Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema para la Gestión de la Información de estudiantes y profesores de la Facultad 6: Módulo Administración

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

Autores: Bárbara Yamili Ávila Rodríguez.
Yitsy Mosqueda Cabrera.

Tutores: Ing. Yamila Mateu Romero.
Ing. Reynaldo Rosado Roselló.

Ciudad de La Habana, junio, 2009.

"Año del 50 Aniversario del Triunfo de la Revolución"

Lo más bello que podemos experimentar es el lado misterioso de la vida, es el sentimiento profundo que se encuentra en la cuna del arte y de la ciencia verdadera.

Albert Einstein

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 firmo la presente a los ____ días del mes de _____ del año _____.

Bárbara Yamili Ávila Rodríguez

_____
Firma del Autor

Yitsy Mosqueda Cabrera

_____
Firma del Tutor

Ing. Yamila Mateu Romero

_____
Firma del Autor

Ing. Reynaldo Rosado Roselló

_____
Firma del Tutor

DATOS DE CONTACTO

Yamila Mateu Romero
ymateu@uci.cu
Ingeniero en Ciencias Informáticas

Reynaldo Rosado Roselló
rrosado@uci.cu
Ingeniero en Ciencias Informáticas

Agradecimientos

A nuestros padres por ser lo más importante para nosotras, sin ellos serían imposibles nuestros logros.
A nuestra familia por hacernos cada día mejores.
A todas las personas que nos han dado un voto de confianza y nos apoyaron en los momentos que los necesitamos.

Dedicatoria

De Bárbara:

A mi madre querida por todo su amor, su dedicación y su apoyo incondicional, por ser la luz que me ilumina cada día y por haberme encaminado correctamente en la vida.
A mi hermanito por ocupar un lugar tan especial en mí vida.
A mi padre, a mis abuelos, a mis tíos y a mis primos por todo el amor, el cariño y la atención que me han dedicado en todo momento.
A Osma por ser como un padre para mí.
A Yanelis y a Maikel por estar para mí en todo momento.
A mi novio por todo el amor, el cariño y el apoyo que me ha brindado en esta etapa de mi vida y por ser mi ángel de la guarda.
A mi dúo de tesis, Yitsy que ha sido mi compañera incondicional en estos cinco años.
A todas mis amistades con las que he compartido mi vida y hoy ocupan un lugar especial en ella.

De Yitsy:

A mi mamá por ser mi guía y parte de cada uno de mis logros, por su confianza en mí, por sentir siempre su presencia aún cuando estaba físicamente lejos.
A mi papá por su apoyo incondicional aunque no siempre estuviera de acuerdo conmigo.
A mi abuela por su infinito amor y por intentar hacerme mejor persona.
A mis tías, mi hermano, mis primas, Chela y Pepe por sus consejos y hacerme saber que siempre están ahí para mí.
A Baby, por ser mi consejera en estos años y mi compañera de tesis.
A mis amigas de toda la vida por sentirlas siempre conmigo.
A todos mis amigos de la UCI, con los que compartí los mejores momentos y que serán imposibles de olvidar.

Resumen

El progresivo desarrollo informático y tecnológico ha aumentado los riesgos en un medio donde prácticamente cualquier intercambio ocurre con la intervención de algún sistema de cómputo. La Universidad de Ciencias Informáticas (UCI), protagonista de estos avances en nuestro país, no escapa a la necesidad de preservar la información.

La Facultad 6, pieza de ésta institución, desarrolla una aplicación como parte del proceso de digitalización que se lleva a cabo actualmente, enfocada en facilitar el trabajo del personal y optimizar el aprovechamiento de las tecnologías con que se cuenta. El Sistema de Gestión de Información de profesores y estudiantes (SIGIPE) constituye la solución de software para la informatización de la misma.

Esta es una aplicación a la cual solo tendrán acceso usuarios autorizados dado que los datos que se manejan en ella son de vital importancia. Por este motivo el objetivo del presente trabajo es implementar el Módulo Administración. Para su realización, se utilizará el framework Symfony y como mecanismo de seguridad el plugin sfGuardPlugin de dicho framework. El mismo se rige por archivos de configuraciones donde se definen los recursos que requieren ser asegurados y quiénes pueden acceder a ellos.

Palabras Claves

Administración, Seguridad, Plugin, Symfony

TABLA DE CONTENIDO

Índice de Figuras

Índice de Tablas

Introducción

El advenimiento de la era digital y el rápido avance de la tecnología, han originado la proliferación de fuentes de información digital. Los revolucionarios sistemas de cómputo y la inesperada velocidad de expansión de las redes de computadoras han facilitado el procesamiento, la distribución y la explotación de este tipo de información.

Cada día las actividades se involucran más con la tecnología, y el intercambio de información se ha convertido en una necesidad primaria para muchos sectores.

Con el avance de las tecnologías en el campo de la informática, el manejo de datos se vuelve cada vez más complicado. En la actualidad se cuenta con enormes volúmenes de información existentes en complejos medios, con capacidades ascendentes de almacenamiento y en soportes más reducidos.

El acceso a la información es necesidad primordial en el mundo de hoy conectado a través de las TICs. La facilidad que ofrece el acceso a la Internet, ha sido la principal causa de esta enorme avalancha de información digital, utilizada por los millones de usuarios que demandan tanto almacenamiento como recuperación. Los niveles de información, de comprensión y el uso que le da el hombre, son problemas que surgen cuando se habla de la Industria de Software. De igual manera es preocupante el hecho de crear aplicaciones cada vez más robustas que garanticen la seguridad de los datos que se manejan en ellas.

Un sistema de computación tiene tres componentes fundamentales: hardware, software, y datos. Los datos son generalmente el activo informático más preciado para cualquier institución. El hardware y el software pueden ser caros pero fáciles de reponer, en cambio, la información contiene la vida de una institución y su valor a veces es incalculable, de ahí la importancia de la seguridad informática.

Se define Seguridad Informática como un conjunto de métodos y herramientas destinados a proteger la información y, por ende, los sistemas informáticos ante cualquier amenaza, un proceso en el cual participan además personas. [1]

Para lograr que un sistema sea seguro es necesario establecer un balance correcto entre los términos confidencialidad, integridad y disponibilidad de los activos informáticos.

El Gobierno Cubano se propuso insertar al país eventualmente entre las principales potencias en la producción de software a través de la creación de instituciones que cumplieran esta tarea, con la finalidad de lograr escalar en el Mercado del Software mundial debido a las ventajas económicas que esto ofrece.

Al calor de la Batalla de Ideas surge la Universidad de las Ciencias Informáticas (UCI) con el objetivo de convertirse en una ciudad digital, estandarte de los procesos de informatización desarrollados en el país y creadora de sistemas informáticos capaces de gestionar gran cantidad de información con un costo mínimo de tiempo de respuesta y con la posibilidad de integración entre estos.

La UCI inició su desarrollo tecnológico, desde el momento de su creación, a nivel de redes físicas ante la necesidad de la comunidad universitaria de servicios como el de correo electrónico, Internet y acceso a información de forma segura. La función informática ha cobrado un papel estratégico en el apoyo a los procesos académicos y de investigación de la universidad.

Con el fin de cubrir las necesidades de la comunidad universitaria de información actualizada y confiable, que permita agilizar y facilitar la gestión académica y administrativa, así como la toma de decisiones encaminadas a lograr el óptimo aprovechamiento de los recursos institucionales, se han creado diversos sistemas de gestión de información en dicha universidad.

La no existencia de un sistema en la Facultad 6 de la UCI que controle la información de estudiantes y profesores de la facultad de forma automatizada, provoca con el paso del tiempo, pérdidas por deterioro tras la constante manipulación de la información archivada en papel y en el caso de los documentos digitalizados, la aparición de virus o fallas en los Sistemas Operativos, lo que ocasiona demoras al realizar búsquedas debido al almacenamiento disipado de los datos y la falta de automatización de los procesos de obtener y brindar información. Por tal motivo, se necesita la creación de un sistema para la gestión de la información de datos de estudiantes y profesores de la facultad, que permita dar solución a problemas reales existentes en la misma. Dicho sistema se compone de seis módulos:

- Módulo Profesores.
- Módulo Estudiantes.
- Módulo Residencia.
- Módulo Extensión.
- Módulo Producción.
- Módulo Tesis.

Debido a que la información que se maneja en la aplicación debe de ser confiable, se impone que el sistema sea seguro y no permita a usuarios que no estén autorizados, a modificar los datos que se manejan en él, proporcionándole mayor confianza a las personas que utilicen el producto y garantizando la seguridad a nivel de aplicación. Por esta razón se crea un módulo para la administración del sistema, de forma que se controlen los usuarios que entran a la aplicación y se establezca el nivel de acceso a los datos de cada uno de ellos, de modo que se evite implementar la

seguridad en cada módulo de dicho sistema y se garantice de esta forma la seguridad centralmente de la solución informática.

Por todo lo antes planteado se define como **Problema Científico**:

¿Cómo contribuir con la administración de los módulos del Sistema para la Gestión de la Información de estudiantes y profesores de la Facultad 6?

Con vista a la solución del problema científico se plantea como **Objeto de estudio**:

El proceso de desarrollo de software para sistemas de gestión de información.

A partir del objeto de estudio se delimita el siguiente **Campo de acción**:

Administración de los módulos del Sistema para la Gestión de la Información de estudiantes y profesores de la Facultad 6.

Se persigue como **Objetivo General:**

Desarrollar el Módulo Administración del Sistema para la Gestión de la Información de estudiantes y profesores de la Facultad 6.

Para cumplir este objetivo general se trazaron los siguientes **Objetivos Específicos**:

1. Fundamentar herramientas y tecnologías.
2. Diseñar el módulo administración del Sistema para la Gestión de la Información de estudiantes y profesores de la Facultad 6.
3. Implementar los componentes para el módulo administración del Sistema de gestión de Información de estudiantes y profesores de la Facultad 6.
4. Identificar los componentes del sistema que requieren ser asegurados.
5. Comprobar la solución propuesta.

Se desarrollarán las siguientes **Tareas** para dar cumplimiento a los objetivos trazados:

1. Estudio de la administración de sistemas de gestión de información similares.
2. Investigación sobre las actividades y artefactos desarrollados por el rol a desempeñar en el equipo de trabajo.
3. Fundamentación de las herramientas y tecnologías a utilizar.
4. Estudio del framework Symfony y uno de sus mecanismos para implementar la seguridad: SfGuardPlugin.
5. Realización de la Vista Lógica.
6. Realización de la Vista Despliegue.
7. Realización de la Vista Implementación.
8. Definición de usuarios, roles y permisos.
9. Identificación de los elementos de la aplicación que requieren ser asegurados.

10. Implementación del módulo Administración.
11. Realizar pruebas a nivel de desarrollador.

El contenido del documento está estructurado en dos capítulos:

Capítulo 1: Fundamentación Teórica.

En este capítulo se hace referencia a los conceptos vinculados al objeto de estudio. Además, se realiza un estudio de los lenguajes, tecnologías y herramientas existentes y se aborda acerca del estado del arte relacionado con herramientas administración que puedan ser utilizadas para la creación del sistema a realizar.

Capítulo 2: Descripción de la solución propuesta.

En este capítulo se realiza una descripción de la solución propuesta al problema científico, a partir de los requerimientos funcionales y no funcionales del Módulo Administración, la creación de la vista Lógica, de Despliegue e Implementación y la realización de pruebas a nivel de desarrollador. Además se explica la definición de usuarios, roles y permisos en la aplicación.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción

En este capítulo se enuncian los principales conceptos asociados al objeto de estudio que son necesarios para un mejor entendimiento de la propuesta de solución. Se estudian puntos claves de herramientas, lenguajes y metodologías a utilizar. Además, se realiza un análisis de herramientas creadas para administrar aplicaciones web y que pueden ser utilizadas en la implementación del Módulo Administración del sistema a elaborar.

1.2 Estado del arte

Las aplicaciones web son de gran importancia en el mundo actual gracias a la evolución de las tecnologías de la información y las comunicaciones, y en especial, de la Internet. Una aplicación web no puede ser tomada como una simple página web, sino como un programa complejo, multiusuario, con conexiones a bases de datos y capaz de ser utilizado en múltiples entornos. Debido a esto, se justifica la importancia de controlar el acceso a la información que se maneja en ellas mediante la administración de los servicios que ofrezca el sistema.

En la actualidad, existen herramientas que, vinculadas a la aplicación, ayudan a la organización administrativa de esta. Dichas herramientas se basan, esencialmente, en distribuir a los usuarios en función de las acciones que puedan realizar en el sistema.

Ejemplo de una de estas herramientas es a2, la cual es una herramienta administrativa configurable que permite consolidar operaciones administrativas bajo un sólido concepto y una amigable interfaz. Su organización administrativa está conformada en módulos o departamentos, todos relacionados de manera lógica.

La herramienta cuenta con un módulo de administración de usuarios que reúne una serie de características, entre ellas están:

- Posibilidad de crear campos adicionales a las bases de datos e incorporarlas fácilmente al sistema y a los formatos de impresión.
- Directorios de datos asignados por el administrador del sistema.
- Administración de números de correlativos y parámetros del sistema.
- Creación de cuentas de usuarios.[2]

Otro ejemplo es el sistema de seguridad del framework ASP.NET. El mismo implementa un sistema de seguridad que consiste en los conceptos de cuentas de usuario, funciones y reglas de acceso, que permite restringir el acceso a los recursos de la aplicación Web sólo a las cuentas de usuario que se

especifique. La configuración de seguridad se establece utilizando una combinación de opciones de configuración y datos almacenados en una base de datos (u otro almacén de datos). Las cuentas de usuario y las funciones que crea se almacenan en la base de datos. La sección Usuarios de la ficha Seguridad permite realizar las tareas siguientes:

- Crear, modificar y eliminar cuentas de usuario registradas para el sitio Web.
- Ver una lista de todas las cuentas de usuario registradas para el sitio Web.
- Cambiar el método de autenticación utilizado por el sitio Web.[3]

Para el desarrollo de la aplicación que se desea realizar, el uso de estas herramientas no es posible debido a que ambas son software propietario y se desea una herramienta desarrollada en software libre.

Después de una investigación previa realizada a diversos módulos de administración se llegó a la conclusión de que los mismos se encargan de llevar el control de los usuarios que entran a la aplicación, los roles que desempeñan, el registro de las actividades que realizan y los permisos, restringiendo de esta forma el acceso a los servicios que brinda el sitio y garantizando a las vez la seguridad de la información que se maneja en el mismo. Por tanto el módulo administración a desarrollar contará con las siguientes secciones:

- ✓ Usuarios
- ✓ Permisos de Usuarios
- ✓ Roles de usuarios
- ✓ Registro de Actividades de Usuarios

1.3 Metodologías de desarrollo

"Una metodología es un conjunto de procedimientos que permiten producir y mantener, implicando esto una definición de pasos a seguir para obtener cierta calidad en el producto final". [4]

Las metodologías se clasifican en ágiles y robustas. La diferencia más notable entre estos dos grupos es que mientras los métodos pesados intentan obtener los resultados apoyándose principalmente en la documentación ordenada, los métodos ágiles tienen como base de sus resultados la comunicación e interacción directa con todos los usuarios involucrados en el proceso. Hoy en día existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo.

1.3.1 Proceso Unificado de Desarrollo (Rational Unified Process, RUP)

Es una metodología robusta. Se sugiere su uso para proyectos nuevos o actualizaciones de sistemas existentes y se recomienda adoptarlo en forma gradual. Es un proceso de desarrollo de software

configurable que se adapta a través de los proyectos variados en tamaños y complejidad. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por casos de uso. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos en el desarrollo de software. El Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. RUP describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto, para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura. Detalla qué entregables producir, cómo desarrollarlos y también provee patrones. Es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas. Se caracteriza básicamente por ser vital la captura de requisitos, iteración actual condicionada por la anterior, se necesita de un buen líder de proyecto para garantizar el trabajo del equipo de desarrollo, se realiza un gran número de artefactos lo que puede provocar retrasos por mala preparación de los analistas, las responsabilidades están divididas y es aplicable a todo tipo de proyecto asumiendo sus extensiones. [5]

Por estas razones se decide utilizar RUP como metodología de desarrollo.

1.4 Rol Implementador

1.4.1 Responsabilidades que desempeña

Este rol desarrolla los componentes de software y efectúa las pruebas de desarrollador para la integración en subsistemas más grandes, de acuerdo con los estándares adoptados de proyecto.

A un implementador se le puede asignar la responsabilidad de implementar una parte estructural del sistema (como un subsistema de implementación o de clases), o una parte funcional del sistema, como la ejecución de casos de uso o sus características.

Tareas que realiza:

- Analizar el comportamiento en tiempo de ejecución.

En esta tarea se describe cómo analizar el comportamiento de un componente durante su ejecución para identificar las mejoras que se pueden realizar.

- Desarrollar productos de trabajo de instalación.

En esta tarea se describe cómo producir todo el software necesario para instalar y desinstalar el producto de forma rápida, sencilla y segura, sin afectar a las demás aplicaciones o características del sistema.

- Implementar elementos de diseño

En esta tarea se describe cómo producir una implementación para una parte del diseño (por ejemplo, una clase, una realización de guión de uso o una entidad de base de datos), o para solucionar uno o varios defectos. El resultado son archivos nuevos o modificados de datos y de código fuente, que se conocen generalmente como elementos de implementación. [6]

1.4.2 Artefactos a generar

- Elemento de implementación

Estos artefactos son los componentes físicos que forman una implementación, que incluyen archivos y directorios. Incluyen los archivos de código de software (binario o ejecutable), los archivos de datos y los archivos de documentación, como los archivos de ayuda en línea.

- Subsistema de implementación

Este artefacto consta de un conjunto de elementos de implementación. Estructura el modelo de implementación dividiéndolo en componentes más pequeños que se pueden integrar y probar separadamente. [6]

1.5 Lenguaje de modelado.

"UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos". [7]

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten, además es independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- ✓ Mayor rigor en la especificación.
- ✓ Permite realizar una verificación y validación del modelo realizado.
- ✓ Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- ✓ Visualizar: UML permite expresar de una forma gráfica un sistema, de modo que otro lo puede entender.
- ✓ Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- ✓ Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.[7]

1.6 Herramienta CASE

CASE es una sigla, que corresponde a las iniciales de: Computer Aided Software Engineering, que significa Ingeniería de Software Asistida por Computación.

El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. [8]

Para el proceso de desarrollo de la aplicación se utilizará como herramienta CASE el software Visual Paradigm for UML 6.1.

Visual Paradigm for UML 6.1

Visual Paradigm para UML 6.1 es una herramienta UML profesional que soporta el ciclo de vida completo del software, análisis orientado a objetos, diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a construir aplicaciones de calidad más rápido, mejor y a menor costo. Puede dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta CASE UML también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

- ✓ Soporte de UML versión 2.1.
- ✓ Diseño de Interfaz de usuario.
- ✓ Proporcionar proyecto basado en la rama y la etiqueta con el trabajo en equipo VP Server, SVN, CVS.
- ✓ Ingeniería inversa.
- ✓ Generación de código - Modelo a código, diagrama a código.

- ✓ Detalles de Casos de Uso Editor - Un todo en un entorno para especificar un caso de uso general, incluyendo los detalles del modelo de pliego de condiciones y descripciones de caso de uso.
- ✓ Diagrama de flujo de datos.
- ✓ Generación de base de datos.
- ✓ División de Respuesta de Emergencia a tablas de base de datos.
- ✓ Inversa de bases de datos.
- ✓ Modelado en colaboración con CVS y Subversion.
- ✓ Informe del generador para la generación de documentación.
- ✓ Diagrama de distribución automática.
- ✓ Exportar diagramas a formato JPG, PNG, SVG, EMF, PDF.
- ✓ Integración de Visio - dibujar diagramas UML con plantillas de Visio.
- ✓ Forma de editor. [9]

1.7 Tecnologías para el desarrollo

El criterio de selección para las herramientas para el proceso de desarrollo de software se basa en que las mismas sean de tipo software libre.

1.7.1 Entorno de desarrollo Integrado (IDE). Eclipse 3.3

Eclipse es un IDE (Entorno de Desarrollo Integrado) tan potente como popular que incorpora varias utilidades para simplificar la labor de los programadores. Aparte de ser un entorno de desarrollo súper completo, una de las particularidades más interesantes para la comunidad es que es de código libre y gratuito.

Para Eclipse existen diversos plugins o añadidos para proveer de nuevas utilidades al programa, enfocadas a diversos usos que los distintos tipos de programadores pueden necesitar. [10]

1.7.2 Lenguaje de programación

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos y una regla principal que resume las demás.

PHP 5

PHP es uno de los lenguajes de programación que nos permiten programar scripts del lado del servidor, insertados dentro del código HTML. Lo interesante de este lenguaje es que es gratuito y puede utilizarse tanto en Linux como en Windows. Es un lenguaje rápido y con una gran cantidad de librerías de funciones y mucha documentación

PHP 5 incluye una orientación total a objetos (serialización incluida), acceso a diversos gestores de bases de datos, orientadas a datos sin necesidad de un RDBMS (Sistema Gestor de Bases de Datos Relacionales) e infinidad de implementación de técnicas para la lectura y escritura de archivos XML.
Otra gran ventaja de PHP 5, es la cantidad de colaboradores que extienden el lenguaje con nuevas funcionalidades como la utilización de XML-RPC, escritura de archivos en PDF u OpenOffice desde PHP, creación de gráficos al vuelo o la utilización de plantillas para separar el código escrito en PHP de la presentación gráfica en pantalla.[11]

1.7.3 Framework Symfony versión 1.2.2

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.
Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.
Se diseñó para que se ajustara a los siguientes requisitos:

- ✓ Fácil de instalar y configurar en la mayoría de plataformas (con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales. Adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. [12]

1.7.4 PostgreSQL 8.2

PostgreSQL es una base de datos relacional, distribuida bajo licencia BSD y con su código fuente disponible libremente. Es un motor de bases de datos de código abierto.

Sus características técnicas la hacen uno de los Sistemas Gestores de Bases de Datos más potentes y robustos. Estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. En los últimos años, su desarrollo, se ha concentrado mucho en la velocidad de proceso y en características demandadas en el mundo empresarial.

PostgreSQL se puede ejecutar en la gran mayoría de sistemas operativos existentes en la actualidad, entre ellos Linux y UNIX en todas sus variantes y Windows. Las características más importantes y soportadas son:

- ✓ Es una base de datos 100% ACID.
- ✓ Llaves ajenas (foreign keys).
- ✓ Joins.
- ✓ Vistas (views).
- ✓ Disparadores (triggers).
- ✓ Reglas (Rules).
- ✓ Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación.
- ✓ Numerosos tipos de datos, posibilidades de definir nuevos tipos.
- ✓ Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido).
- ✓ Herencia de tablas (Inheritance).
- ✓ Tablespaces.
- ✓ Replicación asíncrona.
- ✓ Copias de seguridad en caliente (Online/hot backups).
- ✓ Unicode.
- ✓ Juegos de caracteres internacionales.
- ✓ Multi-Version Concurrency Control (MVCC).
- ✓ Acceso encriptado vía SSL.

- ✓ APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP y muchos otros lenguajes.
- ✓ Completa documentación.[13]

1.7.5 Servidor Web. Apache 2.2.8

Apache HTTP Server 2.2.8 es uno de los más robustos y rápidos servidores web multiplataforma que existen, ha sido creado desde la idea de Open-Source demostrando una vez más que ir asociado a esta idea no es signo de fracaso y lo avala el ser el servidor web más utilizado en todo el mundo (desde pequeña y grandes empresas, pasando por instituciones y universidades) Además, con Apache HTTP Server 2.2.8 podremos ejecutar CGI, Perl, Php3 (o superior) + Bases de datos, SSL y soporte para host virtuales. [14]

1.8 Patrones

De acuerdo con el libro *Pattern-oriented software architecture* de los investigadores Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stal, los patrones arquitectónicos "expresan un esquema de organización estructural para los sistemas de software. Proporcionan un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y lineamientos para organizar la relación entre ellos".

1.8.1 Patrones de Arquitectura

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

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

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. [15]

1.8.2 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos donde existe la comunicación entre los mismos para resolver un problema de diseño en algún contexto. Se clasifican en patrones GRASP y GOF.

Patrones GRASP utilizados por Symfony

- ✓ **Creador**

  En la clase Actions se encuentran las acciones definidas y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Actions es "creador" de dichas entidades.

- ✓ **Experto**

  Este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

- ✓ **Controlador**

  Todas las peticiones Web son manejadas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

- ✓ **Bajo Acoplamiento**

  El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, las conoce y recurre a ellas. Bajo Acoplamiento significa que una clase no depende de muchas clases. La clase Action hereda solamente de sfActions para lograr un bajo acoplamiento de clases.

- ✓ **Alta Cohesión**

  Symfony permite asignar responsabilidades con una alta cohesión. La clase Action, por ejemplo, tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las properties, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios.

Patrones GOF utilizados por Symfony

- En la categoría Creacionales:

- ✓ **Singleton** (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una

llamada a sfContext::getInstance(). En una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony

- ✓ **Abstract Factory** (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

- En la categoría Estructurales:

- ✓ **Decorator** (Envoltorio): Añade funcionalidad a una clase dinámicamente. El archivo layout.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, evitando repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado "decorator".
- ✓ **Composite** (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol .Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.[16]

1.9 Conclusiones

En este capítulo se abordaron los conceptos fundamentales, las condiciones y problemas que rodean el objeto de estudio. Además, se estudió el estado del arte para el Módulo Administración y se mencionaron aspectos esenciales de las tecnologías y herramientas a utilizar para el desarrollo del producto, justificándose así el uso de las mismas.

CAPITULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

2.1 Introducción.

El presente capítulo contiene una descripción de la solución propuesta, a partir de los requerimientos funcionales y no funcionales del Módulo Administración y la creación de la Vista Lógica, de Despliegue e Implementación.

2.2 Análisis de los requerimientos del sistema.

Requerimientos funcionales del Módulo Administración.

RF1 Autenticar Usuario.

RF2 Gestionar Usuario.

- RF2.1 Insertar Usuario.
- RF2.2 Editar Usuario.
- RF2.3 Eliminar Usuario.
- RF2.4 Listar Usuario.
- RF2.5 Filtrar Usuario.
- RF2.6 Importar Usuario.

RF3 Gestionar Rol.

- RF3.1 Insertar Rol.
- RF3.2 Editar Rol.
- RF3.3 Eliminar Rol.
- RF3.4 Listar Rol.
- RF3.5 Filtrar Rol.

RF4 Gestionar Permiso.

- RF4.1 Insertar Permiso.
- RF4.2 Editar Permiso.
- RF4.3 Eliminar Permiso.
- RF4.4 Listar Permiso.
- RF4.5 Filtrar Permiso.

RF5 Gestionar Auditoría.

- RF5.1 Listar Auditoría.
- RF5.2 Eliminar Auditoría.
- RF5.3 Filtrar Auditoría.

Requerimientos no funcionales.

Para el desarrollo:

Sistema Operativo Windows 2000 o superior.

Navegador web Internet Explorer 5.5 o superior o Mozilla Firefox 2.0 o superior.

Servidor web Apache 2.2 o superior.

Servidor de Base de Datos PostgreSQL 8.2.

Para la explotación:

Cliente:

Sistema Operativo Windows 2000 o superior.

Navegador web Internet Explorer (5.5 o superior) o Mozilla Firefox 2.0 o superior.

Servidor:

Sistema Operativo Windows 2000 o superior.

Servidor de Base de Datos PostgreSQL 8.2.

Servidor web Apache 2.2 o superior, PHP 5.2 o superior.

Requerimientos de Hardware:

Pentium III o superior.

256 Mb de memoria RAM o superior.

Disco duro con capacidad de 40 GB o superior.

Para la explotación:

Clientes:

Pentium II o superior.

64 MB de memoria RAM o superior.

Disco duro con capacidad de 20 GB o superior.

Servidor:

Pentium IV o superior a 2.0 GHZ.

2 GB de memoria RAM o superior.

Disco duro con capacidad de 80 GB o superior. [17]

A partir de los requerimientos funcionales se definen los siguientes casos de uso del sistema (CUS):

CUS-1 Autenticar Usuario.

CUS-2 Gestionar Usuario.

CUS-3 Gestionar Rol.

CUS-4 Gestionar Permiso.

CUS-5 Gestionar auditoria.

Descripción de los casos de usos del sistema.

A continuación se muestra la descripción del Caso de Uso Autenticar Usuario.

Caso de Uso:	Autenticar usuario
Actores:	Usuario (Inicia el CU)
Resumen:	El caso de uso se inicia cuando el Usuario introduce los datos para autenticarse en el sistema. El sistema verifica que todos los datos necesarios hayan sido insertados y de forma correcta, dando entrada al sistema según nivel de acceso, finalizando el caso de uso.
Precondiciones:	-
Referencias:	R.1
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el Usuario accede a la aplicación.	2. El sistema muestra el formulario para autenticarse con los campos a llenar para entrar a la aplicación: • Usuario • Clave

3. El usuario llena los campos.	4. El sistema verifica que estos campos estén llenos.
	5. El sistema verifica que este usuario se encuentre en la Base de Datos de la aplicación.
	6. El sistema verifica la clave del usuario en el dominio uci.cu.
	7. El sistema autentica al usuario en la aplicación, finalizando el caso de uso.

Prototipo de interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema informa que estos campos son requeridos para entrar a la aplicación y va a la actividad 1 de la sección **"**signin**"**.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema no encuentra al usuario en la Base de Datos de la aplicación, muestra el mensaje de error "Usuario o contraseña incorrecta" y va a la actividad 1 de la

	Sección **"**signin**".**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	6.1 El sistema comprueba que la contraseña es inválida, muestra el mensaje de error "Usuario o contraseña incorrecta" y va a la actividad 1 de la Sección **"**signin**".**
Postcondiciones	El Usuario queda autenticado con el nivel de acceso correspondiente según el permiso que tenga en la aplicación.

Tabla 1.CUS-1 Autenticar Usuario.

A continuación se muestra la descripción del Caso de Uso Gestionar Usuario.

Caso de Uso:	Gestionar Usuarios
Actores:	Administrador(Inicia el CU)
Resumen:	El CU se inicia cuando el administrador indica realizar alguna de las operaciones siguientes: **Insertar Usuario:** cuando el administrador inserta un nuevo usuario en el sistema, para esto llena los datos necesarios, finalizando el caso de uso. **Editar Usuario:** cuando el administrador necesita modificar datos de un usuario existente, finalizando el caso de uso. **Eliminar Usuario:** el administrador elimina un usuario del sistema, finalizando el caso de uso. **Listar Usuario:** el administrador muestra un listado de todos los usuarios registrados en el sistema, finalizando el caso de uso. **Filtrar Usuario:** el administrador hace una búsqueda de un usuario en el listado de los usuarios, se le especifica el parámetro por el cual buscar y devuelve el usuario, finalizando el caso de uso.
Precondiciones:	- El usuario autenticado debe ser administrador del sistema.

	- Deben existir usuarios insertados siempre que se desee listar, eliminar, editar o filtrar usuarios del sistema.
Referencias:	RF.2.1, RF.2.2, RF.2.3, RF.2.4, RF.2.5
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el Administrador selecciona una de las opciones para gestionar usuarios: • Insertar Usuario. • Editar Usuario. • Eliminar Usuario. • Listar Usuario. • Filtrar Usuario. • Importar Usuario	2. El sistema realiza una de las siguientes acciones: • Si selecciona nuevo usuario, ver sección “Insertar Usuario” • Si selecciona editar, ver sección “Editar Usuario” • Si selecciona eliminar, ver sección “Eliminar Usuario” • Si selecciona listar usuarios, ver sección “Listar Usuarios” • Si selecciona filtrar usuarios, ver sección “Filtrar Usuarios” • Si selecciona importar usuarios, ver sección “Importar Usuarios”
Sección “Insertar Usuario”	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción nuevo usuario.	2. El sistema muestra la interfaz Nuevo Usuario con los campos a llenar del usuario: • Usuario • Usuario Activo • Roles

	• Permisos
3.El administrador llena los campos	4. El sistema verifica que la información en todos los campos sea válida.
	5. El sistema verifica que todos los campos obligatorios estén llenos.
	6. El sistema verifica que el usuario pertenezca al dominio UCI
	7. El sistema verifica que el usuario pertenezca a la Facultad 6.
	8. El sistema adiciona el usuario y se muestra un mensaje informando que el usuario se creó satisfactoriamente, finalizando el caso de uso.

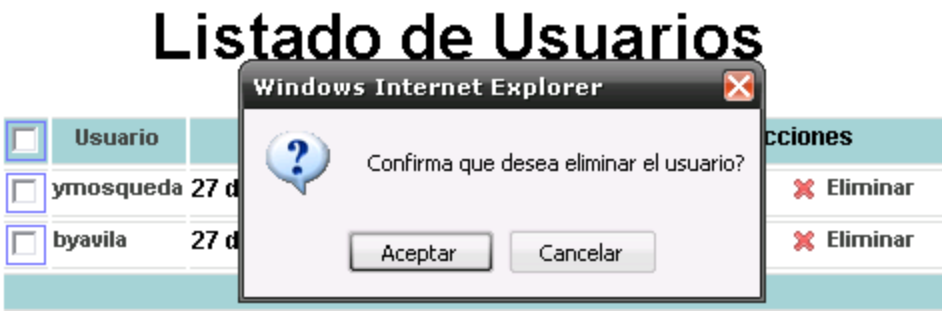
Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1. El sistema muestra un mensaje indicando que el usuario no fue creado debido a algunos errores y va a la actividad 3 de la Sección **"**Insertar Usuario**"**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema informa que se deben llenar los campos y va a la actividad 3 de la Sección **"**Insertar Usuario**"**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	6.1. El sistema informa que el usuario no pertenece al dominio UCI y va a la actividad 3 de la Sección **"**Insertar Usuario**".**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	7.1 El sistema informa que el usuario no pertenece a la Facultad 6 y va a la actividad 3 de la Sección **"**Insertar Usuario**"**.
Sección "Editar Usuario"	
Acción del Actor	**Respuesta del Sistema**

1. El administrador selecciona la opción Editar.	2. El sistema muestra la interfaz Editando Usuario con los campos a modificar del usuario: • Usuario • Usuario Activo • Roles • Permisos
3. El administrador modifica los campos requeridos	4. El sistema verifica que la información en todos los campos sea válida.
	5. El sistema verifica que todos los campos obligatorios estén llenos.
	6. Modifica los datos del usuario y muestra un mensaje informando que el usuario se modificó satisfactoriamente, finalizando el caso de uso

Prototipo de interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje indicando que el usuario no fue modificado debido a algunos errores y va a la actividad 3 de la Sección **"**Editar Usuario**"**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema informa que se deben llenar los campos obligatorios y va a la actividad 3 de la Sección **"**Editar Usuario**"**
Sección "Eliminar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un aviso de confirmación.
3. El administrador acepta el aviso.	4. El sistema elimina el usuario y muestra el Listado de Usuarios, finalizando el caso de uso.
Prototipo de Interfaz	



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1 El administrador cancela el aviso.	4.1 El sistema no elimina al usuario y muestra el Listado de Usuarios.
Sección "Listar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la operación Listado de Usuarios.	2. El sistema muestra un listado de los usuarios del sistema, finalizando el caso de uso.
Prototipo de Interfaz 	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema muestra el mensaje "Sin resultados" si no existen usuarios registrados.
Sección "Filtrar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Filtrar Usuario.	2. El sistema muestra una serie de campos para filtrar los usuarios, en los que se muestran los siguientes parámetros para la

	búsqueda: • Usuario • Tipo • Rol • Permiso
3. El administrador selecciona el criterio de búsqueda	4. El sistema verifica que la información sea válida.
	6. El sistema realiza la búsqueda y muestra el usuario, finalizando el caso de uso.

Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema informa que el formulario no contiene una información válida y va a la actividad 1 de la Sección **"**Filtrar Usuario**"**

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema muestra el mensaje "Sin resultados" si no existen usuarios con los criterios de búsqueda especificados.

<table>
<tr><th colspan="2">Sección “Importar Usuario”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. El administrador selecciona la opción Importar Usuario.</td><td>2. El sistema muestra un formulario para importar los datos, si es estudiante importa por año si es profesor importa todos los profesores</td></tr>
<tr><td>3. El administrador selecciona el criterio de búsqueda</td><td>4. El sistema importa según lo especificado</td></tr>
<tr><td colspan="2">Prototipo de Interfaz
 
</td></tr>
<tr><td>Postcondiciones</td><td>En dependencia de la acción del administrador:
Se insertan nuevos usuarios.
Se editan los datos de los usuarios.
Se eliminan los usuarios.
Se listan los usuarios.
Se filtran los usuarios.
Se importan usuarios</td></tr>
</table>

Tabla 2.CUS-2 Gestionar Usuario.

A continuación se muestra la descripción del Caso de Uso Gestionar Rol.

Caso de Uso:	Gestionar Rol
Actores:	Administrador(Inicia el CU)
Resumen:	El CU se inicia cuando el administrador decide realizar alguna de las operaciones siguientes: **Insertar rol:** cuando el administrador inserta un nuevo rol en el sistema y llena los datos necesarios, finalizando el caso de uso.

	Editar rol: cuando el administrador necesita modificar datos de un rol existente, finalizando el caso de uso. **Eliminar rol:** el administrador elimina un rol del sistema, finalizando el caso de uso. **Listar rol:** el administrador muestra un listado de todos los roles registrados en el sistema, finalizando el caso de uso. **Filtrar rol:** el administrador hace una búsqueda de un rol en el listado, se le especifica el parámetro por el cual buscar y devuelve el rol, finalizando el caso de uso.
Precondiciones:	- El usuario autenticado debe ser administrador del sistema. - Deben existir roles insertados siempre que se desee listar, eliminar, editar o filtrar roles del sistema.
Referencias:	RF.3.1, RF.3.2, RF.3.3, RF.3.4, RF.3.5
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el Administrador selecciona una de las opciones para gestionar roles: • Insertar roles. • Editar roles. • Eliminar roles. • Listar roles. • Filtrar roles.	2. El sistema realiza una de las siguientes acciones: • Si selecciona nuevo rol, ver sección “Insertar Rol” • Si selecciona editar, ver sección “Editar Rol” • Si selecciona eliminar, ver sección “Eliminar Rol” • Si selecciona listar roles, ver sección “Listar Roles” • Si selecciona filtrar roles, ver sección “Filtrar Roles”
Sección “Insertar Rol”	

Acción del Actor	Respuesta del Sistema
1. El Administrador selecciona la opción Nuevo Rol.	2. El sistema muestra la interfaz Nuevo Rol con los campos a llenar del usuario: • Nombre • Descripción • Permisos
3.El administrador llena los campos	4. El sistema verifica que la información en todos los campos sea válida.
	5. El sistema verifica que todos los campos obligatorios estén llenos.
	6. Adiciona el rol y muestra un mensaje informando que el rol se creó satisfactoriamente, terminando el caso de uso.

Prototipo de Interfaz

Flujos Alternos

Acción del Actor	Respuesta del Sistema

	4.1 El sistema muestra un mensaje indicando que el rol no fue creado debido a algunos errores y va a la actividad 3 de la Sección **"**Insertar Rol**"**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema informa que se deben llenar los campos obligatorios y va a la actividad 3 de la Sección **"**Insertar Rol**"**
Sección "Editar Rol"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Editar.	2. El sistema muestra la interfaz Editando Rol con los campos a modificar del usuario: • Nombre • Descripción • Permisos
3. El administrador modifica los campos requeridos	4. El sistema verifica que la información en todos los campos sea válida.
	5. El sistema verifica que todos los campos obligatorios estén llenos.
	6. Modifica los datos del rol y muestra un mensaje informando que el rol se modificó satisfactoriamente, finalizando el caso de uso.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje indicando que el rol no fue modificado debido a algunos errores y va a la actividad 3 de la Sección **"**Editar Rol**"**

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema informa que se deben llenar los campos obligatorios y va a la actividad 3 de la Sección **"**Editar Rol**"**

Sección "Eliminar Rol"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un aviso de confirmación.

3. El administrador acepta el aviso.	4. El sistema elimina el rol y muestra el Listado de Roles, finalizando el caso de uso.

Prototipo de Interfaz

Flujos Alternos

Acción del Actor	Respuesta del Sistema
3.1 El administrador cancela el aviso	4.1 El sistema no elimina el rol y muestra el Listado de Roles.

Sección "Listar Roles"

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción Roles de Usuarios	2. El sistema muestra un listado de los roles del sistema, finalizando el caso de uso.

Prototipo de Interfaz

Flujos Alternos

Acción del Actor	Respuesta del Sistema

	2.1 El sistema muestra el mensaje “Sin resultado” si no existen roles en la Base de Datos.
Sección “Filtrar Rol”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Filtrar Rol.	2. El sistema muestra un campo para filtrar los roles, en el que se muestra el siguiente parámetro para la búsqueda: • Nombre • Permisos
3. El administrador especifica los criterios de búsqueda.	4. El sistema verifica que la información sea válida.
	5. El sistema realiza la búsqueda y muestra el rol, finalizando el caso de uso.
Prototipo de Interfaz 	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema informa que el formulario no contiene una información válida y va a la actividad 2 de la Sección **“**Filtrar Rol**”.**

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema muestra el mensaje sin resultados si no existen roles con este criterio de búsqueda.
Postcondiciones	En dependencia de la acción del administrador: • Se insertan nuevos roles. • Se editan los datos de los roles. • Se eliminan los roles. • Se listan los roles. • Se filtran los roles.

Tabla 3.CUS-3 Gestionar Rol.

A continuación se muestra la descripción del Caso de Uso Gestionar Permiso.

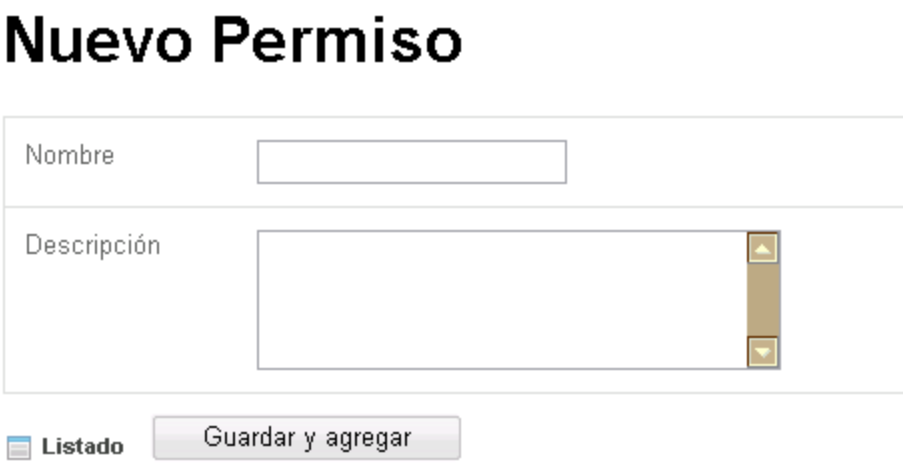
Caso de Uso:	Gestionar Permiso
Actores:	Administrador(Inicia el CU)
Resumen:	El CU comienza cuando el administrador decide realizar alguna de las operaciones siguientes: **Insertar permiso:** cuando el administrador inserta un nuevo permiso en el sistema y llena los datos necesarios, finalizando el caso de uso. **Editar permiso:** cuando el administrador necesita modificar datos de un permiso existente, finalizando el caso de uso. **Eliminar permiso:** el administrador elimina un permiso del sistema, finalizando el caso de uso. **Listar permiso:** el administrador muestra un listado de todos los permisos registrados en el sistema, finalizando el caso de uso. **Filtrar permiso:** el administrador hace una búsqueda de un permiso en el listado, se le especifica el parámetro por el cual buscar y devuelve el permiso, finalizando el caso de uso.
Precondiciones:	- El usuario autenticado debe ser administrador del sistema.

	- Deben existir permisos insertados siempre que se desee listar, eliminar, editar o filtrar roles del sistema.
Referencias:	RF.4.1, RF.4.2, RF.4.3, RF.4.4, RF.4.5
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el Administrador selecciona una de las opciones para gestionar permisos: • Insertar permisos. • Editar permisos. • Eliminar permisos. • Listar permisos. • Filtrar permisos.	2. El sistema realiza una de las siguientes acciones: • Si selecciona nuevo permisos, ver sección “Insertar Permiso” • Si selecciona editar, ver sección “Editar Permisos” • Si selecciona eliminar, ver sección “Eliminar Permisos” • Si selecciona listar permisos, ver sección “Listar Permisos” • Si selecciona filtrar permisos, ver sección “Filtrar Permisos”
Sección “Insertar Permiso”	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción nuevo permiso.	2. El sistema muestra la interfaz Nuevo Permiso con los campos a llenar del permiso: • Nombre • Descripción
3.El administrador llena los campos	4. El sistema verifica que la información en todos los campos sea válida.

	5. El sistema verifica que todos los campos obligatorios estén llenos.
	6. El sistema adiciona el permiso y muestra un mensaje informando que el permiso se creó satisfactoriamente, finalizando el caso de uso.

Prototipo de Interfaz



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje indicando que el permiso no fue creado debido a algunos errores y va a la actividad 3 de la Sección **"**Insertar Permiso**"**

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema informa que se deben llenar los campos obligatorios y va a la actividad 3 de la Sección **"**Insertar Permiso**"**

Sección “Editar Permiso”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Editar.	2. El sistema muestra la interfaz Editando Permiso con los campos a modificar del permiso: • Nombre • Descripción
3. El administrador modifica los campos requeridos	4. El sistema verifica que la información en todos los campos sea válida.
	5. El sistema verifica que todos los campos obligatorios estén llenos.
	6. Modifica los datos del permiso y muestra un mensaje informando que el permiso se modificó satisfactoriamente, finalizando el caso de uso.

Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje indicando que el permiso no fue

	modificado debido a algunos errores y va a la actividad 3 de la Sección **"**Editar Permiso**"**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema informa que se deben llenar los campos y va a la actividad 3 de la Sección **"**Editar Permiso**"**
Sección "Eliminar Permiso"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	7. El sistema muestra un aviso de confirmación.
3. El administrador acepta el aviso.	4. El sistema elimina el permiso y va a la actividad 2 de la sección "Listar Permisos", finalizando el caso de uso.
Prototipo de Interfaz 	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1 El administrador cancela la operación	4.1 El sistema no elimina el permiso y muestra el Listado de Permisos.

Sección “Listar Permisos”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Permisos de Usuarios.	2. El sistema muestra un listado de los permisos del sistema, finalizando el caso de uso.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema muestra el mensaje “Sin resultados” si no existen permisos en la Base de Datos.

Sección “Filtrar Permisos”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Filtrar Permiso.	2. El sistema muestra un campo para filtrar permisos, en el que se muestra el siguiente parámetro para la búsqueda: • Nombre
3. El administrador especifica los criterios de búsqueda.	4. El sistema realiza la búsqueda y muestra el permiso, finalizando el caso de uso.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra el mensaje “Sin Resultados” si no existen permisos con estos criterios de búsqueda.
Postcondiciones	En dependencia de la acción del administrador: • Se insertan nuevos permisos. • Se editan los datos de los permisos. • Se eliminan los permisos. • Se listan los permisos. • Se filtran los permisos

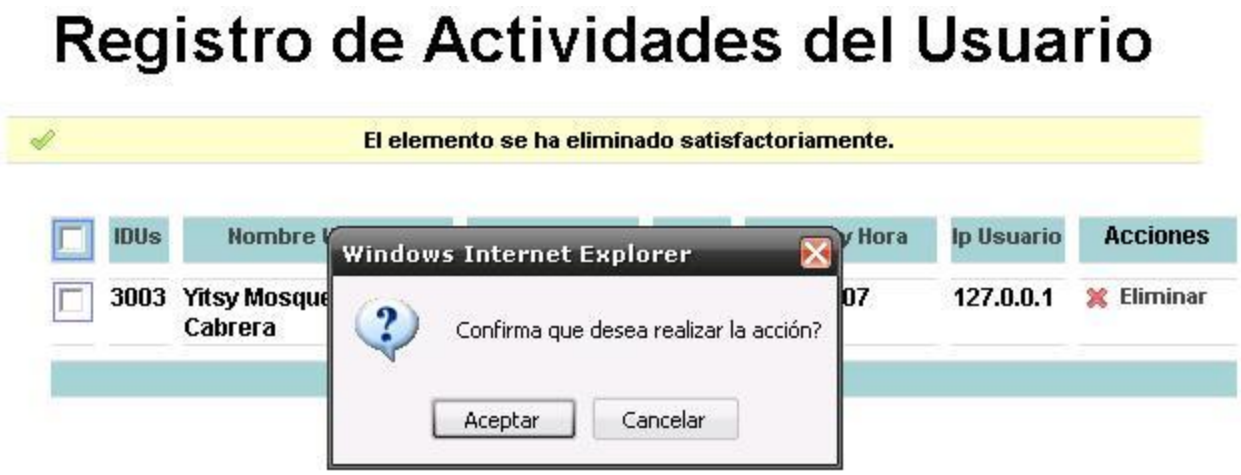
Tabla 4.CUS-4 Gestionar Permiso.

A continuación se muestra la descripción del Caso de Uso Gestionar Auditoría.

Caso de Uso:	Gestionar Auditoría
Actores:	Administrador(Inicia el CU)
Resumen:	El CU comienza cuando el administrador decide realizar alguna de las operaciones siguientes: **Listar Auditoría:** cuando el administrador muestra un listado de todas las actividades realizadas por el usuario en la aplicación, finalizando el caso de uso. **Eliminar Auditoría:** el administrador elimina algún registro de actividades del usuario, finalizando el caso de uso. **Filtrar Auditoría:** el administrador hace una búsqueda de algún registro de actividades del usuario en el listado, para esto se le especifica el id, el usuario, el módulo, la acción o la fecha, finalizando el caso de uso.

Precondiciones:	- El usuario autenticado debe ser administrador del sistema.
Referencias:	RF.5.1, RF.5.2, RF.5.3
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el Administrador selecciona una de las opciones para gestionar auditoria: • Eliminar auditoría. • Listar auditoría. • Filtrar auditoría.	2. El sistema realiza una de las siguientes acciones: • Si selecciona eliminar, ver sección “Eliminar Auditoría” • Si selecciona listar auditoria, ver sección “Listar Auditoría” • Si selecciona filtrar auditoria, ver sección “Filtrar Auditoría”
Sección “Eliminar Auditoria”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un aviso de confirmación.
3. El administrador acepta el aviso.	4. El sistema elimina el registro de ese usuario y muestra el Registro de Actividades del Usuario, finalizando el caso de uso.
Prototipo de Interfaz	



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1 El administrador cancela el aviso	4.1 El sistema no elimina la auditoría y muestra el Registro de Actividades del Usuario, finalizando el caso de uso.
Sección "Listar Auditoria"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción registro de actividades del usuario.	2. El sistema muestra un listado de los registros de los usuarios que han realizado actividades en la aplicación, finalizando el caso de uso.

Prototipo de Interfaz

Flujos Alternos

Acción del Actor	Respuesta del Sistema
	2.1 El sistema muestra el mensaje sin resultados si no existen auditorías en la Base de Datos.
Sección "Filtrar Auditoria"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Filtrar.	2. El sistema muestra una serie de campos para filtrar los registros del usuario, en la que se muestran los siguientes parámetros para la búsqueda: • Id Usuario • Usuario • Módulo • Acción • Fecha
3. El administrador especifica los criterios de búsqueda.	4. El sistema verifica que la información sea válida.
	5. El sistema realiza la búsqueda y muestra el registro buscado.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1 El sistema muestra el mensaje “Sin resultados” si no existen auditorías con estos criterios de búsqueda.
Postcondiciones	En dependencia de la acción del administrador: • Se eliminan las auditorías. • Se listan las auditorías. • Se filtran las auditorías.

Tabla 5.CUS-5 Gestionar Auditoría.

2.3 Vista Lógica.

La vista lógica muestra cómo la funcionalidad es diseñada dentro del sistema, define la estructura y el comportamiento del mismo. [18]

El framework Symfony define un diseño, conceptos, componentes y herramientas de administración que se reutilizarán en el desarrollo del sistema. Symfony implementa el patrón MVC por lo que la estructura organizacional del Módulo Administración se representa a través de tres elementos: el controlador, el modelo y las vistas.

A continuación se muestran los diagramas de clases del diseño por cada caso de uso del sistema.

2.3.1 Diagramas de clases del diseño.

Los paquetes y subsistemas de diseño que conforman la vista lógica son:

Paquete Controlador: Agrupa las clases que implementan la lógica de la aplicación.

Paquete Vista: Produce las plantillas, formadas por código HTML y PHP, que se muestran como resultado de las acciones.

Paquete Modelo: Contiene las clases que encapsulan la lógica del dominio y se encarga del acceso a datos almacenados en el gestor de base de datos.

Subsistema Propel: Es un ORM para PHP que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la Base de Datos mediante objetos, con la que se puede recuperar, insertar y modificar datos. [19]

Subsistema Componentes Symfony: Representa las funcionalidades del Symfony que serán utilizadas para el funcionamiento del sistema.

En la figura 1 se muestra el diagrama de clases del diseño correspondiente al **CUS-1 Autenticar Usuario**.

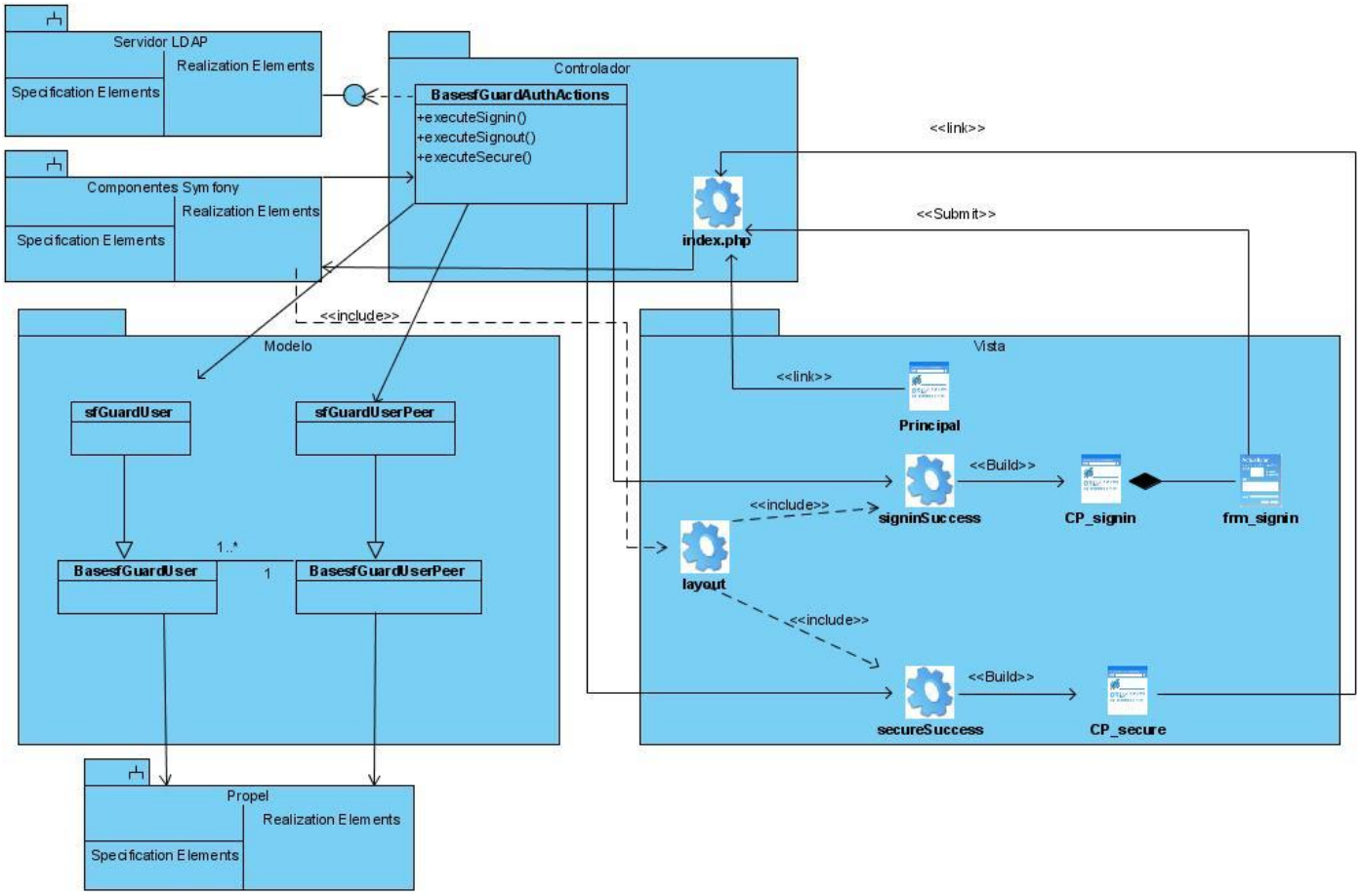


Figura 1.Diagrama de clases del diseño CUS-1 Autenticar Usuario

A continuación se describen los paquetes y subsistemas de diseño que conforman este caso de uso:

Subsistema LDAP: Representa al Servidor Web LDAP de la UCI. Se utiliza para la validación de la contraseña del usuario.

En el Paquete Controlador:

- La clase controlador frontal *index.php* es el único punto de entrada al sistema, carga la configuración inicial y determina la acción a ejecutarse. Esta clase es común a todos los diagramas que se muestran en este epígrafe.
- La clase BasesfGuardAuthActions contiene el código específico de las acciones del módulo sfGuardAuth.

  Las acciones de esta clase son:

✓ executeSignin() La acción de la vista signin.

✓ executeSignout() La acción que permite la salida del sistema del usuario autenticado. Redirecciona a la template página principal.

- ✓ executeSecure() La acción de la vista secure.

Para mejor comprensión se muestra el paquete Controlador en la figura 2.

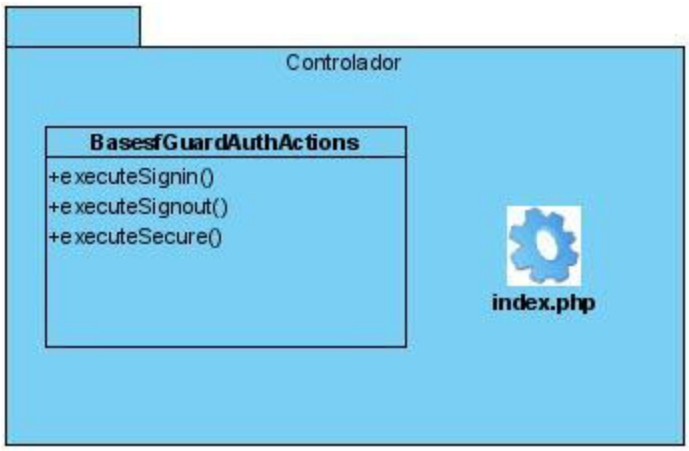


Figura 2. Paquete Controlador del CUS-1 Autenticar Usuario

En el Paquete Vista:

- La clase *layout* contiene los elementos que se muestran de forma idéntica para las páginas web a lo largo de toda la aplicación. Incluye el código de las páginas *signinSuccess*, *signoutSuccess* y *secureSuccess*.
- La página *Portada* es la página inicial del sistema.
- La página *signinSuccess* contiene el script que interactúa con los recursos del servidor a través de operaciones que validan si el usuario está correctamente autenticado. Esta página construye la página *CP_signin,* que contiene el formulario de entrada al sistema.
- La página *signoutSuccess* contiene el script que, ejecutado por el servidor, permite al usuario terminar la sesión, previamente iniciada, en el sistema.
  - La página *SP_secure* se ejecuta cuando el usuario intenta acceder a una página para la cual tiene que estar previamente autenticado. Esta página construye otra de tipo *CP_secure*, la que muestra en el navegador que el usuario no tiene permiso para acceder a la URL a la que intenta entra

En la figura 3 se muestra el paquete Vista.

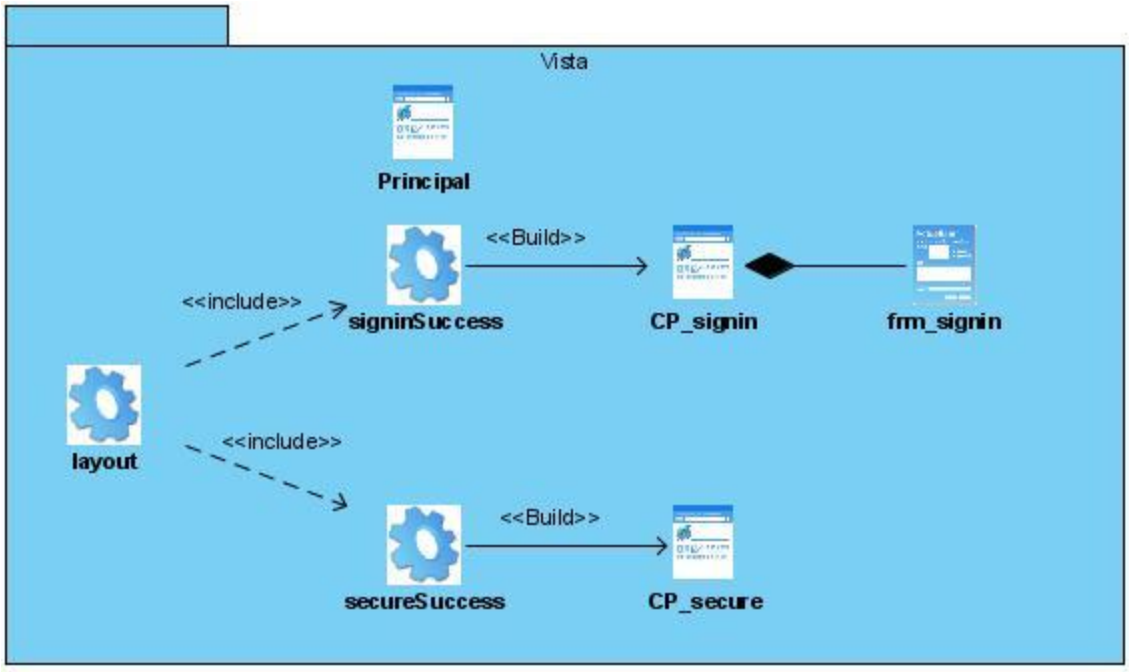


Figura 3. Paquete Vista del CUS-1 Autenticar Usuario

En el Paquete Modelo:

La clase *sfGuardUser* contiene todos los atributos de los usuarios.

La clase *sfGuardUserPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *sfGuardUser*.

Una instancia de la clase *BasesfGuardUserPeer* puede crear una o más instancias de la clase *BasesfGuardUser*.

En la figura 4 se muestra el paquete Modelo:

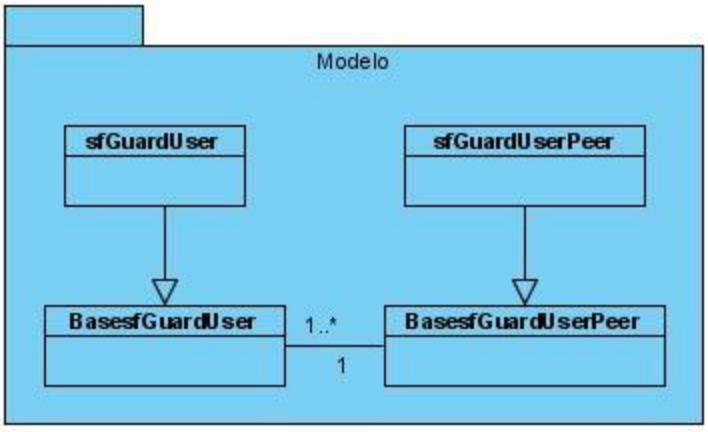


Figura 4. Paquete Modelo del CU-1 Autenticar Usuario

En la figura 5 se muestra el diagrama de clases del diseño correspondiente al **CUS_2 Gestionar Usuario.**

Figura 5.Diagrama de clases del diseño CUS-2 Gestionar Usuario

Servidor Identidad: Representa al Servidor Web Identidad de la UCI, el cual se emplea para obtener un usuario dado su nombre de usuario del dominio UCI.

Servidor Assets: Representa al Servidor Web de Capital Humano de la UCI, el cual se utiliza para verificar si el usuario, de tipo Profesor o Trabajador, pertenece a la Facultad 6 de la UCI.

Servidor Akademos: Representa al Servidor Web de Gestión Académica de la UCI, el cual se utiliza para verificar si el usuario, de tipo Estudiante, pertenece a la Facultad 6 de la UCI.

En el Paquete Controlador:

La clase *sfGuardUserActions* hereda de la clase Base*sfGuardUserActions*, clase que contiene el código específico de las acciones del módulo sf_guard_user.

Las acciones de esta clase son:

- ✓ executeIndex() La acción de la vista index
- ✓ executeFilter() Actualiza los filtros
- ✓ executeNew() La acción de la vista new
- ✓ executeCreate() Crea un nuevo elemento
- ✓ executeEdit() La acción de la vista edit
- ✓ executeUpdate() Actualiza un elemento
- ✓ executeDelete() Borra un elemento
- ✓ executeBatch() Ejecuta una acción por lote
- ✓ executeBatchDelete() Ejecuta la acción por lote _delete
- ✓ processForm() Procesa el formulario del elemento
- ✓ getFilters() Devuelve los filtros actuales
- ✓ setFilters() Establece los filtros
- ✓ getPager() Devuelve el paginador de la lista
- ✓ getPage() Obtiene la página de la lista
- ✓ setPage() Establece la página de la lista
- ✓ buildCriteria() Construye el Criteria para la lista
- ✓ addSortCriteria() Agrega un Criteria ordenado para la lista

- ✓ getSort() Devuelve la columna utilizada para ordenar
- ✓ setSort() Establece la columna utilizada para ordenar

Estas acciones también son válidas para las clases BasesfGuardGroupActions, BasesfGuardPermissionActions y auditoriaActions de los módulos sf_guard_group, sf_guard_permission y auditoria respectivamente.

Para este módulo en particular se implementa:

- ✓ CrearProfes() Importa los usuarios de los profesores de la Facultad 6, les asigna el rol Profesor y los permisos asociados.
- ✓ Crear Estudiantes() Importa los usuarios de los estudiantes de la Facultad 6 y el año especificado, les asigna el rol Estudiante y los permisos asociados.
- ✓ EliminarUsuariosProfes() Elimina los usuarios del sistema de tipo Profesor.
- ✓ EliminarUsuariosEstudiantes() Elimina los usuarios del sistema de tipo Estudiante y del año especificado.
- ✓ executeImportar() La acción de la vista importar.
- ✓ executeMensajeImportar() La acción de la vista mensajeImportar.
- ✓ executeEnviarDatos() Procesa las operaciones de usuarios de tipo Estudiante de la vista importar.
- ✓ executeEnviarDatosP() Procesa las operaciones de usuarios de tipo Profesor de la vista importar.

Para mejor comprensión se muestra el paquete Controlador en la figura 6.

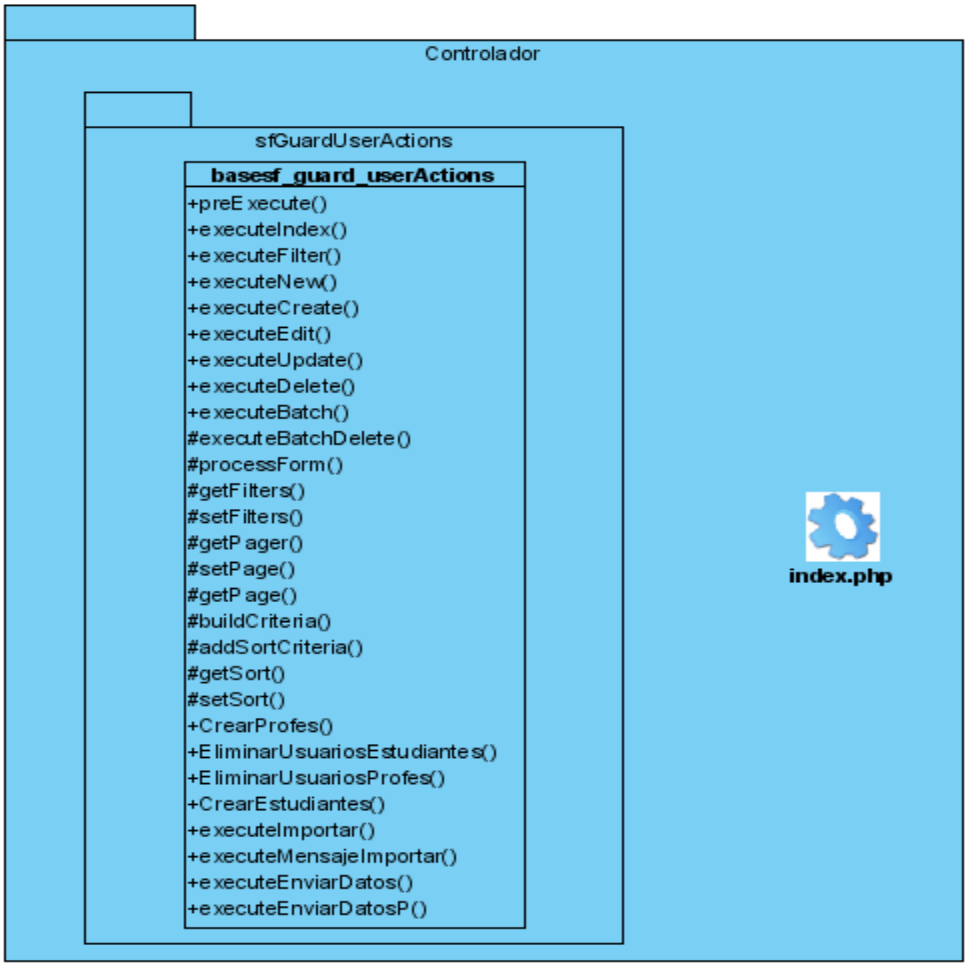


Figura 6.Paquete Controlador del CUS-2 Gestionar Usuario

En el Paquete Vista:

- La clase *layout* contiene los elementos que se muestran de forma idéntica para las páginas web a lo largo de toda la aplicación. Incluye el código de las páginas *indexSuccess*, *newSuccess* y *ediSuccesst*.
- La página *Principal* es la página principal del sistema.
- La página *indexSuccess* contiene toda la información de los usuarios.
- La página *CP_index,* que es construida por la página *indexSuccess,* contiene los formularios *frm_sf_admin_content* que muestra el listado de los usuarios y brinda la opción de eliminar al mismo de la aplicación y el frm_sf_admin_filter que permite filtrar el usuario.
- La página *newSuccess* se encarga de agregar un nuevo usuario a la aplicación.
- La página *CP_new,* que es construida por la página *newSuccess,* contiene el formulario *frm_sf_admin_form* que permite insertar el nuevo usuario.

