Keeper y validar su integración como un módulo. Las pruebas de aceptación con el usuario final mostraron como resultado la satisfacción del cliente, demostrando que el módulo desarrollado cumple con sus expectativas.

Conclusiones

Después de realizar la fundamentación teórica que sustentó este trabajo, de realizar la implementación y la validación del mismo, se pueden plantear las siguientes conclusiones:

- El estudio del estado del arte relacionado con el proceso de registro de las modificaciones efectuadas por los administradores, sobre las entidades persistentes de una base de datos, a través de la IW de administración, permitió contar con los elementos teóricos necesarios y elegir las herramientas, tecnologías y la metodología para guiar el proceso de desarrollo.
- El diseño del módulo mediante la arquitectura MVC que propone symfony, garantizó mantener un estándar en el desarrollo y organizar el código del módulo de forma eficiente.
- La implementación del módulo para el registro de acciones de los administradores en la IW de Smart Keeper provee al sistema Smart Keeper de una herramienta capás de ofrecer registros de las acciones de los administradores.
- El desarrollo de los diferentes tipos de pruebas permitió validar el correcto funcionamiento del módulo y demostrar la compatibilidad de la propuesta con Smart Keeper.

Ante los elementos anteriores puede afirmarse que se cumplieron los objetivos propuestos, lo cual se materializa en la aplicación obtenida.

Recomendaciones

Con la culminación del presente trabajo se obtuvo un módulo para Smart Keeper en forma de *plugin*, que permite el registro de las acciones realizadas, a las entidades persistentes, por parte de los administradores. Aunque este *plugin* se encuentra totalmente funcional e integrado a Smark Keeper las autoras proponen posibles mejoras a tener en cuenta para próximas versiones las cuales se citan a continuación:

- Implementación de una tarea que permita la habilitación y deshabilitación del registro en los diferentes módulos de Smart Keeper.
- Implementación de una tarea que permita añadir y eliminar los campos de los objetos que no se tendrán en cuenta para el registro de las acciones.
- Implementar una acción que permita acceder al objeto modificado desde el listado de registro.
- Implementar una acción que permita la recuperación de un objeto eliminado.
- Incluir en la solución para el registro el *plugin* paranoidbehavior lo que impide la eliminación de forma persistente de los objetos en la base de datos.

Lista de acrónimos

AUP	Políticas de Uso Aceptable.
WWW	Gran Telaraña Mundial.
TIC	Tecnologías de la Información y las Comunicaciones.
TCP	Protocolo de Control de Transmisión.
URL	Localizador Uniforme de Recursos.
IW	Interfaz Web.
IDE	Entorno de Desarrollo Integrado.
MVC	Modelo Vista Controlador.
GRASP	Patrones de Software de Asignación de Responsabilidades Generales.
UCI	Universidad de las Ciencias Informáticas.
IAW	Interfaz de Administración Web.
XSS	*Cross-site scripting*.
SSL	Capa de *Sockets* Seguro.
SQL	Lenguaje de Consulta Estructurado.
CSRF	*Cross Site Request Forguery*.
IP	Protocolo de Internet.
IU	Interfaz de Usuario.
W3C	Consorcio de la Gran Telaraña Mundial.
BOCA	Administrador de Concurso en Línea Bombonera.
MIT	Instituto Tecnológico de Massachusetts.

UML Lenguaje Unificado de Modelado.

RUP Proceso Unificado de Rational.

ORM Mapeo Objeto-Relacional.

HTTPS Protocolo de Transferencia de Hipertexto Seguro.

PHP PHP Pre-procesador de Hypertexto.

Referencias bibliográficas

[1] Fuentes M. Sociedad de la información: concepto, desarrollo e impacto [en línea]. 2011. Disponible en: http://suite101.net/article/sociedad-de-la-informacion-concepto-desarrollo-e-impacto-a42967 [Accedida 15/06/2012].

[2] Federal Communications Commission. Telecommunications act of 1996 [en línea]. 1996. Disponible en: http://transition.fcc.gov/Reports/tcom1996.txt [Accedida 5/06/2012].

[3] Creswell J. W. Educational research: Planning, conducting, and evaluating quantitative and. 2012. http://www.scis.nova.edu/~nasutif/MCTE690-syllabus-summer2003.pdf.

