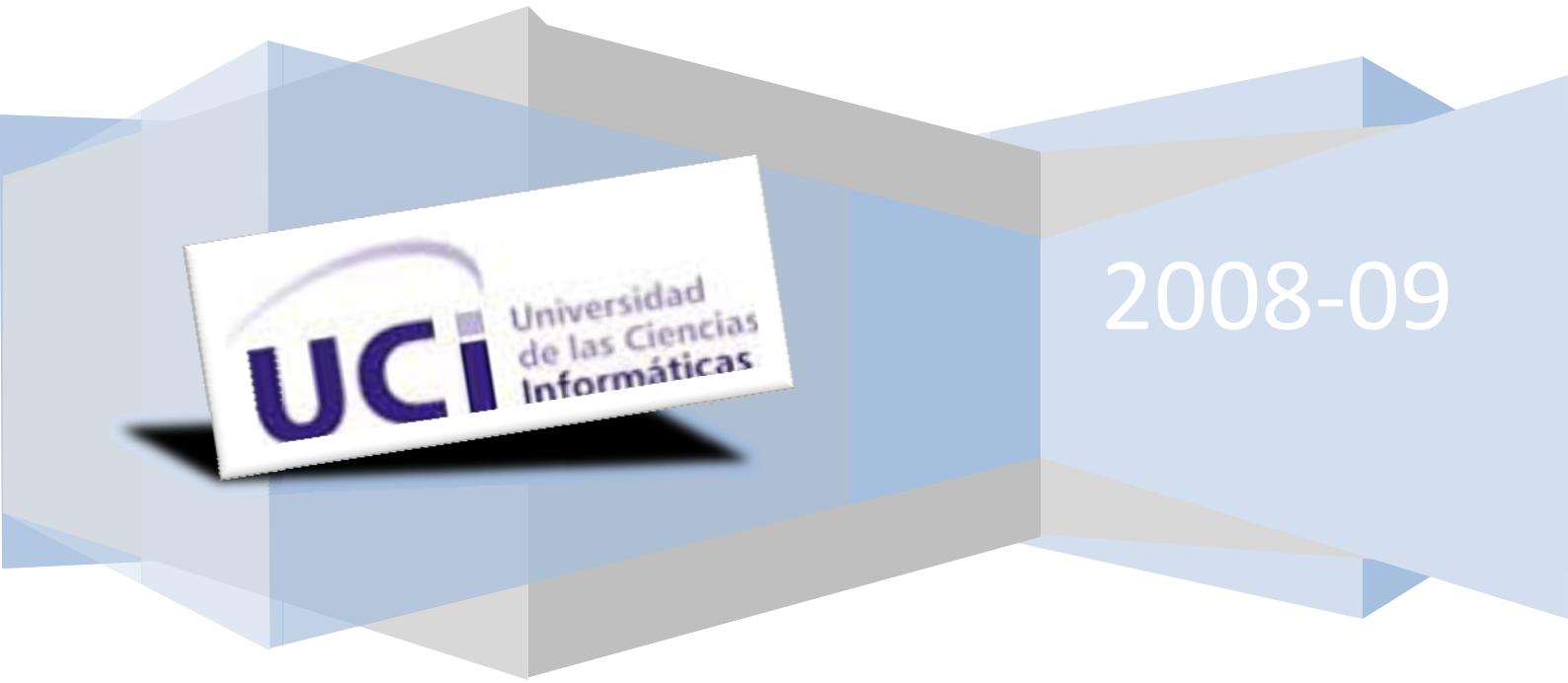
Universidad de las Ciencias Informáticas
Facultad V.

Título: Sistema Automatizado para la organización del Taller de IS en la UCI.

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

Autor: Adrian Gattorno Gil.

Tutor: MSc. Yamilis Fernández Pérez.



2008-09

DATOS DE CONTACTO

Yamilis Fernández Pérez. Máster en ciencias en Informática Aplicada.

- ✓ Graduada de Ingeniera en Sistema Automatizado de Dirección, en 1992 en el ISPJAE
- ✓ Profesora asistente desde 1995.
- ✓ MSc en Informática Aplicada en 1995.
- ✓ Imparte docencia en universidades desde 1992.
- ✓ Ha desarrollado trabajos con Universidades extranjeras en Brasil, Bolivia, Canadá.
- ✓ Es la jefa de departamento docente central de Ingeniería y Gestión de Software de la UCI desde su fundación.

e-mail: yamilisf@uci.cu

AGRADECIMIENTOS

A la UCI por ser para mí más que una escuela, que con sus cosas buenas y malas me ha hecho ser una mejor persona, me ha preparado y me ha enseñado mucho. Por darme tantos y tan buenos amigos.

A mi tutora Yamilis que no tengo cómo agradecerle su incondicional y constante ayuda. Por atenderme en cualquier momento y aconsejarme tanto.

A todas mis amistades que me soportaron todos estos años, que me brindaron su ayuda y su apoyo en los momentos en que lo necesité. A todos esos que estuvieron conmigo desde primer año y que al final aprendimos a soportarnos, a Vanegas, Anniester, Yeisnier (el loco) o sea a toda esa pandilla de locos y locas.

A mi bruja, la Leya, por brindarme su apoyo y darme ánimos y fuerzas para seguir.

A mis padres: por su amor, sus enseñanzas, sus años de sacrificio, su confianza, sus consejos, y apoyo. A mi hermana que siempre me está ayudando y apoyando y que lo da todo por mí. A Adalito que es como otro hermano.

A la Revolución y a Fidel: por el orgullo de participar en este proyecto, por sus ideas y su ejemplo.

DEDICATORIA

A mis padres y mi hermana que son todo para mí al igual que yo lo soy para ellos. Por guiarme para ser la persona que soy hoy. Que han sufrido mi carrera tanto o más que yo mismo.

A toda mi gente de la UCI por ayudarme a llegar hasta aquí.

Resumen:

La Universidad de las Ciencias informáticas se plantea elevar la calidad de la investigación científica del centro. Con este objetivo la universidad ha estado convocando, organizando y realizando diversos eventos científicos, entre ellos el Taller de Ingeniería, Arquitectura y Gestión de Software. En este evento se realizan un conjunto de tareas que conllevan a manejar un gran flujo de información que debe procesarse de forma rápida y segura. Actualmente estas tareas se realizan de forma manual produciendo pérdida de tiempo y de calidad en el evento.

Este trabajo propone un sistema automatizado para la organización del Taller de Ingeniería, Arquitectura y Gestión de Software. En él se desarrolla todo el ciclo de vida del software, comenzando con la identificación de los principales procesos del Taller, que son objeto a automatizar, y finalizando con la implementación de un sistema informático que gestione dichos procesos. Para esto se utiliza como metodología de desarrollo de software a RUP, y una serie de tecnologías libres que propician una buena calidad y eficiencia al sistema.

Índice:

Índice de Figuras:

Índice de Tablas:

Introducción:

La Universidad de las Ciencias Informáticas (UCI) es un centro creado a raíz de la batalla de ideas, cuya misión es formar de manera continua profesionales integrales comprometidos con la patria, y ser el soporte de la informatización del país y la competitividad internacional de la industria cubana del software. Para ello, la Dirección de Investigación trabaja incansablemente por la organización y el buen desempeño de la actividad científica del centro. Sus esfuerzos están dirigidos a organizar las investigaciones en la Universidad estimulando la participación de profesores y estudiantes, a establecer alianza estratégica con el MES, el CITMA y el Polo Científico, e incorporar el centro al sistema científico nacional. Con esto se quiere obtener resultados científico-técnicos de impacto, convirtiéndose la Universidad en el espacio de referencia nacional en innovación tecnológica en las TIC, logrando la transferencia del 50% de los resultados aplicables de la investigación en negocios concretos.

Para lograr esto, la universidad ha estado convocando, organizando y realizando múltiples eventos científicos con el fin de incentivar la investigación en la universidad, tanto por parte de estudiantes como de profesores, y de tener un espacio donde estas investigaciones se expongan a la comunidad. Algunos de esos eventos son: UCIENCIA, los Fórum de Ciencia y Técnica, las Jornadas Científicas Estudiantiles (JCE), y otros tantos.

Entre los talleres que se realizan en el marco de UCIENCIA, se encuentra el Taller de Ingeniería, Arquitectura y Gestión de Software. La responsabilidad de la organización y control de este evento recae sobre el Departamento Docente Central de Ingeniería y Gestión de Software y la Dirección de Calidad de Software. Entre sus obligaciones está la recepción, aceptación o no y evaluación de todos los trabajos científicos. Estos procesos son realizados actualmente de forma manual pero se están haciendo demasiado agotadores y trabajosos debido al crecimiento de las investigaciones científicas, así como de la cantidad de talleres y exposiciones que se realizan en los eventos. Los datos que se exponen a continuación pueden dar una idea de este crecimiento de la actividad científica en la Universidad:

- En **UCIENCIA 2005** se realizaron 15 convocatorias de talleres, en los cuales fueron aprobados por las comisiones y expuestos 126 trabajos científicos. En ese marco tuvo lugar el 1er Taller de Ingeniería y Gestión de Software, en el cual se expusieron un total de 8 trabajos científicos y sesionó una sola comisión. (www.investigaciones.uci.cu).

- En **UCIENCIA 2006** fueron 20 las convocatorias de talleres, en los cuales fueron 359 los trabajos científicos que fueron aprobados por las comisiones. Se desarrolló el II Taller de Ingeniería y Gestión de Software, en el cual se expusieron 17 trabajos científicos, todos en una única comisión. (www.investigaciones.uci.cu).

- En **UCIENCIA 2007** los talleres convocados fueron 19, en los cuales se aprobaron y se expusieron 574 trabajos científicos. Durante este evento se desarrolló el III Taller Calidad e ISW con 59 trabajos científicos expuestos entre 7 comisiones. (www.investigaciones.uci.cu).

- Para este año, en el marco de **UCIENCIA 2008,** se convocaron 25 talleres. Se desarrolló el IV Taller de Ingeniería, Arquitectura y Gestión de Software en el cual se expusieron un total de 108 trabajos científicos en 11 comisiones. (www.uciencia.uci.cu).

Todo esto hace que sea muy difícil realizar los procesos anteriormente mencionados con la rapidez y calidad que se necesita, lo que provoca mal funcionamiento del evento debido a que las comisiones no pueden hacer su mejor trabajo, lo cual trae consigo que haya también problemas de divulgación, organización y control del evento virtual.

Para dar solución a esta situación se define como problema científico lo siguiente:

¿Cómo gestionar los procesos de divulgación, funcionamiento, organización y control de un evento virtual de Ingeniería de Software (IS) en la UCI?

El objeto de estudio radica en la gestión de información asociada a un evento virtual de Ingeniería de Software.

El campo de acción de este trabajo se centra en la gestión de información asociada a los procesos de divulgación, funcionamiento, organización y control de un evento virtual de Ingeniería de software en la UCI.

El objetivo general de este trabajo es diseñar e implementar una aplicación Web para la organización, divulgación y funcionamiento de un evento virtual de Ingeniería de software (IS) en la UCI.

Para dar cumplimiento a este objetivo se desarrollarán diferentes tareas tales como:

- Caracterizar los eventos virtuales y la gestión de estos en el mundo.
- Caracterizar los procesos elementales del negocio.
- Realizar un estudio del estado del arte de la tecnología a utilizar, analizando la posibilidad de su uso para implementar el software.
- Definir la arquitectura a utilizar en la aplicación a desarrollar.
- Implementar la Aplicación Web haciendo uso de una metodología de desarrollo de Software.

Como idea a defender se plantea que si se desarrolla una aplicación Web que automatice la organización, divulgación y funcionamiento de un evento virtual de Ingeniería de Software (IS), mejorará la calidad y organización de dicho evento.

A su vez, para guiar la investigación científica se utilizarán los métodos:

— Teóricos:

- Analítico _ Sintético: Se estudian y analizan las teorías y la documentación sobre los lenguajes de programación a utilizar, así como las metodologías y conceptos. Lo cual permite determinar las metodologías más apropiadas y aplicar las mejores prácticas de programación.

- Inductivo _ deductivo: Son formas de razonamiento que permiten llegar a un grupo de conocimientos generalizadores, tanto desde el

análisis de lo particular a lo general, como desde el análisis de los elementos generalizadores a uno de menor nivel de generalización.

— Empíricos:

- Observación: Registro visual de lo que ocurre en una situación real, en un fenómeno determinado, clasificando y consignando los hechos y acontecimientos pertinentes de acuerdo con algún esquema previsto.

1 CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este Capítulo se detallan los procesos elementales del negocio y se realiza un estudio y análisis de las tecnologías apropiadas para desarrollos de aplicaciones Web, escogiendo la mejor opción para la construcción del sistema propuesto.

1.2 Sistemas automatizados existentes vinculados al objeto de estudio y al campo de acción:

Los sistemas de gestión de eventos virtuales de Ingeniería de Software existentes en el mundo tienen entre sus características más importantes las siguientes:

- Sistema de promoción que permite, tanto brindar información al público sobre el evento mediante anuncios o noticias, como informar a los usuarios de la realización y progreso de las actividades del evento.
- Recibir artículos y evaluar por un tribunal de personas correctamente calificadas si debe ser publicado.
- Son aplicaciones hechas a medida.
- Son sistemas sujetos a derechos de autor y realizados con aplicaciones propietarias.