- La página *editSuccess* es la encargada de modificar la información del usuario.
- La página *CP_edit,* que es construida por la página *editSuccess,* contiene el formulario *frm__sf_admin_form* que permite realizarle modificaciones a los datos del usuario.
- La página *importarSuccess* es la encargada de importar los usuarios de estudiantes o profesores de la Facultad 6.
- La página *CP_importar,* que es construida por la página *importarSuccess,* contiene el formulario frm_importar que permite entrar al sistema los parámetros para importar los usuarios de tipo Profesor o Estudiante.

En la figura 7 se muestra el paquete Vista.

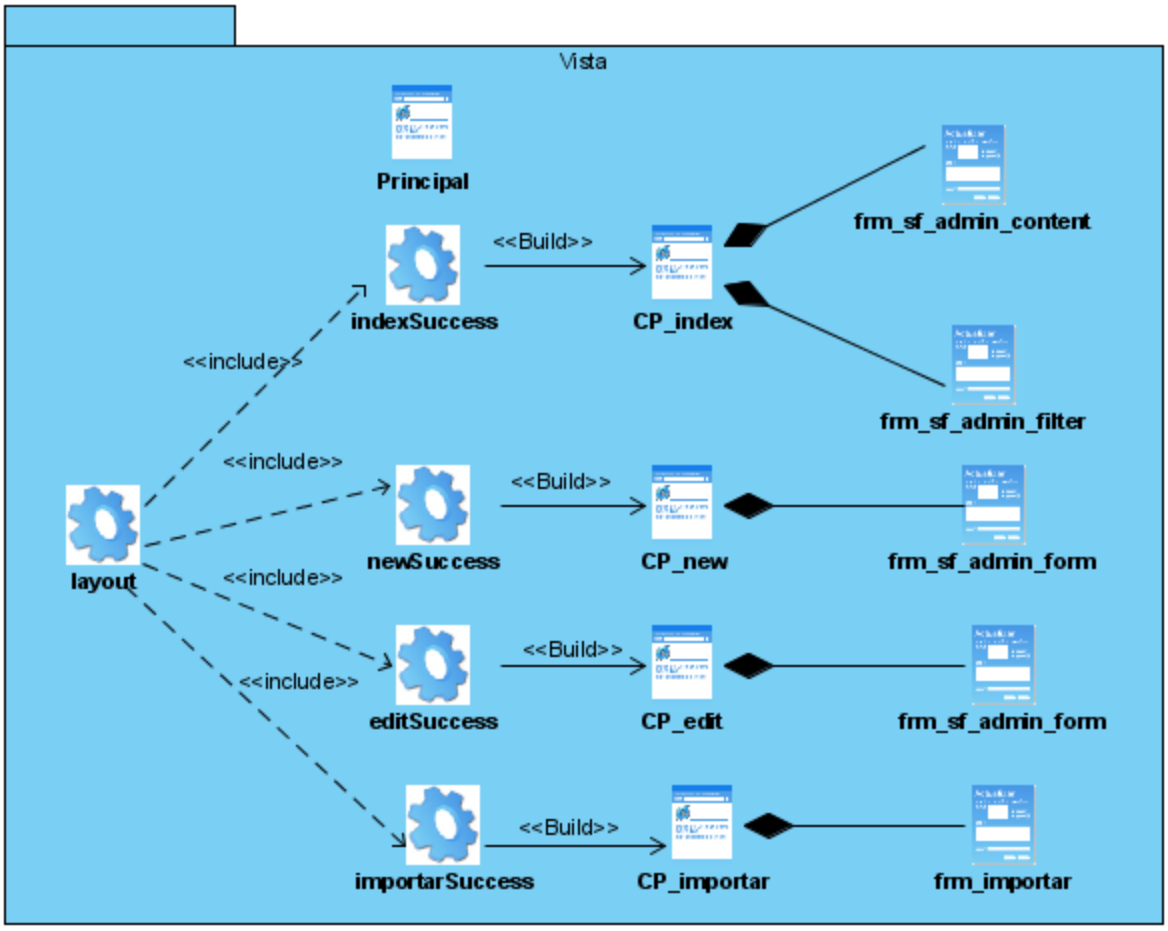


Figura 7.Paquete Vista del CUS-2 Gestionar Usuario

En el Paquete Modelo:

La clase *sfGuardUser* contiene todos los atributos de los usuarios.

La clase *sfGuardUserPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *sfGuardUser*.

Una instancia de la clase *BasesfGuardUserPeer* puede crear una o más instancias de la clase *BasesfGuardUser.*

La clase *sfGuardUserGroup* contiene la relación entre los usuarios y los grupos.

La clase *sfGuardUserGroupPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *sfGuardUserGroup*.

Una instancia de la clase *BasesfGuardUserGroupPeer* puede crear una o más instancias de la clase *BasesfUserGuardGroup.*

La clase *sfGuardUserPermission* contiene la relación entre los usuarios y los permisos.

La clase *sfGuardUserPermissionPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *sfGuardUserPermission.*

Una instancia de la clase *BasesfGuardPermissionPeer* puede crear una o más instancias de la clase *BasesfGuardPermission.*

En la figura 8 se muestra el paquete Modelo.

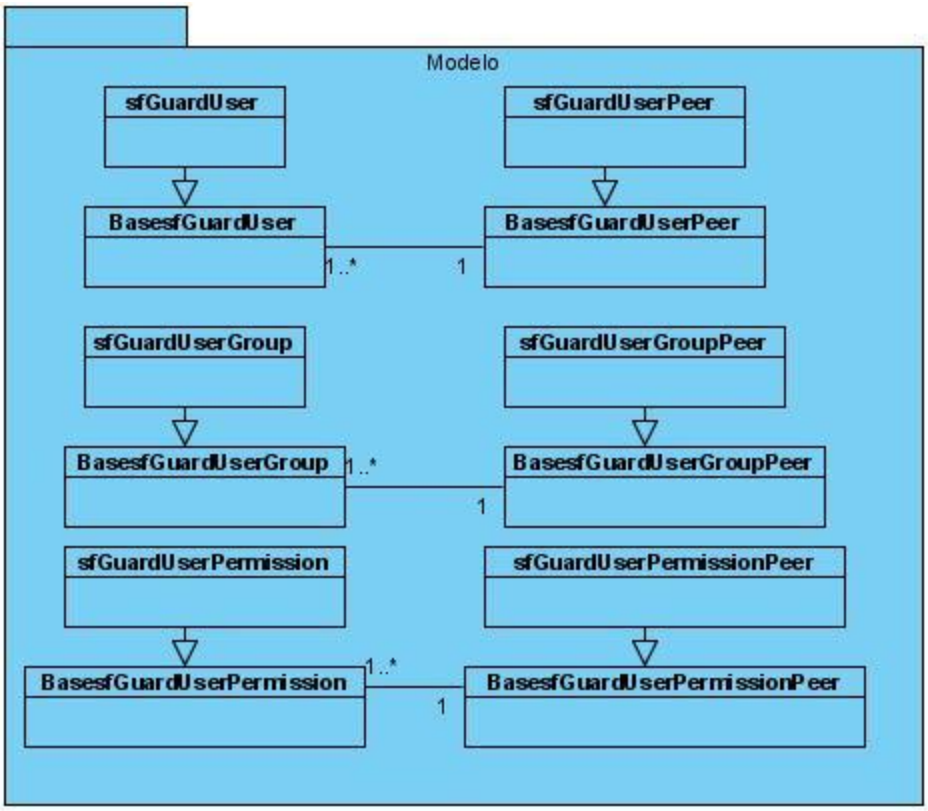


Figura 8.Paquete Modelo del CUS-2 Gestionar Usuario

En la figura 9 se muestra el diagrama de clases del diseño correspondiente al **CUS_3 Gestionar Rol.**

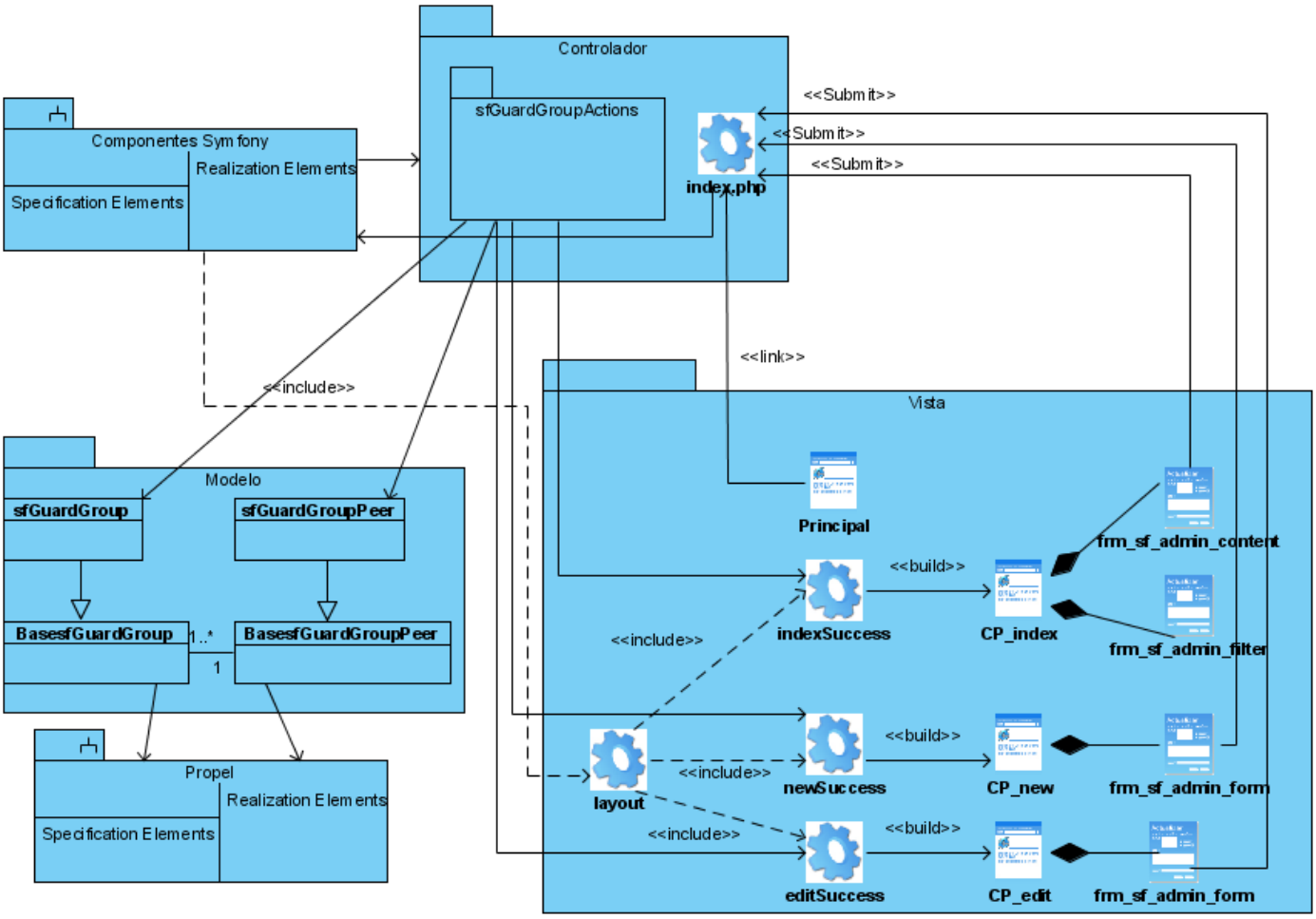


Figura 9.Diagrama de clases del diseño CUS-3 Gestionar Rol

En el Paquete Controlador:

La clase *sfGuardGroupActions* hereda de la clase Bases*fGuardGroupActions, clase que* contiene el código específico de las acciones del módulo sf_guard_group.

Para mejor comprensión se muestra el paquete Controlador en la figura 10.

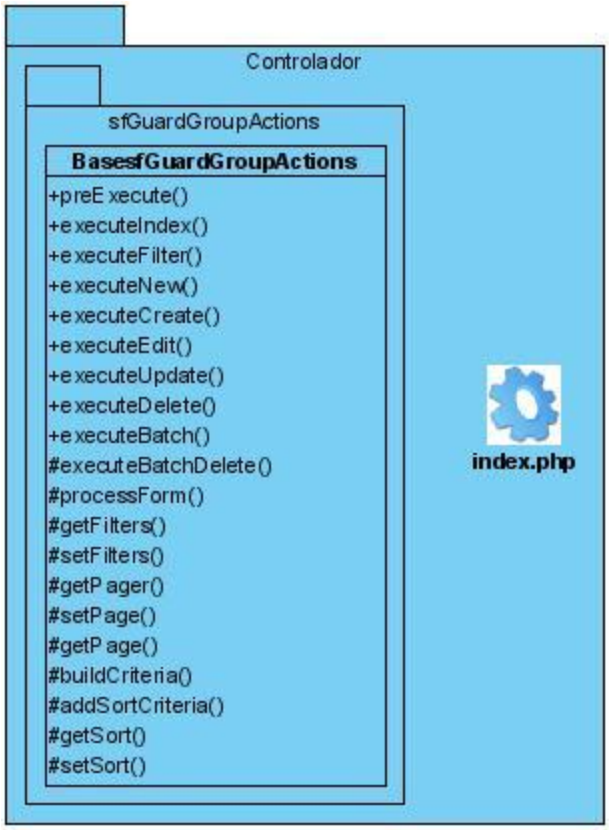


Figura 10.Paquete Controlador del CUS-3 Gestionar Rol

En el Paquete Vista:

- La clase *layout* contiene los elementos que se muestran de forma idéntica para las páginas web a lo largo de toda la aplicación. Incluye el código de las páginas *indexSuccess*, *newSuccess* y *ediSuccesst*.
- La página *Principal* es la página principal del sistema.
- La página *indexSuccess* contiene toda la información de los roles.
- La página *CP_index,* que es construida por la página *indexSuccess,* contiene los formularios *frm_sf_admin_content* que muestra el listado de los roles y brinda la opción de eliminar al mismo de la aplicación y el sf_admin_filter que permite filtrar el rol.
- La página *newSuccess* se encarga de agregar un nuevo rol a la aplicación.
- La página *CP_new,* que es construida por la página *newSuccess,* contiene el formulario *frm_sf_admin_form* que permite insertar el rol.
- La página *editSuccess* es la encargada de modificar la información del rol.
- La página *CP_edit,* que es construida por la página *editSuccess,* contiene el formulario *frm_sf_admin_form* que permite realizarle modificaciones a los datos del rol.

En la figura 11 se muestra el paquete Vista.

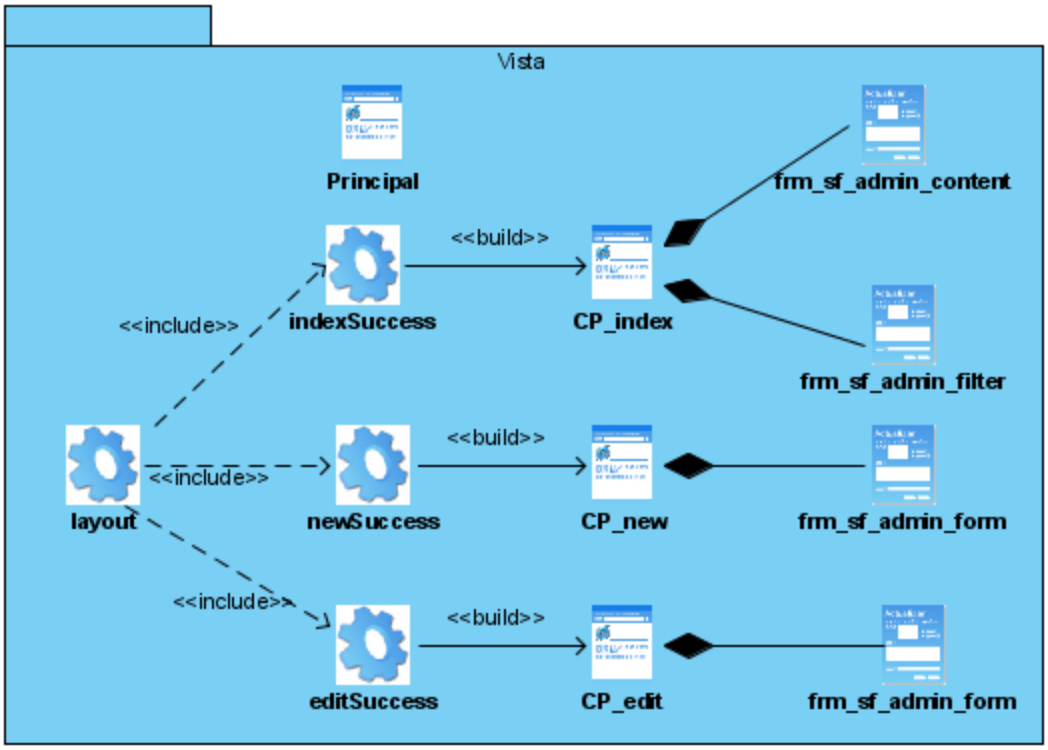


Figura 11.Paquete Vista del CUS-3 Gestionar Rol

En el Paquete Modelo:

Las clases *sfGuardGroup* contiene la lista de roles.

La clase *sfGuardGroupPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *sfGuardGroup*.

Una instancia de la clase *BasesfGuardGroupPeer* puede crear una o más instancias de la clase *BasesfGuardGroup.*

En la figura 12 se muestra el paquete Modelo.

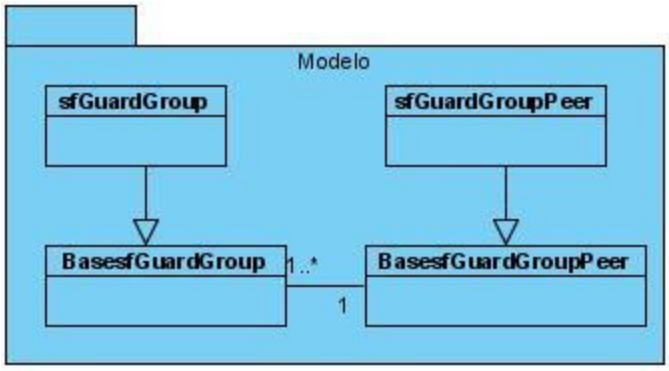


Figura 12.Paquete Modelo del CUS-3 Gestionar Rol

En la figura 13 se muestra el diagrama de clases del diseño correspondiente al **CUS_4 Gestionar Permisos.**

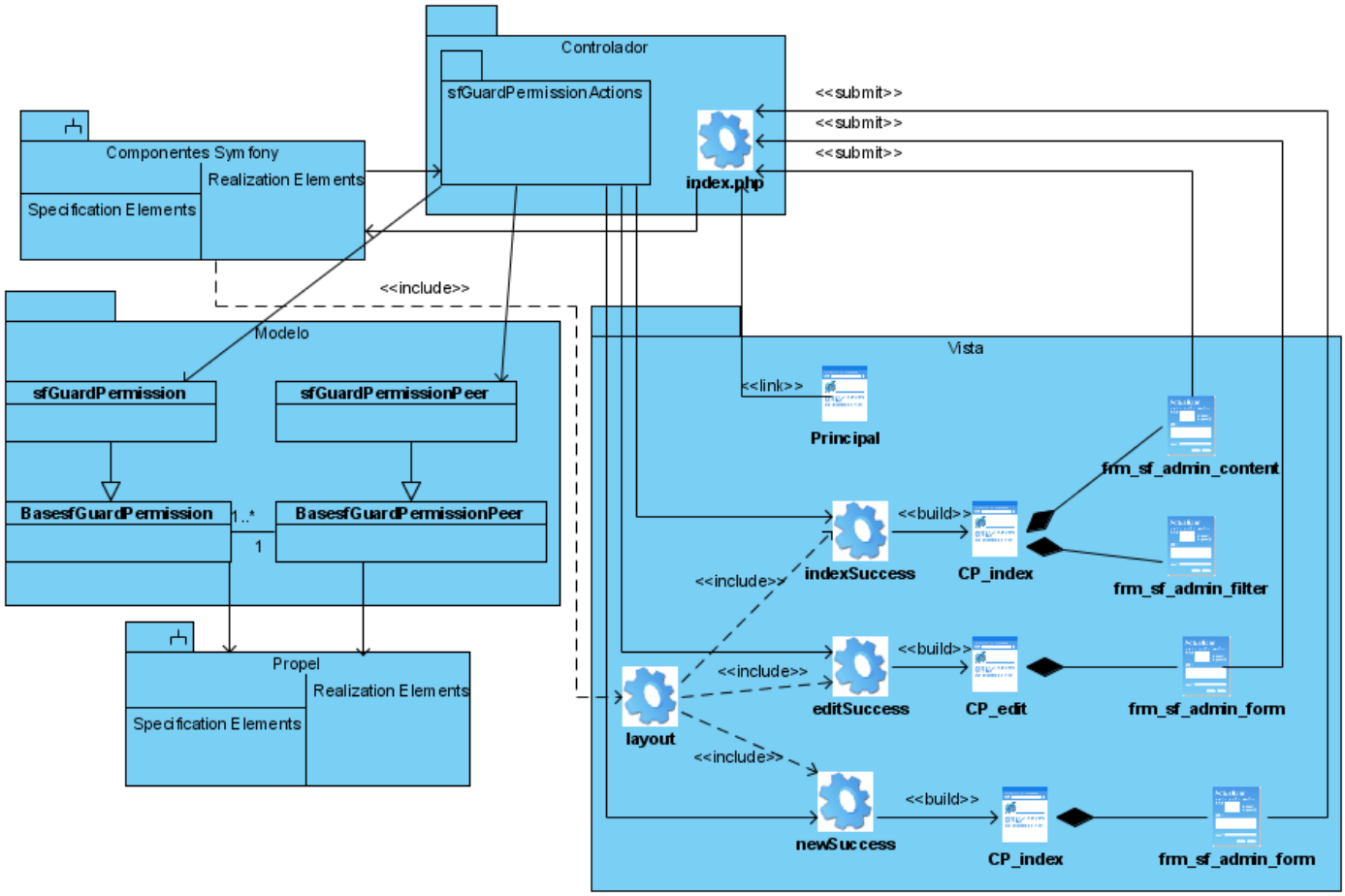


Figura 13.Diagrama de clases del diseño CUS-4 Gestionar Permisos

En el Paquete Controlador:

La clase *sfGuardPermissionActions* hereda de la clase Base*sfGuardPermissionActions, clase que* contiene el código específico de las acciones del módulo sf_guard_permission.
Para mejor comprensión se muestra el paquete Controlador en la figura 14.

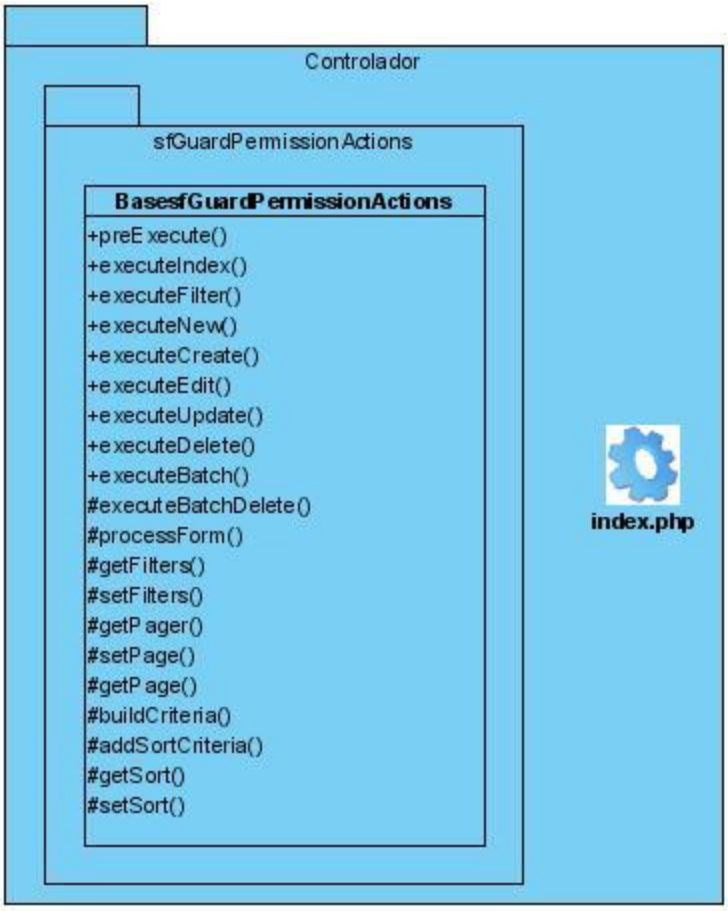


Figura 14.Paquete Controlador del CU-4 Gestionar Permisos

En el Paquete Vista:

- La clase *layout* contiene los elementos que se muestran de forma idéntica para las páginas web a lo largo de toda la aplicación. Incluye el código de las páginas *indexSuccess*, *newSuccess* y *editSuccess*.
- La página *Principal* es la página principal del módulo.
- La página *newSuccess* se encarga de agregar un nuevo permiso a la aplicación.
- La página *CP_new,* que es construida por la página *newSuccess,* contiene el formulario *frm_sf_admin_form* que permite insertar el permiso.
- La página *editSuccess* es la encargada de editar la información del permiso.

- La página *CP_edit,* que es construida por la página *editSuccess,* contiene el formulario *frm_sf_admin_form* que edita los datos del permiso y permite realizarle modificaciones.
- La página *indexSuccess* contiene toda la información de los permisos.
- La página *CP_index,* que es construida por la página *indexSuccess,* contiene los formularios *frm_sf_admin_content* que muestra el listado de los usuarios y brinda la opción de eliminar al mismo de la aplicación y el frm_sf_admin_filter que permite filtrar un usuario.

En la figura 15 se muestra el paquete Vista.

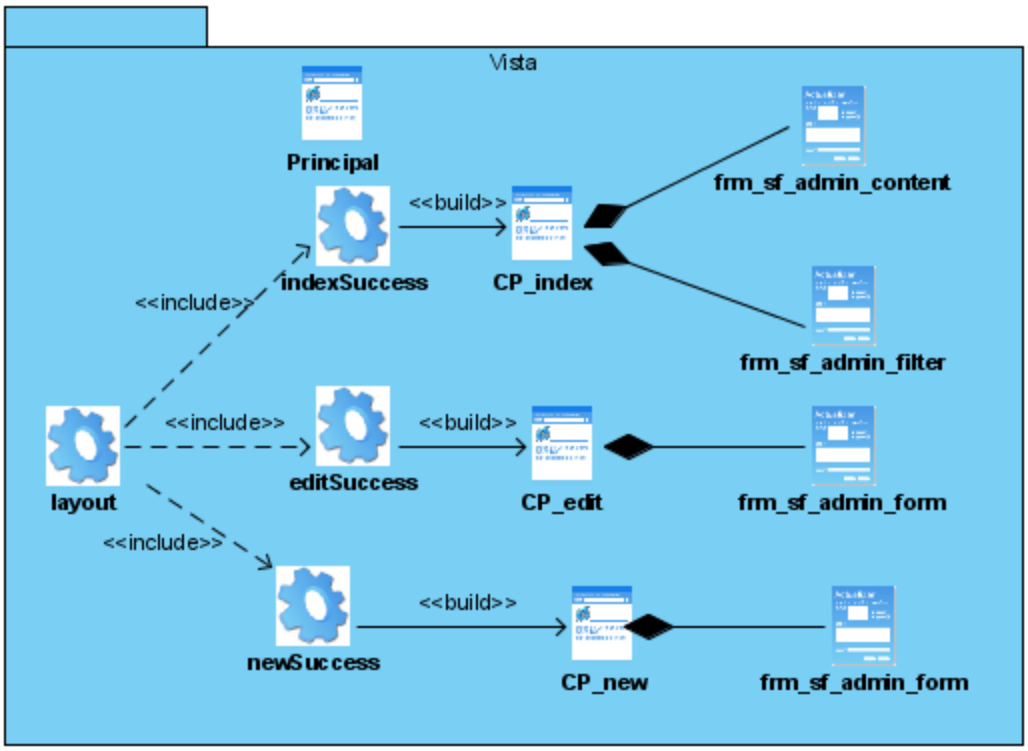


Figura 15.Paquete Vista del CUS-4 Gestionar permisos

En el Paquete Modelo:

La clase *sfGuardPermission* contiene la lista de permisos.

La clase *sfGuardPermissionPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *sfGuardPermission*.

Una instancia de la clase *BasesfGuardPermissionPeer* puede crear una o más instancias de la clase *BasesfGuardPermission.*

En la figura 16 se muestra el paquete Modelo.

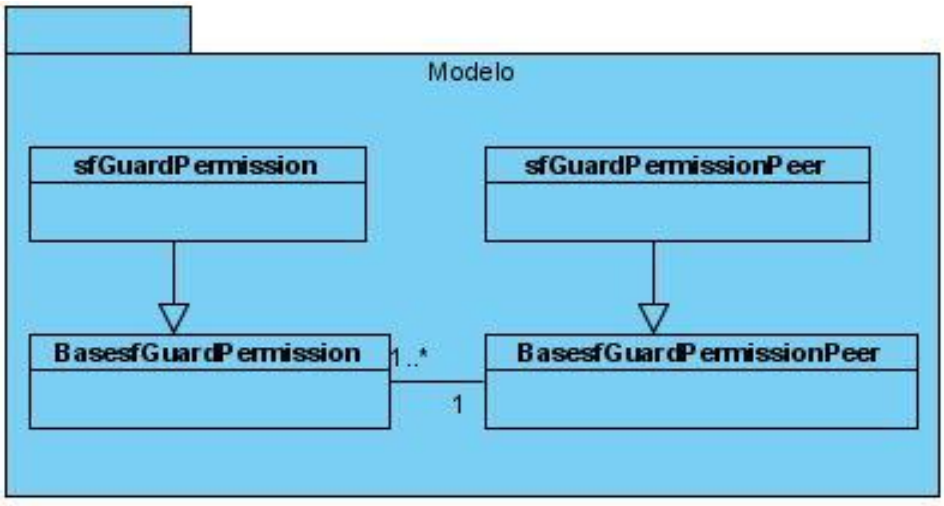


Figura 16.Paquete Modelo del CUS-4 Gestionar Permisos

En la figura 17 se muestra el diagrama de clases del diseño correspondiente al **CUS_5 Gestionar Auditoría.**

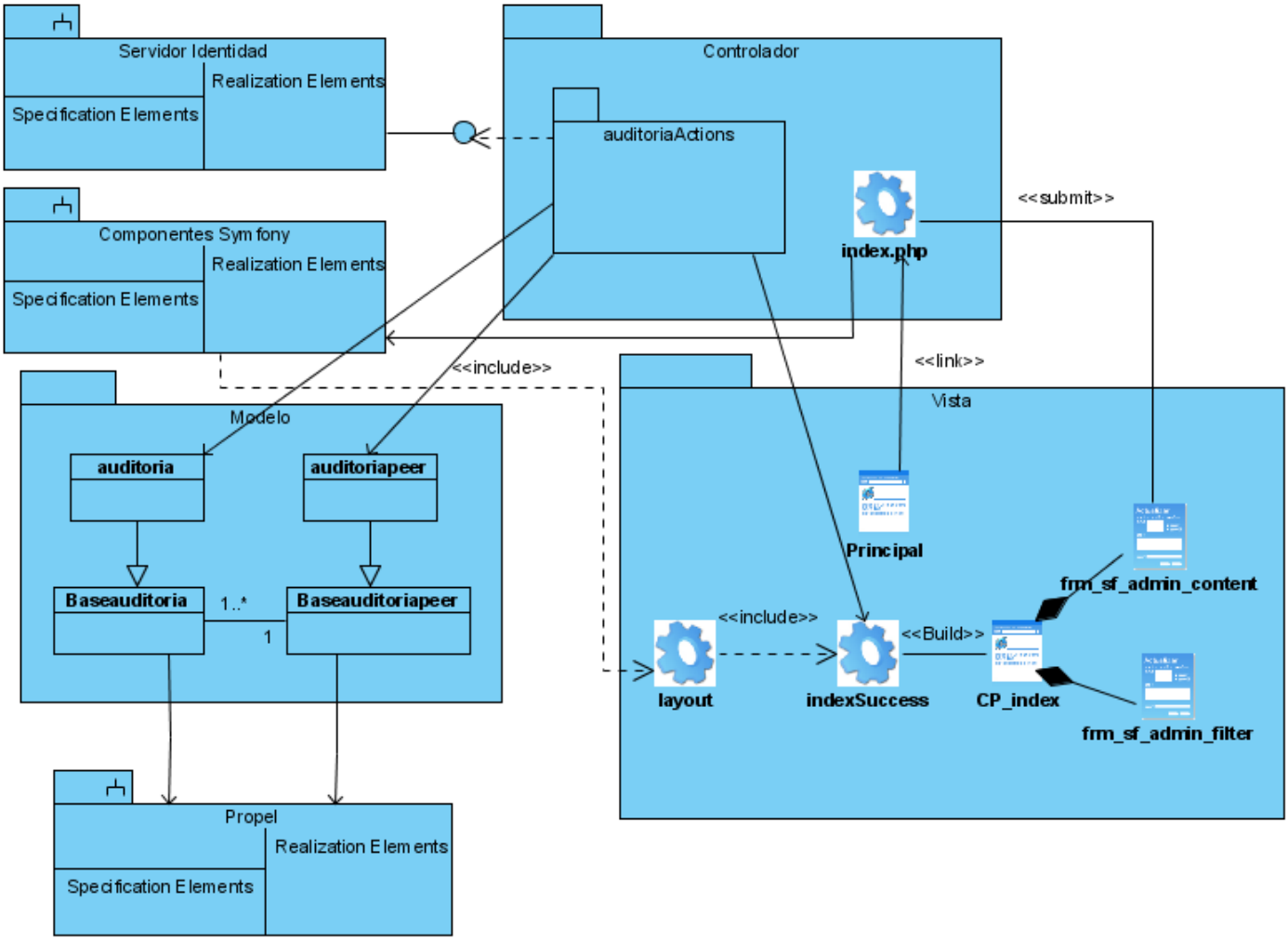


Figura 17.Diagrama de clases del diseño CUS-5 Gestionar Auditoría

Servidor Identidad: Representa al Servidor Web Identidad de la UCI, el cual se emplea para obtener un usuario dado su nombre de usuario del dominio UCI.