[4] Guerrero A. y Rodríguez S. Sistemas de filtrado de contenidos. Revista Digital de las Tecnologías de la Información y las Comunicaciones, 2007. Vol 6, issn: 1729-3804.

[5] Gerken T. y Ratschiller T. Creación de aplicaciones web con PHP 4. *Paperback, Mayo*, 2001. p. 10.

[6] Hermosilla J. R. y Sánchez L. E. Interfaz de administración web para el sistema de filtrado filpacon, June 2008. Universidad de las Ciencias Informáticas.

[7] Hermosilla J. R. y Sánchez L. E. Sistema cubano de filtrado de contenidos web utilizando tecnologías libres, 2010. Universidad de las Ciencias Informáticas.

[8] Symfony license [en línea]. 2012. Disponible en: http://www.symfony-project.com/license [Accedida 08/03/2012].

[9] Microsoft. Database properties (changetracking page) [en línea]. 2012. Disponible en: http://msdn.microsoft.com/en-us/library/bb895205.aspx [Accedida 16/03/2012].

[10] Upscene Productions. Mssql logmanagertm product family [en línea]. 2012. Disponible en: http://www.upscene.com/products.audit.mssqllm_main.php [Accedida 18/03/2012].

[11] Zaninotto F. y Potencier F. Symfony [en línea]. 2009. Disponible en: http://www.librosweb.es/symfony_1_1 [Accedida 15/01/2012].

[12] Propelorm. Pre and post hooks for save() and delete() methods [en línea]. 2012. Disponible en: http://www.propelorm.org/documentation/07-behaviors.html [Accedida 8/06/2012].

[13] Techmedianetwork [en línea]. 2012. Disponible en: http://internet-filter-review.toptenreviews.com/netnanny-review.html [Accedida 25/11/2011].

[14] Gamma E. Elements of reusable object-oriented software [en línea]. Disponible en: http://www.exciton.cs.rice.edu/JavaResources/DesignPatterns/book [Accedida 01/12/2011].

[15] Reyes T. A. Sistema de gestión de información odontológica utilizando orm para el departamento de bienestar universitario de la utn. 2011. http://repositorio.utn.edu.ec/bitstream/123456789/571/1/Tesis.pdf.

[16] Propel [en línea]. Disponible en: http://Propelorm.org [Accedida 06/12/2011].

[17] Tupe C. y Cisneros J. Evaluación y selección de framework de desarrollo php: Symfony, kumbia, cakephp y zend. 2008. http://sites.google.com/site/laleydelacarretera/Home/Evaluaci%C3%B3nySelecci%C3%B3ndeFrameworkDesarrolloPHP-CakePhp,Zend,KumbiaySymfony.pdf.

[18] An introduction to netbeans [en línea]. Disponible en: http://netbeans.org/about/index.html. [Accedida 16/11/2011].

[19] Pressman R. *Ingienería de Software un Enfoque Práctico*. Quinta Edición, 2002. ISBN: 84-481-3214-9. Capítulo 1: El producto. p. 20.

[20] Jacobson G. y Rumbaugh J. Booch I. *El proceso unificado de desarrollo de software*. McGraw-Hill, 2002. Captura de requisitos: de la visión a los requisitos. p 107.

[21] Scribd Inc. Arquitectura basada en componentes [en línea]. 2001. Disponible en: http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes [Accedida 7/06/2012].

[22] Jacobson G. y Rumbaugh J. Booch I. *El proceso unificado de desarrollo de software*. McGraw-Hill, 2002. Captura de requisitos: de la visión a los requisitos. p 110.

[23] Booch I. Jacobson G. y Rumbaugh J. *El proceso unificado de desarrollo de software*. McGraw-Hill, 2002. Captura de requisitos como casos de uso. p 128.

[24] Booch I. Jacobson G. y Rumbaugh J. *El proceso unificado de desarrollo de software*. McGraw-Hill, 2002. Captura de requisitos como casos de uso. p 39.

[25] Pressman R. *Ingeniería del Software. Un enfoque práctico*. McGraw-Hill, 2002. ISBN: 84-4813-214-9. Diseño. pp. 238-289.