Luego de un estudio realizado a estos sistemas, se determinó que ninguno cumplía con los requisitos que se necesitan para el Sistema Automatizado para la Organización del Taller Virtual de IS en la UCI. Esto es debido a que algunas de las características anteriormente mencionadas están en oposición a las políticas de la universidad, y otras como el ser aplicaciones hechas a medida, hacen que dichos sistemas sean prácticamente imposibles de adaptar a las necesidades existentes. Además, no se encontró ningún CMS o aplicación libre que brindara las prestaciones que se necesitan. Se pueden citar a: APSEC 08 y 2009 World Congress on Software Engineering (WCSE 2009), SEPGLA, SEPG.

En la actualidad, en la Universidad no se ha llegado a realizar ningún software relacionado con este campo de acción que tenga las características

que se necesitan, ya que aunque existen sitios relacionados con la investigación científica, estos se pueden incluir en el mismo caso de los internacionales que solo cumplen con algunas características y que su implementación no permite reutilizar el código o extenderlo para cumplir con los requisitos planteados. Se pueden citar a UCIENCIA y al sitio de la Dirección de Investigaciones que son meramente informativos y al sitio de la Serie Científica que se realizó para atender necesidades específicas.

1.3 Procesos elementales del negocio:

Para la creación del Sistema Automatizado para la Organización del Taller Virtual de IS en la UCI, se detallaron diferentes procesos a automatizar que cubren las necesidades fundamentales de dicho sistema. Este trabajo se propone, partir de esos procesos, a los cuales se le realizarían algunas modificaciones pertinentes para la construcción del sistema de forma que cubra la totalidad de las necesidades y preocupaciones de los organizadores del evento, para con esto lograr una excelente calidad en estos Talleres Virtuales.

A continuación se muestran los principales procesos a automatizar:

- **Organización del evento:** Se realizan diferentes actividades como selección del Comité Organizador y Comité Académico. Se hacen plegables con información sobre el evento. Se organiza la logística del evento y se decide el primer llamado.
- **Promoción del evento virtual:** Para una buena promoción del evento, los organizadores necesitan contar con un sistema de promoción de tipo informativo que permita mantener constantemente informados de los pormenores del evento a participantes, Comité Académico, organizadores, entre otros, donde uno de los métodos de información es el envío de correos electrónicos con información sobre el evento.
- **Recepción de los trabajos investigativos:** El evento necesita contar con un sistema que permita, de forma automática, recepcionar los trabajos. Una vez recibidos los trabajos, si son aceptados, se distribuyen por la comisión que le corresponda para que sean revisados y evaluados.

- **Gestionar las publicaciones:** El Jefe de cada taller comprueba las publicaciones asignadas, las acepta o rechaza, y le asigna revisores para que sean revisadas y evaluadas.
- **Notificar al jurado de la comisión:** Cuando se asigna una publicación a una comisión, se le envía una notificación a cada uno de los revisores a su buzón de correo y a su perfil en el sistema.
- **Evaluación de las publicaciones:** Cada uno de los revisores de la comisión, una vez analizado el documento, procede a evaluar la publicación. Se envía un correo electrónico al autor principal notificando la aceptación o el rechazo del trabajo.
- **Formar el programa académico:** El Comité Académico se dispone a conformar el plan de trabajo del evento, a organizar los talleres y demás eventos en los salones disponibles y en los horarios posibles.

1.4 Tecnologías para el desarrollo:

1.4.1 Plataforma de desarrollo y lenguajes de programación Web:

- **Plataforma de desarrollo Web: Symfony.**

(…)Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (…) (Potencier, y otros, 2009)

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web. Añade una capa por encima de PHP y proporciona herramientas que simplifican el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

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.

Symfony se diseñó para que se ajustara a los siguientes requisitos (entre otros):

- 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.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web
- Lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como por ejemplo:

• La capa de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework.

• Los formularios incluyen validación automatizada y relleno automático de datos (“repopulation”). Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.

• La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.

• El soporte de e-mail incluido y la gestión de APIs, permiten a las aplicaciones web interactuar más allá de los navegadores.

• Los plugins, las factorías (patrón de diseño “Factory”) y los “mixin”, permiten realizar extensiones a medida de Symfony.

• Las interacciones con Ajax son muy fáciles de implementar mediante los *helpers* que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código.

Toda la filosofía de desarrollo de Symfony está creada sobre la base de ciertos conceptos básicos como: **OOP, ORM, DRY, TDD, YAML y PEAR,** entre otros.

- **PHP:**

"PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. Este lenguaje permite escribir, a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil."

PHP es uno de los lenguajes de lado servidor más extendidos en la Web. Puede ser utilizado en cualquiera de los principales sistemas operativos del mercado: Microsoft Windows y la mayoría de las variantes Unix (Solaris y Linux, Mac OS X). PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server y muchos otros.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Algunas de las bases de datos que están soportadas actualmente son:

Oracle (OCI7 and OCI8).	ODBC.
PostgreSQL.	mSQL.
MySQL.	Hyperwave.

También cuenta con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), así que puede conectarse a cualquier base de datos que soporte tal estándar.

PHP tiene características muy útiles para el procesamiento de texto, desde expresiones regulares POSIX extendidas o tipo Perl, hasta procesadores de documentos XML. (PHP Documentation Group)

- **AJAX**

El término AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como "JavaScript asíncrono + XML".

"Ajax no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes." (Jesse James Garrett).

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano. Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor. (Pérez, 2007)

- **Hojas de Estilo en Cascada (Cascading Style Sheets, CSS):**

Es un lenguaje de hoja de estilo que permite que los autores y los usuarios asocien un estilo a los documentos estructurados (por ejemplo, documentos HTML y aplicaciones XML). Separando el estilo de presentación del contenido

de los documentos, CSS simplifica la creación y mantenimiento de los sitios Web.

Éste brinda soporte específico a hojas de estilo específicas para cada medio, de modo que los autores puedan adaptar la presentación de sus documentos a los navegadores, a los dispositivos sonoros, a las impresoras, a los dispositivos de braille, de mano, etc. También soporta el posicionamiento de contenidos, disposición de tablas, y algunas características relacionas con la interfaz del usuario. (W3C Candidate Recommendation)

- **YAML**

Según el sitio web oficial de YAML (http://www.yaml.org/), YAML es el acrónimo de "*YAML Ain't Markup Language*" ("YAML No es un Lenguaje de Marcado"); es "*un formato para serializar datos que es fácil de procesar por las máquinas, fácil de leer para las personas y fácil de interactuar con los lenguajes de script*". Dicho de otra forma, YAML es un lenguaje muy sencillo que permite describir los datos como en XML, pero con una sintaxis mucho más sencilla.

YAML es mucho más rápido de escribir que XML y es mucho más poderoso que los tradicionales archivos .ini (ya que estos últimos no soportan la herencia y las estructuras complejas). Symfony utiliza el formato YAML como el lenguaje preferido para almacenar su configuración. (Potencier, y otros, 2009)

1.4.2 Herramientas de gestión y diseño de bases de datos:

- **ERwin:**

ALLFusion Erwin Data Modeler es una herramienta de diseño de bases de datos que ayuda a generar, mantener alta calidad y permite gran rendimiento en las aplicaciones de bases de datos, desde un modelo lógico dado por los requerimientos de información y las reglas de negocio al modelo físico optimizado por las características específicas de las bases de datos. ALLFusion Erwin Data Modeler permite visualizar la estructura y elementos clave, además optimizar el diseño de las bases de datos. Automáticamente

genera tablas y cientos de líneas de procedimientos almacenados y código trigger para las bases de datos. La tecnología “complete-compare” permite el desarrollo iterativo para que los modelos estén siempre sincronizados con la base de datos. Esta tecnología permite:

- Fácil acceso a cualquier base de datos relacional.
- Comparación comprensiva entre el modelo de datos y la base de datos.
- Soporta la separación del modelo lógico y del físico. (MANAGEMENT 2008)

- **PostgreSQL**

Es un sistema de gestión de base de datos de objetos relacionales. Soporta una gran parte del estándar SQL. Permite, mediante un sistema denominado MVCC (Acceso concurrente multi versión, por sus siglas en inglés), que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

Se puede integrar con una amplia variedad de lenguajes de programación como:

- C/C++.
- Java PL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.

Entre las ventajas de este sistema de gestión de bases de datos se pueden nombrar las siguientes (entre otras):

- Estabilidad y confiabilidad legendarias: Es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad.

- Extensible: El código fuente está disponible para todos sin costo. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo, que también extienden PostgreSQL todos los días.

- Multiplataforma: PostgreSQL está disponible en casi cualquier Unix y de Windows.

- Por la licencia libre a la que está sujeto, PostgreSQL puede ser usado, modificado, y distribuido por todo el mundo libre de cargo para cualquier propósito, ya sea privado, comercial, o académico. (The PostgreSQL Global Development Group)

- **EMS SQL Manager for PostgreSQL.**

EMS SQL Manager for PostgreSQL es una herramienta de alto rendimiento para la administración y desarrollo de servidores de base de datos en PostgreSQL. Soporta todas las últimas características del sistema, incluyendo parámetros de factor de relleno en tablas e índices, construcción de índices concurrentes, creación de dominios basados en otros dominios, entre otros.

Ofrece muchas herramientas poderosas como el Visual Database Designer, Visual Query Builder, y un poderoso editor de BLOB para satisfacer todas las necesidades. SQL Manager for PostgreSQL tiene una nueva interfaz gráfica de usuario enormemente amigable e intuitiva. Permite una administración y navegación rápida de las bases de datos. Fácil administración de todos los objetos de PostgreSQL. (EMS Database Management Solutions, Inc.)

1.4.3 Herramientas de modelado:

- **UML (Lenguaje Unificado de Modelado)**

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. UML es un estándar para describir un modelo, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Está respaldado por el OMG (Object Management Group).(OMG, 2009)

Pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. (Object Management Group, Inc. (OMG), 2007)

1.4.4 Metodología de desarrollo de Software.

- **El Proceso Unificado de Desarrollo de Software (RUP)**:

El Proceso Unificado de Desarrollo de Software es una de las metodologías más generales y más usadas de las que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además una propuesta de proceso para el desarrollo de software orientado a objeto.

El RUP provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales.

El Proceso Unificado se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas. Éste además usa el Lenguaje Unificado de Modelado (UML) en la preparación de todos los planos del sistema. Los aspectos distintivos del Proceso Unificado están capturados en tres conceptos clave:

- Dirigido por casos de uso (use-case driven): Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Los casos de uso capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema y dirige el proceso de desarrollo.

- Centrado en la arquitectura (architecture-centric): El concepto de arquitectura de software involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, tal y como las interpretan los usuarios y otros stakeholders, y tal y como están reflejadas en los casos de uso. La arquitectura es la vista del diseño completo con las características más importantes.

- Iterativo e incremental: Es práctico dividir el trabajo en pequeños pedazos o mini-proyectos. Cada mini-proyecto es una iteración que finaliza en un incremento. Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. (ENRÍQUEZ, 2005)

1.4.5 Herramientas de desarrollo Web:

- **PhpDesigner 6.1**

Es un completo ambiente integrado de desarrollo para PHP tanto para desarrolladores principiantes como profesionales. Mejora el proceso de edición, análisis, corrección de fallos del programa y la publicación de aplicaciones y sitios Web creados con PHP y otros lenguajes Web. Diseñado para mejorar la productividad y simplificar la codificación mediante un conjunto de herramientas inteligentes de edición. No sólo soporta PHP sino también otros lenguajes web como HTML, XHTML, MySQL, XML, CSS, JavaScript, VBScript, Java, C#, Perl, Python y Ruby. En PhpDesigner 6.1 se puede trabajar con proyectos e integrar cualquier PHP framework al proyecto y acceder a todos los archivos, clases, funciones, interfaces, variables y constantes declaradas en el proyecto en cualquier momento.

La corrección de errores es fácil debido a la tecnología Xdebug integrada en phpDesigner 6.1, la cual permite analizar el código paso a paso usando puntos de ruptura (breakpoints), esperas (watches), evaluaciones etc. Esta tecnología permite encontrar cuellos de botella en el código y determinar si alguna parte es muy lenta. (MPSOFTWARE)

1.4.6 Otras herramientas propuestas:

Para confeccionar la propuesta de este trabajo, se propone utilizar como editor de páginas Web el Macromedia Dreamweaver CS3, por ser la herramienta de creación de sitios Web más utilizada en la actualidad y poseer un amplio soporte para la creación y utilización de CSS, logrando un diseño sencillo y óptimo.

Para modelar la aplicación, se propone utilizar Rational Rose Enterprise Edition 2003, herramienta líder para este propósito, la cual facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue).