En el Paquete Controlador:

La clase *auditoriaActions* contiene el código específico de las acciones del módulo auditoria.

Para mejor comprensión se muestra el paquete Controlador en la figura 18.

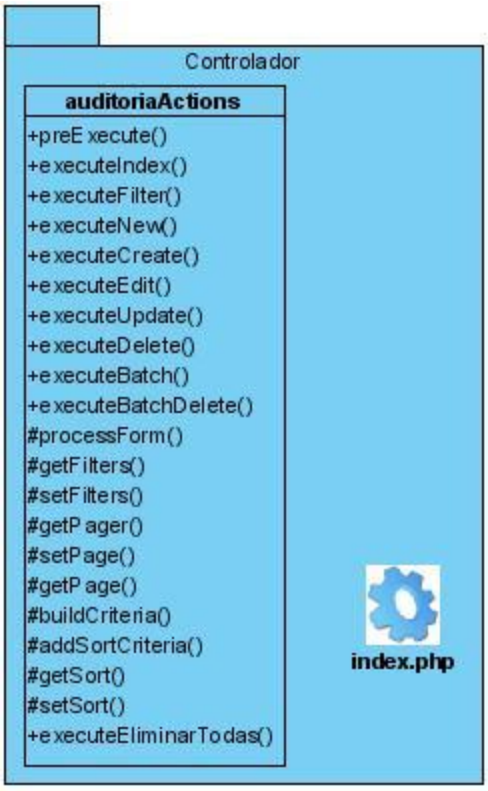


Figura 18.Paquete Controlador del CU-5 auditoria

En el Paquete Vista:

- La clase *layout* contiene los elementos que se muestran de forma idéntica para las páginas web a lo largo de toda la aplicación. Incluye el código de la página *indexSuccess*.
- La página *Principal* es la página principal del módulo.
- La página *indexSuccess* contiene toda la información de los registros de las actividades de los usuarios.
- La página *CP_index,* que es construida por la página *indexSuccess,* contiene los formularios *frm_sf_admin_content* que muestra el listado de los registros y brinda la opción de eliminar el mismo de la aplicación y el frm_sf_admin_filter que permite filtrar un registro.

En la figura 19 se muestra el Paquete Vista.

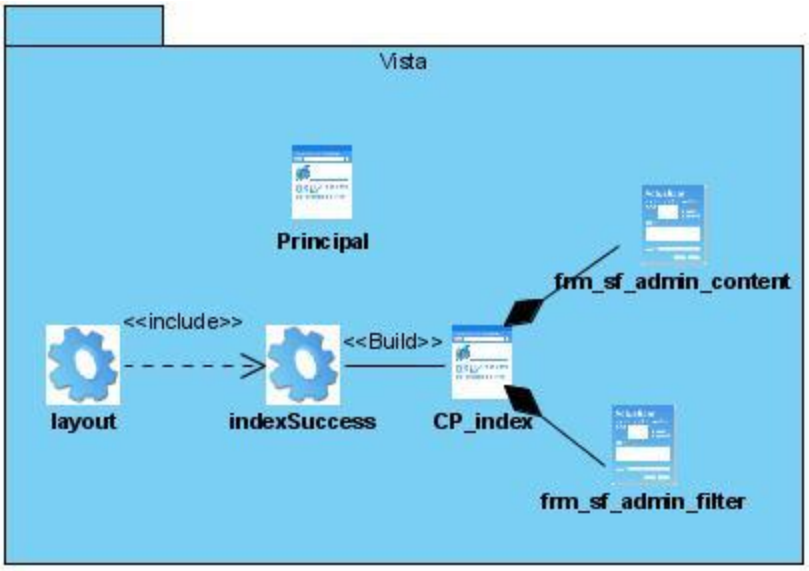


Figura 19.Paquete Vista del CUS-5 auditoria

En el Paquete Modelo:

La clase *auditoria* contiene la lista de los registros.

La clase *auditoriaPeer* contiene métodos estáticos que devuelven un objeto o una colección de objetos de la clase *auditoria*.

Una instancia de la clase *BaseauditoriaPeer* puede crear una o más instancias de la clase *Baseauditoria.*

En la figura 20 se muestra el paquete Modelo.

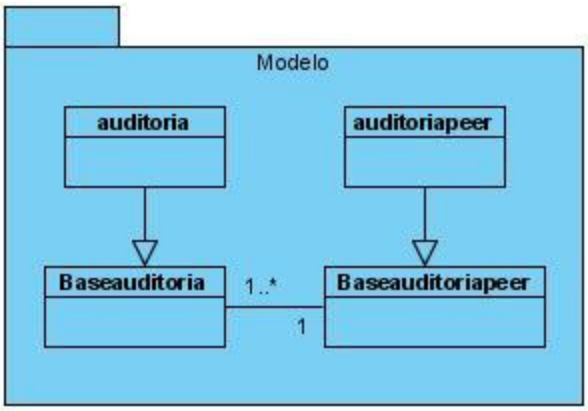


Figura 20.Paquete Modelo del CUS-5 auditoria

2.3.2 Diagramas de secuencia.

Los diagramas de secuencia muestran en detalle cómo interactúan las clases y guían al programador en la fase de implementación. A continuación se exponen los mismos:

La Figura 21 muestra la interacción entre las clases que intervienen en la entrada del usuario a la aplicación. En este diagrama aparece la clase sfController que se encarga de determinar la acción a ejecutarse y de cargar la configuración, cada diagrama de secuencia que se presenta a continuación muestra esta clase. La interacción concluye al comprobar si el usuario tiene acceso o no a los servicios que brinda la aplicación.

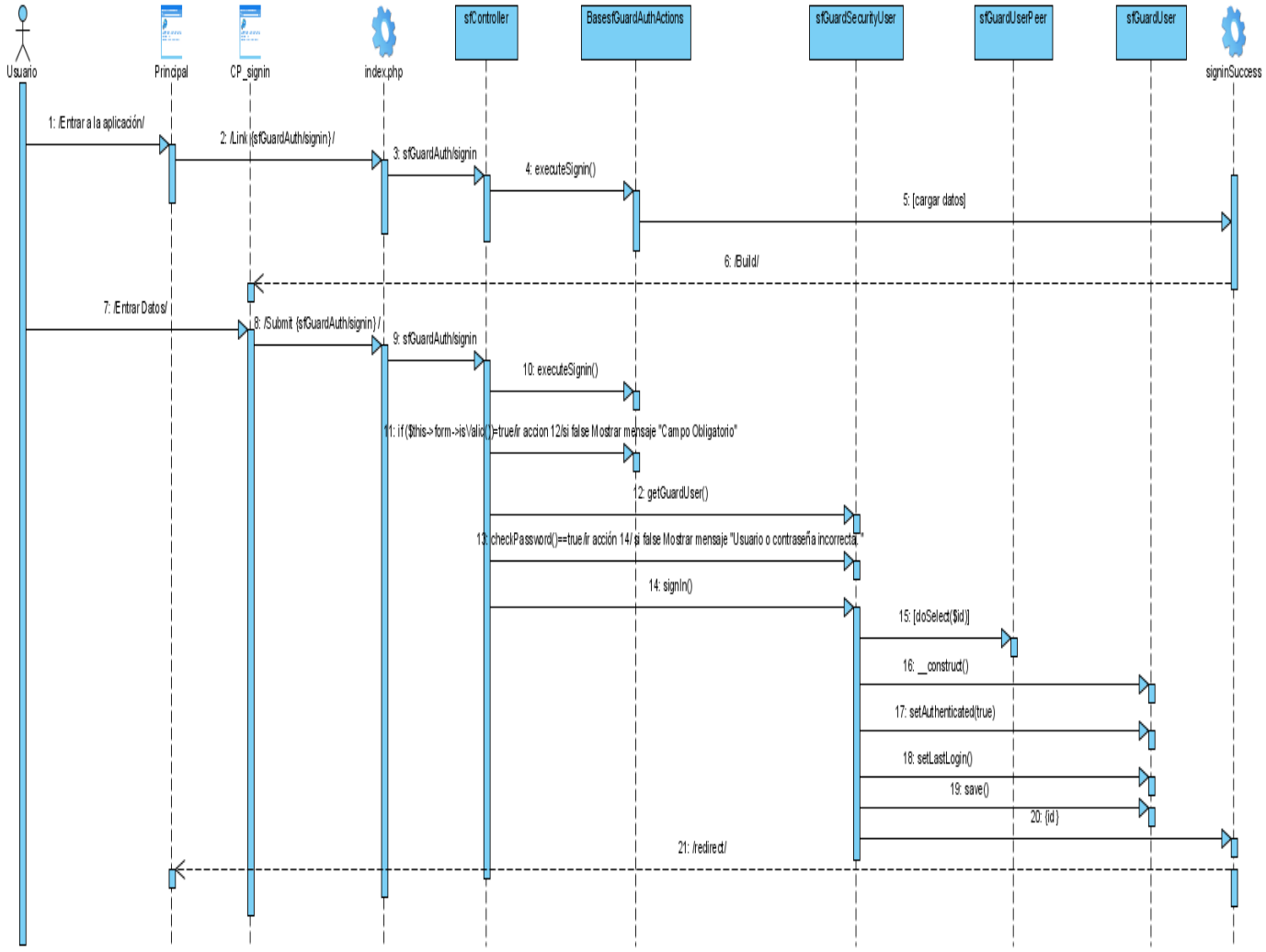


Figura 21.CUS Autenticar Usuario Escenario signin.

La figura 22 muestra la interacción entre las clases que intervienen en la creación de un nuevo usuario. Este escenario se encarga de validar los datos del usuario: comprobar que no se encuentre registrado anteriormente en la Base de Datos y que la contraseña cumpla con los requerimientos necesarios.

Además, permite la asignación de los roles en los que se desempeñará el usuario en la aplicación y de los permisos necesarios para su interacción con la misma. Este escenario concluye cuando se muestra el mensaje “Usuario creado satisfactoriamente. Puede adicionar otro” en caso de que los datos proporcionados sean correctos. En caso contrario muestra el mensaje: “El usuario no fue creado debido a algunos errores” y en el formulario se señalan los campos incorrectos.

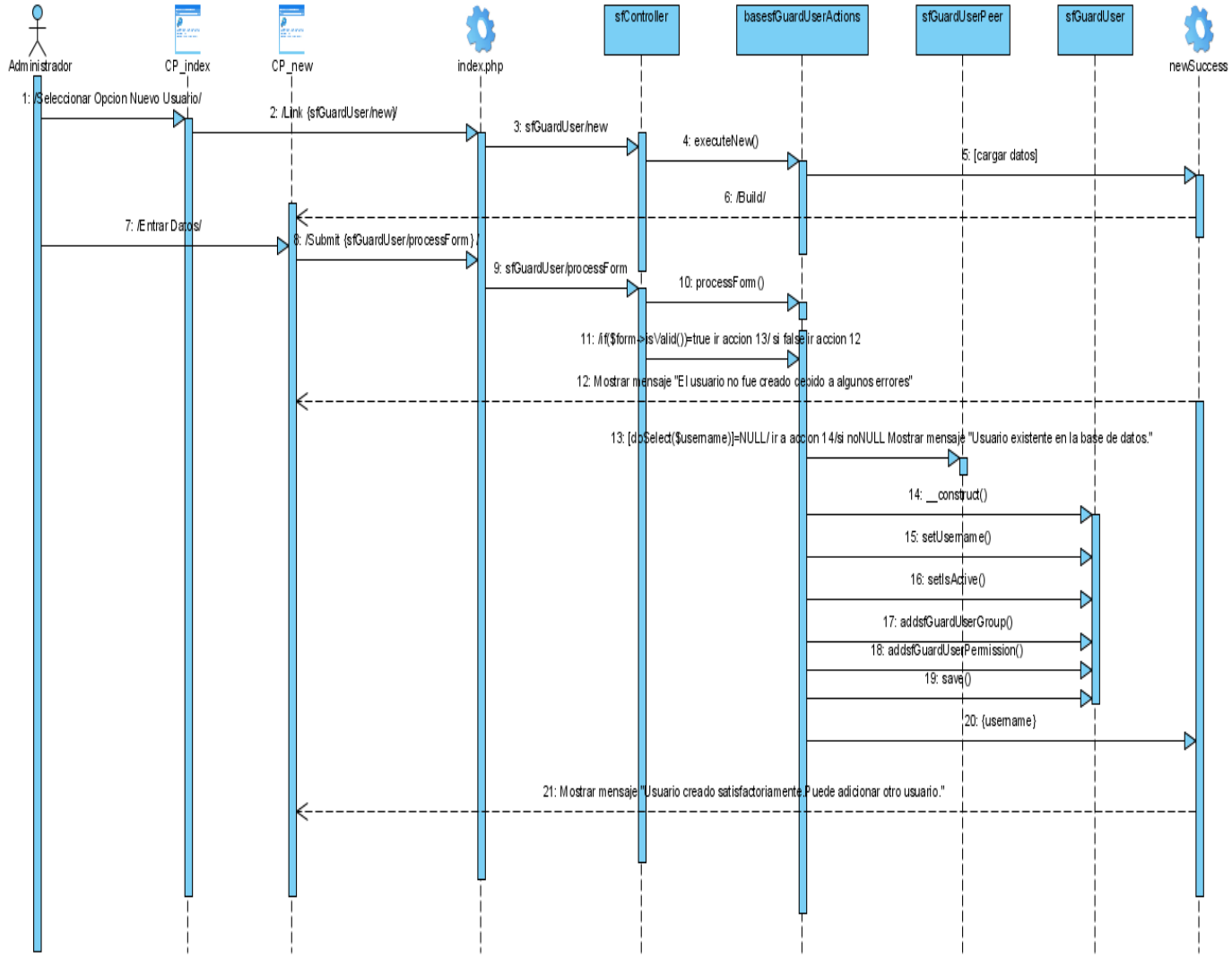


Figura 22.CUS Gestionar Usuario Escenario Nuevo Usuario

La figura 23 se muestra la interacción entre las clases que intervienen cuando el administrador edita un usuario para realizar algún cambio en sus datos. Cuando esta acción es ejecutada satisfactoriamente, el sistema muestra el mensaje “Usuario modificado satisfactoriamente”. En caso contrario muestra el mensaje “El usuario no fue creado debido a algunos errores” y en el formulario se señalan los campos incorrectos.

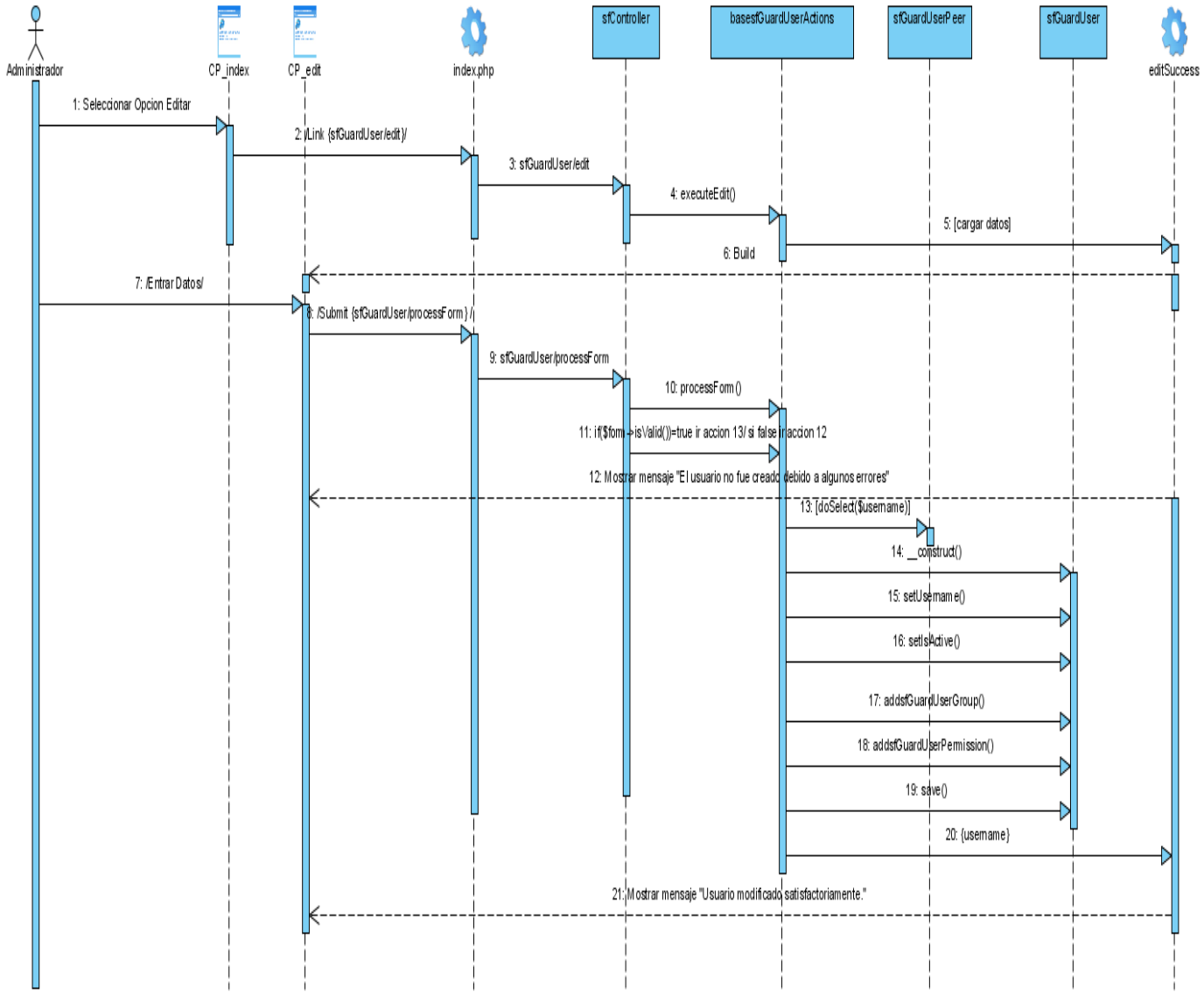


Figura 23.CUS Gestionar Usuario Escenario Editar

La figura 24 muestra la interacción entre las clases que intervienen en el proceso de eliminar un usuario. Al seleccionar la opción de eliminar, el sistema muestra un mensaje para la confirmación de la acción, en caso de elegir la opción Aceptar es que se procede a ejecutar la acción de eliminar el usuario. El escenario concluye cuando el usuario ya no se encuentra en la Base de Datos. Cuando se realiza esta operación se muestra el mensaje" Usuario eliminado satisfactoriamente".

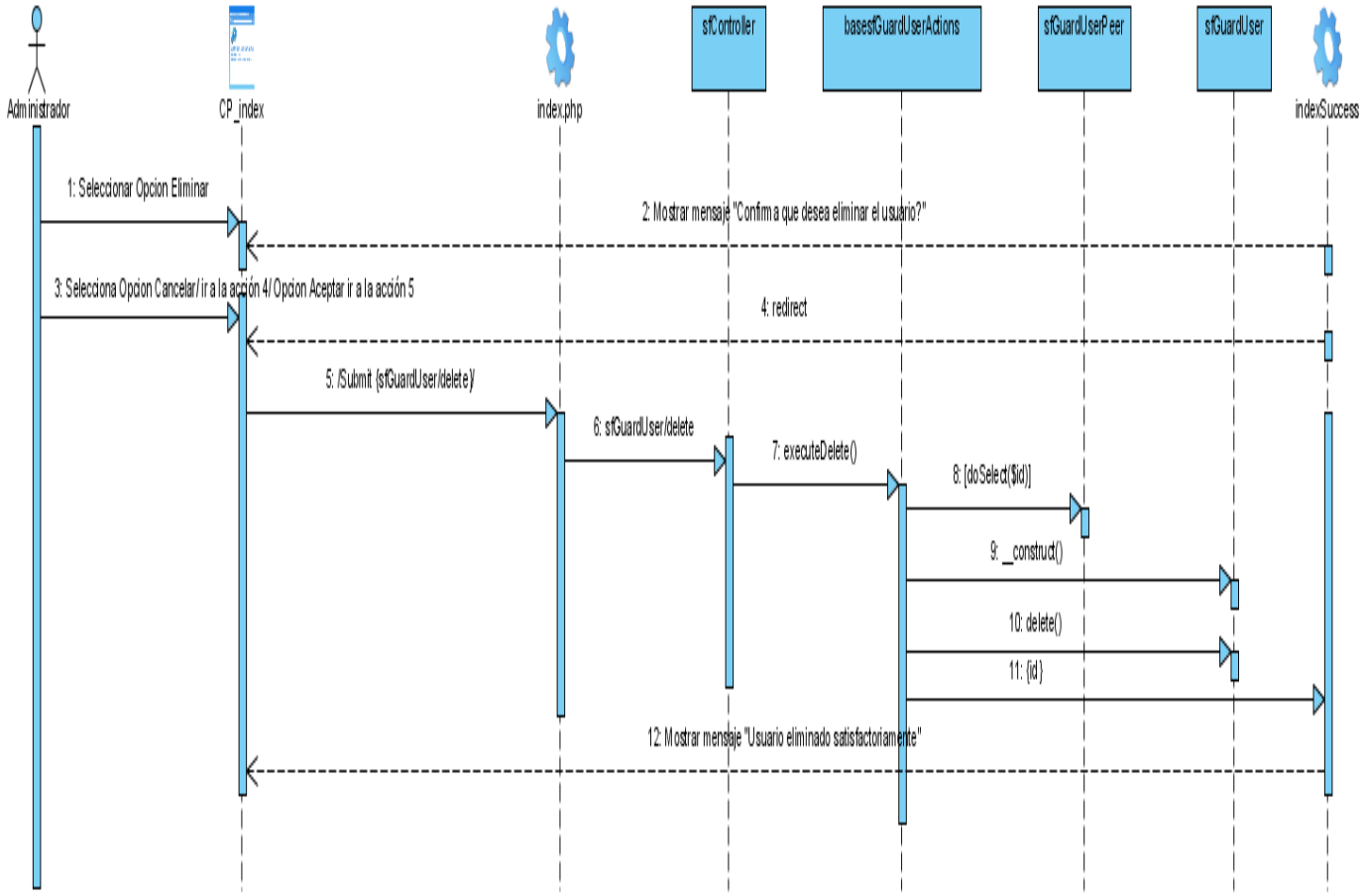


Figura 24.CUS Gestionar Usuario Escenario Eliminar

La figura 25 muestra la interacción entre las clases que intervienen en la muestra del listado de todos los usuarios que están registrados en la Base de Datos de la aplicación.

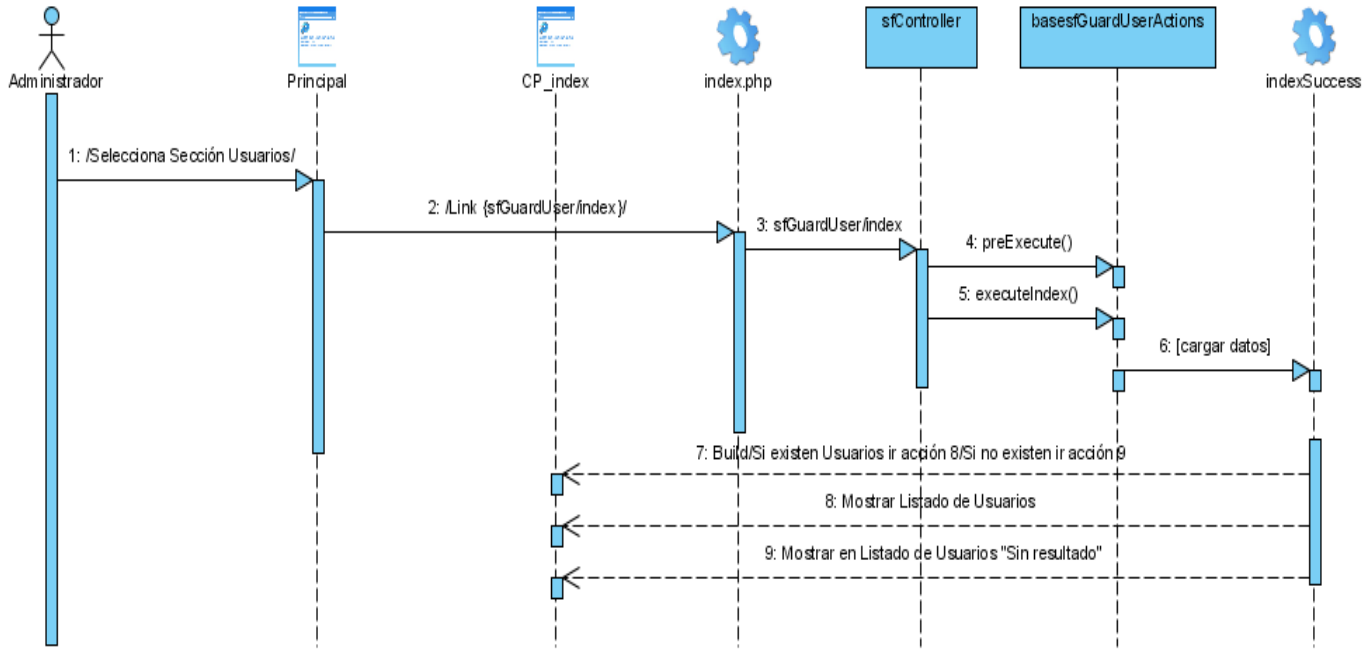


Figura 25.CUS Gestionar Usuario Escenario Listar Usuario

La figura 26 muestra cómo interactúan las clases que intervienen en el proceso de filtrar usuarios. El administrador puede obtenerlos por los parámetros de nombre de usuario, roles o permisos que del sistema. Los datos que devuelve la búsqueda se muestran en el listado de usuarios, en caso de que no lo encuentre muestra el mensaje “Sin resultado”.

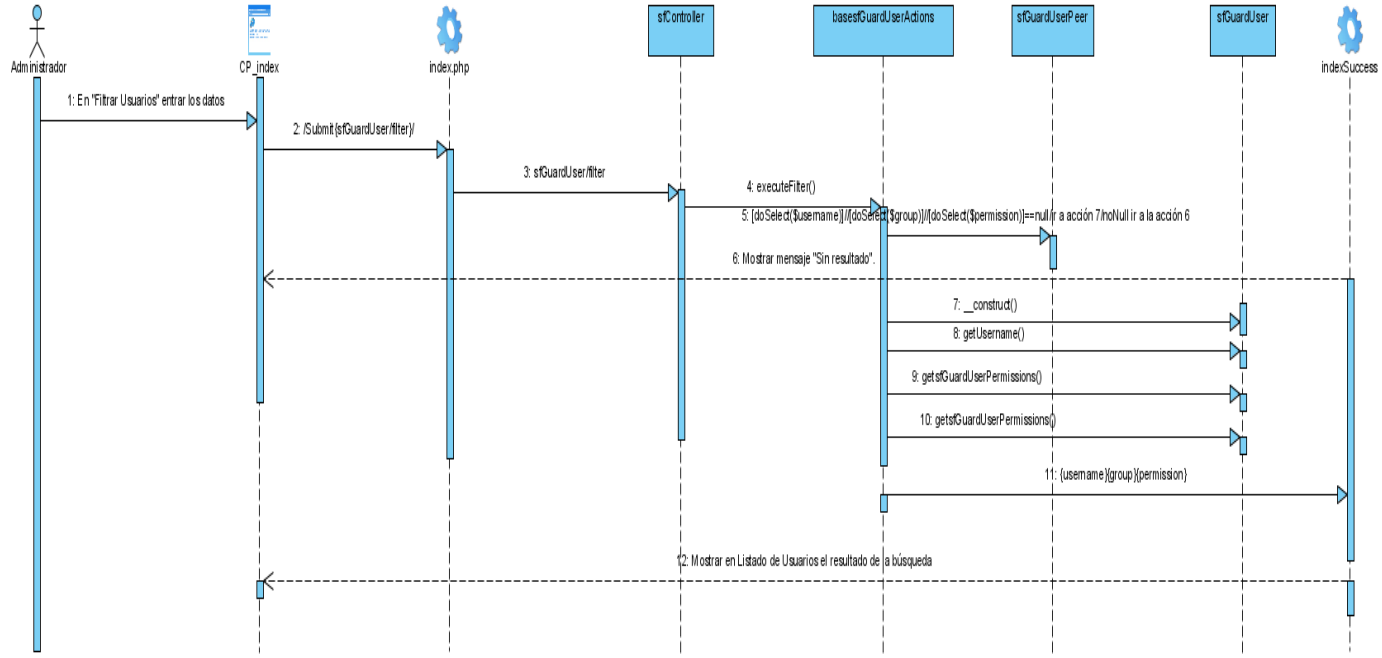


Figura 26.CUS Gestionar Usuarios Escenario Filtrar

La figura 27 muestra la interacción entre las clases que intervienen para importar la información de los usuarios de tipo Estudiante o Profesor. Si es estudiante, permite escoger el año de los usuarios que se desean importar y si es profesor importa la información de todos los profesores de la Facultad 6.

Figura 27.CUS Gestionar Usuario Escenario Importar Usuario

La figura 28 muestra la interacción entre las clases que intervienen en la creación de un nuevo rol. Este escenario se encarga de validar que el nuevo rol no se encuentre anteriormente en la Base de Datos. Concluye cuando se muestra el mensaje “Rol creado satisfactoriamente. Puede adicionar otro” en caso de que los datos proporcionados sean correctos. En caso contrario muestra el mensaje: “El rol no fue creado debido a algunos errores” y en el formulario se señalan los campos incorrectos.

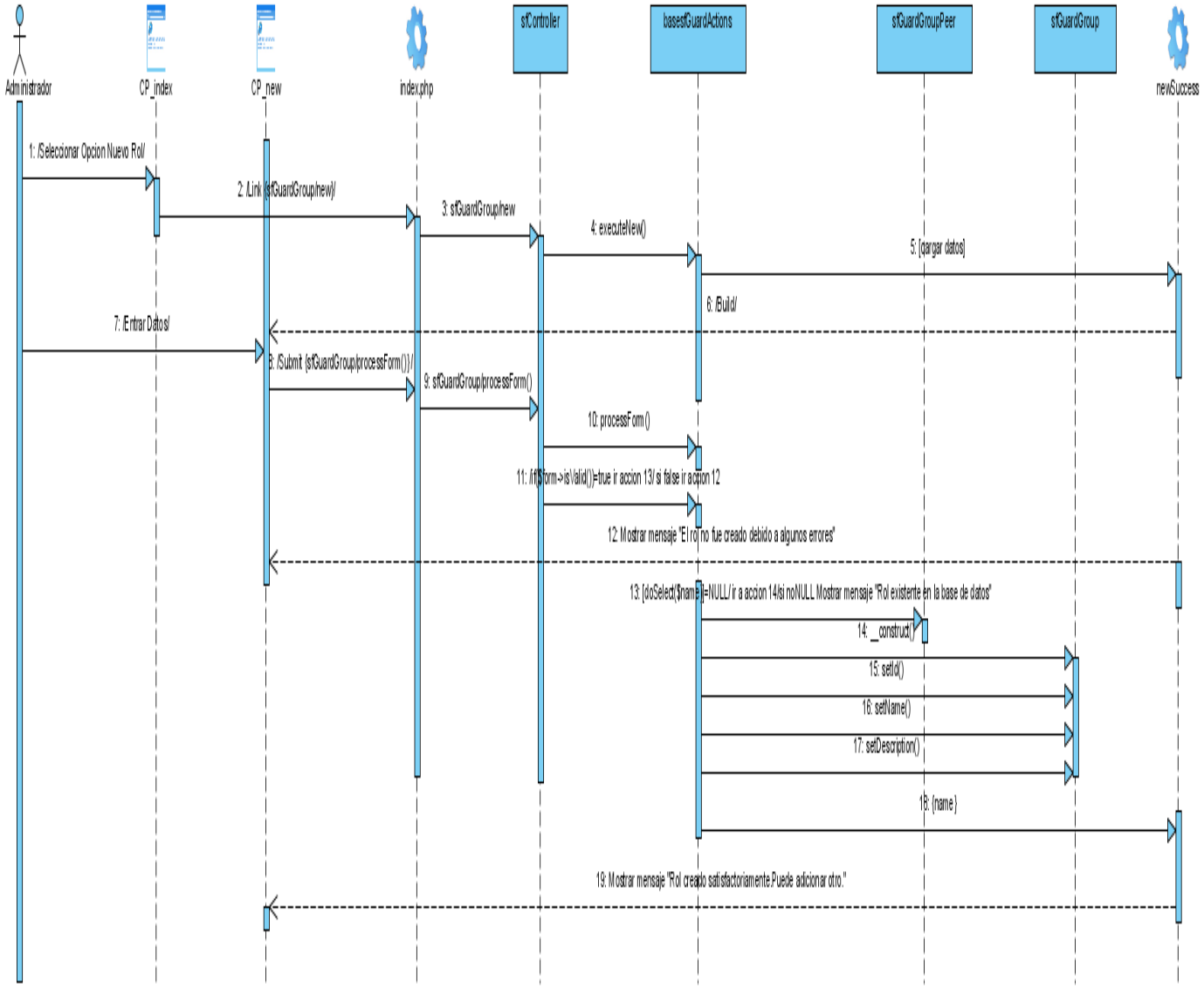


Figura 28.CUS Gestionar Rol Escenario Nuevo Rol

La figura 29 muestra la interacción entre las clases que intervienen cuando el administrador edita un rol para realizar algún cambio en los datos del mismo. Cuando esta acción es ejecutada satisfactoriamente muestra el mensaje “Rol modificado satisfactoriamente”. En caso contrario muestra el mensaje: “El rol no fue creado debido a algunos errores” y en el formulario se señalan los campos incorrectos.