[26] Pallett D. *Taking a look at ten different PHP frameworks*. Disponible en: http://www.phpit.net/article/ten-different-php-frameworks [Accedida 20/12/2011].

[27] Krucheten P. *The Rational Unified Process: An Introduction*. 2da Edición, 2000. ISBN: 978-0201707106. p. 34.

[28] Scott W. Introduction to uml 2 component diagrams [en línea]. 2010. Disponible en: http://www.agilemodeling.com/artifacts/componentDiagram.htm. [Accedida 2/05/2012].

[29] Pressman R. *Ingeniería del Software. Un enfoque práctico*. McGraw-Hill, 2002. ISBN: 84-4813-214-9. Métodos convencionales para la ingeniería de software. pp. 294-318.

[30] O. Jordán Enríquez and O. Vázquez Ruiz. Generación de casos de prueba a partir de casos de uso en las pruebas de software. *Ingeniería Industrial*, 27(1):4–pág, 2010. http://rii.cujae.edu.cu/index.php/revistaind/article/download/101/80.

Bibliografía

- Pressman R. *Ingienería de Software un Enfoque Práctico*. Quinta Edición. ISBN: 84-481-3214-9, 2002.
- Krucheten P. *The Rational Unified Process: An Introduction*. 2da Edición, 2000.
- Booch I. Jacobson G. y Rumbaugh J. *El proceso unificado de desarrollo de software*. McGraw-Hill, 2002. ISBN: 84-4813-214-9.
- Pallett D. *Taking a look at ten different PHP frameworks*. Disponible en: http://www.phpit.net/article/ten-different-php-frameworks [Accedida 20/12/2011].
- Jacobson G. y Rumbaugh J. Booch I. *El proceso unificado de desarrollo de software*. McGraw-Hill, 2002. ISBN: 84-4813-214-9.
- Pressman R. *Ingeniería del Software. Un enfoque práctico*. McGraw-Hill, 2002.

Anexo A

Casos de usos

	Listar registro de acciones de los administradores	
Objetivo	El objetivo que persigue el actor es acceder al listado de los registros de las acciones de los administradores.	
Actores	Administrador	
Resumen	El caso de uso inicia cuando el administrador entra al módulo.	
Complejidad	Alta	
Prioridad	Crítico	
Referencia	RF14	
Precondiciones	Usuario autenticado en el sistema con los permisos para acceder al módulo.	
Postcondiciones		
Flujo de eventos		
Flujo básico: Listar registro de acciones de los administradores		
Num	**Actor**	**Sistema**
Continúa en la próxima página		

1.	Accede a la página del listado de los registros.	1.1 Muestra la página del listado de los registros con una cantidad mínima de 10 elementos dando al usuario la posibilidad de paginar si así se requiere. Para cada registro aparece: el objeto, usuario, tipo de acción realizada, fecha y hora. También incluye: ▪ para cada elemento del listado la acción detalles ▪ un formulario para filtrar los resultados, con las opciones filtrar y restablecer Con esto se termina el caso de uso.

Tabla A.1: Descripción del caso de uso Listar registro de acciones de los administradores.

Filtrar los resultados del listado de acciones	
Objetivo	El objetivo que persigue el actor es filtrar los resultados del listado de acciones.
Actores	Administrador
Resumen	El caso de uso inicia cuando el administrador selecciona la acción filtrar.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Usuario autenticado en el sitema con los permisos para acceder al módulo.
Referencia	RF14
Postcondiciones	