1.5 Conclusiones:

Luego de haber realizado un profundo análisis de las principales características de las tecnologías más utilizadas en la actualidad, se acuerda utilizar:

- PHP como lenguaje de programación del lado del servidor, debido a las ventajas que presenta: multiplataforma, multisistema operativo, con una sintaxis familiar y bien definida a los programadores. Es bastante sencillo y fácil de aprender. Dispone de gran cantidad de documentos y ejemplos en Internet y además es libre. Como ambiente de desarrollo se escoge Symfony, y como IDE de desarrollo PhpDesigner por la fortaleza y facilidades que brindan para el desarrollo de aplicaciones.
- Como sistema gestor de bases de datos se selecciona PostgreSQL por la gran fortaleza, estabilidad, fiabilidad y seguridad con los datos que brinda este sistema. No es tan rápido como otros sistemas, pero es capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta y presenta mayor cantidad de tipos de datos, lo que permite optimizar espacio en la Base de Datos. Es software libre, multiplataforma y es el estándar que se quiere adoptar en la universidad.

- Como modelador de datos el ERwin, ya que permite visualizar todos los datos de una base de datos.
- Como lenguaje de marcado YAML, ya que es mucho más rápido y sencillo de escribir que XML, y es mucho más poderoso que los tradicionales archivos .ini.
- Como metodología de desarrollo RUP, por todas las ventajas de organización que brinda y por venir acompañada de una potente herramienta que soporta todos los procesos básicos de RUP: Suite del Rational. Además se utilizará a UML como lenguaje Unificado de Modelado.

2 CAPITULO 2: REQUISITOS.

2.1 Introducción:

Después de haber analizado y refinado cada uno de los requisitos imprescindibles para el funcionamiento del nuevo sistema, este capítulo se propone mostrar a partir de estos requisitos el Diagrama de Caso de Uso del Sistema, el cual representa las relaciones de los actores que interactúan con el sistema y el flujo de actividades con las que interactúan.

"(…) Básicamente, un requisito del software es una característica que se debe exhibir para solucionar un cierto problema del mundo real. Por lo tanto, un requisito del software es una característica que se debe exhibir por el software desarrollado o adaptado para solucionar un problema particular. Por lo tanto, los requisitos de software son típicamente una combinación compleja de requisitos de diversa gente en diversos niveles de una organización y del ambiente en el cual el software funcionará. (IEEE Computer Society, 2004)

2.2 Requisitos Funcionales.

Los requisitos funcionales describen las funciones que el software va a ejecutar; por ejemplo, ajustarse a un formato de texto o modular una señal. Se conocen también como capacidades. (IEEE Computer Society, 2004)

RF1. Gestionar Comité Organizador (CO).

- **RF1.1.** Adicionar integrantes del CO.
- **RF1.2.** Eliminar integrantes del CO.
- **RF1.3.** Modificar datos de integrante del CO.
- **RF1.4.** Mostrar integrantes del CO.

RF2. Gestionar Comité Académico (CA).

- **RF2.1.** Adicionar integrante del CA.
- **RF2.2.** Eliminar integrante del CA.
- **RF2.3.** Modificar datos de integrante del CA.
- **RF2.4.** Mostrar integrantes del CO.

RF3. Gestionar Comisiones:

RF3.1. Adicionar comisión.

RF3.2. Modificar datos comisión.

RF3.3. Eliminar comisión.

RF3.4. Mostrar comisión.

RF4. Publicar artículos:

RF4.1. Mostrar planilla “Normas de Publicación”.

RF4.2. Validar y almacenar datos de la Planilla.

RF4.3. Almacenar el documento de la publicación.

RF4.4. Asignar el artículo a la comisión del tema que le corresponde.

RF4.5. Mostrar listado de artículos.

RF5. Gestionar artículos:

RF5.1. Mostrar y revisar los artículos nuevos.

RF5.2. Mostrar los artículos aceptados o no.

RF5.3. Aceptar o rechazar artículos.

RF5.4. Asignarle revisores al artículo.

RF5.5. Notificarles a los revisores.

RF6. Evaluar artículos:

RF6.1. Mostrar artículos por evaluar.

RF6.2. Abrir artículo para revisarlo.

RF6.3. Completar planilla para evaluar artículo.

RF6.4. Validar y almacenar los datos correctamente.

RF6.5. Mostrar artículos evaluados (título, autor, nota, fecha).

RF6.6. Enviar notificación al Jefe de la Comisión y al autor del artículo.

RF7. Gestionar el programa académico:

RF7.1. Adicionar actividad.

RF7.2. Eliminar actividad.

RF7.3. Modificar actividad.

RF8. Autentificar Usuario.

RF8.1. Verificar usuario y contraseña en el sistema.

RF8.2. Chequear y almacenar privilegios de acceso a una página abierta.

RF9. Gestionar Usuarios.

RF9.1. Adicionar un nuevo usuario al sistema (nombre, email, grupo, permisos).

RF9.2. Modificar usuario existente.

RF9.3. Eliminar usuario.

RF9.4. Mostrar usuarios.

RF10. Gestionar Grupos De Usuarios:

RF10.1. Adicionar un nuevo grupo al sistema (nombre, descripción, permisos)

RF10.2. Eliminar un grupo existente.

RF10.3. Modificar un grupo existente.

RF10.4. Mostrar grupos.

RF11. Gestionar Permisos.

RF11.1. Adicionar permiso (tipo, descripción).

RF11.2. Modificar permisos.

RF11.3. Eliminar permiso.

RF11.4. Mostrar permisos.

RF12. Gestionar promoción.

RF12.1. Publicar noticias (título, descripción, fecha).

RF12.2. Modificar noticias.

RF12.3. Eliminar noticias publicadas.

RF12.4. Mostrar Noticias.

RF12.5. Subir plegable.

RF12.6. Enviar plegable.

2.3 Definición de los requerimientos no funcionales.

Un requisito no funcional es un requisito que especifica los criterios que se pueden utilizar para juzgar el funcionamiento de un sistema o de funciones específicas. Esto debe contrastarse con las necesidades funcionales que definen el comportamiento o funciones específicas.

En general, los requisitos funcionales de un sistema definen qué es lo que se supone que haga, mientras que los requisitos no funcionales definen cómo un sistema se supone que sea. Los requisitos no funcionales son a menudo llamados cualidades de un sistema. Otros términos de requisitos no funcionales son "limitaciones", "los atributos de calidad", "objetivos de calidad" y "calidad del servicio". (Object Management Group, Inc. (OMG))

A continuación se detallan los requerimientos no funcionales del sistema:

1. **Apariencia o interfaz externa:**
   - **RNF1.1.** Diseño orientado a llamar la atención del usuario y con una navegación sencilla.
   - **RNF1.2.** Construcción de enlaces rápidos o anclas para los documentos muy largos.
2. **Usabilidad:**
   - **RNF2.1.** El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
3. **Rendimiento:**
   - **RNF3.1.** Tiempos de respuestas no mayores de 10 segundos, al igual que la velocidad de procesamiento de la información.
4. **Soporte:**
   - **RNF4.1.** Se requiere un servidor de bases de datos con las siguientes características:
     - **RNF4.1.1.** Soporte para medianos volúmenes de datos y velocidad de procesamiento.
     - **RNF4.1.2.** Tiempo de respuesta de no más de 7 segundos en accesos concurrentes.

5. Portabilidad:

RNF5.1. Necesidad de que el sistema sea multiplataforma.

6. Seguridad:

RNF6.1. Identificar al usuario antes de que pueda realizar cualquier acción sobre el contenido del sistema.

RNF6.2. Garantizar que la información sea editada únicamente por quien tiene derecho a editarla.

RNF6.3. Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de acceso del usuario activo.

RNF6.4. Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

RNF6.5. Verificación sobre acciones irreversibles (eliminaciones).

7. Legales:

RNF7.1. La plataforma escogida para el desarrollo de la aplicación, está basada en la licencia GNU/GPL.

8. Confiabilidad:

RNF8.1. La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

9. Funcionalidad:

RNF9.1. Guardar en caché páginas de contenido para agilizar la navegación del portal.

10. Software:

RNF10.1. Navegador compatible o superior con Internet Explorer 4, o NetsCape Navegator.

RNF10.2. Macromedia Dreamweaver 8.

RNF10.3. PostGreSQL 8.2.x

RNF10.4. Apache 2.x.x

RNF10.5. PHP 5.x.x.

2.4 Actores del sistema.

Un actor modela un tipo de rol realizado por una entidad que interactúa con un sistema, un subsistema, o clase. Los actores pueden representar papeles jugados por usuarios humanos, hardware externo u otros sujetos. Caracterizan las interacciones que los usuarios exteriores pueden tener con el sistema.

Pueden ser definidos en jerarquías de generalización. Un actor se representa como un icono de un hombre con rayas finas y el nombre cerca (usualmente debajo o encima) de icono. (Object Management Group, Inc. (OMG))

Tabla 1. Descripción de Actores

Nombre del actor	Descripción
Participante	Aquel usuario que solo puede publicar artículos o ver algunos datos de los ya publicados.
Presidente Comité Organizador	Es el encargado de gestionar y organizar el Comité Organizador.
Secretario Comité Organizador	Es el encargado de realizar el primer llamado a la lista de contactos. De administrar el sistema de promoción del evento, gestionando noticias y el envío de plegables.
Presidente Comité Académico	Es responsable de gestionar y organizar el Programa Académico, el Comité Académico y las comisiones.
Jefe de Comisión	Es el encargado de aceptar o no los trabajos asignados a su comisión, y de enviarlos a revisión. Además de supervisar el trabajo de la comisión.
Revisor	Es el encargado de revisar los artículos y evaluarlos
Administrador	Es el responsable de instalación, soporte y mantenimiento del sistema, de gestionar roles y usuarios.

2.4.1 Vista Global de actores.

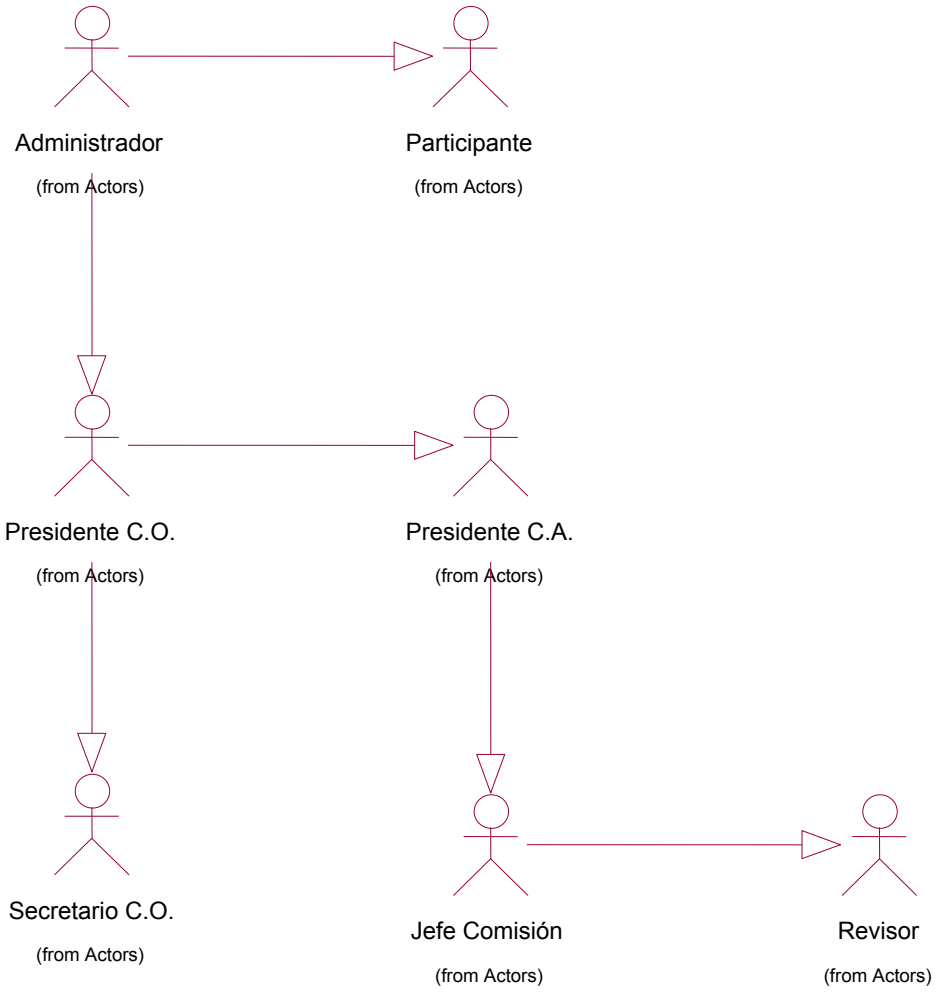


Figura 1 Vista Global de actores

2.5 Diagrama de Caso de Uso.

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. (Object Management Group, Inc. (OMG))

2.5.1 Diagrama de Paquetes

Un paquete es una parte de un modelo. Cada parte del modelo debe pertenecer a un paquete. Para ser funcional, la asignación debe seguir un

cierto principio Rational, tal como funcionalidad común, implementación relacionada y punto de vista común. Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas, agrupando elementos de modelado. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general de los modelos de UML, por lo que un diagrama de paquetes permite dividir al sistema orientado a objetos, organizándolo en subsistemas y detallando sus relaciones. (Object Management Group, Inc. (OMG))