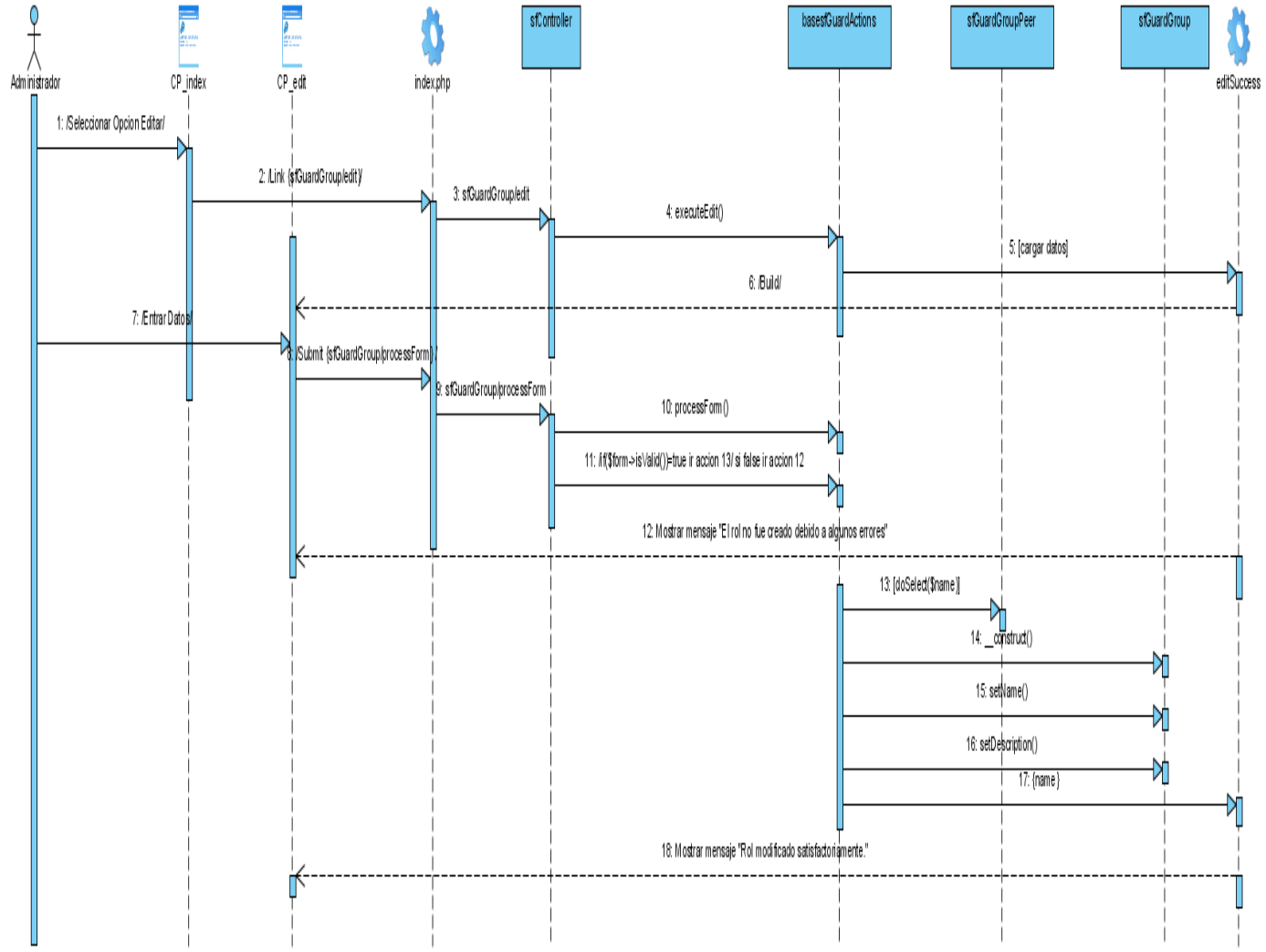


Figura 29.CUS Gestionar Rol Escenario Editar

La figura 30 muestra la interacción entre las clases que intervienen en el proceso de eliminar un rol. Al seleccionar la opción de eliminar, el sistema muestra un mensaje para la confirmación de la acción, en caso de elegir la opción Aceptar es que se procede a ejecutar la acción de eliminar el usuario. El

escenario concluye cuando el rol ya no se encuentra en la Base de Datos. Cuando se realiza esta operación se muestra el mensaje" Rol eliminado satisfactoriamente".

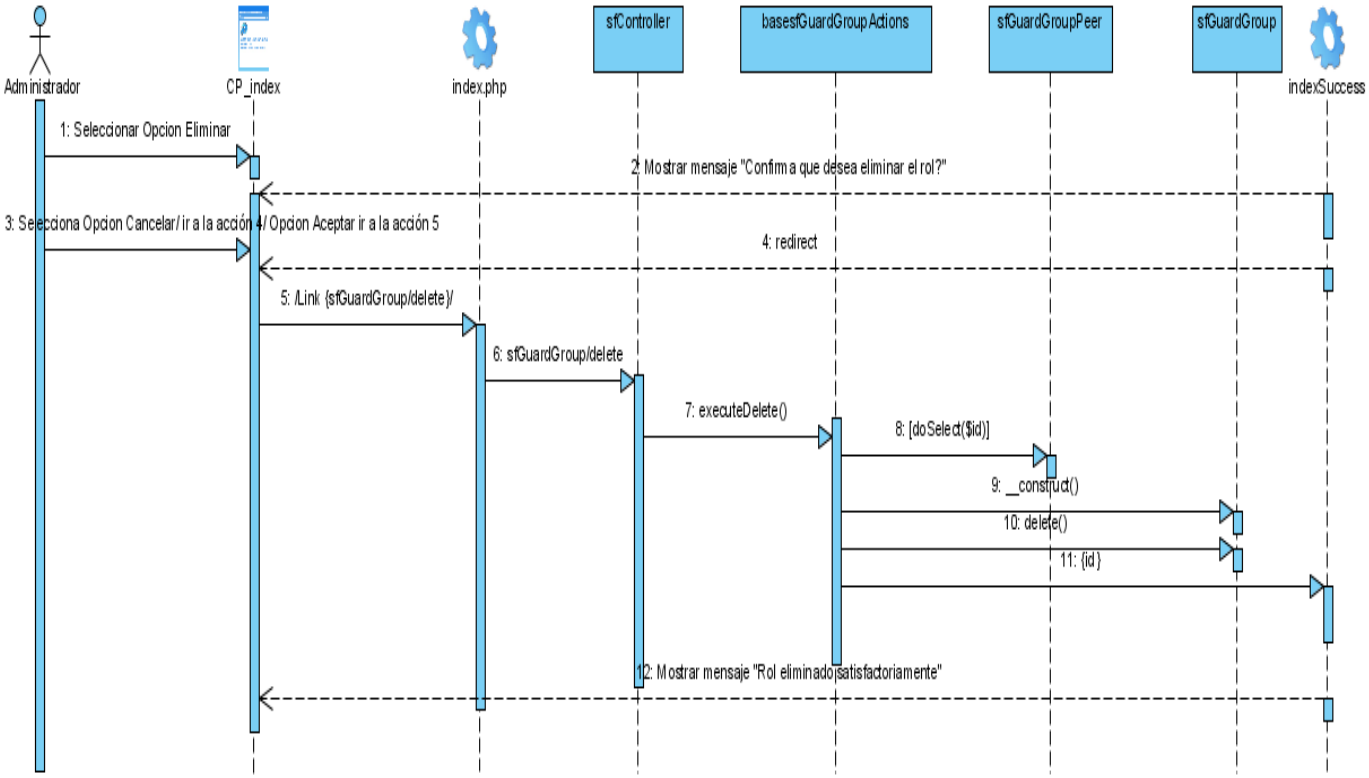


Figura 30.CUS Gestionar Rol Escenario Eliminar

La figura 31 muestra la interacción entre las clases que intervienen en la muestra del listado de todos los roles de los usuarios que están registrados en la Base de Datos de la aplicación.

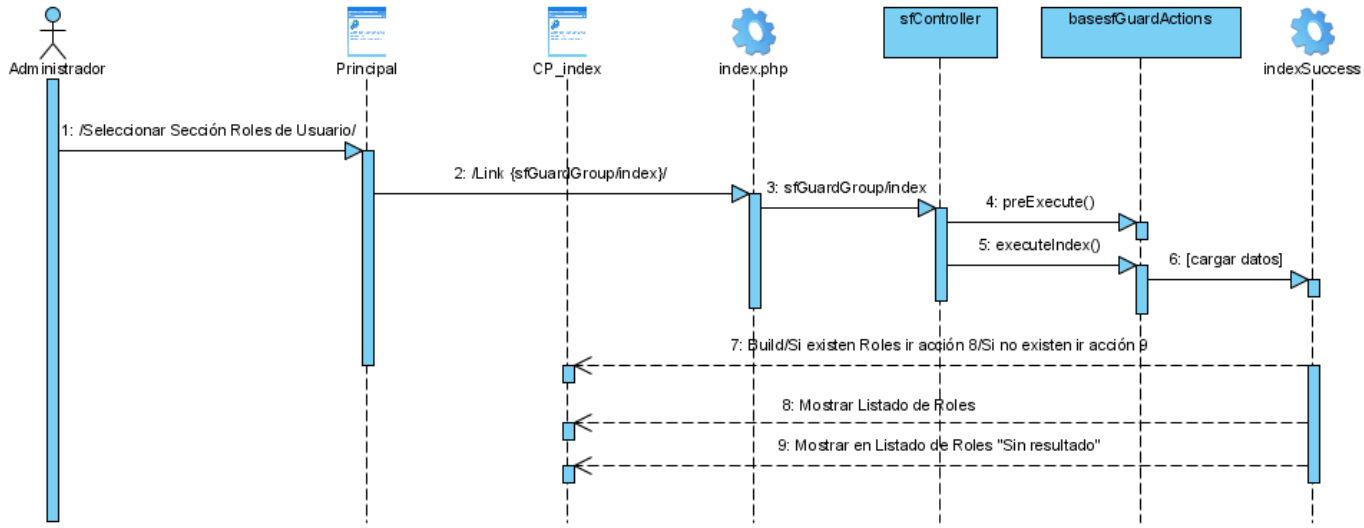


Figura 31.CUS Gestionar Rol Escenario Listar Rol

La figura 32 muestra cómo interactúan las clases que intervienen en el proceso de filtrar un rol. El administrador puede obtenerlos por el parámetro de nombre del rol. Los datos que devuelve la búsqueda se muestran en el listado de roles, en caso de que no lo encuentre muestra el mensaje “Sin resultado”.

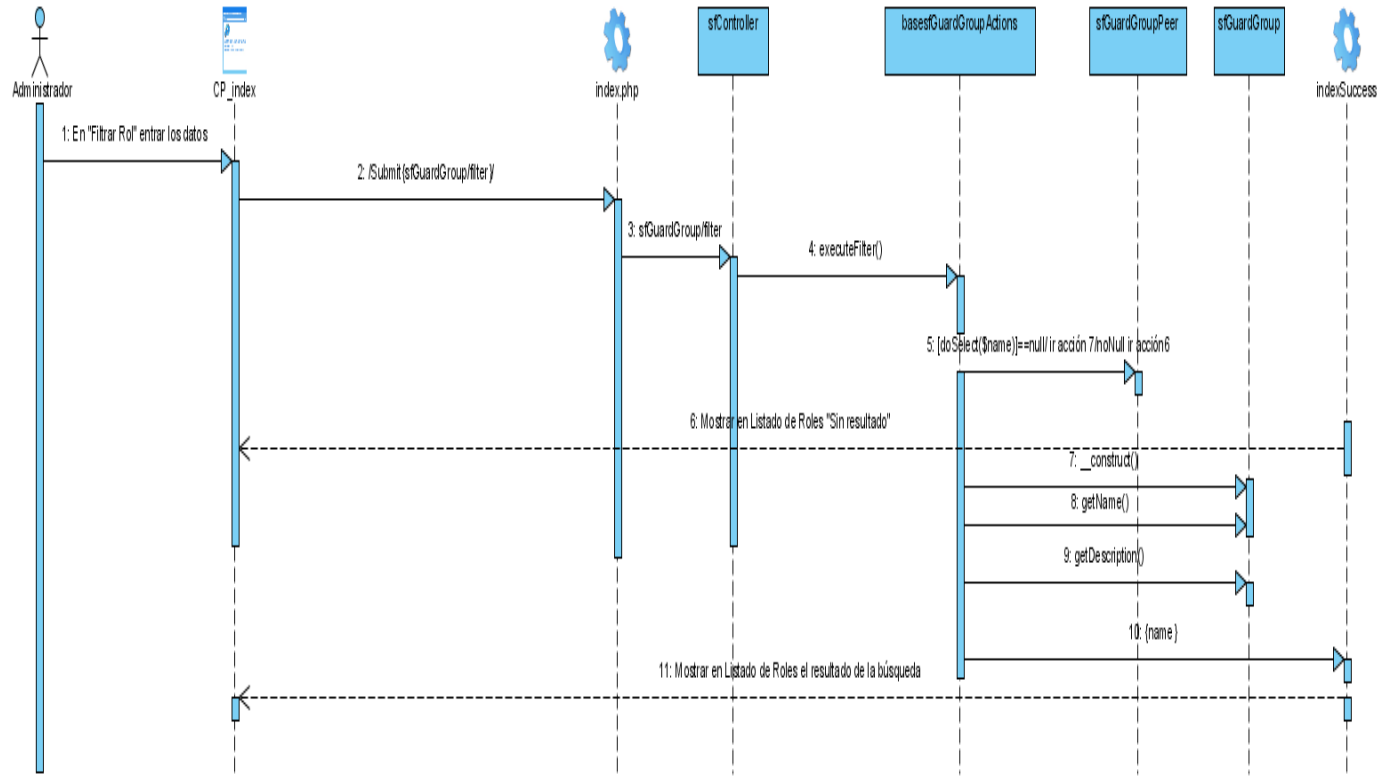


Figura 32.CUS Gestionar Rol Escenario Filtrar

La figura 33 muestra la interacción entre las clases que intervienen en la creación de un nuevo permiso. Este escenario se encarga de validar que el nuevo permiso no se encuentre anteriormente en la Base de Datos. Concluye cuando se muestra el mensaje “Permiso creado satisfactoriamente. Puede adicionar otro” en caso de que los datos proporcionados sean correctos. En caso contrario muestra el mensaje: “El permiso no fue creado debido a algunos errores” y en el formulario se señalan los campos incorrectos.

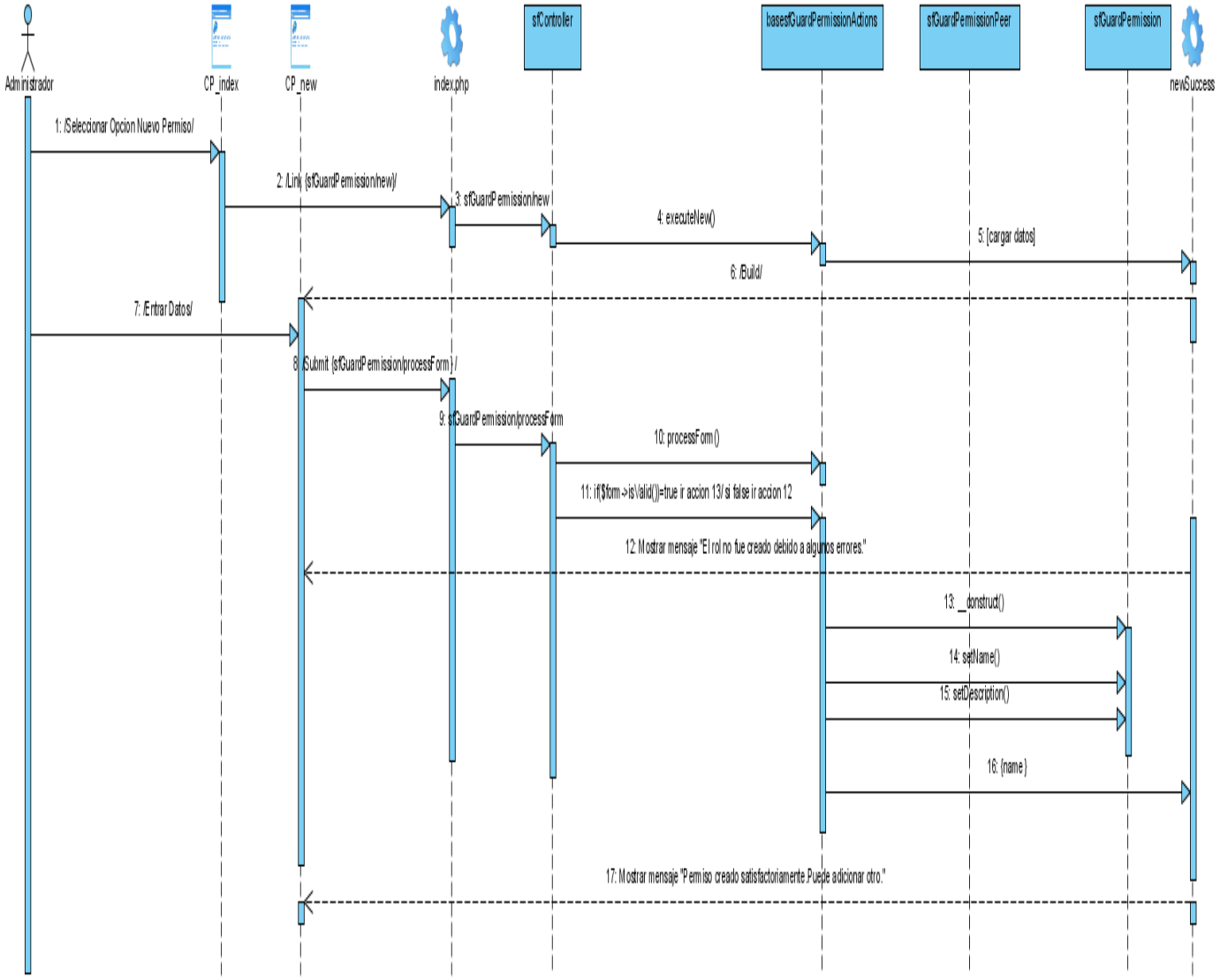


Figura 33.CUS Gestionar Permiso Escenario Nuevo Permiso

La figura 34 muestra la interacción entre las clases que intervienen cuando el administrador edita un permiso para realizar algún cambio en sus datos. Cuando esta acción es ejecutada satisfactoriamente, el sistema muestra el mensaje “Permiso modificado satisfactoriamente”. En caso contrario muestra el mensaje “El permiso no fue creado debido a algunos errores” y en el formulario se señalan los campos incorrectos.

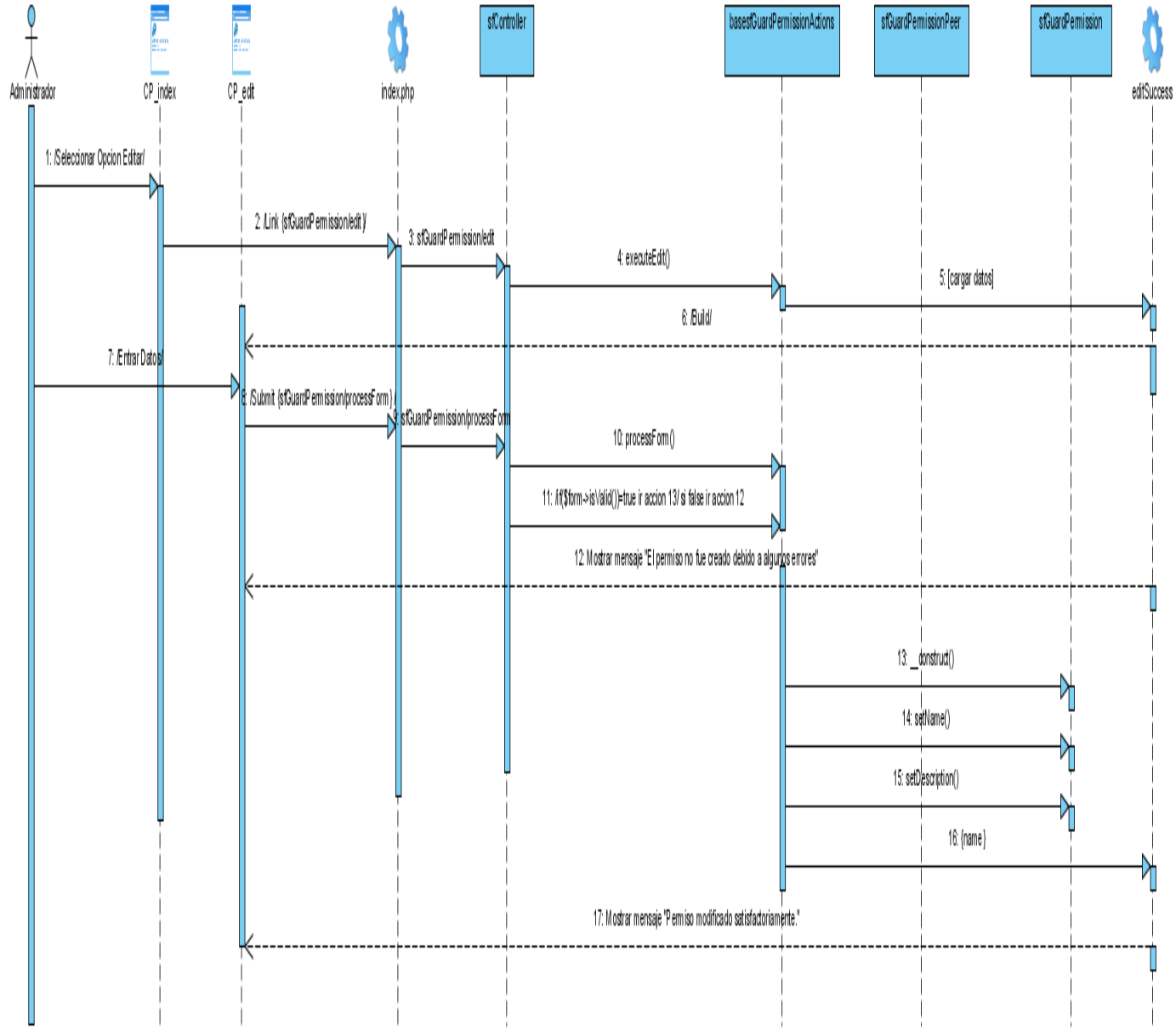


Figura 34.CUS Gestionar Permiso Escenario Editar

La figura 35 muestra la interacción entre las clases que intervienen en el proceso de eliminar un permiso. Al seleccionar la opción de eliminar, el sistema muestra un mensaje para la confirmación de la acción, en caso de elegir la opción Aceptar es que se procede a ejecutar la acción de eliminar el usuario. El escenario concluye cuando el permiso ya no se encuentra en la Base de Datos. Cuando se realiza esta operación se muestra el mensaje" Permiso eliminado satisfactoriamente".

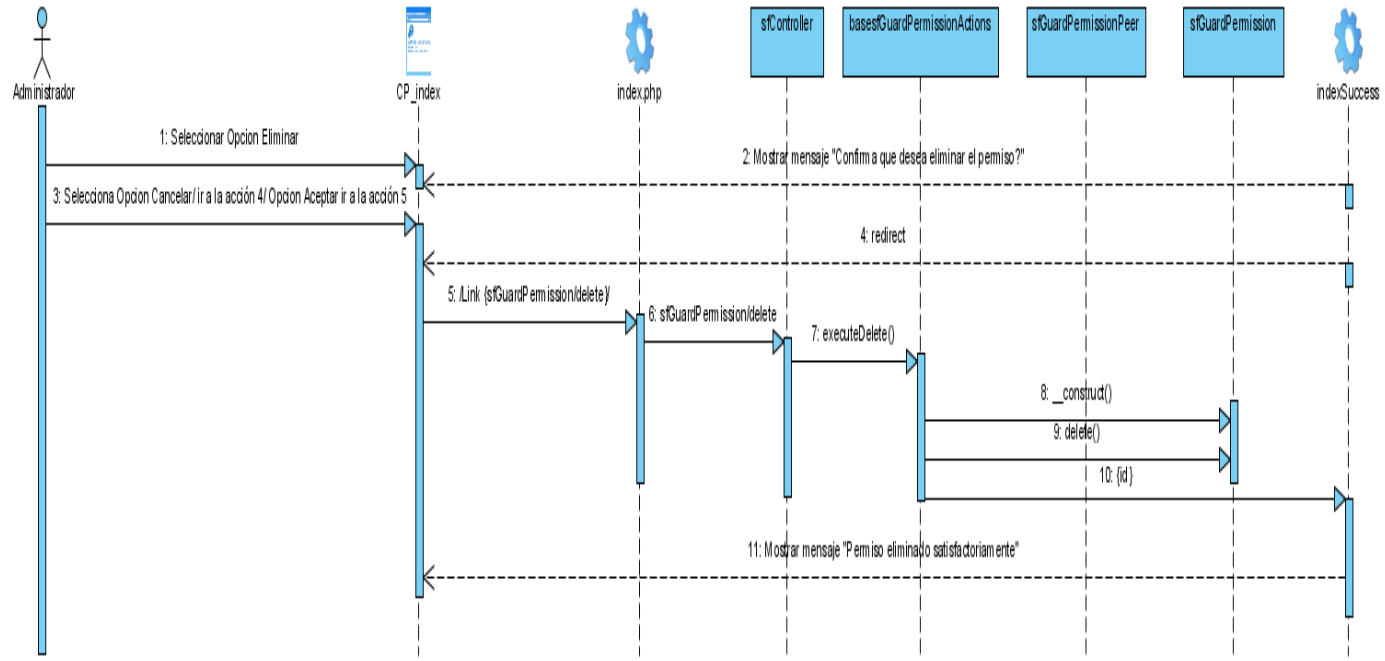


Figura 35.CUS Gestionar Permiso Escenario Eliminar

La figura 36muestra la interacción entre las clases que intervienen en la muestra del listado de todos los permisos que estén registrados en la Base de Datos de la aplicación.

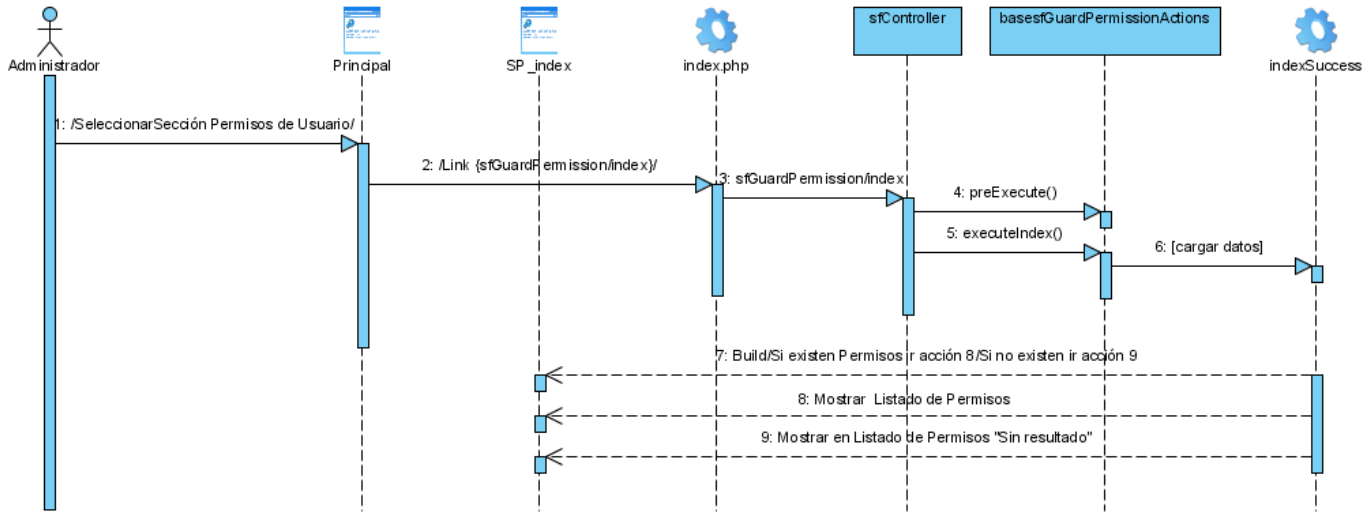


Figura 36.CUS Gestionar Permiso Escenario Listar Permiso

La figura 37 muestra cómo interactúan las clases que intervienen en el proceso de filtrar un permiso. El administrador puede obtenerlos por el parámetro de nombre de permiso. Los datos que devuelve la búsqueda se muestran en el listado de permisos, en caso de que no lo encuentre muestra el mensaje “Sin resultado”.

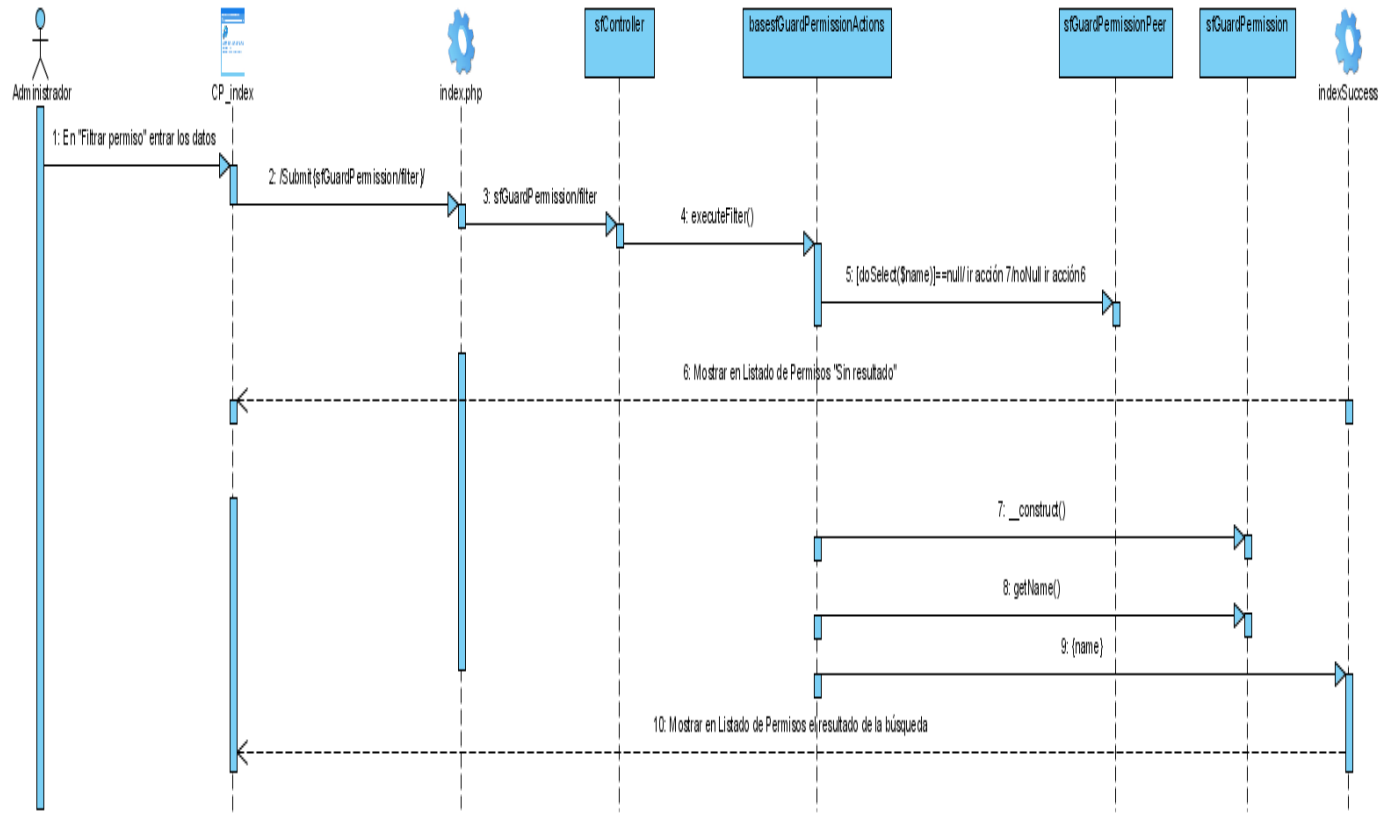


Figura 37.CUS Gestionar Permiso Escenario Filtrar

La figura 38 muestra la interacción entre las clases que intervienen en el proceso de eliminar una auditoria. Al seleccionar la opción de eliminar, en el Registro de Actividades del Usuario el sistema muestra un mensaje para la confirmación de la acción, en caso de elegir la opción Aceptar es que se procede a ejecutar la acción de eliminar la auditoria. El escenario concluye cuando la auditoria ya no se encuentra en la Base de Datos. Cuando se realiza esta operación se muestra el mensaje" el elemento se ha eliminado satisfactoriamente".

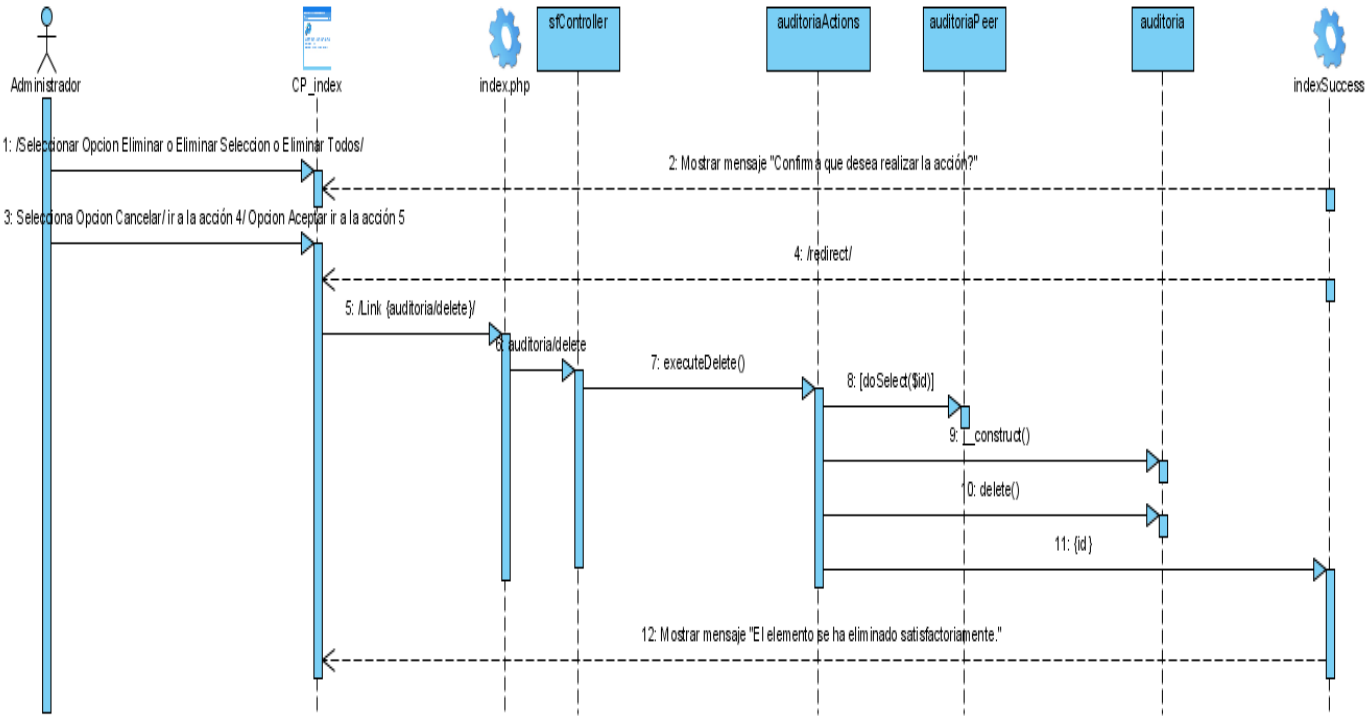


Figura 38.CUS Gestionar auditoría Escenario Eliminar

La figura 39 muestra la interacción entre las clases que intervienen en la muestra del listado de todas las auditorias que están registradas en la Base de Datos de la aplicación, las mismas se muestran en el Registro de Actividades del Usuario.