Continúa en la próxima página

<table>
<tr><th colspan="3">Flujo de eventos</th></tr>
<tr><th colspan="3">Flujo básico: Filtrar los resultados del listado de los registros</th></tr>
<tr><th>Num</th><th>Actor</th><th>Sistema</th></tr>
<tr><td>1.</td><td>Introduce los datos en el formulario y selecciona la opción filtrar.</td><td>1.1 Muestra la página del listado de los usuarios con una cantidad mínima de 10 elementos dando al usuario la posibilidad de paginar si así se requiere. También incluye:
para cada elemento del listado la acción detalles
Además muestra un formulario con los campos: objeto, usuario, tipo de operación y fecha para aplicar restricciones a los valores de las columnas del listado de los registros. Incluye en el formulario las opciones filtrar y restablecer. Con esto se termina el caso de uso.</td></tr>
<tr><td>2.</td><td>Introduce los datos en el formulario y selecciona la opción filtrar.</td><td>2.1 Verifica que los datos son correctos y en función de sus valores muestra la página del listado de los elementos que cumplan con las restricciones, terminando el caso de uso.</td></tr>
<tr><th colspan="3">Flujos Alternos</th></tr>
<tr><th colspan="3">Nº 2.1 : los datos son incorrectos</th></tr>
<tr><th>Num</th><th>Actor</th><th>Sistema</th></tr>
<tr><td></td><td></td><td>2.2 Muestra los mensajes de errores asociados a los campos incorrectos y repite el paso 1.1 del flujo normal de eventos de la sección actual.</td></tr>
</table>

Continúa en la próxima página

Flujos Alternos		
Nº 2 : Selecciona la opción restablecer		
Num	**Actor**	**Sistema**
	Selecciona la opción restablecer.	2.1 Restablece los campos del formulario a sus valores originales (vacíos o con valores por defecto) y muestra la página del listado de los elementos sin las restricciones, terminando el caso de uso.

Tabla A.2: Descripción del caso de uso Filtrar resultados del listado de acciones.

	Mostrar detalles de acciones registradas
Objetivo	El objetivo que persigue el actor es ver los detalles de un registro.
Actores	Administrador
Resumen	El caso de uso inicia cuando el administrador selecciona la acción detalles.
Complejidad	Alta
Prioridad	Crítico
Referencia	RF15
Precondiciones	Usuario autenticado en el sistema con los permisos para acceder al módulo.
Postcondiciones	
Flujo de eventos	
Flujo básico: Mostrar detalles de acciones registradas	
Num	**Actor**	**Sistema**

Continúa en la próxima página

1.	En al página del listado de los registros accede a la acción detalles del registro correspondiente.	1.1 Muestra los datos del registro según el tipo del evento y el tipo de objeto sobre el que se realizó la acción. Si el evento es de tipo adicionar o eliminar , se muestra loa datos del objeto asociado a la acción. Si es de tipo editar, se muestra el objeto antes de editarlo y el objeto editado. Ofrece las opción: volver atrás.
2.	Selecciona la opción ir atrás.	2.1 Envía a la página del listado de los usuarios terminando el caso de uso.

Tabla A.3: Descripción del caso de uso Mostrar detalles de acciones registradas.