2.5.1.1 Paquete: Promoción del Evento

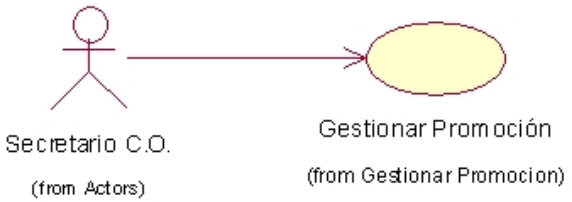


Figura 2 Paquete Promoción del Evento

2.5.1.2 Paquete: Administración

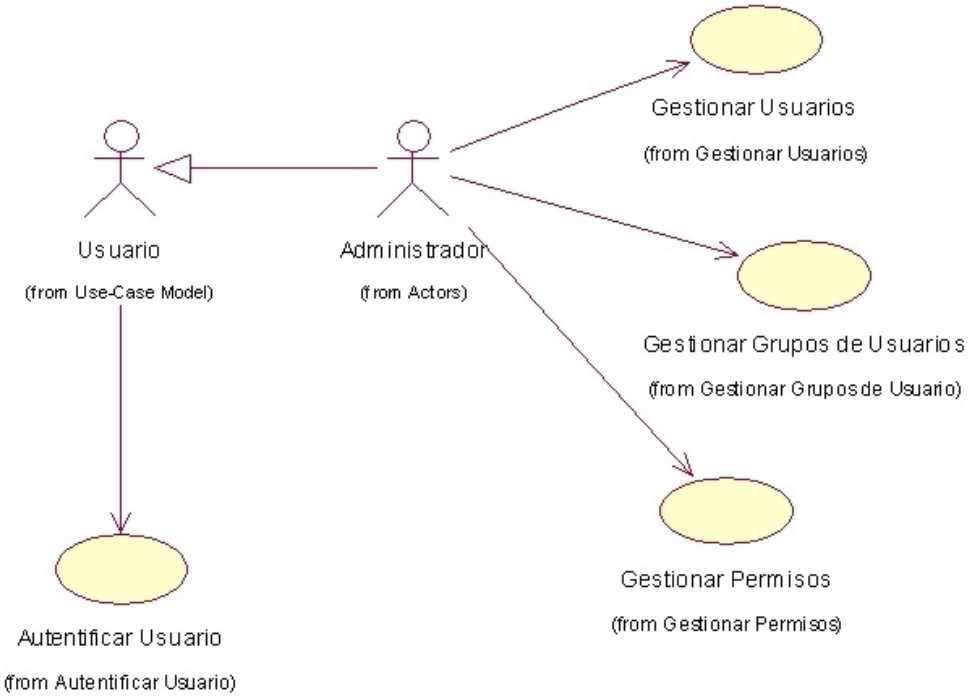


Figura 3 Paquete Administración

2.5.1.3 Paquete: Evaluación de las publicaciones.

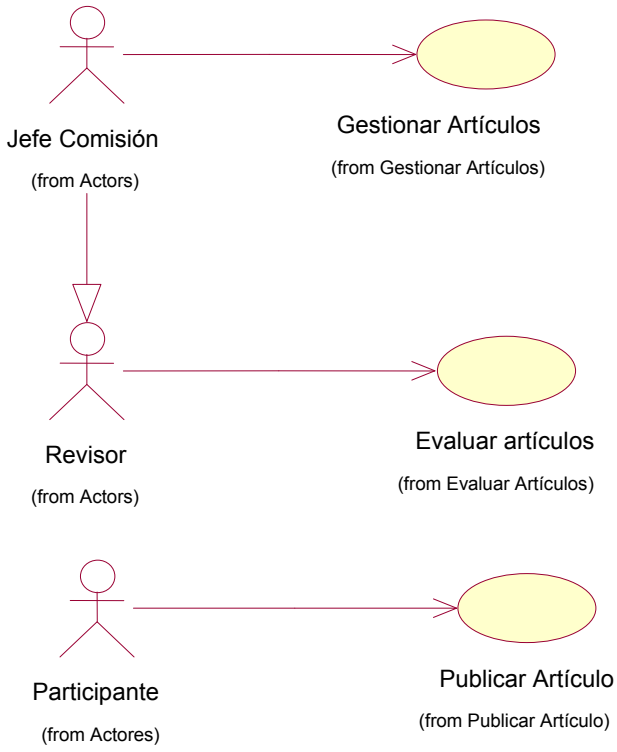


Figura 4 Paquete Evaluación de las publicaciones

2.5.1.4 Paquete: Organización del Evento.

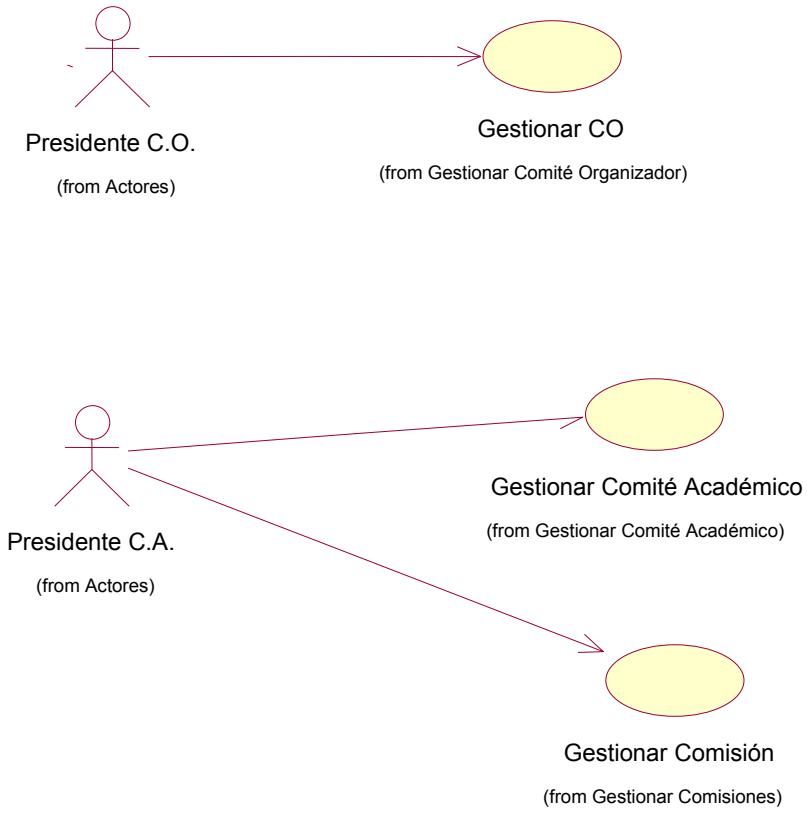


Figura 5 Paquete Organización del Evento

2.6 Descripción de casos de uso

Tabla 1 Gestionar Comité Organizador (CO)

CU-1	Gestionar Comité Organizador (CO)
Actor	Presidente Comité Organizador
Resumen	El caso de uso se inicia cuando el Presidente del CO consulta el listado de integrantes del CO, pudiendo adicionar integrantes introduciendo datos como: nombre, correo, cargo. Se puede volver a iniciar si decide modificar algunos de los datos o eliminar algunos de los integrantes del CO. El caso de uso culmina cuando se muestra el CO, pudiendo imprimirlo y enviarlo por correo.
Referencia	RF1, (RF1.1, RF1.2, RF1.3, RF1.4)
Precondiciones	El Presidente Comité Organizador debe estar autenticado en el sitio.
Pos condiciones	Quedan creados los integrantes del CO con sus roles correspondientes

Tabla 2 Gestionar Comité Académico (CA)

CU-2	Gestionar Comité Académico(CA)
Actor	Presidente del Comité Académico
Resumen	Este caso de uso se inicia cuando el presidente del CA consulta el listado de integrantes del CA, los cuales pueden ser Presidente de Comisión o Revisor, pudiendo adicionar integrantes asignándoles roles y distribuyéndolos por las comisiones. Se puede volver a iniciar si decide modificar algunos de los datos o eliminar alguno de los integrantes del CA.
Referencia	RF2, (RF2.1, RF2.2, RF2.3, RF2.4)
Precondiciones	El Presidente del CA debe estar autenticado en el sistema.
Pos condiciones	Queda organizado el CA y distribuido el personal por las comisiones.

Tabla 3 Gestionar Comisiones

CU-3	Gestionar Comisiones
Actor	Presidente del Comité Académico
Resumen	Este caso de uso se inicia cuando el presidente del CA consulta el listado de comisiones existentes, pudiendo adicionar comisiones. Se puede volver a iniciar si decide modificar algunos de los datos o eliminar alguna de las comisiones. Quedan conformadas así todas las comisiones.
Referencia	RF3, (RF3.1, RF3.2, RF3.3, RF3.4)
Precondiciones	El Presidente del CA debe estar autenticado en el sistema.
Pos condiciones	Quedan conformadas las comisiones.

Tabla 4 Publicar artículos

CU-4	Publicar artículos
Actor	Participante.
Resumen	El caso de uso se inicia cuando el participante acede a ver el listado de artículos que han sido subidos a la aplicación pudiendo decidir publicar un artículo, para lo cual introduce los datos de la planilla de publicación (titulo, autor, resumen, entre otros) y sube el documento. El caso de uso culmina cuando se muestra una notificación de que los datos se procesaron correctamente o de si ocurrió algún error.
Referencia	RF4, (RF4.1, RF4.2, RF4.3, RF4.4, RF4.5)
Precondiciones	
Pos condiciones	Se envían los datos de la publicación.

Tabla 5 Gestionar artículos

CU-5	Gestionar artículos

Actor	Presidente del Comité Académico (CA).
Resumen	El caso de uso inicia cuando Presidente del CA revisa si existen artículos nuevos, donde revisa la descripción del artículo y lo acepta directamente si se corresponde con la comisión seleccionada. Al enviar el artículo o en caso contrario, selecciona la comisión que realmente le corresponde. Una vez aceptado el artículo, se le asignan revisores y se envía para la comisión. El caso de uso culmina cuando se les envía una notificación a los revisores, al presidente de la comisión, y al autor del artículo.
Referencia	RF5, (RF5.1, RF5.2, RF5.3, RF5.4, RF5.5)
Precondiciones	El Presidente del CA debe estar autenticado en el sistema. Deben existir artículos nuevos subidos.
Pos condiciones	Quedan los artículos listos para revisión.

Tabla 6 Evaluar artículos

CU-6	Evaluar artículos
Actor	Revisor.
Resumen	El caso de uso inicia cuando el revisor comprueba la lista de artículos por revisar asignados a él, seleccionando uno para proceder a su evaluación. El caso de uso culmina cuando del artículo es evaluado, enviándole una notificación a su autor.
Referencia	RF6, (RF6.1, RF6.2, RF6.3, RF6.4, RF6.5, RF6.6)
Precondiciones	El revisor debe estar autenticado en el sistema y tener artículos asignados pendientes de revisión.
Pos condiciones	Son evaluados los artículos.

Tabla 7 Gestionar Programa Académico

CU-7	Gestionar Programa Académico
Actor	Presidente del Comité Académico (CA).

Resumen	El caso de uso se inicia cuando el presidente del CA se dispone a adicionar las actividades que se deberán realizar en el evento, pudiendo luego modificarle los datos (como fecha, lugar) o eliminarla si es necesario.
Referencia	RF7, (RF7.1, RF7.2, RF7.3)
Precondiciones	El Presidente del CA debe estar autenticado en el sistema.
Pos condiciones	Queda conformado el programa académico.

Tabla 8 Gestionar Promoción

CU-8	Gestionar Promoción
Actor	Secretario Comité Organizador
Resumen	Este caso de uso se inicia cuando el Secretario Comité Organizador adiciona, edita o elimina noticias en el sistema. El caso de uso se puede iniciar también si decide enviar un plegable a la lista de distribución del evento.
Referencia	RF12, (RF12.1, RF12.2, RF12.3, RF12.4, RF12.5, RF12.6)
Precondiciones	El Secretario del CO debe estar autenticado en el sistema. Debe existir una lista de distribución para enviar el correo con los plegables.
Pos condiciones	Se garantiza la promoción del evento y sus actividades.

Tabla 9 Autentificar Usuario

CU-9	Autentificar Usuario
Actor	Usuario
Resumen	El caso de uso se inicia cuando el usuario solicita entrar al sistema, introduciendo nombre y contraseña, verificando sus credenciales en el sistema para permitir el acceso.
Referencia	RF8, (RF8.1, RF8.2)
Precondiciones	El usuario debe haber sido adicionado previamente al sistema.

Pos condiciones	El usuario es autenticado en el sistema

Tabla 10 Gestionar Usuarios

CU-10	Gestionar Usuarios
Actor	Administrador
Resumen	El caso de uso inicia cuando el Administrador decide Crear Nuevo Usuario, Modificar un Usuario, Eliminar Usuario o Mostrar Usuarios para controlar los usuarios que pueden acceder al sistema y el nivel de acceso que estos poseen.
Referencia	RF9, (RF9.1, RF9.2, RF9.3, RF9.4)
Precondiciones	El administrador debe estar autenticado en el sistema
Pos condiciones	Quedan creados los usuarios con sus credenciales correspondientes.