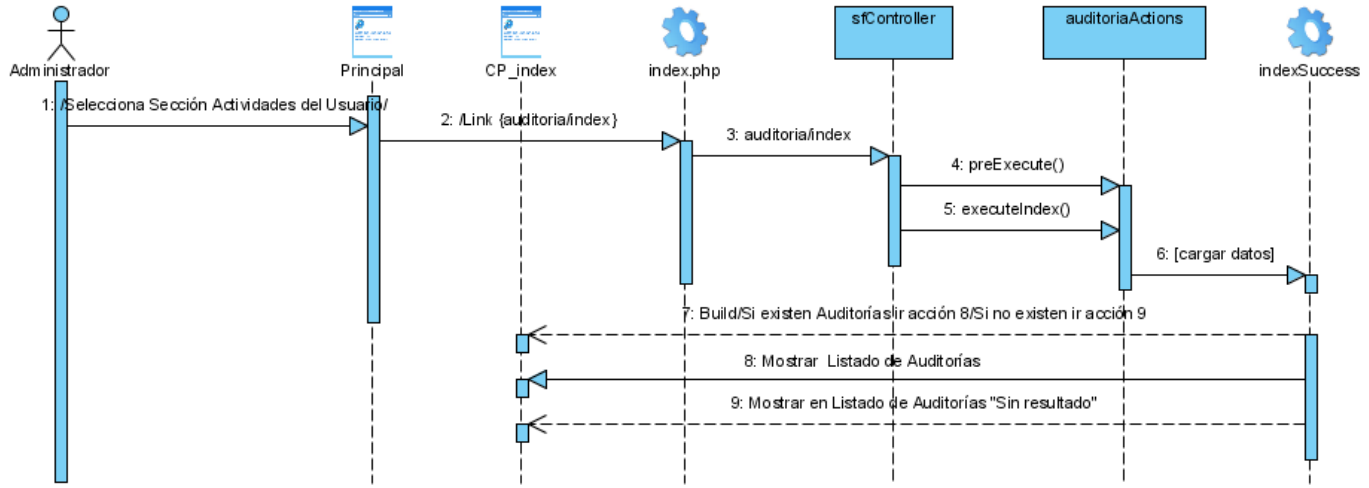


Figura 39.CUS Gestionar auditoría Escenario Listar auditoria

La figura 40 muestra cómo interactúan las clases que intervienen en el proceso de filtrar una auditoria. El administrador puede obtenerlos por el parámetro de id del usuario, usuario, módulo acción y fecha. Los datos que devuelve la búsqueda se muestran en el registro de actividades del usuario, en caso de que no lo encuentre muestra el mensaje “Sin resultado”.

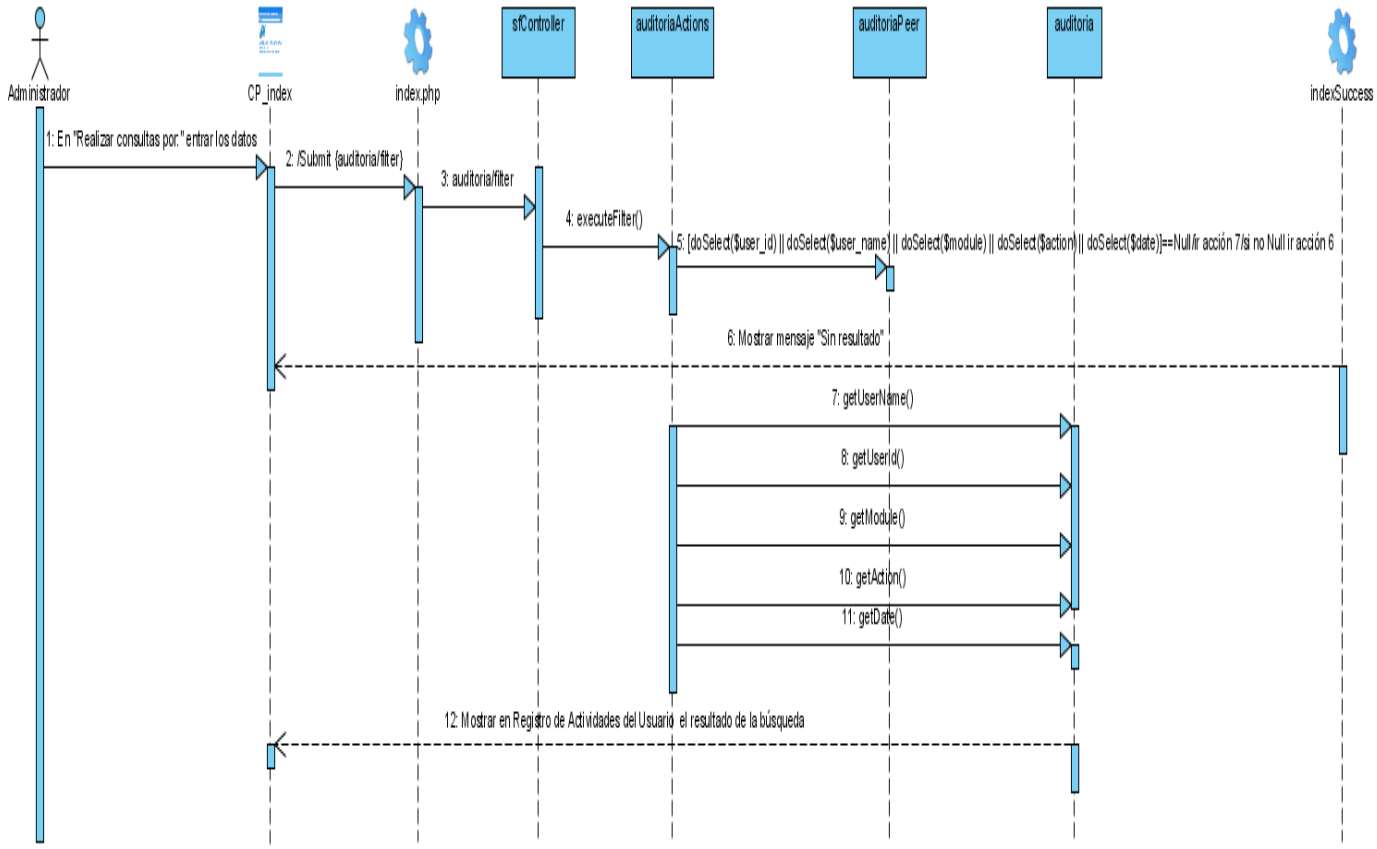


Figura 40.CUS Gestionar auditoría Escenario Filtrar

2.3.3 Mapa de Navegación

Un mapa de navegación es la representación gráfica de la organización de la información de una estructura web. Expresa todas las relaciones de jerarquía y secuencia y permite elaborar escenarios de comportamiento de los usuarios. [19]

En la figura 41 se muestra el mapa de navegación del Módulo Administración:

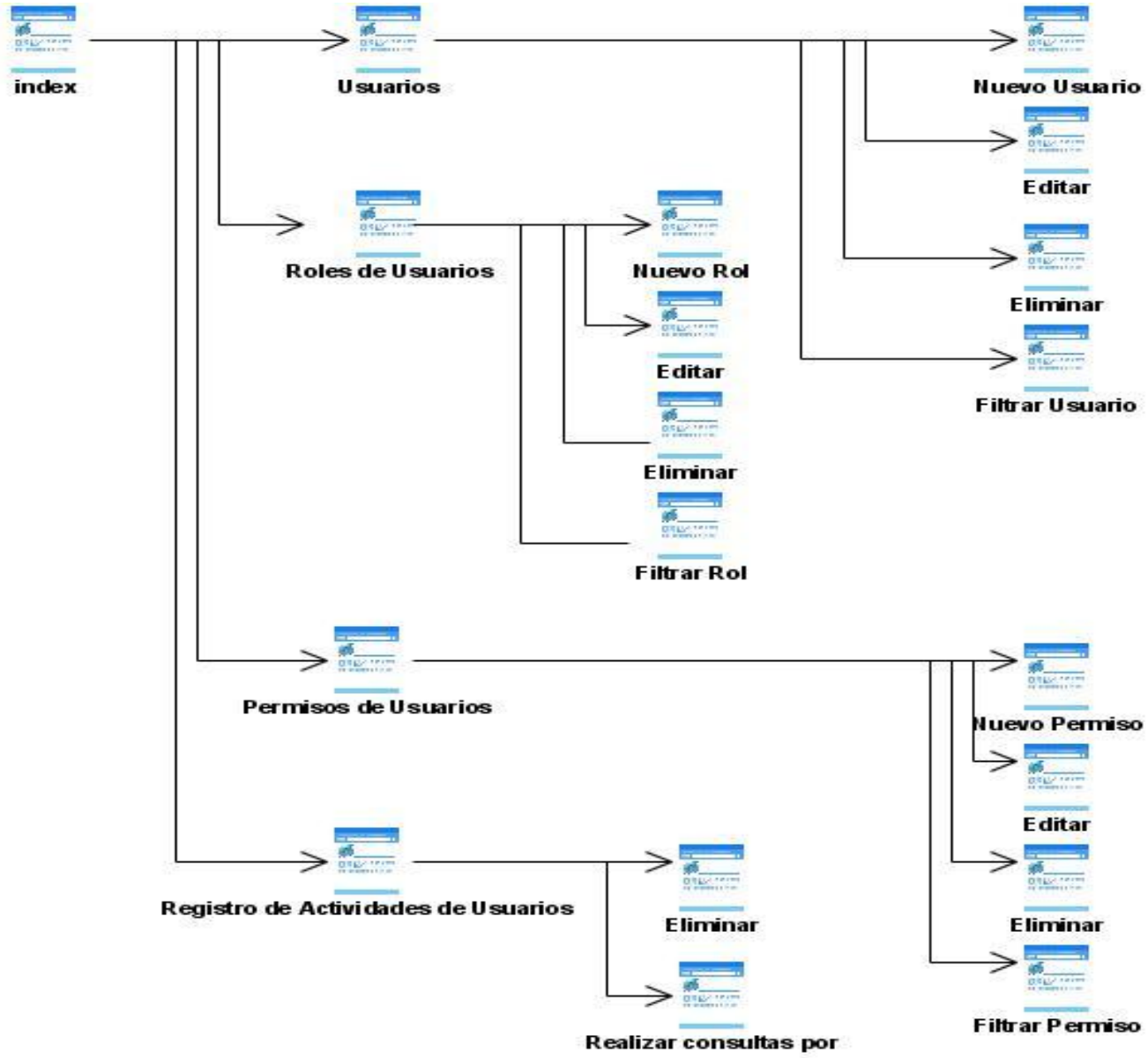


Figura 41. Mapa de Navegación

2.4 Vista Despliegue.

El modelo de despliegue, que se realiza en esta vista, muestra la configuración de los nodos de proceso en el tiempo de ejecución, los enlaces de comunicación entre ellos, y las instancias de componente y los objetos que residen en ellas.

En la figura 42 se muestra el modelo de despliegue.

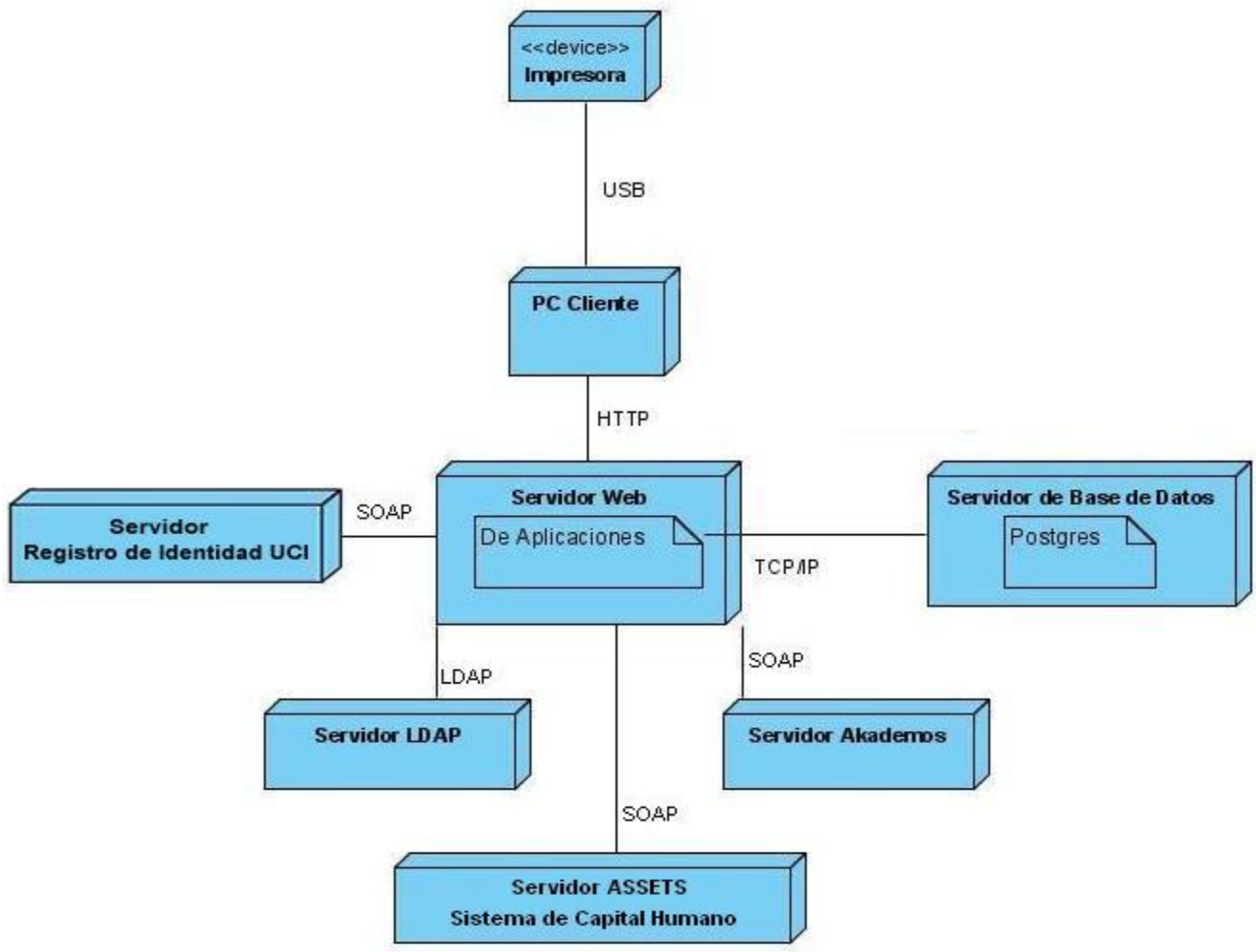


Figura 42.Modelo de Despliegue

Descripción de los nodos del modelo:

PC Cliente: Estación de trabajo desde donde interactúa el cliente con la aplicación.

Servidor Web Apache + PHP: Contiene la aplicación Web.

Servidor BD PostgreSQL: Servidor donde se almacenan los datos de la aplicación.

Servidor LDAP UCI: Servidor al que se le realizan peticiones para la validación de los usuarios.

Servidor Akademos: Servidor al que se le realizan peticiones para los datos de los estudiantes

Servidor Assets: Servidor de Capital Humano de la UCI.

Servidor Registro de Identidad UCI: Contiene la información de todas las personas registradas bajo el dominio UCI.

Impresora: Dispositivo que permite editar los reportes generados en la aplicación.

2.5 Vista Implementación.

La vista de implementación muestra el software como los elementos físicos que lo integran: componentes, ficheros, librerías. [20]

Un componente es una unidad física de implementación que encapsula una o más clases del diseño y se pueden agrupar en subsistemas de implementación para mayor organización del diagrama.

Symfony establece una estructura de carpetas para sus proyectos, dentro del mismo las operaciones se agrupan de forma lógica en aplicaciones. Cada aplicación está formada por uno o más módulos.

La vista de implementación del sistema queda conformada por un grupo de componentes y subsistemas de implementación que modelan el empaquetamiento físico real del sistema. Estos subsistemas de implementación representan a las carpetas del proyecto. A continuación se expone la descripción de los componentes y subsistemas de implementación que conforman la vista de implementación.

Componente index.php: Es el controlador frontal del sistema, este componente implementa la clase *index.php* del diagrama de clases del diseño.

Componente layout.php: Contiene los elementos que se muestran de forma idéntica a lo largo de toda la aplicación, este componente implementa la clase *layout.php* del diseño.

Componente login.class.php: Contiene los métodos que permiten interactuar con el Servidor Web LDAP de la UCI.

Componente integracionWS.class.php: Contiene los métodos que permiten interactuar con el Servidor Web Identidad de la UCI.

Componente assetsWSIntegracion.class.php: Contiene los métodos que permiten interactuar con el Servidor Web de Capital Humano de la UCI.

Componente akademos.ws.integration.class.php: Contiene los métodos que permiten interactuar con el Servidor Web de Gestión Académica de la UCI.

Componente BD: Encapsula todos los datos del sistema.

Subsistema de implementación model: Cada componente del model se corresponde con un archivo php que implementa una clase de diseño del Paquete Modelo del diseño. Agrupa los componentes que definen el modelo.

Subsistema de implementación modules: Encapsula los diferentes *Subsistemas de implementación módulo* correspondientes a cada módulo del sistema.

Subsistemas de implementación módulo: Encapsula los componentes de un módulo. Administración cuenta con los siguientes módulos: auditoría, sfGuardAuth, sf_guard_user, sf_guard_group y sf_guard_permission.

Componente actions.class.php define las acciones que incluyen el código específico del controlador para cada página del módulo, implementa la clase *actions.php* del diseño.

Subsistema de implementación templates: Agrupa los componentes que implementan el código correspondiente a cada plantilla que utiliza el módulo. Se corresponde con las plantillas de la Vista del diseño de cada módulo.

Subsistema de implementación lib: Contiene los componentes que implementan cada módulo:

Componente *BasesfGuardAuthActions.class.php*: Contiene la implementación de todos los métodos del módulo sfGuardAuth.

Componente *basesfGuardUserActions.class.php*: Contiene la implementación de todos los métodos del módulo sf_guard_user.

Componente B*asesfGuardGroupActions.class.php*: Contiene la implementación de todos los métodos del módulo sf_guard_group.

Componente *BasesfGuardPermissionActions.class.php*: Contiene la implementación de todos los métodos del módulo sf_guard_permission.

Subsistema de implementación config: Encapsula los componentes de tipo archivos de texto en formato YML que definen la configuración específica de cada módulo.

Componente *security.yml*: Archivo de configuración que permite restringir el acceso a determinadas acciones del módulo.

Componente *generator.yml*: Permite configurar las funcionalidades más utilizadas del módulo.

Subsistemas de implementación Symfony: Encapsula todos los componentes propios del framework. Se especifica el **componente Propel**, por ser ORM que utiliza Symfony, Propel utiliza el **componente Creole** como capa de abstracción a la base de datos.

En la Figura 43 se muestra la vista de los principales elementos del diagrama de componentes del módulo sfGuardAuth.

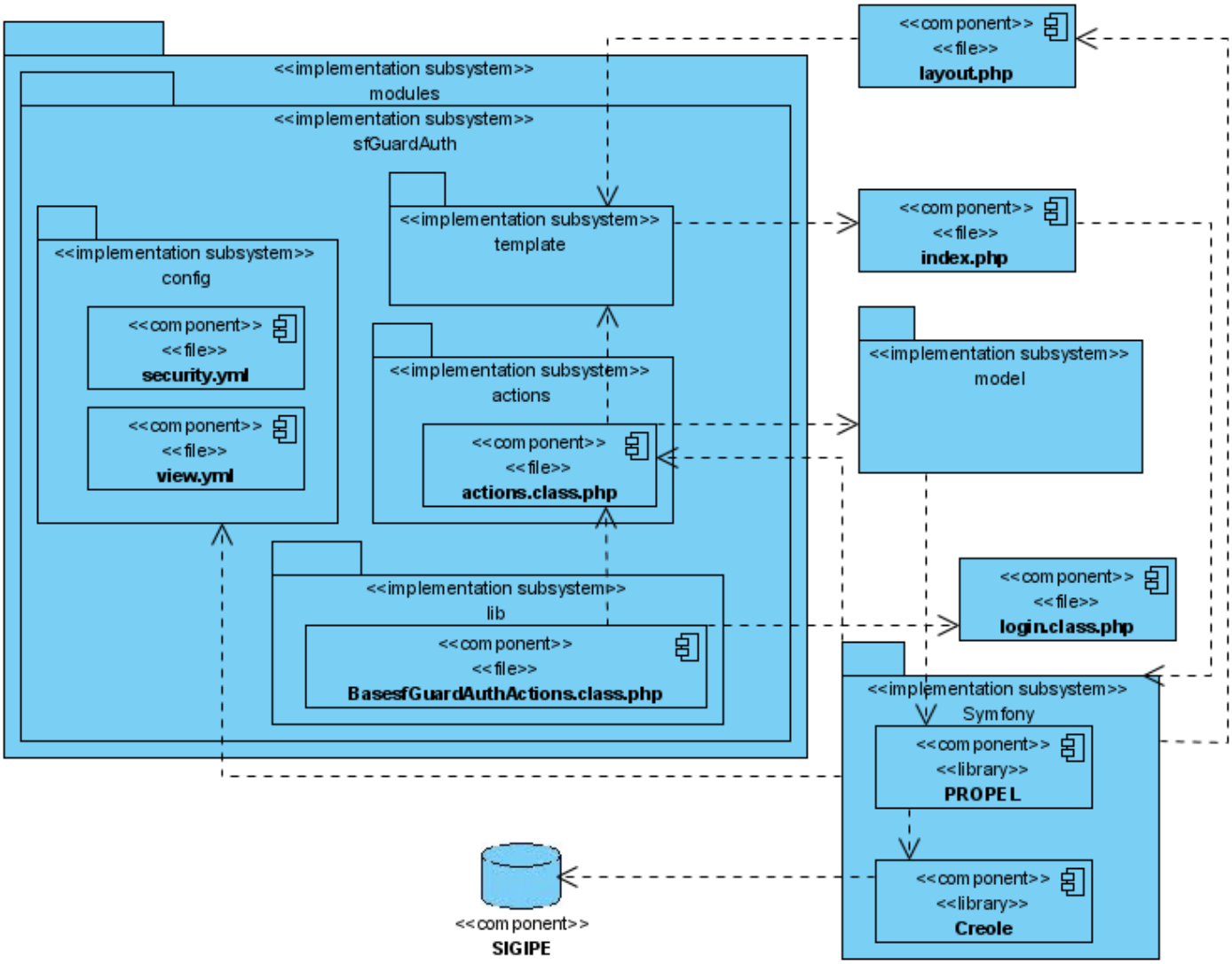


Figura 43.Diagrama de Componentes Módulo sfGuardAuth

En la Figura 44 se muestran las clases que conforman el paquete model del Módulo sfGuardAuth.

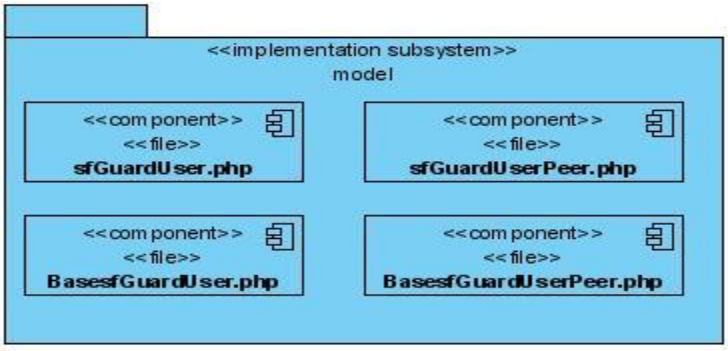


Figura 44.Paquete model del Módulo sfGuardAuth

Las templates de este módulo son:

- ✓ secureSuccess.php Se muestra cuando el usuario no tiene acceso a la página solicitada.
- ✓ signinSuccess.php Se muestra con cuando el usuario no está autenticado. Contiene el formulario sfGuardSignin que contiene los campos de usuario y contraseña para la entrada al sistema.

En la Figura 45 se muestran las clases que conforman el paquete templates del Módulo sfGuardAuth.

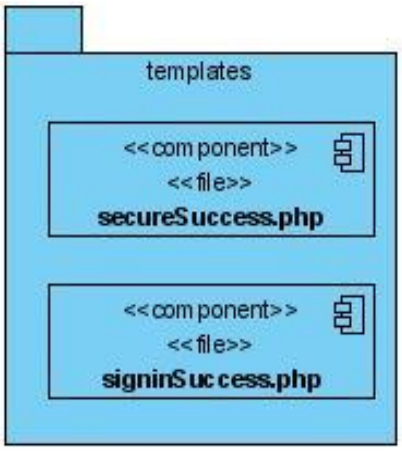


Figura 45.Paquete templates del Módulo sfGuardAuth

En la Figura 46 se muestra la vista de los principales elementos del diagrama de componentes del módulo sf_guard_user.

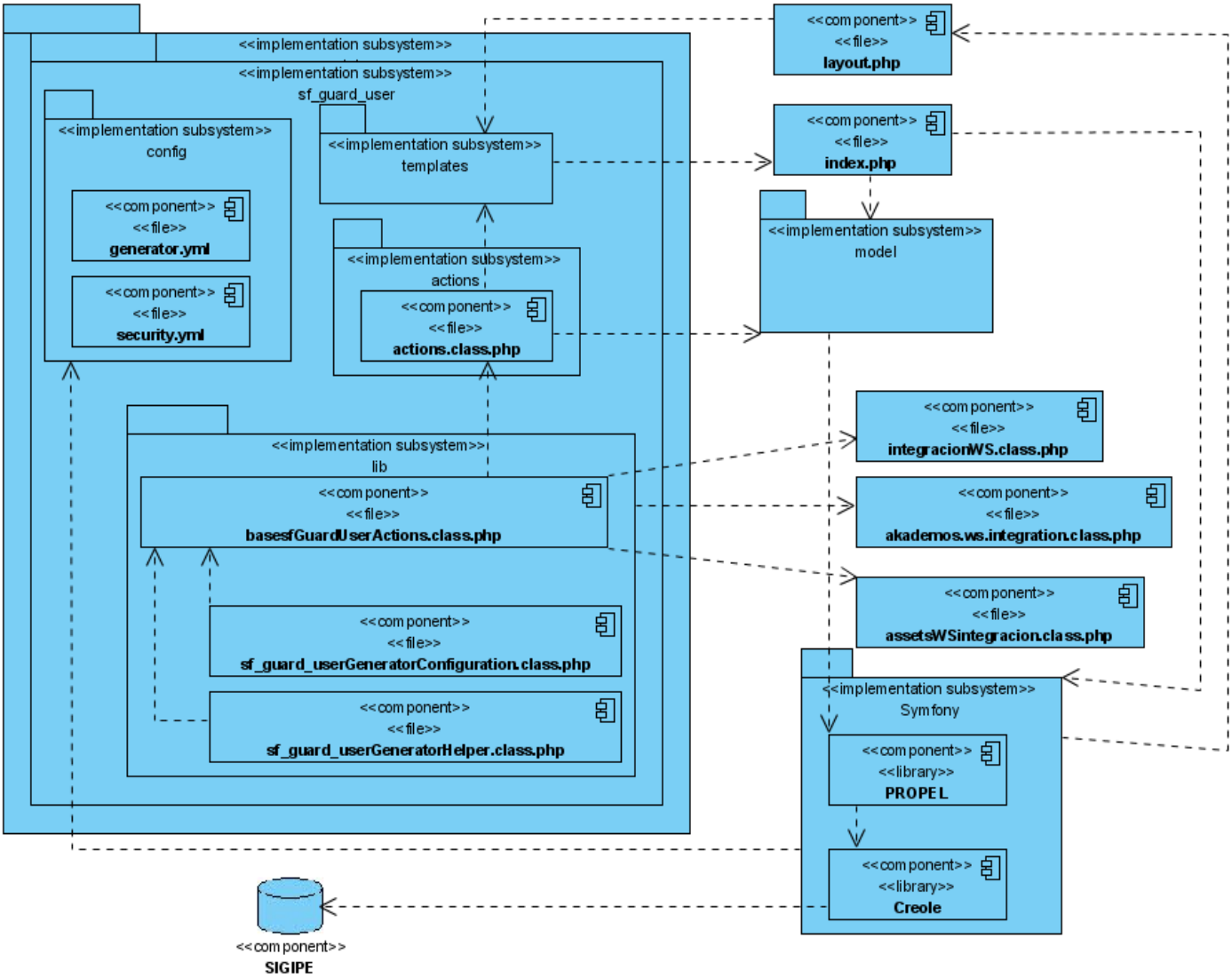


Figura 46.Diagrama de componentes Módulo sf_guard_user

En la Figura 47 se muestran las clases que conforman el paquete model del Módulo sf_guard_user.

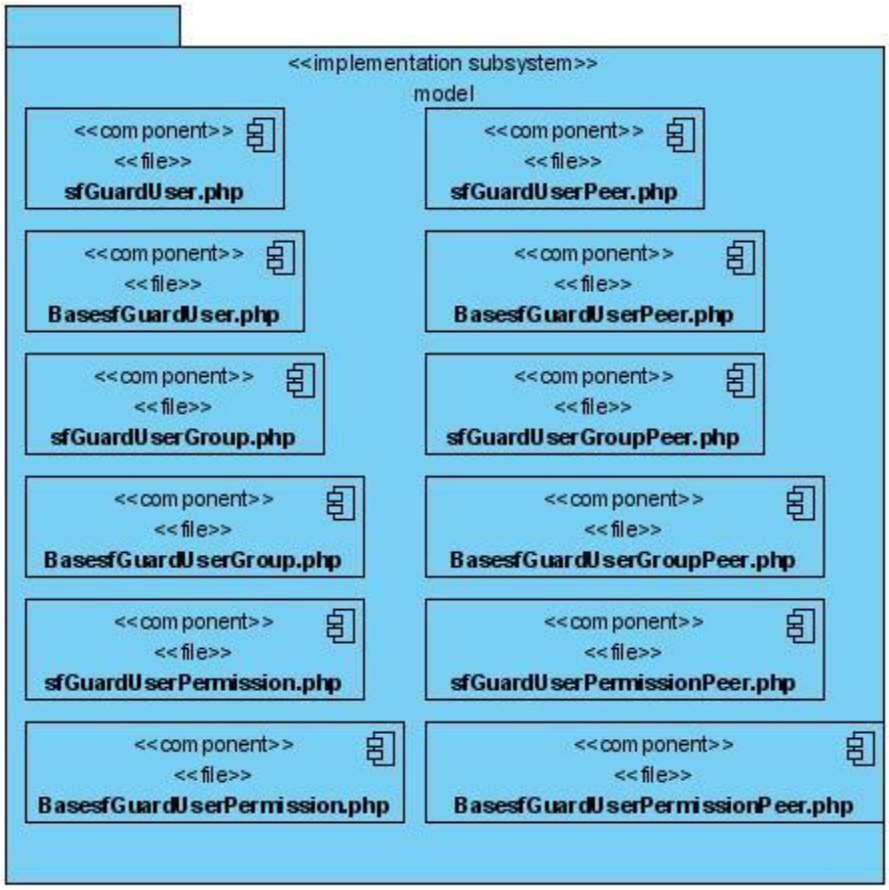


Figura 47.Paquete model del Módulo sf_guard_user

Las templates de este módulo son:

- ✓ _assets.php — Muestra CSS y JS a usar por las plantillas
- ✓ _filters.php — Muestra los filtros
- ✓ _filters_field.php — Muestra un único filtro de campo
- ✓ _flashes.php — Muestra los mensajes flash
- ✓ _form.php — Muestra el formulario
- ✓ _form_actions.php — Muestra las acciones del formulario
- ✓ _form_field.php — Muestra un único campo de formulario
- ✓ _form_fieldset.php — Muestra un fieldset de formulario
- ✓ _form_footer.php — Muestra el pie de página del formulario
- ✓ _form_header.php — Muestra cabecera del formulario
- ✓ _list.php — Muestra la lista
- ✓ _list_actions.php — Muestra las acciones de lista

- ✓ _list_batch_actions.php Muestra la lista de acciones por lotes
- ✓ _list_field_boolean.php Muestra un único campo booleano en la lista
- ✓ _list_footer.php Muestra el pie de página de lista
- ✓ _list_header.php Muestra la cabecera de lista
- ✓ _list_td_actions.php Muestra las acciones de objeto para una fila
- ✓ _list_td_batch_actions.php Muestra la casilla de verificación para una fila
- ✓ _list_td_stacked.php Muestra el stacked layout para una fila
- ✓ _list_td_tabular.php Muestra un único campo de lista
- ✓ _list_th_stacked.php Muestra un solo nombre de columna para la cabecera
- ✓ _list_th_tabular.php Muestra un solo nombre de columna para la cabecera
- ✓ _pagination.php Muestra la paginación de lista
- ✓ editSuccess.php Muestra la vista edit
- ✓ indexSuccess.php Muestra la vista index
- ✓ newSuccess.php Muestra la vista new

Estas templates también son válidas para los módulos sf_guard_group, sf_guard_permission y auditoria.

Para este módulo se añaden:

- ✓ importarSuccess.php Muestra la vista importar
- ✓ mensajeImportarSuccess.php Muestra la vista mensajeImportar

En la Figura 48 se muestra las clases que conforman el paquete templates del Módulo sf_guard_user.

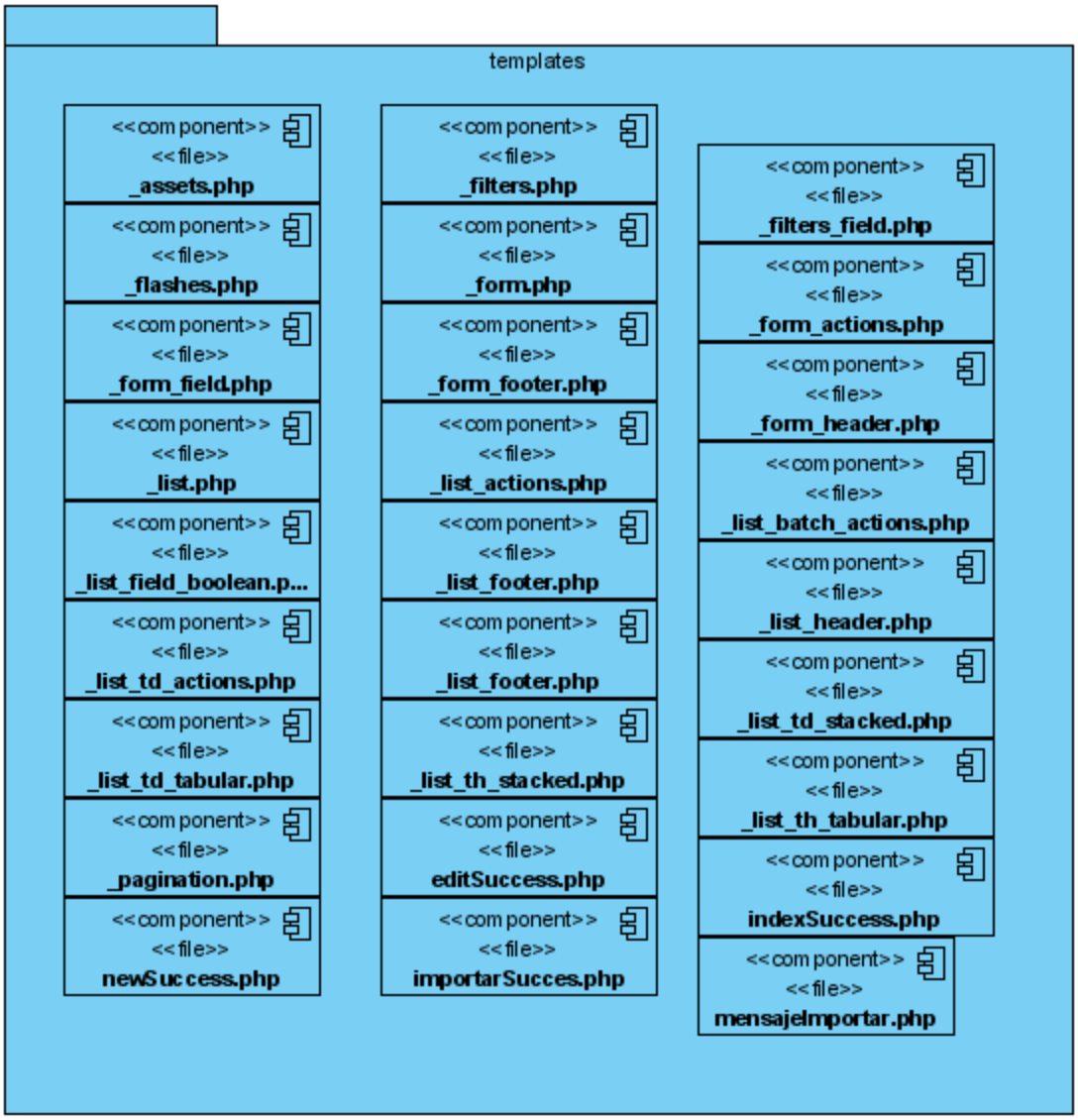


Figura 48. Paquete templates del Módulo sf_guard_user

En la Figura 49 se muestra la vista de los principales elementos del diagrama de componentes del módulo sfGuardGroup.