Anexo B

Diagramas de clases del diseño

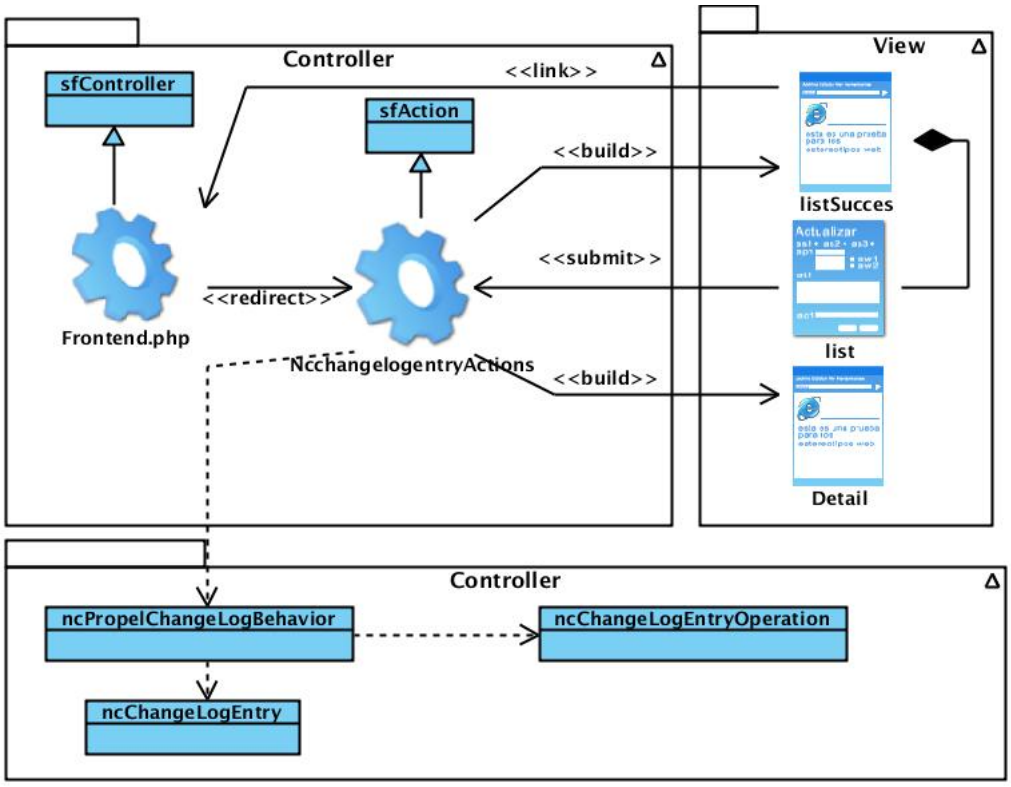

Figura B.1: Diagrama de clases del diseño CU Listar registro de acciones de los administradores, CU Filtra resultados del listado de acciones y CU Mostrar detalles de acción registrada.

Anexo C

Diagramas de interacción

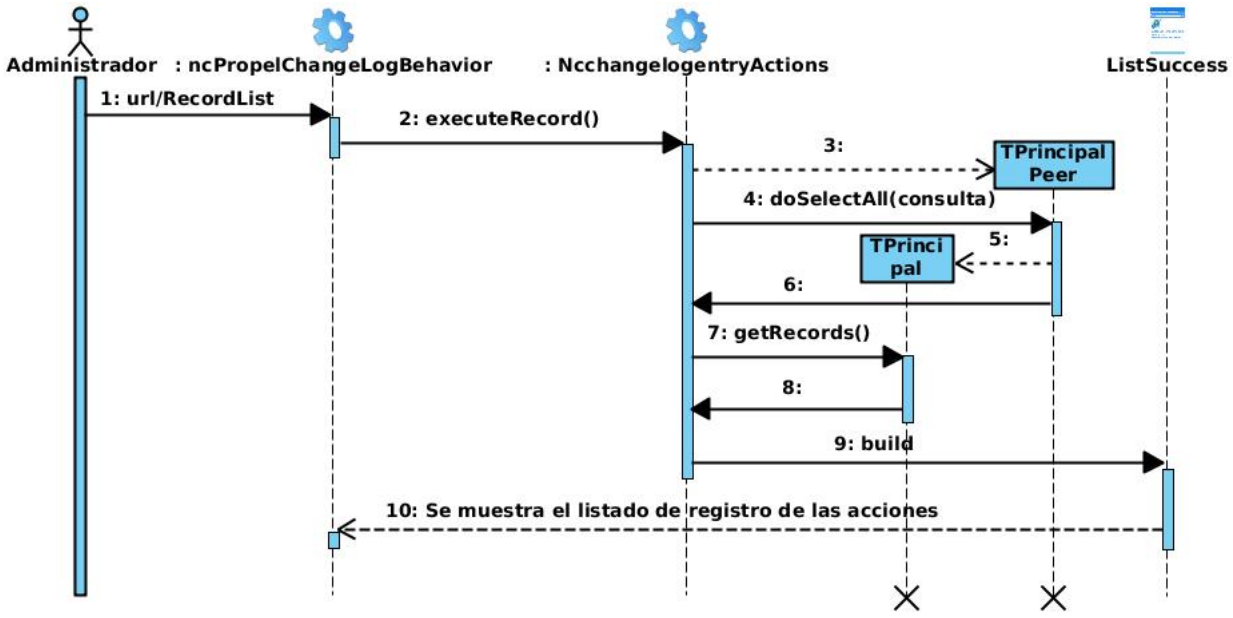

Figura C.1: Diagrama de secuencia CU Listar registro de acciones de los administradores.