Tabla 11 Gestionar Grupos de Usuarios

CU-11	Gestionar Grupos de Usuarios
Actor	Administrador
Resumen	Este caso de uso se inicia cuando el administrador decide crear los diferentes Grupos a los cuales pertenecerán los usuarios, pudiendo eliminar un grupo o modificar los datos.
Referencia	RF10, (RF10.1, RF10.2, RF10.3, RF10.4)
Precondiciones	El administrador debe estar autenticado en el sistema
Pos condiciones	Se crean los distintos grupos que se les asignarán a los usuarios.

Tabla 12 Gestionar Permisos

CU-12	Gestionar Permisos
Actor	Administrador
Resumen	Este caso de uso se inicia cuando el administrador decide

	crear los diferentes permisos que se le asignarán a los usuarios o grupos de usuarios para determinar su accesibilidad en el sistema, pudiendo además eliminar o modificar los datos de los permisos.
Referencia	RF11, (RF11.1, RF11.2, RF11.3, RF11.4)
Precondiciones	El administrador debe estar autenticado en el sistema.
Pos condiciones	Se crean los distintos permisos que se les asignarán a los usuarios o grupos de usuarios.

2.7 Conclusiones

En este capítulo se ha realizado una breve descripción de los actores del sistema y sus responsabilidades. Se han refinado tambien los requerimientos funcionales, los cuales son de gran importancia para el desarrollo del sistema, ya que constituyen la funcionalidad del mismo. También se realizó un refinamiento de los requerimientos no funcionales. Además se presentó el diagrama de casos de uso por paquetes y se detallaron los casos de usos arquitectónicamente significativos para la implementación del sistema propuesto.

3 CAPÍTULO3: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción:

En el presente capítulo se realiza el diseño de la propuesta de solución, seleccionando una arquitectura y modelándose los artefactos que contribuyen al desarrollo de la aplicación Web, con el uso de Frameworks Manipuladores de Contenidos flexibles. Se elabora el modelo de datos adecuado. Se analiza cómo va a estar distribuido el sistema.

3.2 Descripción de la arquitectura:

Según RUP la arquitectura de software "representa la estructura o las estructuras del sistema, que consta de componentes de software, las propiedades visibles externamente y las relaciones entre ellas. Es la organización o la estructura de los componentes importantes del sistema que interactúan mediante interfaces, con componentes compuestos de interfaces y componentes cada vez más pequeños". (IBM Corporation)

Así mismo se define que la arquitectura de software comprende:

- Las decisiones significativas acerca de la organización de un sistema de software
- La selección de los elementos estructurales y sus interfaces.
- El estilo de arquitectura que guía la organización, los elementos y sus interfaces, sus colaboraciones y su composición. (IBM Corporation, 2009)

3.2.1 Arquitectura Modelo Vista Controlador (MVC)

El Sistema para la organización de un taller virtual en la UCI ha sido desarrollado sobre el patrón clásico de diseño web conocido como arquitectura Modelo Vista Controlador (MVC), el cual 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.

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo; el código de la presentación en la vista; y la lógica de la aplicación en el controlador. Al separar la lógica de negocio (el modelo) y la presentación (la vista), se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (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. (Potencier, y otros, 2009) La figura 6 ilustra el funcionamiento del patrón MVC.

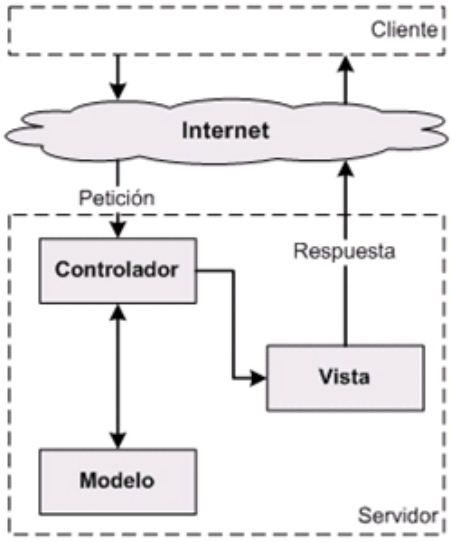


Figura 1 El patrón MVC.

Las capas del modelo, la vista y el controlador, se subdividieron en más capas utilizando además otros patrones de diseño, lo cual permitió que la programación se simplificara.

La capa del modelo se dividió en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos, no utilizan sentencias ni consultas que dependen de una base de

datos específica, sino que utilizan otras funciones para realizar las consultas; logrando con ésto independencia del sistema gestor de bases de datos y que las funciones creadas en la capa de abstracción de la base de datos, se puedan reutilizar en otras funciones del modelo que necesiten acceder a la base de datos.

La vista se separa en un layout y en una plantilla. El layout es global en toda la aplicación o al menos en un grupo de páginas. La plantilla se encarga de visualizar las variables definidas en el controlador.

El controlador está compuesto por un controlador frontal, el cual es un patrón de control de entrada de datos que asume las dos responsabilidades que tiene todo controlador de entrada: manipular la petición http y decidir qué hacer con ella. Con este patrón se logra centralizar toda la manipulación http, ya que es único para cada aplicación y para las acciones, que incluyen el código específico del controlador de cada página. Un controlador frontal ofrece un punto de entrada único para toda la aplicación, evitando así la necesidad de reconfigurar el servidor web cada vez que cambie la estructura de la acción del sitio y posibilitando tener fácil control sobre el flujo de información, tanto para la seguridad de la aplicación, como para el funcionamiento de la misma. (Fowler, 2003).

En la aplicación se implementó el patrón de diseño web MVC a través del framework Symfony, el cual toma lo mejor del patrón MVC y lo implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. En la figura 7 se muestra la implementación que hace Symfony de este patrón de diseño web.

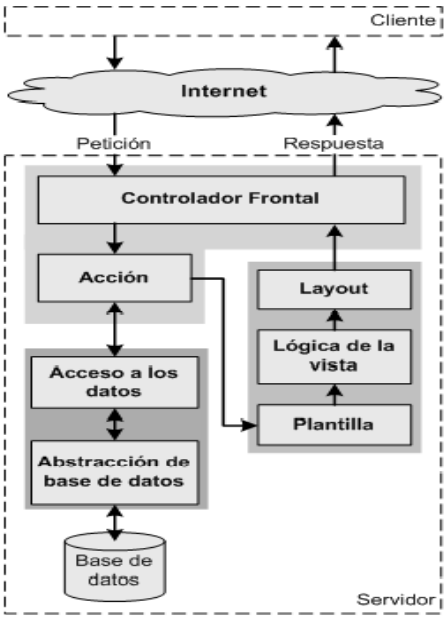


Figura 2 Flujo de trabajo en Symfony.

3.3 Patrones de diseño:

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, no son más que una solución estándar para un problema común de programación. Son una técnica para flexibilizar el código, haciéndolo satisfacer ciertos criterios. (PRESSMAN, 2005)

3.3.1 Patrón Decorator:

El comportamiento del layout y la plantilla en la vista, representan una implementación del patrón de diseño de aplicaciones web llamado “Decorator”. El contenido se muestra con una plantilla que después se decora con un layout global. Este procedimiento es la solución para el problema de no repetir el código de las páginas. La figura 8 muestra el funcionamiento del patrón al decorar una plantilla.

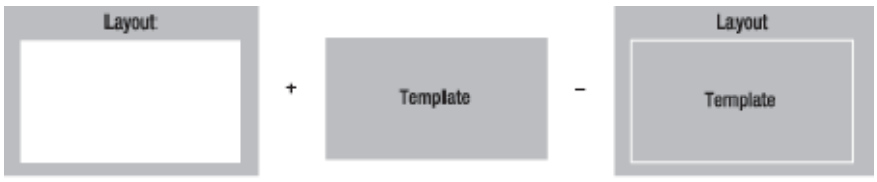


Figura 3 Patrón Decorator.

3.4 Principios de diseño de Interfaz de Usuario

Para el diseño de la interfaz de usuario de este sistema se han seguido los siguientes principios:

- Teniendo presente el principio de Mínimo Acceso, permitir al usuario acceder solamente a las opciones a las que, dado su rol, puede ejecutar.
- No mostrar funcionalidades que no estén completamente implementadas o a las que el usuario no pueda acceder ya sea por políticas de seguridad u otro motivo.
- Lograr que la navegación por la aplicación sea simple, amigable e intuitiva.
- Permitir que su utilización desde el primer momento, no esté condicionada por avanzados conocimientos de informática.
- Requerir de los usuarios un mínimo esfuerzo para alcanzar sus objetivos.

3.4.1 Estándares de la interfaz de la aplicación.

Con el objetivo de lograr un diseño consistente de la interfaz de la aplicación, se organizó el contenido en todas las páginas mediante el esquema Cabecera-Navegador-Contenido-Pie de página, poniendo así en práctica el estándar de diseño web que plantea ubicar el contenido por su importancia de arriba hacia abajo y de izquierda a derecha. La cabecera contiene el logotipo del Grupo de Investigación de Ingeniería de Software en la UCI en la esquina superior izquierda, el nombre de la aplicación en el medio y enlaces a algunas funcionalidades del sistema. En el navegador se incluyen los enlaces a las distintas secciones del sistema así como a otros sitios y áreas vinculadas con el sistema. En el área del contenido se muestran los formularios de entrada, las salidas, las respuestas del sistema ante acciones realizadas por los usuarios, los listados y cualquier otro tipo de información que se le muestre al usuario. En el Pie de página se muestran enlaces a diferentes secciones del sistema así como información adicional sobre el desarrollo, autoría y soporte técnico de la aplicación.

Para el diseño se utilizan las capas, las tablas y las plantillas, se utiliza también hojas de estilos para establecer la configuración del diseño de todas las páginas. Estas hojas de estilos establecen el tipo y tamaño de las fuentes de los distintos elementos de cada página. Se utiliza en general la fuente Arial, de tamaño entre 10 y 16 píxeles, según la importancia de la información mostrada. Las hojas de estilos también establecen el color de fondo y de todas las secciones de la aplicación, el formato de las tablas, entre otros. Se utilizó una combinación de colores con tonos suaves, usando tonos de gris y blanco para las secciones de la aplicación y azul para el fondo y algunas letras logrando así una armonía tal que ninguna de las secciones del sistema logra que la atención del usuario se centre solo en ella sino en toda la aplicación por igual.

3.4.2 Tratamiento de excepciones.

El tratamiento de errores posibilita el buen funcionamiento de una aplicación dándole una mejor apariencia ante los clientes. Para prevenir errores por parte del usuario, sólo se le brindan las opciones necesarias a la hora de efectuar cualquier operación, por ejemplo: se deshabilitan los accesos si el usuario no se ha autenticado en el sistema. Una vez determinado su rol, se le da acceso a las páginas correspondientes, y en caso de querer acceder a alguna sección de la aplicación a la que no tiene permiso, se le muestra un mensaje indicándole que no tiene acceso a esa sección.

Mediante el uso de los widgets de formularios y la validación que de estos se puede hacer mediante el uso de funciones brindadas por el framework, se

garantiza que los datos suministrados por los usuarios tengan la estructura establecida, se almacenen íntegros y no existan inconsistencias. Para esto, se verifican los campos obligatorios, y se revisa el tipo de datos, mostrándose en caso de algún error, mensajes que informan al usuario dónde está el error y de qué tipo es.

3.4.3 Estándares de codificación

Resulta muy ventajoso utilizar un estándar para escribir código, pues se reducen considerablemente los errores, y los códigos resultan más compresibles y fáciles de leer. Con vistas a garantizar la homogeneidad de dicho código, se establece el estilo descrito a continuación:

Comentarios: Los comentarios utilizados en las secciones de código php, jscript o css se definen comenzando con los caracteres ***/**** y terminando con ****/*** para los comentarios de varias líneas, y comenzando con los caracteres ***//*** para los de una sola línea. En las secciones de código HTML los comentarios utilizados se definen comenzando con los caracteres <!— y terminando con -->.

Todas las clases del framework utilizado utilizan el prefijo sf haciendo referencia al nombre del framework “Symfony”, lo cual también hacen todas las variables principales de Symfony en las plantillas. De esta forma, se evitan las *colisiones* entre los nombres de clases y variables de Symfony y los nombres de las clases y variables propias de los programadores, además de que las clases del framework son más fáciles de reconocer. La norma seguida por el código es el estándar *"UpperCamelCase"* para el nombre de las clases y el estándar *"CamelCase" para las* variables. Solamente existen dos excepciones: las clases del núcleo de Symfony empiezan por sf (por tanto en minúsculas) y las variables utilizadas en las plantillas que utilizan la sintaxis de separar las palabras con guiones bajos.

3.5 Modelo de diseño:

El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que abarca todas las clases de

diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos. (IBM Corporation).

3.5.1 Diagrama de clases del diseño:

En los diagramas de clases para Aplicaciones Web, son más importantes la modelación de la lógica y estado del negocio que los detalles de presentación. Para esto se utilizarán los estereotipos Web, es por eso que difieren un poco del resto de las aplicaciones que se acostumbra a construir. Para obtener un nivel correcto de abstracción y detalle que permita obtener un resultado final, es mejor modelar los artefactos del sistema, es decir, modelar las páginas, los enlaces entre éstas, así como el contenido dinámico de las mismas, una vez que estén en el navegador del cliente. Estos son los artefactos que se necesitan modelar para la implementación del producto final.