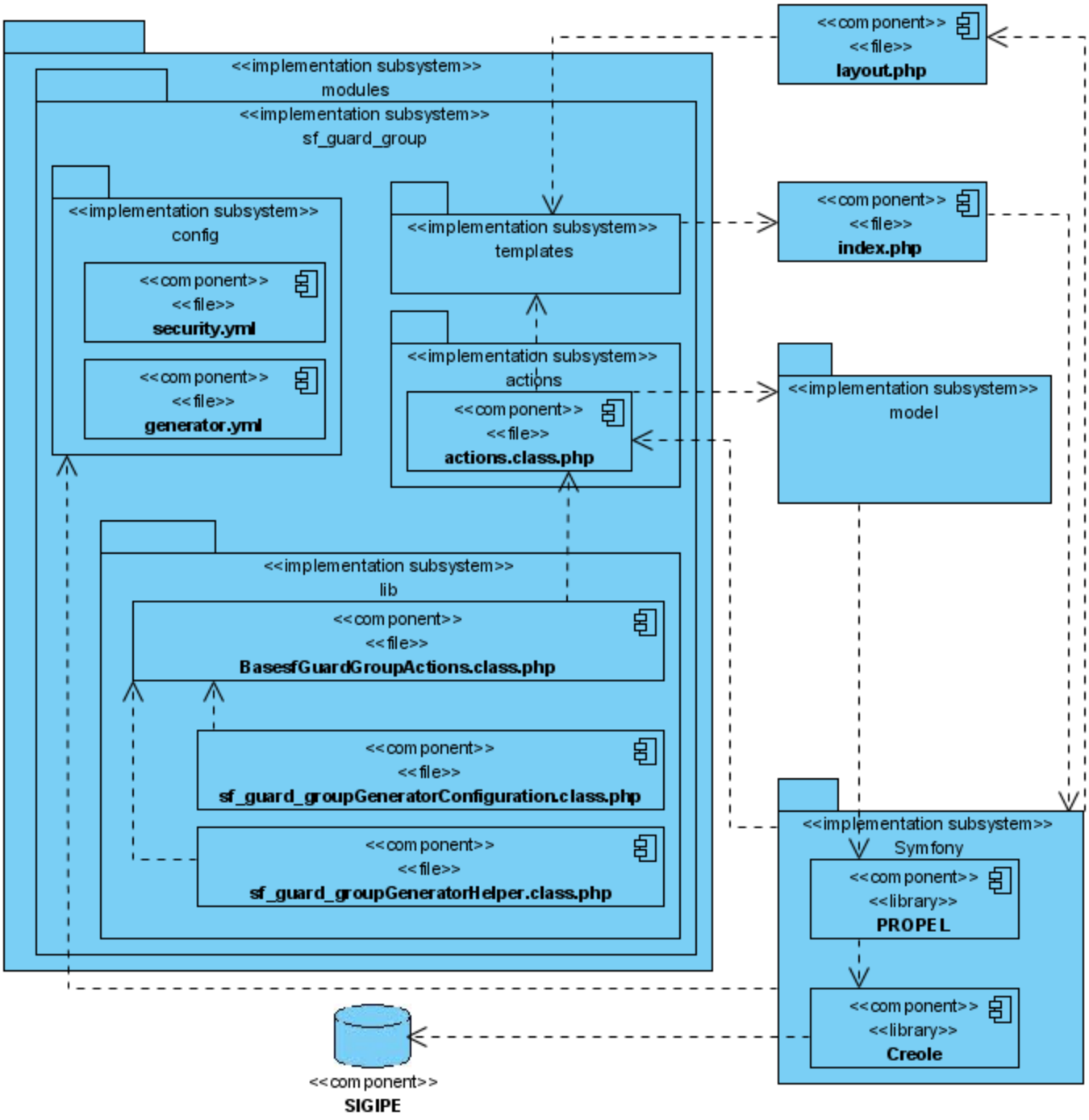


Figura 49.Diagrama de Componentes Módulo sf_guard_group

En la Figura 50 se muestran las clases que conforman el paquete model del Módulo sf_guard_group.

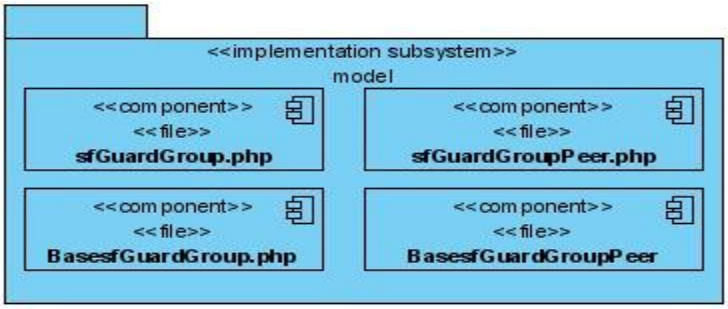


Figura 50.Paquete model del Módulo sf_guard_group

En la Figura 51 se muestran las clases que conforman el paquete templates del Módulo sf_guard_group.

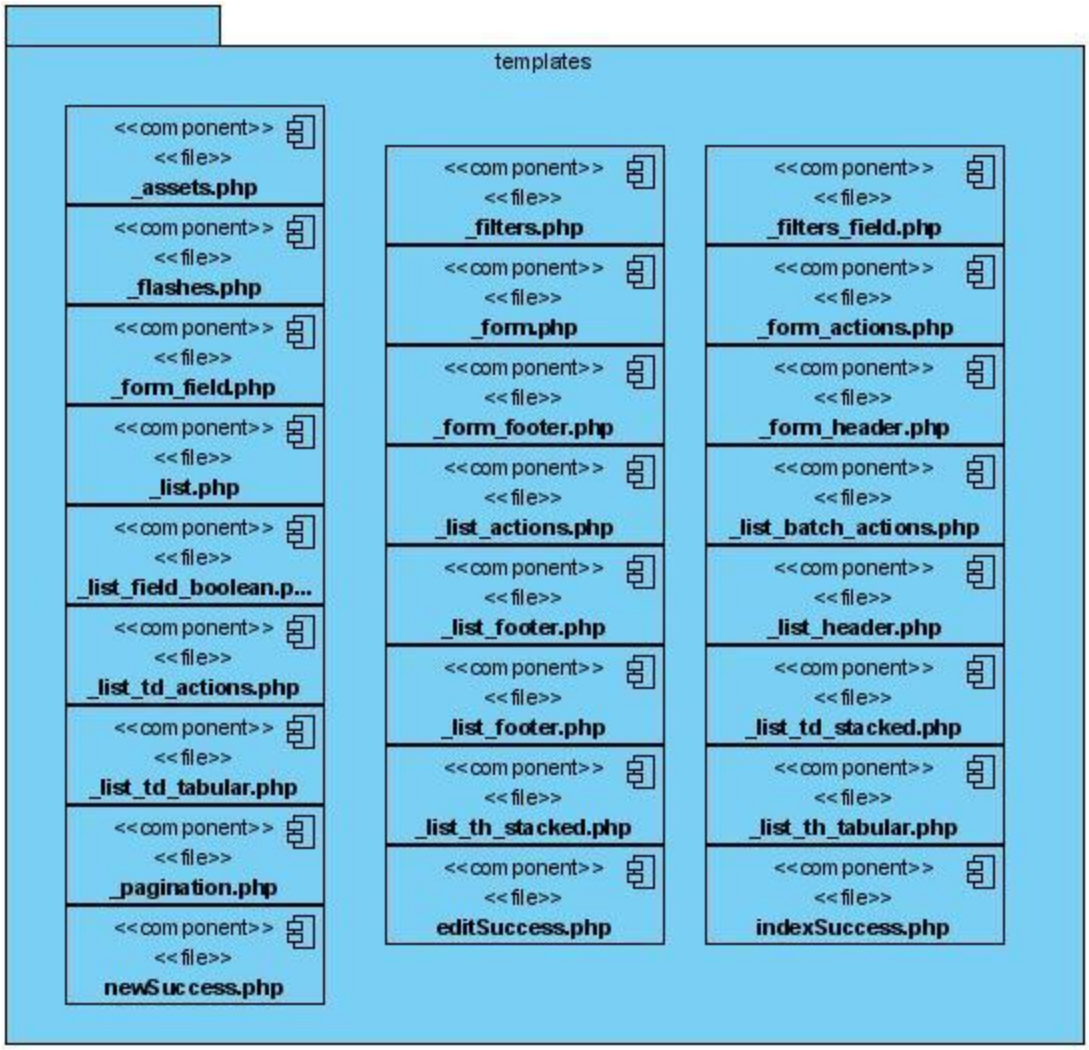


Figura 51.Paquete templates del Módulo sf_guard_group

En la Figura 52 se muestra la vista de los principales elementos del diagrama de componentes del módulo sfGuardPermission.

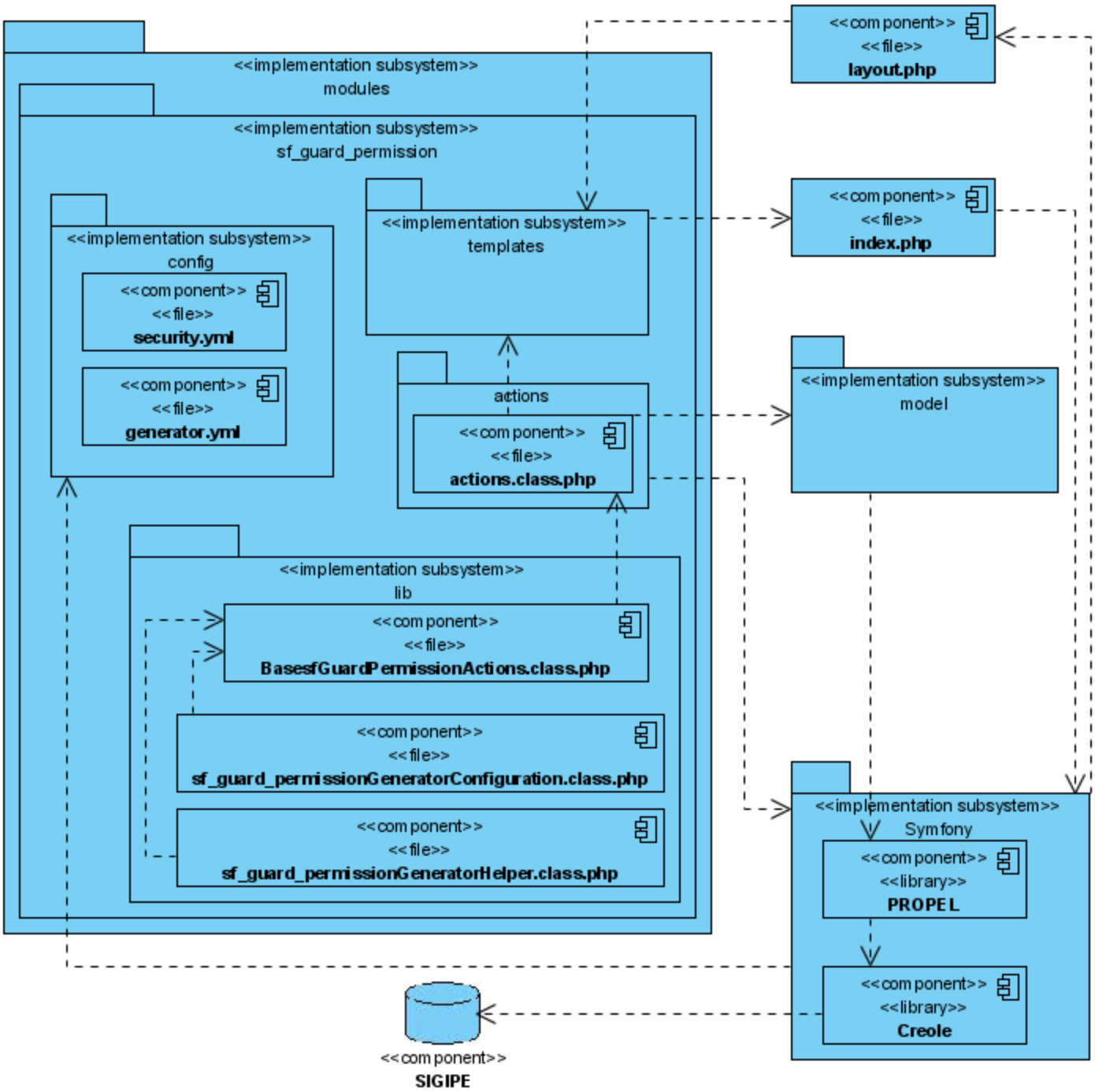


Figura 52.Diagrama de Componentes Módulo sf_guard_permission

En la Figura 53 se muestran las clases que conforman el paquete model del Módulo sf_guard_permission.

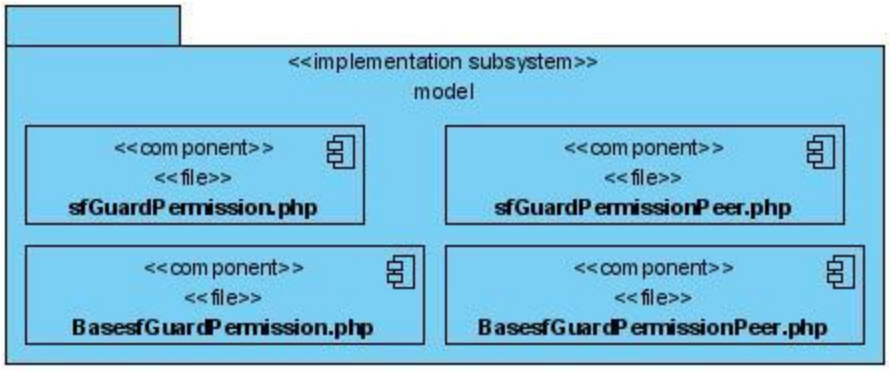


Figura 53.Paquete model del Módulo sf_guard_permission

En la Figura 54 se muestran las clases que conforman el paquete templates del Módulo sf_guard_permission.

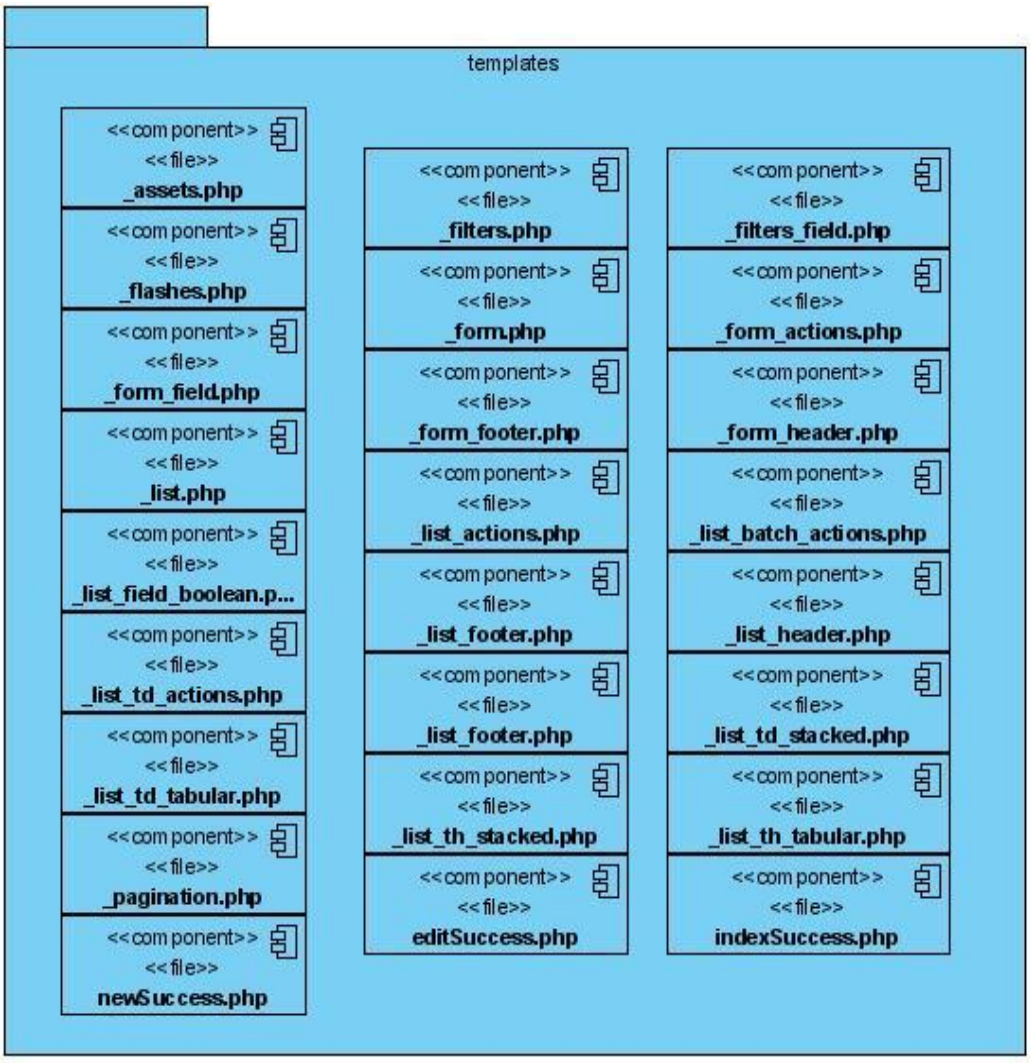


Figura 54.Paquete templates del Módulo sf_guard_permission

En la Figura 55 se muestra la vista de los principales elementos del diagrama de componentes del módulo auditoría.

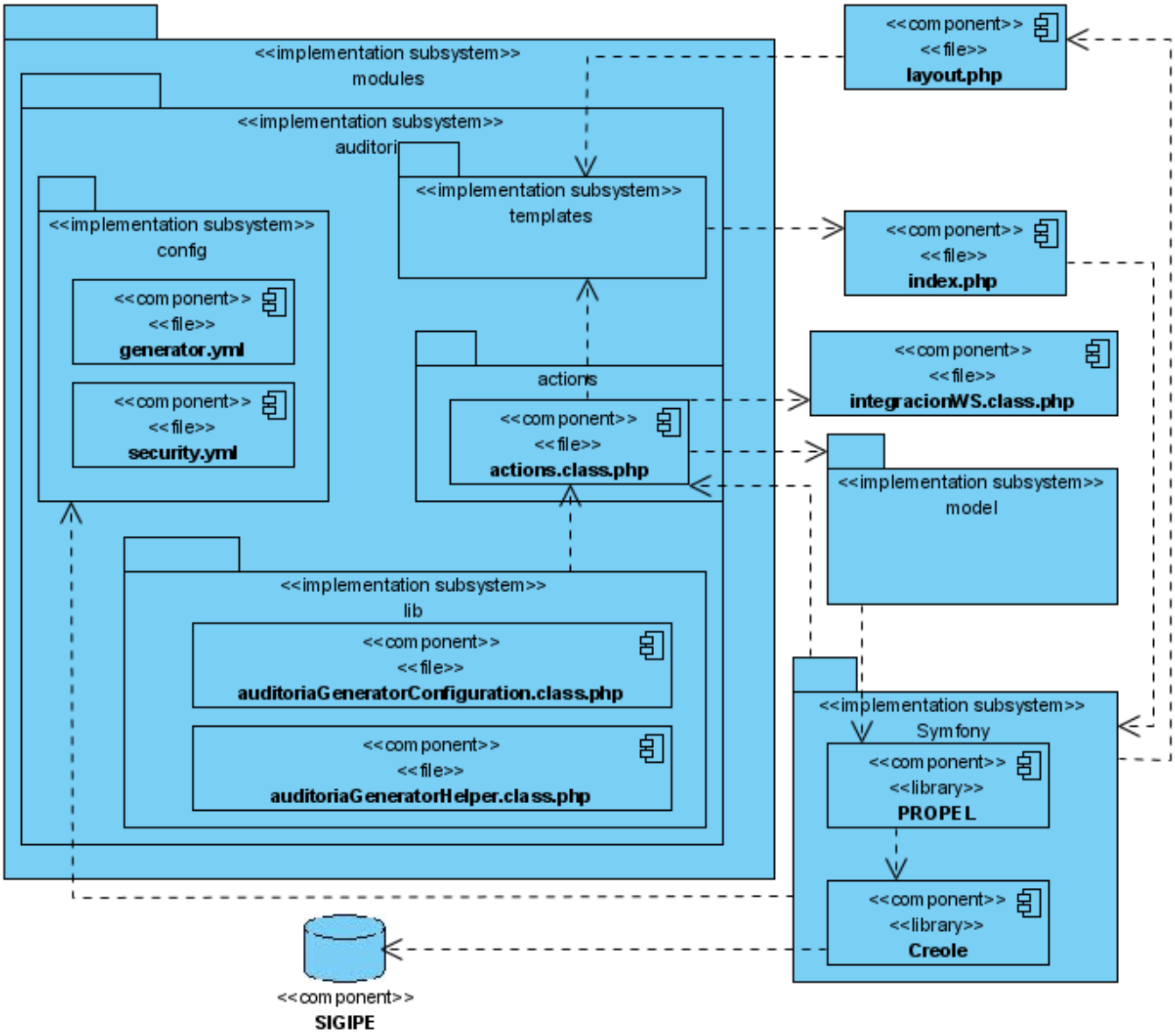


Figura 55.Diagrama de Componentes Módulo auditoria

En la Figura 56 se muestran las clases que conforman el paquete model del Módulo auditoria.

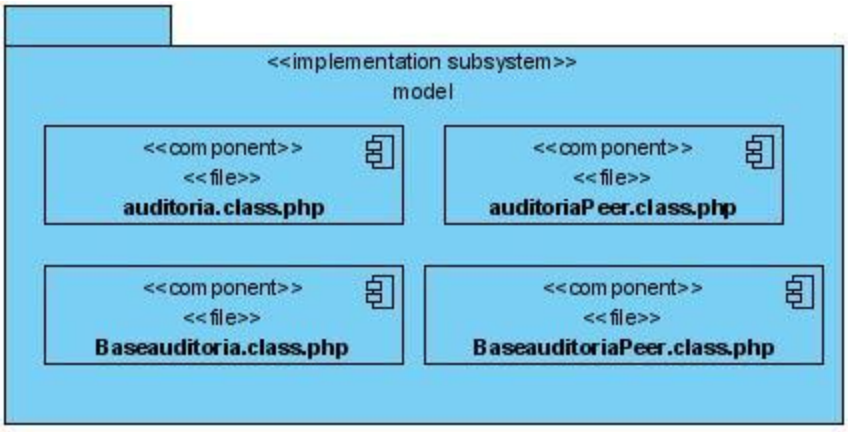


Figura 56. Paquete model del Módulo auditoria

En la Figura 57 se muestran las clases que conforman el paquete templates del Módulo auditoria.

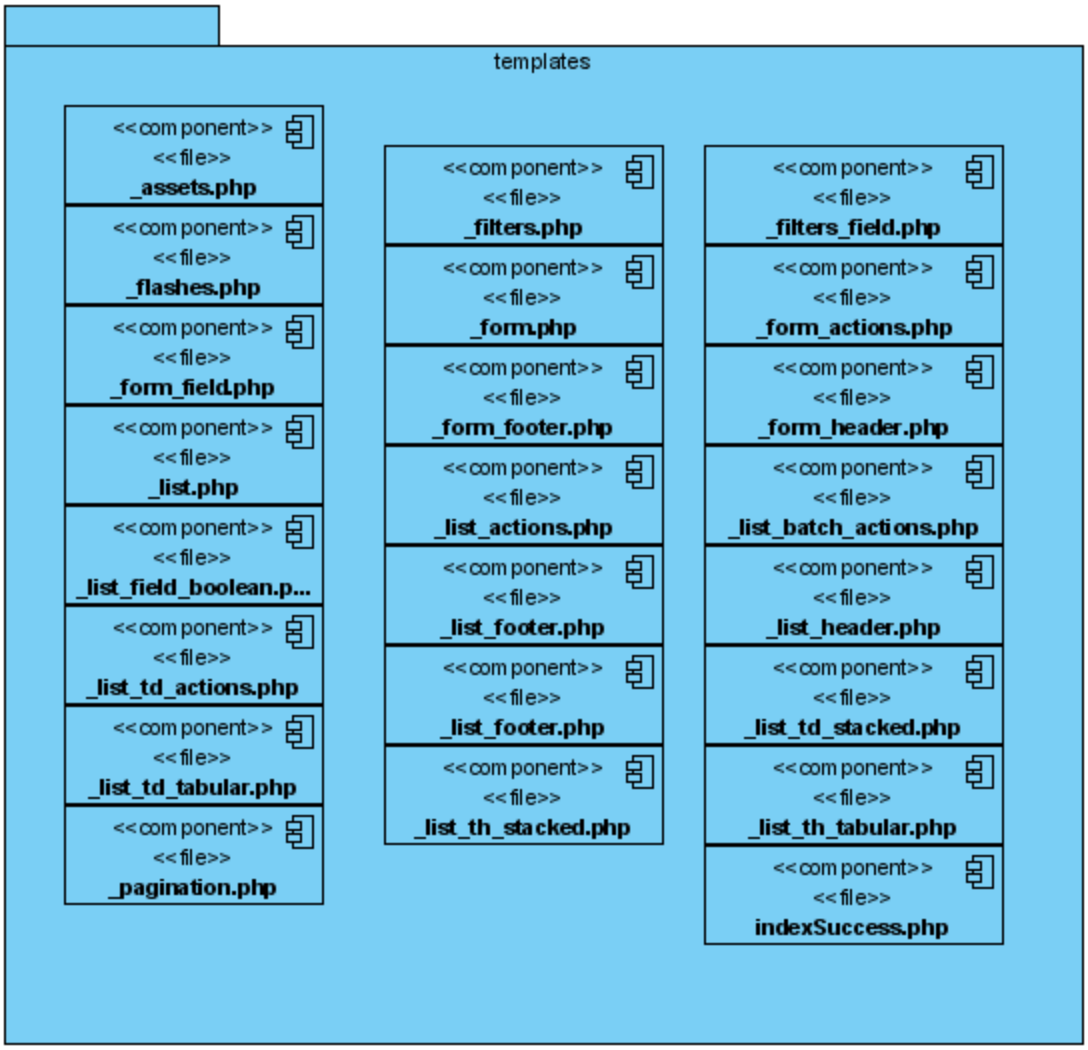


Figura 57. Paquete templates del Módulo auditoria

2.6 Principales funciones.

A continuación se muestran ejemplos de las principales clases utilizadas en el desarrollo del Módulo Administración y algunos de sus métodos.

La clase BasesfGuardAuthActions es la que contiene los métodos de comprobación para la autenticación de usuarios, la salida de los mismos del sistema y ejecuta la acción que se muestra cuando el usuario intenta acceder a una parte de la aplicación para la que no tiene permisos asignados. En esta clase se evidencia el patrón de Bajo Acoplamiento, debido a que esta clase sólo hereda de la clase sfAction, lo que permite un bajo acoplamiento entre estas clases. En el ejemplo se muestran las funciones que permiten la autenticación, el método executeSignin(), y la salida del usuario del sistema, el método executeSignout().

```
class BasesfGuardAuthActions extends sfActions
{
  public function executeSignin($request)
  {
          $user = $this->getUser();
          if ($user->isAuthenticated())
    {
      return $this->redirect('homepage');
    }

    $class = sfConfig::get('app_sf_guard_plugin_signin_form', 'sfGuardFormSignin');
    $this->form = new $class();

    if ($request->isMethod('post'))
    {
      $this->form->bind($request->getParameter('signin'));
      if ($this->form->isValid())
      {
        $values = $this->form->getValues();
        $this->getUser()->signin($values['user'], array_key_exists('remember', $values) ?
$values['remember'] : false);

        // always redirect to a URL set in app.yml
        // or to the referer
        // or to the homepage
        $signinUrl = sfConfig::get('app_sf_guard_plugin_success_signin_url', $user-
>getReferer('@homepage'));

        return $this->redirect('homepage');
      }
    }
    else
```

```
    {
      if ($request->isXmlHttpRequest())
      {
        $this->getResponse()->setHeaderOnly(true);
        $this->getResponse()->setStatusCode(401);

        return sfView::NONE;
      }

      // if we have been forwarded, then the referer is the current URL
      // if not, this is the referer of the current request
      $user->setReferer($this->getContext()->getActionStack()->getSize() > 1 ? $request->getUri() :
$request->getReferer());

      $module = sfConfig::get('sf_login_module');
      if ($this->getModuleName() != $module)
      {
        return $this->redirect($module.'/'.sfConfig::get('sf_login_action'));
      }

      $this->getResponse()->setStatusCode(401);
    }
  }

  public function executeSignout($request)
  {
    $this->getUser()->signOut();

$signoutUrl = sfConfig::get('app_sf_guard_plugin_success_signout_url', $request->getReferer());

    $this->redirect('homepage');
  }
}
```

La clase myUser es la que crea al usuario del sistema, esta clase puede ser utilizada a través de la variable $sf_user desde cualquiera de las templates del sistema o $this->getUser() desde las acciones de la aplicación. En esta clase se evidencia el uso del patrón Singleton (Instancia única), ya que con ella se garantiza la existencia de una única instancia para esta clase y la creación de un mecanismo de acceso global a dicha instancia. En el ejemplo se muestra el método Auditoria(), que garantiza el registro de la actividad que esté realizando el usuario en el sistema en un momento dado.

```
class myUser extends sfGuardSecurityUser
{
public function Auditoria()
  {
```

```
        $i=sfContext::getInstance();
        $m=$i->getModuleName();
        $a=$i->getActionName();
        $prueba=new auditoria();
        $username=$this->getUsername();
        $nombre=$this->getUsuario();
        $id=$this->getId();
        $fecha=$this->getLastRequestTime();
        $prueba->setUserId($id);
        $prueba->setUserName($nombre);
        $prueba->setModule($m);
        $prueba->setAction($a);
        $prueba->setDate($fecha);
        $prueba->save();
  }
}
```

La clase sfGuardSecurityUser es la que contiene los métodos que comprueban los parámetros de seguridad de los usuarios. En esta clase se evidencia el uso del patrón Alta Cohesión, debido a que tiene la responsabilidad de definir acciones que pueden ser utilizadas por las plantillas de la aplicación y colabora con otras clases para realizar diferentes operaciones, logrando una alta cohesión entre ella y las clases con las que colabora. A continuación se ejemplifican los métodos más utilizados en la creación de la aplicación: la función hasPermission() que comprueba la credencial del usuario, getUsername() que devuelve el nombre de usuario de quien está autenticado en el sistema.

```
class sfGuardSecurityUser extends sfBasicSecurityUser
{
  public function hasPermission($name)
  {
    return $this->getGuardUser() ? $this->getGuardUser()->hasPermission($name) :     false;
  }
public function getUsername()
  {
    return $this->getGuardUser()->getUsername();
  }
}
```

La clase basesf_guard_userActions es la que contiene los métodos específicos del módulo sf_guard_user. En esta clase se evidencia el uso del patrón Creador debido a que, en esta clase se encuentran las acciones definidas y se ejecutan cada una de ellas. En las acciones se crean los objetos de la clase que representa la entidad sf_guard_user, evidenciando de este modo que dicha

clase es "creador" de esta entidad. A continuación se muestra la función processForm(), que se ejecuta cuando se intenta crear un nuevo usuario en el sistema o modificar uno existente.

```
class basesf_guard_userActions extends sfActions
{
 protected function processForm(sfWebRequest $request, sfForm $form)
  {
        $form->bind($request->getParameter($form->getName()), $request->getFiles($form->getName()));
    if ($form->isValid())
    {

        $obj = integracionWS::getInstance();
                        $persona = $obj->getPersonaByUsuario($form['username']->getValue());
                        $ci= $persona->CI;

                        if($persona != null)
                        {
                        $id=$persona->CI;
                          $pertenece=false;
                        if ($persona->TipoPersona=='Profesor')
                    {
                    $nuevo=assetsWSintegracion::getInstance()->getPersonasDadoIdArea();
                    foreach ($nuevo as $prof)
                    {
                        if ($prof->CI==$ci)
                       {
                         $pertenece=true;
                        }
                    }
                    }
                    if ($persona->TipoPersona=='Estudiante')
                    {
                        $otro=AkademosWSIntegration::getInstance()->getEstudianteByCI($ci);
                                if ($otro->Area->IdArea =='Q0000')
                           {
                                        $pertenece=true;
                                }
                    }
                    if ($pertenece==true)
                    {
                        $username=$form['username']->getValue();
                       $this->getUser()->setFlash('notice', $form->getObject()->isNew() ? 'Usuario creado satisfactoriamente' : 'Usuario modificado satisfactoriamente.');
                            $sf_guard_user=$form->save();
                            if ($request->hasParameter('_save_and_add'))
            {
            $this->getUser()->setFlash('notice', $this->getUser()->getFlash('notice').' Puede adicionar otro usuario.');
```

```
            $this->redirect('@sf_guard_user_new');
            }
            else
            {
            $this->redirect('@sf_guard_user_edit?id='.$sf_guard_user->getId());
            }
                    }
                        else
                       {
                                $this->getUser()->setFlash('error', 'El usuario no pertenece a la Facultad
6.');;
                        }
                        }
                        else
                        {
                                //$this->getUser()->setFlash('notice', $this->getUser()->getFlash('notice').'
El usuario no fue creado debido a algunos errores.');
                                $this->getUser()->setFlash('error', 'El usuario no pertenece al dominio
UCI');
                        }
                        $this->dispatcher->notify(new sfEvent($this, 'admin.save_object', array('object'
=> $sf_guard_user)));
    }
    else
    {
      $this->getUser()->setFlash('error', 'El usuario no fue creado debido a algunos errores.');
    }
  }
```

2.7 Definición de Usuarios, Roles y Permisos.

Limitar el acceso a distintas áreas de la aplicación, constituye una tarea crítica a resolver mediante la implementación del Módulo Administración, garantizando que los usuarios puedan acceder solamente a los módulos del sistema para los que posean permisos asignados por el rol que desempeñen. Para su cumplimiento, se decidió utilizar el sfGuardPlugin de Symfony, que brinda un modelo de usuarios, permisos asignados a los mismos y los agrupa según los roles que desempeñen y que permite gestionar la seguridad de la aplicación de forma más eficiente que con el empleo de las funciones de seguridad estándar de dicho framework.