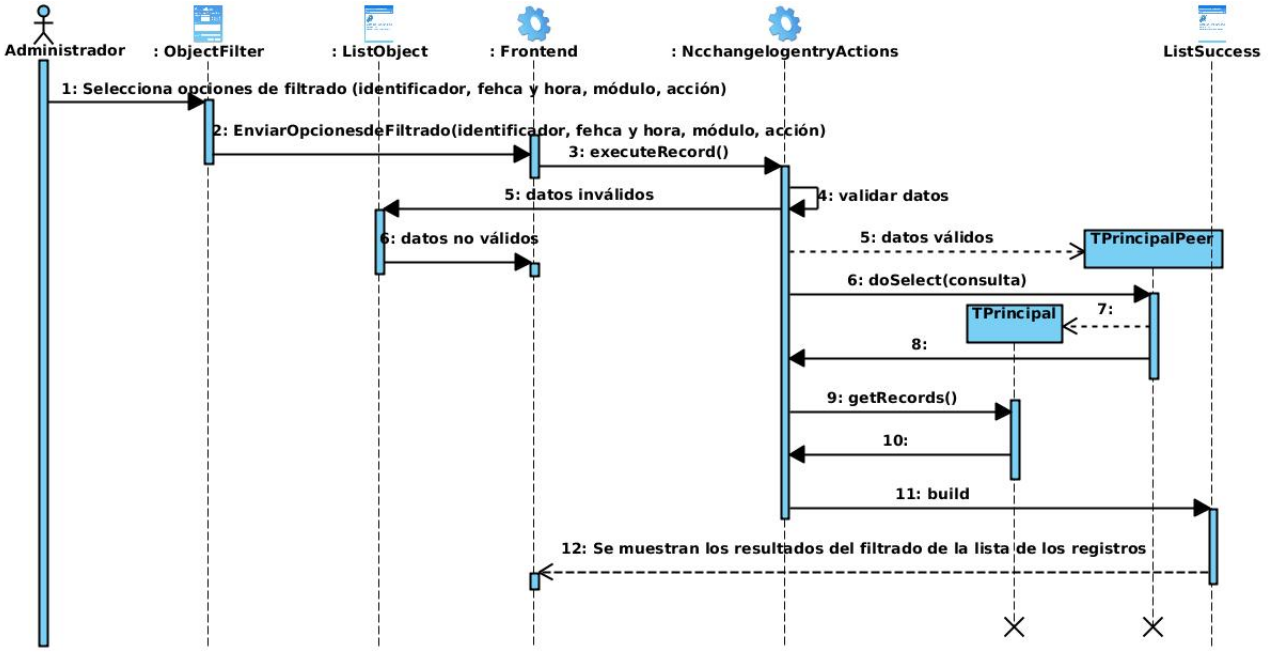

Figura C.2: Diagrama de secuencia CU Filtra resultados del listado de acciones.

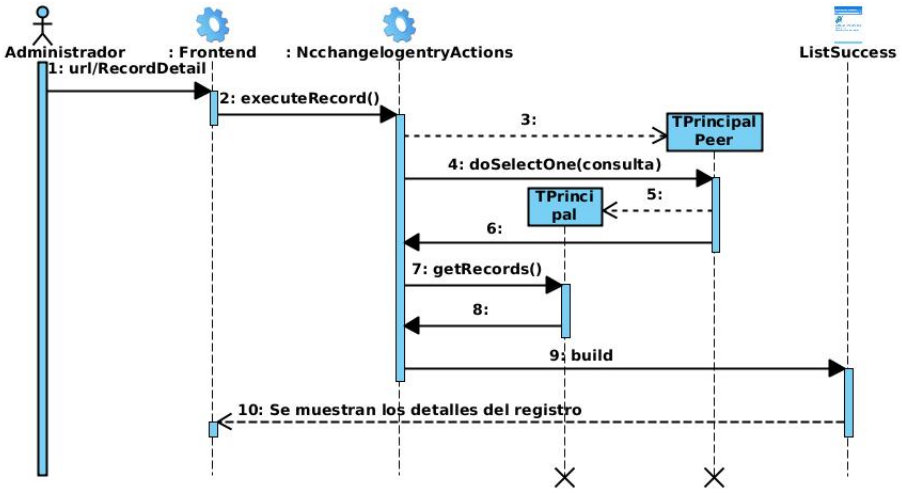

Figura C.3: Diagrama de secuencia CU Mostrar detalles de acción registrada.