De acuerdo a la forma en que se ha organizado el contenido del trabajo, se presentarán solamente aquellos Casos de Uso más importantes para el Módulo Organización del Evento, Evaluación de las publicaciones y para el Administrativo. Los demás diagramas de estos módulos y los del módulo Promoción del Evento se pueden encontrar en los Anexos.

3.5.1.1 Módulo Organización del Evento:

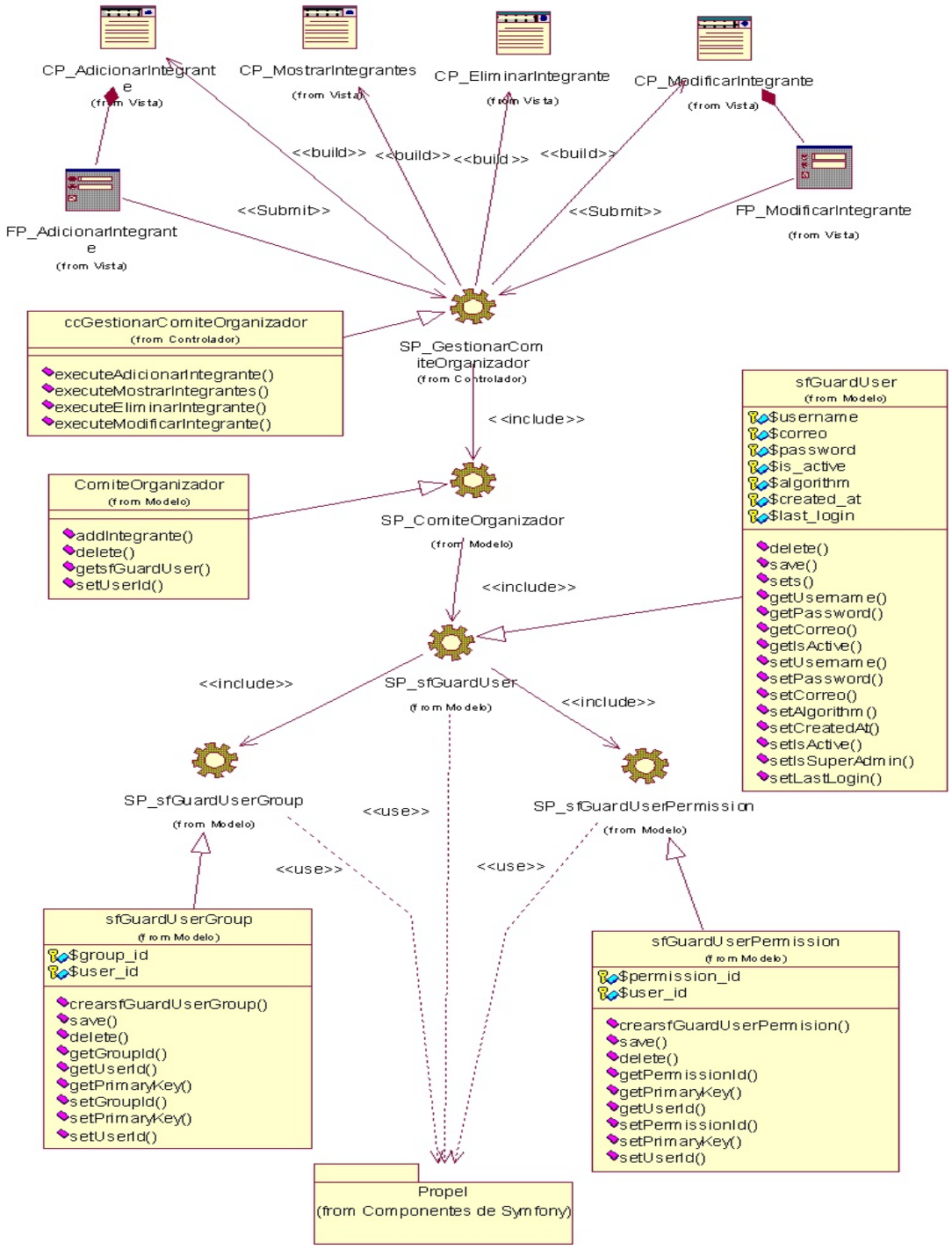


Figura 4 Diagrama de Clases: Gestionar Comité Organizador

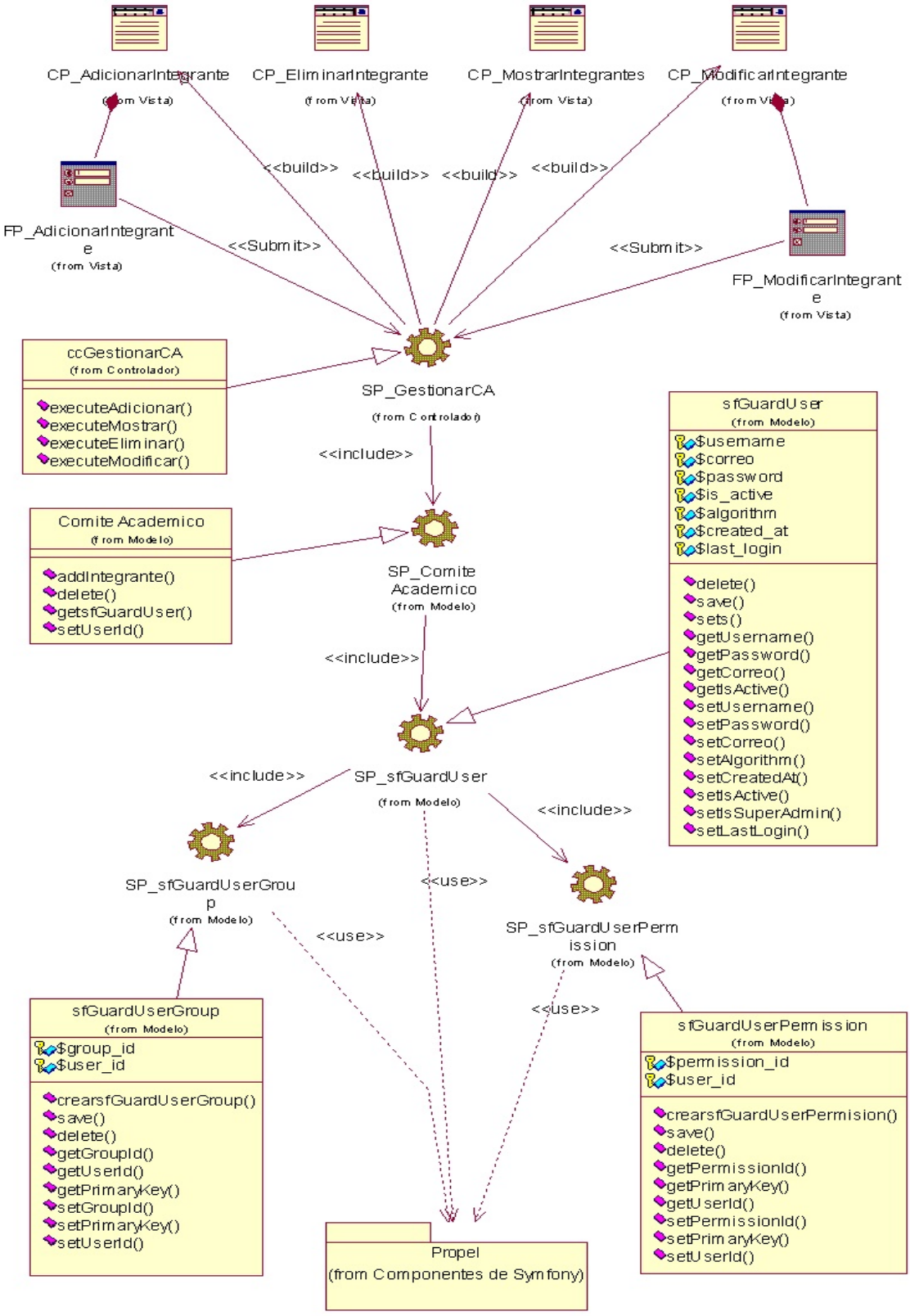


Figura 5 Diagrama de Clases: Gestionar Comité Académico.