Para el desarrollo del Módulo Administración se utilizaron 6 tablas de las que define sfGuardPlugin (se identifican por el prefijo sf_guard) y una tabla auditoria por ser necesaria para el completamiento de la solución. A continuación se detallan los atributos de cada tabla y una breve descripción de cada uno de ellos:

Tabla sf_guard_user: Contiene toda la información referente al usuario.

Atributos	**Descripción**
id	Es la llave primaria de la tabla. Contiene el identificador del usuario.
username	Contiene el nombre de usuario. El valor del campo es único para cada usuario que se registre.
algorithm	Contiene el algoritmo de encriptación de la contraseña. Para este sistema se utilizó el algoritmo sha1.
salt	El valor de este campo se genera a partir del valor del campo password.
password	Contiene la contraseña del usuario de forma encriptada.
created_at	Contiene la fecha de creación del usuario.
last_login	Guarda la fecha de la última vez que el usuario entró al sistema.
is_active	Especifica si es un usuario activo o no.

Tabla 6.Tabla sf_guard_user

Tabla sf_guard_group: Contiene toda la información referente a los roles de los usuarios.

Atributos	**Descripción**
id	Es la llave primaria de la tabla. Contiene el identificador del rol.
name	Contiene el nombre de rol. El valor del campo es único para cada rol del sistema.

description	Contiene una descripción del rol. Puede o no llenarse este campo.

Tabla 7.Tabla sf_guard_group

Tabla sf_guard_permission: Contiene toda la información referente a los permisos de los usuarios.

Atributos	**Descripción**
id	Es la llave primaria de la tabla. Contiene el identificador del permiso.
name	Contiene el nombre de permiso. El valor del campo es único para cada permiso del sistema.
description	Contiene una descripción del permiso. Puede o no llenarse este campo.

Tabla 8.Tabla sf_guard_permission

Tabla sf_guard_user_permission: Almacena la relación entre usuarios y permisos asignados a este.

Atributos	**Descripción**
user_id	Contiene el identificador del usuario.
permission_id	Contiene el identificador del permiso asignado al usuario.

Tabla 9.Tabla sf_guard_user_permission

Tabla sf_guard_user_ group: Almacena la relación entre usuarios y roles asignados a este.

Atributos	**Descripción**
user_id	Contiene el identificador del usuario.
group_id	Contiene el identificador del grupo asignado al usuario.

Tabla 10.Tabla sf_guard_user_ group

Tabla sf_guard_group_permission: Almacena la relación entre roles y permisos.

Atributos	**Descripción**
group_id	Contiene el identificador del rol.
permission_id	Contiene el identificador del permiso.

Tabla 11.Tabla sf_guard_group_permission

Tabla auditoria: Almacena los registros de actividad del usuario en el sistema

Atributos	**Descripción**
id	Es la llave primaria de la tabla. Contiene el

	identificador de la auditoría.
user_id	Contiene el identificador del usuario que realiza la acción.
user_name	Contiene el nombre del usuario que realiza la acción.
module	Contiene el nombre del módulo en que está el usuario.
action	Contiene la acción que está ejecutando el usuario.
date	Contiene la fecha y la hora en que se ejecuta la acción.

Tabla 12.Tabla auditoria

En la figura 58 se presenta el Diagrama de clases persistentes que muestra las relaciones que se establecen entre las tablas.

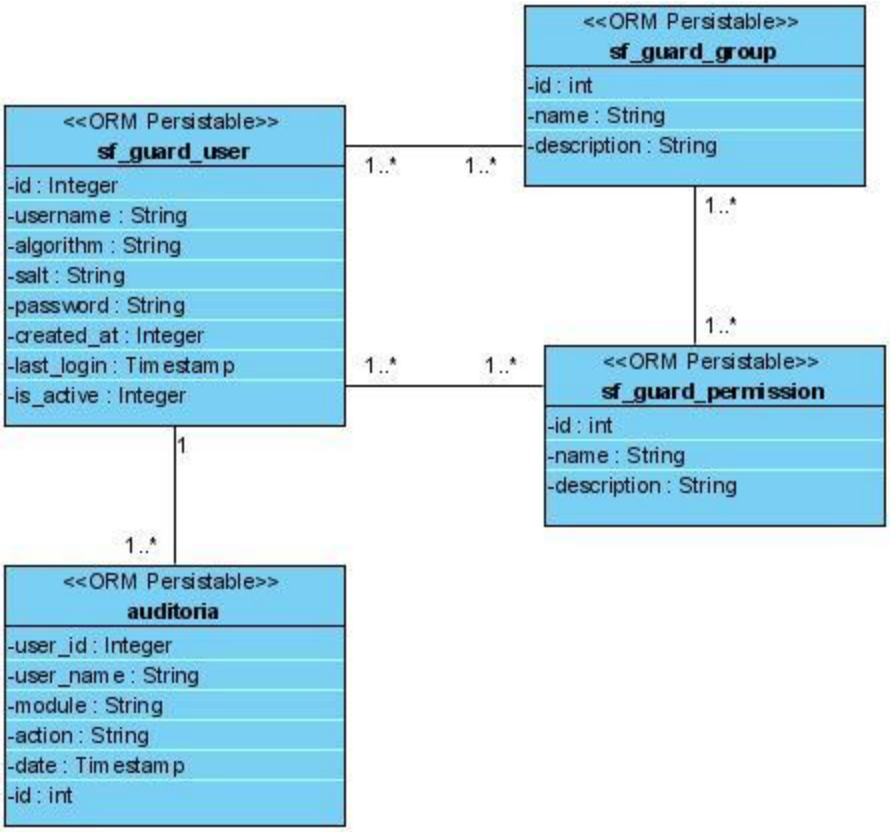


Figura 58.Diagrama de clases persistentes

La figura 59 presenta el modelo de datos el cual muestra las tablas y las relaciones que se generan a partir de la relación de las clases persistentes.

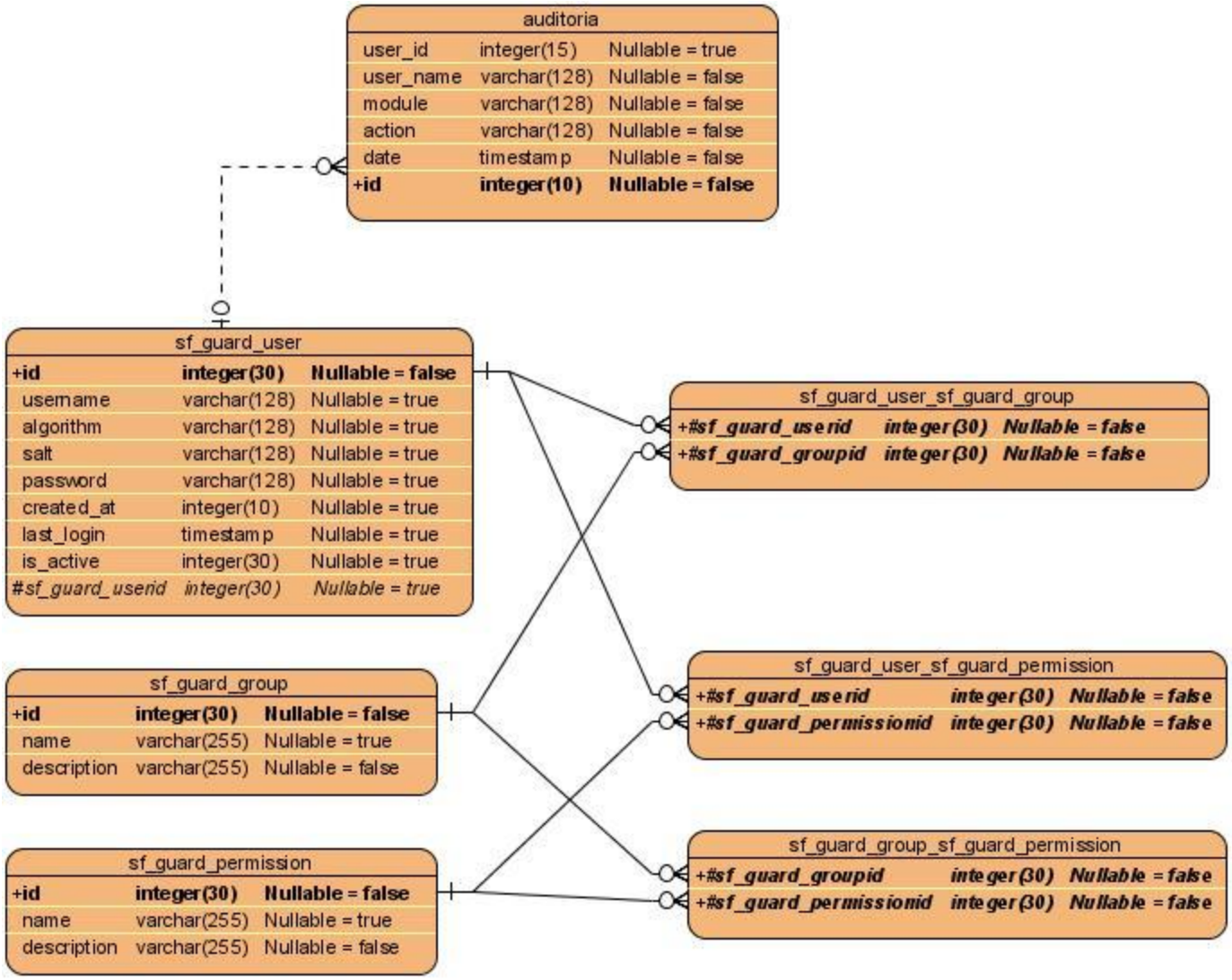


Figura 59. Modelo de datos

Para la entrada a la aplicación se utilizarán los usuarios del sistema de la UCI. Para la verificación de la contraseña será a través de consultas al servidor LDAP de la UCI.

Los roles de usuario se definen por el cargo en el cual se desempeñe el usuario dentro de la facultad 6 de la UCI.

Los permisos se definen de la siguiente forma:

- ✓ Un Administrador del Sistema que se encarga del mantenimiento de la aplicación y la asignación de roles y permisos a los usuarios. Se identifica por el nombre *Admin_Sistema*.
- ✓ Un Administrador por cada módulo de la aplicación. Se identifica como Administrador_NombreMódulo, ejemplo: *Administrador_Tesis*.

- ✓ Un Editor y un Consultor por cada funcionalidad del módulo correspondiente. Se identifica como Editor_NombreMódulo_Funcionalidad y Consultor__NombreMódulo_Funcionalidad, ejemplo: *Editor_Estudiante_AlumnoAyudante, Consultor_Estudiantes_Reportes.*

Para el acceso a las páginas de la aplicación se necesita tener los permisos correctos, previamente definidos en el archivo *security.yml* que se encuentra en cada módulo del sistema. Lo que sucede cuando un usuario trata de acceder una acción restringida depende de sus credenciales:

- ✓ Si el usuario está autenticado y tiene las credenciales apropiadas, entonces la acción se ejecuta.
- ✓ Si el usuario no está autenticado, es redireccionado a la acción de entrada al sistema (sfGuardAuth/signin).
- ✓ Si el usuario está autenticado, pero no posee las credenciales apropiadas, será redirigido a la acción segura por defecto (sfGuardAuth/secure) y se le informa que no tiene acceso a la página solicitada.

2.8 Pruebas

En Symfony se pueden crear dos tipos diferentes de pruebas automáticas: las pruebas unitarias y las pruebas funcionales. Estas pruebas no garantizan la ausencia de errores en la aplicación, sólo sirven como guía al programador para comprobar si los resultados obtenidos son los esperados.

Las pruebas unitarias comprueban que todas las funciones y todos los métodos funcionan correctamente. Cada una de las pruebas unitarias debe ser completamente independiente de las demás.

Por otra parte, las pruebas funcionales verifican que la aplicación funciona correctamente en su conjunto. Las pruebas en Symfony se guardan en el directorio test/ del proyecto. El directorio contiene a su vez dos subdirectorios, uno para las pruebas unitarias (test/unit/) y otro para las pruebas funcionales (test/functional/) como se muestra en la figura 60.

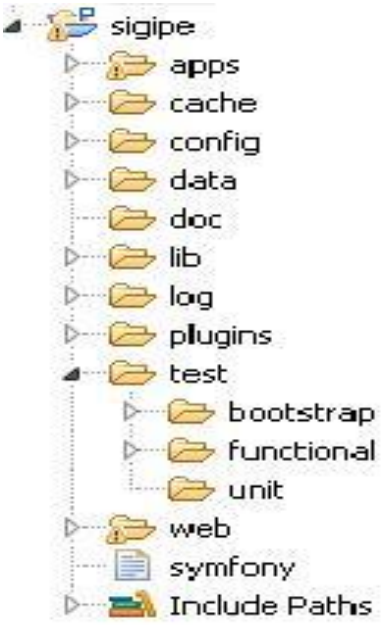


Figura 60. Localización de las pruebas en la aplicación

Las pruebas funcionales son la mejor forma de probar la aplicación de extremo a extremo: desde la petición realizada por un navegador hasta la respuesta enviada por el servidor. Las pruebas funcionales prueban todas las capas de la aplicación: el sistema de enrutamiento, el modelo, las acciones y las plantillas. Son similares a las pruebas que realiza el desarrollador manualmente al probar un escenario correspondiente al caso de uso que acaba de implementar en la aplicación y fueron las desarrolladas para el Módulo Administración. Estas pruebas se almacenan en el subdirectorio test/functional/frontend.

A continuación se muestra el contenido de la prueba funcional *auditoriaActionsTest,* en la misma se comprueba la página principal del módulo *auditoria.*

```
<?php

include(dirname(__FILE__).'/../../bootstrap/functional.php');

$browser = new sfTestFunctional(new sfBrowser());

$browser->
get('/auditoria/index')->
```

```
  with('request')->begin()->
    isParameter('module', 'auditoria')->
    isParameter('action', 'index')->
  end()->

  with('response')->begin()->
    checkElement('body', '!/This is a temporary page/')->
  end()
;
```

Para este caso, la prueba realizada arrojó el resultado mostrado en la figura 61, lo que comprueba que la página principal del módulo no redirecciona a ninguna otra página.

```
D:\Users\Baby\Tesis\Sigipe>symfony test:functional frontend auditoriaActions
# get /auditoria/index
ok 1 - request parameter module is auditoria
ok 2 - request parameter action is index
ok 3 - response selector body does not match regex /Esta es una pagina temporal/

1..3
 Looks like everything went fine.
```

Figura 61. Prueba funcional a auditoriaActionsTest

Además se realizó la prueba funcional sf_guard_userActionsTest, que comprueba que para la acción de editar un usuario registrado en la tabla sf_guard_user, esta función obtiene correctamente el identificador del usuario, el código de estado de la respuesta y la cabecera de la página.

```
<?php

include(dirname(__FILE__).'/../../bootstrap/functional.php');

$b = new sfTestBrowser();
$b->get('sf_guard_user/2/edit');
$request  = $b->getRequest();
$context  = $b->getContext();
$response = $b->getResponse();
```

```
// Acceder a los métodos de lime_test mediante el método test()
$b->test()->is($request->getParameter('id'), 2);
$b->test()->is($response->getStatuscode(), 401);
$b->test()->is($response->getHttpHeader('content-type'), 'text/html; charset=utf-8');
;
```

El resultado de esta prueba se muestra en la figura 62.

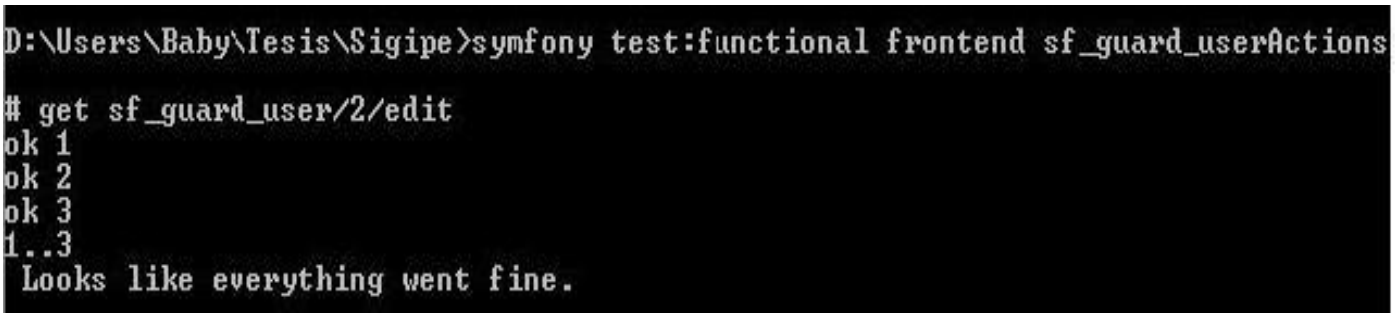


Figura 62. Prueba funcional a sf_guard_userActionsTest

2.9 Conclusiones

En el capítulo se realizó el diseño del sistema, como resultado, se obtienen los diagramas de clases del diseño y sus correspondientes diagramas de secuencia por cada escenario de los Casos de Uso identificados para el Módulo Administración. Además, se modeló el diagrama de despliegue, a través del cual, se especificaron los nodos en los que se distribuye la aplicación, las conexiones de red y protocolos que los unen. También se desarrolló el modelo de implementación, que describe como se organizan los componentes en el sistema. Por último, se explicó en qué forma se definieron los usuarios que tendrán interacción con la aplicación y los roles y permisos de los mism

Conclusiones Generales

Con la realización del Módulo Administración del Sistema de Gestión de Información de Estudiantes y Profesores (SIGIPE):

- ✓ Se fundamentaron herramientas y tecnologías a utilizar.
- ✓ Se diseñó el Módulo.
- ✓ Se implementaron los componentes a utilizar.
- ✓ Se identificaron los componentes del sistema que requieren ser asegurados.
- ✓ Se comprobó la solución propuesta con la realización de pruebas propuestas por Symfony para desarrolladores.

Recomendaciones

Se recomienda:

Continuar en la profundización del estudio de sfGuardPlugin del framework Symfony como herramienta de seguridad para el desarrollo de aplicaciones web.

Referencias Bibliográficas

1. **Ramió Aguirre, Jorge.** *Libro Electrónico de Seguridad Informática y Criptografía Versión 4.1*. [en línea]. 1 de marzo 2006. [Consultado: 15 de abril de 2009]. Disponible en: http://www.criptored.upm.es/guiateoria/gt_m001a.htm
2. *Herramienta Administrativa Configurable*. [en línea]. [Consultado: 27 de febrero de 2009]. Disponible en*:* http://www.rssoftsistemas.com/info_a2_herramientaadmin.htm
3. *Herramienta Administración de sitios Web, Seguridad (Ficha)*. [en línea]. [Consultado: 15 de abril de 2009]. Disponible en: http://msdn.microsoft.com/es-es/library/ssa0wsyf.aspx
4. **Aguirre, Sebastián.** *Metodologías de desarrollo de software 1*. [en línea]. 18 enero 2009. [Consultado en: 23 de febrero 2009]. Disponible en: http://www.autorneto.com/Tecnolog%C3%ADa/Software/Metodolog%C3%ADas-de-desarrollo-de-software-1.464053
5. *Procesos de Desarrollo: RUP, XP y FDD.* [en línea] 2002. [Consultado: 27 de febrero de 2009] http://www.javahispano.org/contenidos/es/procesos_de_desarrollo.
6. RUP_Help_ES. [Consultado: 28 de febrero de 2009].
7. **Ferré Grau, Xavier & Sánchez Segura, María Isabel.** *Desarrollo Orientado a Objetos con UML*. [en línea]. 2004. [Consultado: 23 de febrero 2009]. Disponible en: http://www.clikear.com/manuales/uml/index.aspx.
8. **Perissé, Marcelo Claudio.** *Una Metodología Simplificada.* [en línea]. febrero del 2001. [Consultado: 27 de febrero 2009]. Disponible en: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.
9. *Visual Paradigm for UML (SE) description.* [en línea].1 de agosto 2007. [Consultado: 16 de marzo 2009]. Disponible en: http://fivesign.com/downloads/program/Visual-Paradigm-for-UML-SE_23094_98.html%20-%2032k%20-
10. **Angel Álvarez, Miguel.** *PDT: Eclipse + PHP.* [en línea]. 29 de marzo 2008. [Consultado: 27 de febrero 2009]. Disponible en: http://www.desarrolloweb.com/articulos/pdt-eclipse-php.html.
11. **Rguez. Antonio.** *Hablamos con Luis M. Cabezas, autor del "Manual Imprescindible de PHP 5".* [en línea]. 13 de enero 2005. [Consultado en: 27 de febrero 2009]. Disponible en: http://www.sinuh.org/content/view/210/26/.

12. **Potencier,Fabien & Zaninotto,François.** *Symfony 1.2, la guía definitiva*. [en línea].30 de diciembre 2008. [Consultado en: 27 de febrero 2009].Disponible en: http://www.librosweb.es/symfony_1_2/capitulo1/symfony_en_pocas_palabras.html.
13. *Introducción a PostgreSQL - Instalación e inicialización*. [en línea].8 de junio 2007. [Consultado en: 28 de febrero 2009].Disponible en: http://www.linux-es.org/node/536.
14. *Descripción de Apache HTTP Server 2.2.8.* [en línea]. 4 de febrero 2008. [Consultado en: 1 de abril 2009]. Disponible en: http://www.boxsoftware.net/programas/apache-http-server-2-2-8.asp
15. **Potencier,Fabien & Zaninotto,François.** *Symfony 1.2, la guía definitiva*. [en línea].30 de diciembre 2008. [Consultado en: 27 de febrero 2009].Disponible en: http://www.librosweb.es/symfony_1_2/capitulo2/el_patron_mvc.html
16. [en línea]. 18 de mayo 2006. [Consultado en: 27 de febrero 2009].Disponible en: http://groups.google.com/group/symfony-users/msg/cd94d2ddb2057355?pli=1
17. **León Valdés, Esley & Sotolongo Vázquez, Duanis.** *Sistema para la gestión de la información de profesores y estudiantes: Arquitectura e Integración.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba, junio 2007. [Consultado en: 13 de abril 2009].
18. *ADS - Sesion2 - Presentation Transcript.* [en línea]. [Consultado en: 16 de abril 2009].Disponible en: http://www.slideshare.net/willy0303/ads-sesion
19. **Sacristán, Luis.** *Propel: ORM para PHP.* [en línea]. [Consultado en: 23 de mayo 2009]. Disponible en: http://sentidoweb.com/2008/10/06/propel-orm-para-php.php
20. **Colectivo de Autores.** *Propuesta del diseño arquitectónico de una plataforma para la predicción de actividad biológica de compuestos orgánicos.* [en línea]. [Consultado en: 23 de mayo 2009]. Disponible en: http://informatica2009.sld.cu/Members/ymlopez/propuesta-del-diseno-arquitectonico-de-una-plataforma-para-la-prediccion-de-actividad-biologica-de-compuestos-organicos/at_download/trabajo
21. *Mapa de navegación.* [en línea]. [Consultado en: 12 de abril 2009].Disponible en: http://arquitecturadeinformacion.cl/como/mapa.html

Bibliografía

1. **Ramió Aguirre, Jorge.** *Libro Electrónico de Seguridad Informática y Criptografía Versión 4.1*. [en línea]. 1 de marzo 2006. [Consultado: 15 de abril de 2009]. Disponible en: http://www.criptored.upm.es/guiateoria/gt_m001a.htm
2. *Herramienta Administrativa Configurable*. [en línea]. [Consultado: 27 de febrero de 2009]. Disponible en*:* http://www.rssoftsistemas.com/info_a2_herramientaadmin.htm
3. *Herramienta Administración de sitios Web, Seguridad (Ficha)*. [en línea]. [Consultado: 15 de abril de 2009]. Disponible en: http://msdn.microsoft.com/es-es/library/ssa0wsyf.aspx
4. **Aguirre, Sebastián.** *Metodologías de desarrollo de software 1*. [en línea]. 18 enero 2009. [Consultado en: 23 de febrero 2009]. Disponible en: http://www.autorneto.com/Tecnolog%C3%ADa/Software/Metodolog%C3%ADas-de-desarrollo-de-software-1.464053
5. *Procesos de Desarrollo: RUP, XP y FDD.* [en línea] 2002. [Consultado: 27 de febrero de 2009] http://www.javahispano.org/contenidos/es/procesos_de_desarrollo.
6. RUP_Help_ES. [Consultado: 28 de febrero de 2009].
7. **Ferré Grau, Xavier & Sánchez Segura, María Isabel.** *Desarrollo Orientado a Objetos con UML*. [en línea]. 2004. [Consultado: 23 de febrero 2009]. Disponible en: http://www.clikear.com/manuales/uml/index.aspx.
8. **Perissé, Marcelo Claudio.** *Una Metodología Simplificada.* [en línea]. febrero del 2001. [Consultado: 27 de febrero 2009]. Disponible en: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.
9. *Visual Paradigm for UML (SE) description.* [en línea].1 de agosto 2007. [Consultado: 16 de marzo 2009]. Disponible en: http://fivesign.com/downloads/program/Visual-Paradigm-for-UML-SE_23094_98.html%20-%2032k%20-
10. **Angel Álvarez, Miguel.** *PDT: Eclipse + PHP.* [en línea]. 29 de marzo 2008. [Consultado: 27 de febrero 2009]. Disponible en: http://www.desarrolloweb.com/articulos/pdt-eclipse-php.html.
11. **Rguez. Antonio.** *Hablamos con Luis M. Cabezas, autor del "Manual Imprescindible de PHP 5".* [en línea]. 13 de enero 2005. [Consultado en: 27 de febrero 2009]. Disponible en: http://www.sinuh.org/content/view/210/26/.

12. **Potencier,Fabien & Zaninotto,François.** *Symfony 1.2, la guía definitiva*. [en línea].30 de diciembre 2008. [Consultado en: 27 de febrero 2009].Disponible en: http://www.librosweb.es/symfony_1_2/capitulo1/symfony_en_pocas_palabras.html.
13. *Introducción a PostgreSQL - Instalación e inicialización*. [en línea].8 de junio 2007. [Consultado en: 28 de febrero 2009].Disponible en: http://www.linux-es.org/node/536.
14. *Descripción de Apache HTTP Server 2.2.8.* [en línea]. 4 de febrero 2008. [Consultado en: 1 de abril 2009]. Disponible en: http://www.boxsoftware.net/programas/apache-http-server-2-2-8.asp
15. **Potencier,Fabien & Zaninotto,François.** *Symfony 1.2, la guía definitiva*. [en línea].30 de diciembre 2008. [Consultado en: 27 de febrero 2009].Disponible en: http://www.librosweb.es/symfony_1_2/capitulo2/el_patron_mvc.html
16. [en línea]. 18 de mayo 2006. [Consultado en: 27 de febrero 2009].Disponible en: http://groups.google.com/group/symfony-users/msg/cd94d2ddb2057355?pli=1
17. **León Valdés, Esley & Sotolongo Vázquez, Duanis.** *Sistema para la gestión de la información de profesores y estudiantes: Arquitectura e Integración.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba, junio 2007. [Consultado en: 13 de abril 2009].
18. *ADS - Sesion2 - Presentation Transcript.* [en línea]. [Consultado en: 16 de abril 2009].Disponible en: http://www.slideshare.net/willy0303/ads-sesion
19. **Sacristán, Luis.** *Propel: ORM para PHP.* [en línea]. [Consultado en: 23 de mayo 2009]. Disponible en: http://sentidoweb.com/2008/10/06/propel-orm-para-php.php
20. **Colectivo de Autores.** *Propuesta del diseño arquitectónico de una plataforma para la predicción de actividad biológica de compuestos orgánicos.* [en línea]. [Consultado en: 23 de mayo 2009]. Disponible en: http://informatica2009.sld.cu/Members/ymlopez/propuesta-del-diseno-arquitectonico-de-una-plataforma-para-la-prediccion-de-actividad-biologica-de-compuestos-organicos/at_download/trabajo
21. *Mapa de navegación.* [en línea]. [Consultado en: 12 de abril 2009].Disponible en: http://arquitecturadeinformacion.cl/como/mapa.html
22. **Potencier,Fabien & Zaninotto,François.** *Symfony 1.2, la guía definitiva*. [en línea].30 de diciembre 2008. [Consultado en: 27 de febrero 2009].Disponible en: http://www.librosweb.es/symfony_1_2
23. http://groups.google.com/group/symfony-users

24. *sfGuard plugin - extra documentación.* [en línea]. [Consultado en: 25 de febrero 2009].Disponible en: http://trac.symfony-project.com/wiki/sfGuardPluginExtraDocumentation
25. **Molina de Armas, Elvismary & Orosco Fonseca, Eyleen.** *Sistema Informático para la Red Nacional de Genética Médica: Definición de la Arquitectura de Software.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba, junio 2008. [Consultado en: 2 de abril 2009].

Anexos

Anexo 1: Página de inicio del Sistema para la Gestión de la Información de Profesores y Estudiantes.

Anexo 2: Página de Usuarios del Sistema para la Gestión de la Información de Profesores y Estudiantes.

Anexo 3: Página de Permisos del Sistema para la Gestión de la Información de Profesores y Estudiantes.

Anexo 4: Página de Roles del Sistema para la Gestión de la Información de Profesores y Estudiantes.

Anexo 5: Página de Auditoría del Sistema para la Gestión de la Información de Profesores y Estudiantes.

Glosario de Términos

TICs: Tecnologías de la información y la comunicación, son un conjunto de servicios, redes, software y dispositivos.

Framework: Conjunto de conceptos, metodologías y herramientas de administración y diseño para el desarrollo de forma estandarizada de una aplicación.

Plugin: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

RUP: Metodología de desarrollo cuyas siglas vienen dadas por su nombre en inglés *Rational Unified Process.*

UML: Lenguaje visual para especificar, construir y documentar un software. Sus siglas vienen dadas por su nombre en inglés *Unified Modeling Language*.

Herramienta CASE: Las siglas CASE vienen dadas por su nombre en inglés *Computer Aided Software Engineering* que se conoce como Ingeniería de Software Asistida por Computadoras.

SVN: Sistema de control de versiones que sustituye al CVS.

CVS: Sistema de control de versiones, cuyas siglas vienen dadas por su nombre en inglés *Concurrent Version System.*

Ingeniería Inversa: Método que obtiene información de un producto ya liberado con el propósito de determinar qué elementos lo componen, que utiliza para su funcionamiento y cómo fue fabricado.

IDE: Entorno de desarrollo integrado cuyas siglas vienen dadas por su nombre en inglés *Integrated Development Environment.*

PHP: Lenguaje de programación interpretado diseñado para la creación de páginas web dinámicas cuyas siglas vienen dadas por su nombre en inglés *Hypertext Pre-processor*.

HTML: Lenguaje de marcas de hipertexto cuyas siglas vienen dadas por su nombre en inglés *Hyper Text Markup Language* y es empleado para la construcción de páginas web.

RDBMS: Sistema Administrador de Bases de Datos Relacionales cuyas siglas vienen dadas por su nombre en inglés *Relational Data Base Management System*.

XML: Estándar de información cuyas siglas vienen dadas por su nombre en inglés *eXtensible Markup Language.*

XML-RPC: Protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes cuyas siglas viene dadas por su nombre en inglés *eXtensible Markup Language- Remote Procedure Call*.

PDF: Formato de almacenamiento de documentos cuyas siglas viene dadas por su nombre en inglés *Portable Document Format.*

Aplicación Web: Sistema informático utilizado por los usuarios que acceden a un servidor web a través de Internet o de una Intranet.

MySQL: Sistema gestor de Base de datos.

PostgreSQL: Sistema Gestor de Base de Datos.

Oracle: Sistema de gestión de base de datos relacional.

SQL: Lenguaje de consulta estructurado cuyas siglas vienen dadas por su nombre en inglés *Structured Query Language.*

Unix: Sistema operativo.

Linux: Sistema operativo de código abierto.

Windows: Sistema operativo propietario.

BSD: Distribución de Software Berkeley cuyas siglas vienen dadas por su nombre en inglés *Berkeley Software Distribution* y se utiliza para identificar un sistema operativo

ACID: Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Atomicidad, Consistencia, Aislamiento y Durabilidad y sus siglas vienen dadas por su nombre en inglés *Atomicity, Consistency, Isolation and Durability*.

SSL: Protocolo de Capa de Conexión Segura cuyas siglas vienen dadas por su nombre en inglés *Secure Sockets Layer***.** Es un protocolo criptográfico que proporciona comunicaciones seguras por una red.

MVCC: Técnica avanzada para mejorar las prestaciones de una base de datos en un entorno multiusuario. Sus siglas vienen dadas por su nombre en inglés *Multiversion Concurrency Control*.

API: Interfaz de programación de aplicaciones o *Application Programming Interface*, conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

HTTP: Protocolo de transferencia de hipertexto cuyas siglas vienen dadas por su nombre en inglés *HyperText Transfer Protocol*, es el protocolo usado en cada transacción de la Web.

CGI: Interfaz de entrada común que permite a un cliente solicitar datos de un programa ejecutado en un servidor web. Sus siglas vienen dadas por su nombre en inglés *Common Gateway Interface*.

MVC: Patrón arquitectónico *Modelo-Vista-Controlador*. Identifica tres componentes fundamentales que se corresponden con su nombre, en los que se separan la interfaz de usuario, la lógica del negocio y los datos de la aplicación.

URL: Localizador uniforme de recurso que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización. Sus siglas vienen dadas por su nombre en inglés *Uniform Resource Locator.*

ADO: Mecanismos que usan los programas de computadoras para comunicarse con las bases de datos. Sus siglas vienen dadas por su nombre en inglés *ActiveX Data Objects*.

USB: *Universal Serial Bus* o Conductor Universal en Serie, es un puerto que sirve para conectar periféricos a una computadora.

***GRASP*:** Patrones generales de software para asignación de responsabilidades, sus siglas vienen dadas por su nombre en inglés *General Responsibility Assignment Software Patterns*.

***GOF*:** Expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software, sus siglas vienen dadas por su nombre en inglés *Gang of Four*.

LDAP: *Lightweight Directory Access Protocol*, implementa un servicio de directorio jerárquico y distribuido para accesar depósitos de información referente a usuarios, contraseñas y otras entidades en un entorno de red, ofreciendo una amplia capacidad de filtrado sobre la información que está siendo solicitada.