3.5.1.2 Módulo Evaluación de las publicaciones.

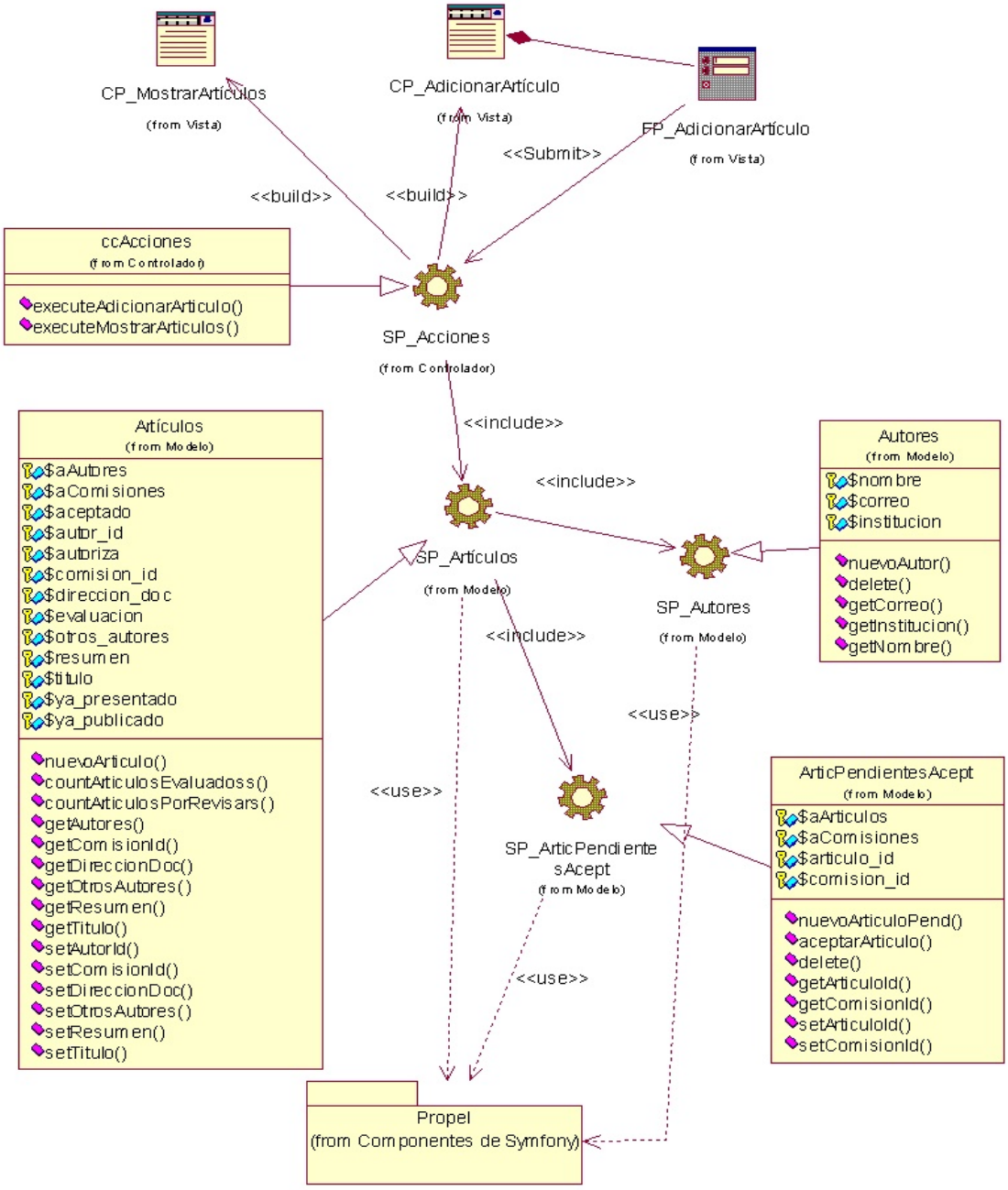


Figura 6 Diagrama de Clases: Publicar Artículos

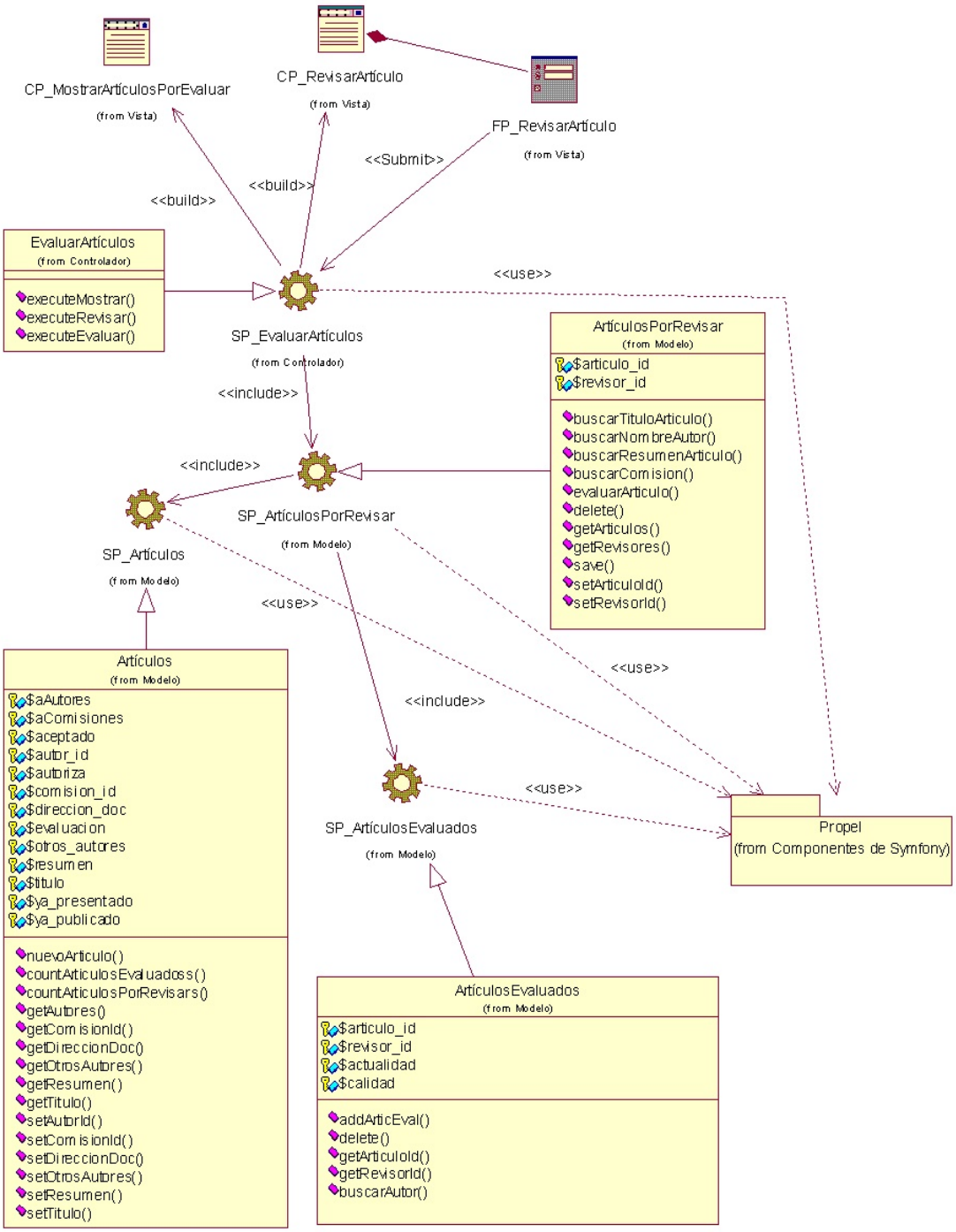


Figura 7 Diagrama de Clases: Evaluar Artículos

3.5.1.3 Modulo administración

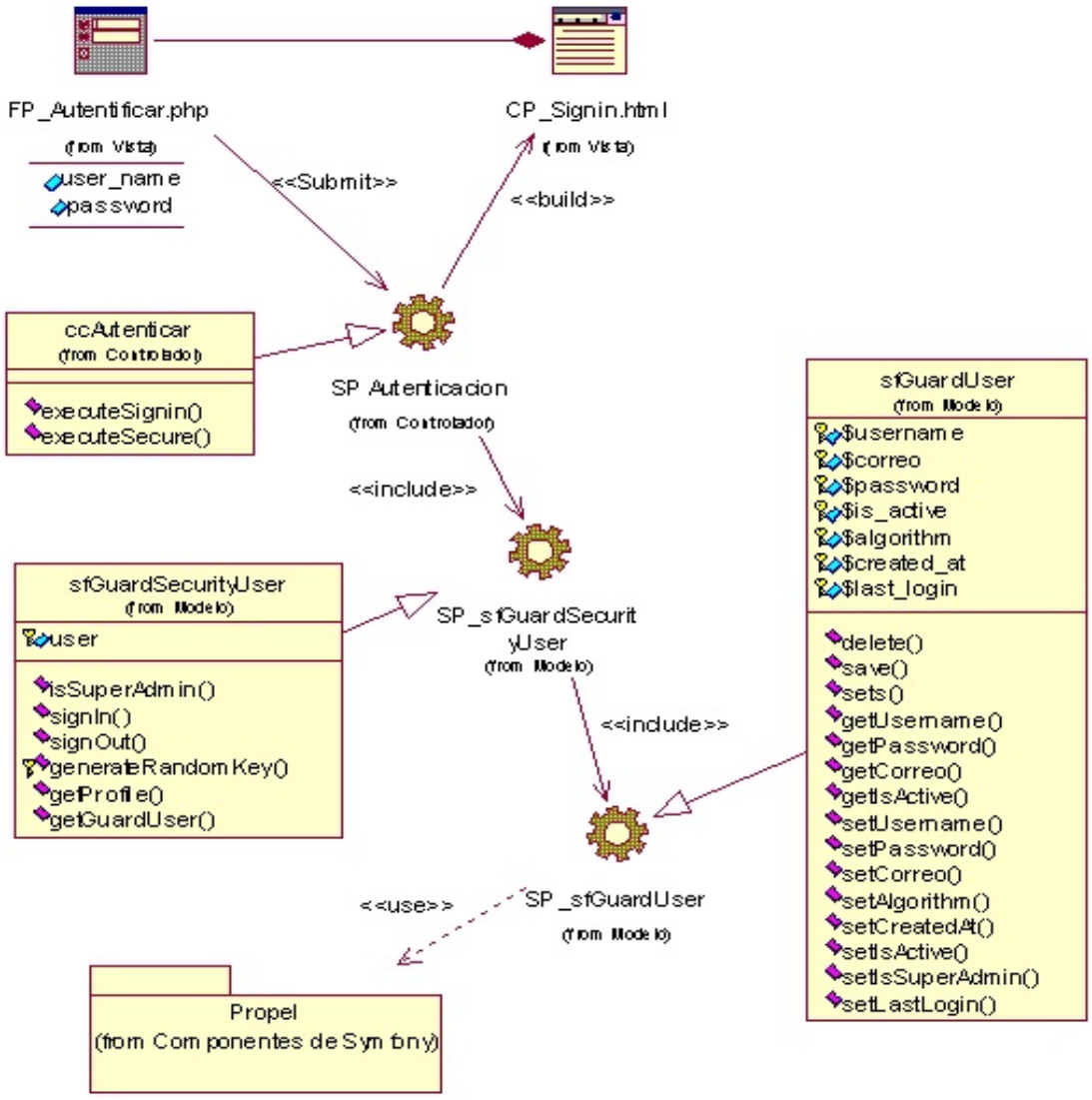


Figura 8 Diagrama de Clases: Autenticar Usuario.

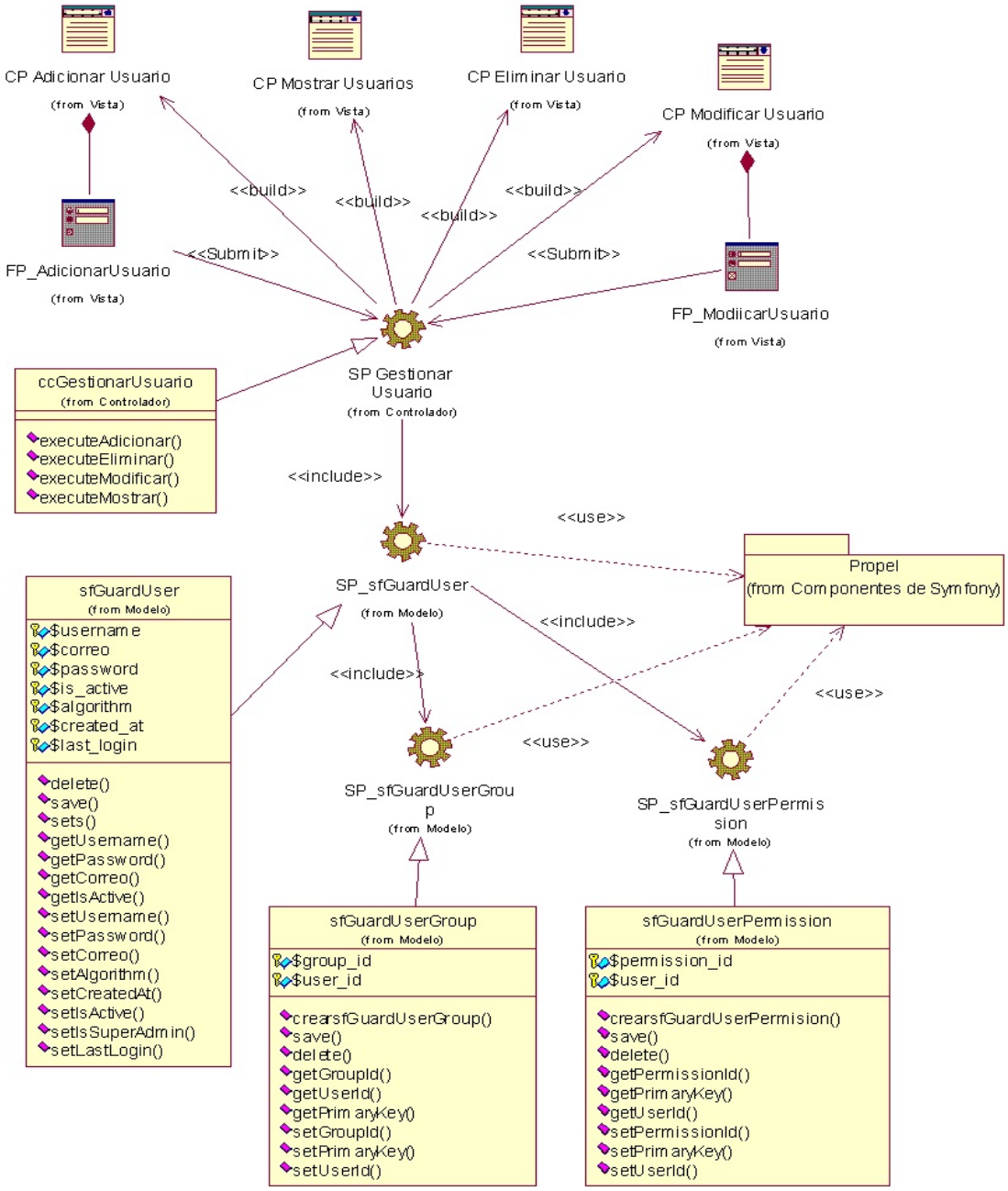


Figura 9 Diagrama de Clases: Gestionar Usuario.

3.6 Diseño de la base de datos:

Para diseñar la base de datos del sistema se utilizó el diagrama del modelo de datos, el cual se confeccionó basado en la modelación de las clases persistentes que son utilizadas en el modelo del diseño, mediante el componente de Symfony que se encarga de gestionar el modelo el cual es una capa de tipo ORM (object/relational mapping) realizada mediante el proyecto Propel.

El modelo de datos describe, de una forma abstracta, cómo se representan los datos en un sistema de gestión de base de datos. Es una herramienta para especificar los tipos de datos y la organización de los mismos, que son permisibles en una base de datos específica. Este modelo es una guía para el diseño de la base de datos y es el elemento clave en el diseño de la arquitectura de un manejador de bases de datos. (Ver Anexo 6)

3.7 Diagrama de despliegue:

El diagrama de despliegue permite apreciar de forma visual cómo se encuentran relacionados físicamente los componentes de la aplicación. En este caso, el usuario accede al sistema desde un navegador Web por medio del protocolo HTTP. La aplicación se encuentra hospedada en un servidor Web, el cual se conecta al servidor de base de datos (PostgreSQL) mediante el protocolo TCP/IP. (Ver figura 9)

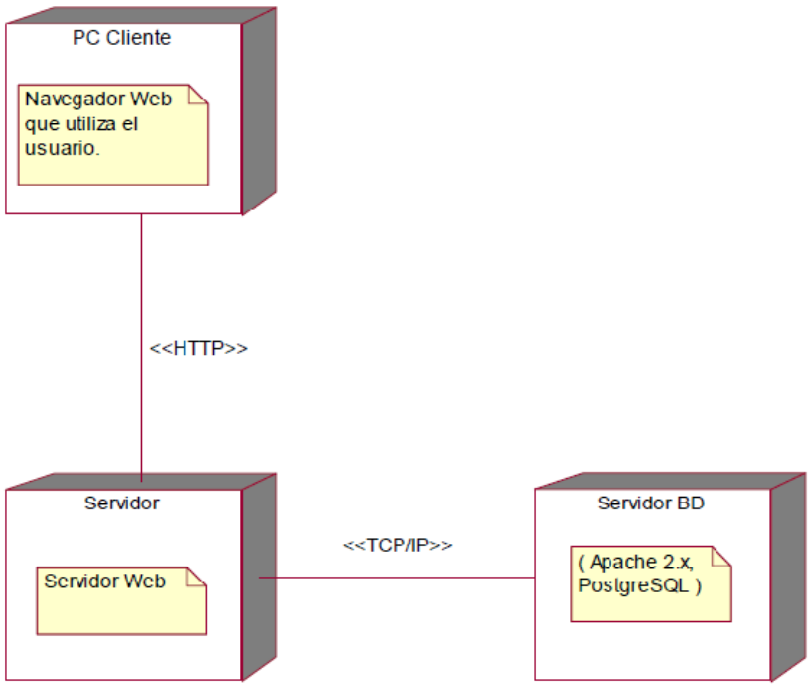


Figura 10: Diagrama de despliegue.

3.8 Conclusiones:

En el presente capítulo se obtuvo la solución al sistema propuesto, describiendo la arquitectura y patrones utilizados para el desarrollo de la aplicación. Se mostraron los resultados de la etapa de diseño, para ello se

mostraron los diagramas de clases por módulos, lo cual brindó una mejor comprensión del sistema. Se modelaron los componentes como clases, utilizando las extensiones Web del UML. Además, se realizó el modelo de datos, que permite una mejor vista para realización de la base de datos. Se realizó el diseño del diagrama de despliegue, donde se describen los nodos de procesamiento en tiempo real donde se ejecutará la aplicación y los vínculos entre ellos. Todos estos elementos obtenidos son claves para la correcta implementación del sistema propuesto.

CONCLUSIONES GENERALES

Al culminar el trabajo, se llegó a las siguientes conclusiones:

- Se logró la implementación de una aplicación que brinda una serie de funcionalidades importantes para la organización y gestión de los procesos de un taller virtual en la UCI.

- Se logró establecer pautas con respecto a la seguridad de la información almacenada de los miembros del taller virtual.

- Se logró un sistema provisto de un ambiente cómodo, fácil de entender y que cumple los estándares de diseño.

- Para el desarrollo de la propuesta, se realizó el análisis de las tecnologías más usadas en la actualidad, concluyéndose en la utilización de PHP 5 como lenguaje de programación. El gestor de bases de datos utilizado fue PostgreSQL, junto a otras tecnologías como las hojas de estilo CSS. Todas estas tecnologías se utilizaron enmarcadas en la plataforma de desarrollo web Symfony.

- La aplicación se desarrolló siguiendo la metodología RUP y se utilizaron representaciones UML para la modelación de todas las fases del proyecto.

- Se identificaron los procesos principales del negocio. Se definieron los requerimientos del sistema, tanto funcionales como no funcionales, estructurándose además el modelo de casos de uso del sistema.

- Se realizó el diseño del sistema, a través de diagramas de clases del diseño, de secuencia, de despliegue.

- Por todo lo anterior, se concluye que el objetivo propuesto ha sido cumplido satisfactoriamente, incluyendo una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

RECOMENDACIONES:

Al concluir este trabajo se recomienda lo siguiente para versiones futuras:

- Profundizar en el estudio de los procesos y tareas del Taller Virtual de Ingeniería de Software, creándose un grupo de desarrollo que permita seguir perfeccionando la aplicación.

- Continuar el desarrollo de este sistema, adicionándole nuevas funcionalidades y dándole seguimiento a las actividades del Taller, adecuándolo a los cambios que puedan surgir en etapas posteriores, tanto en un proceso de pruebas como de puesta en práctica de la aplicación.

- Realizar una etapa de pruebas para refinar los procesos automatizados y detectar y corregir cualquier error en diseño o implementación así como implementar algún requerimiento que no haya sido tomado en cuanta inicialmente.
- Implantar el sistema en la red de la Universidad de las Ciencias Informáticas para prestar los servicios implementados dándole solución inmediata a los problemas detectados en la etapa de estudio preliminar.

REFERENCIAS BIBLIOGRÁFICAS:

EMS Database Management Solutions, Inc. EMS SQL Manager. [En línea] http://www.sqlmanager.net/.

ENRÍQUEZ, BARRIENTOS. 2005. El proceso Unificado de Modelado (RUP). [En línea] 2005. http://www.monografias.com/trabajos16/lenguaje-modelado-unificado/lenguaje-modelado-unificado.shtml#PROCESO.

Fowler, Martin. 2003. *Patterns of Enterprise Application Architecture.* s.l. : Addison-Wesley, ISBN: 0-32112-742-0, 2003.

IBM Corporation. 2009. *Rational Method Composer.* [En línea] 2009. http://www-306.ibm.com/software/awdtools/rmc/.

—. RUP en developerWorks. [En línea] http://www-136.ibm.com/developerworks/rational/products/rup.

MPSOFTWARE. [En línea] http://www.mpsoftware.dk/.

Object Management Group, Inc. (OMG). 2007. *OMG UML Superstructure Specification, v2.1.2.* 2007.

OMG. 2009. UML® Resource Page. [En línea] 2009. http://www.uml.org/.

Pérez, Javier Eguíluz. 2007. *Introducción a AJAX.* 2007.

PHP Documentation Group. Offical Php Web Site. [En línea] http://www.php.net/docs.php.

Potencier, Fabien y Zaninotto, François. 2009. *Symfony la guía definitiva.* 2009. pág. 425.

PRESSMAN, R. S. 2005. Ingeniería del Software, Un enfoque práctico. [ed.] Mc Graw Hil. *Ingeniería del Software, Un enfoque práctico.* 5a. ed. La Habana. 2005, pág. 601.

The Institute of Electrical and Electronics Engineers, Inc (IEEE Computer Society). 2004. *(SWEBOK) Guide to the Software Engineering Body of Knowledge.* [ed.] Alain Abran y James W. Moore. 2004.

The PostgreSQL Global Development Group. Official PostgreSQL Web Site. [En línea] http://www.postgresql.org/.

UCI. investigaciones. [En línea] www.investigaciones.uci.cu.

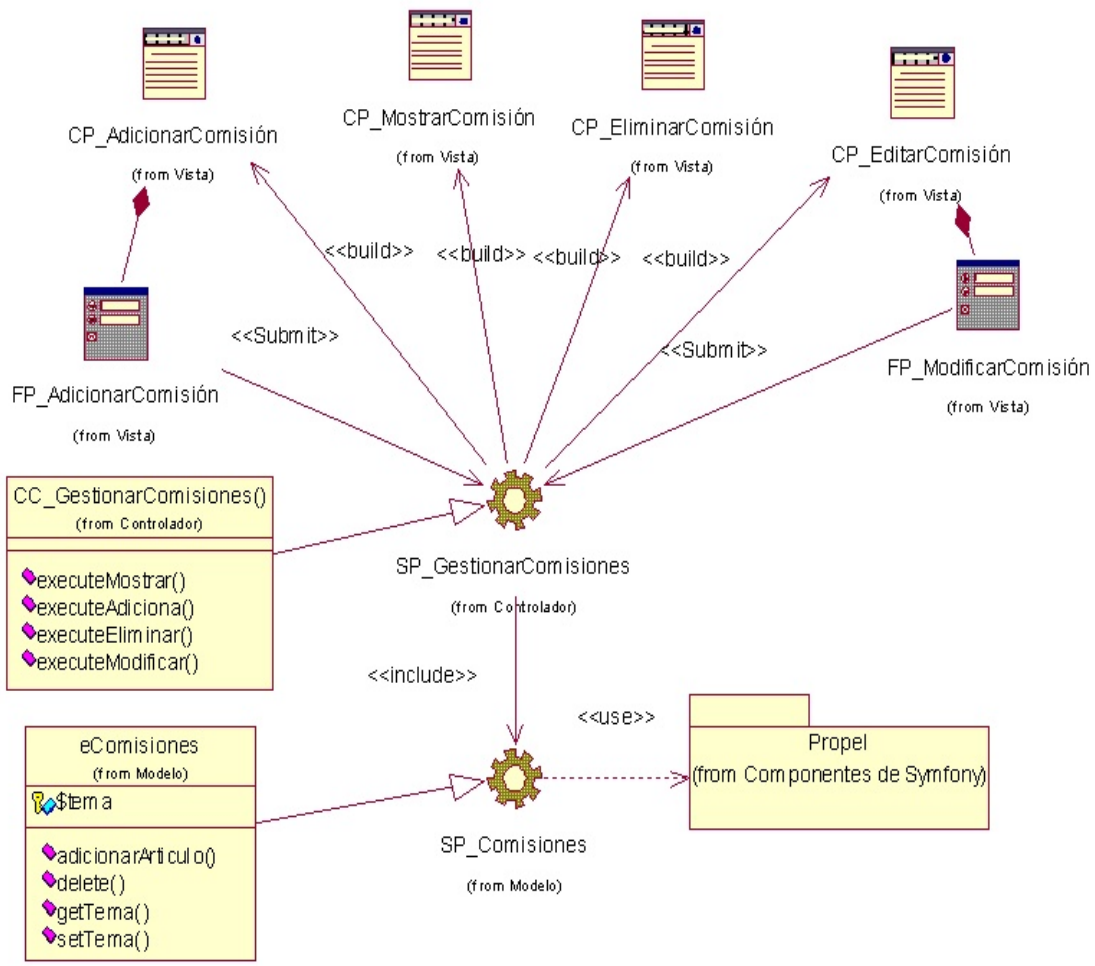
—. UCIENCIA. [En línea] www.uciencia.uci.cu.

W3C Candidate Recommendation. W3C. [En línea] http://www.w3.org/TR/2007/CR-CSS21-20070719.

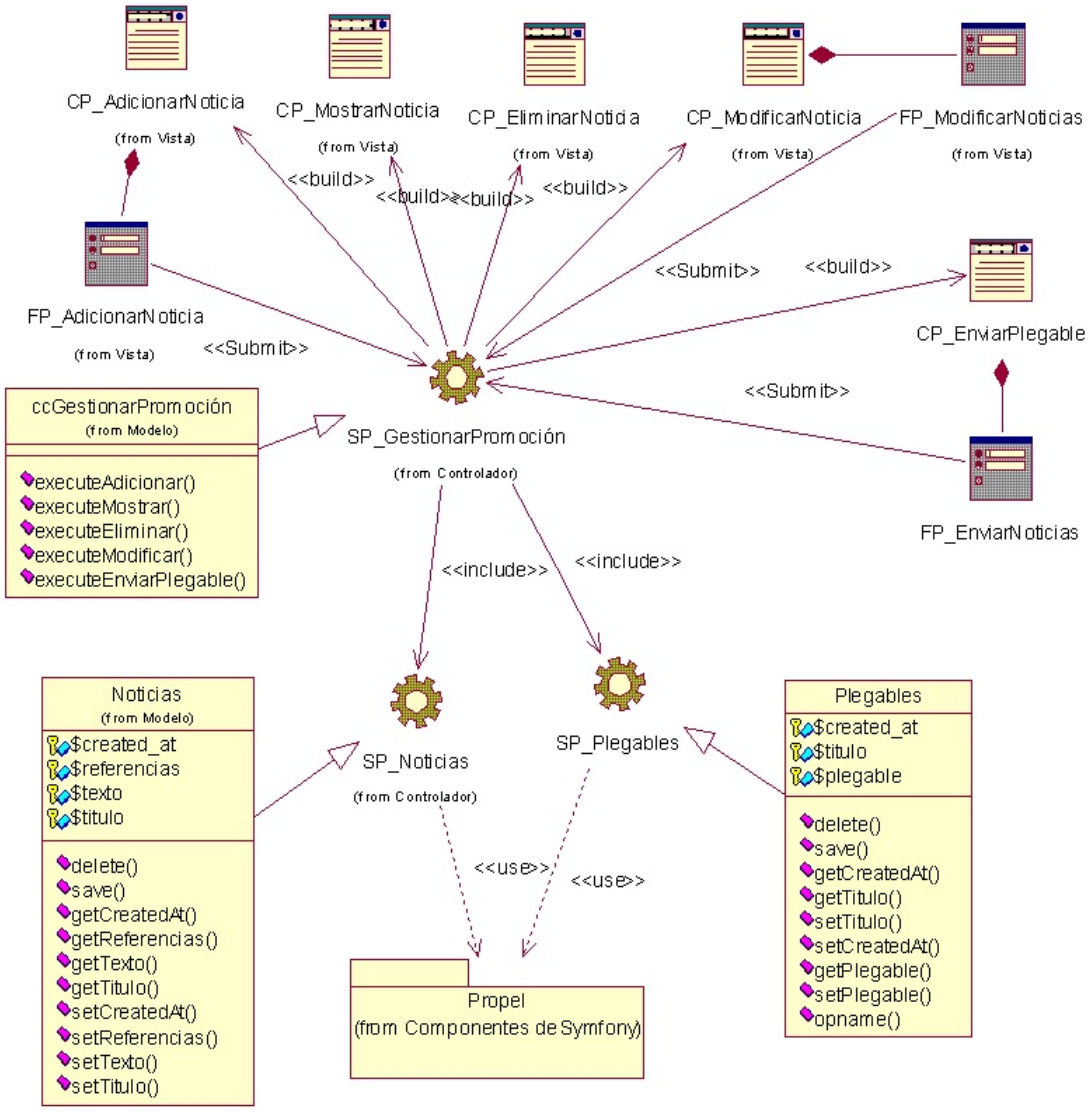
2009. YAML. [aut. libro] Fabien Potencier y François Zaninotto. *Symfony Guia Definitiva.* 2009, págs. 15-17.

ANEXOS

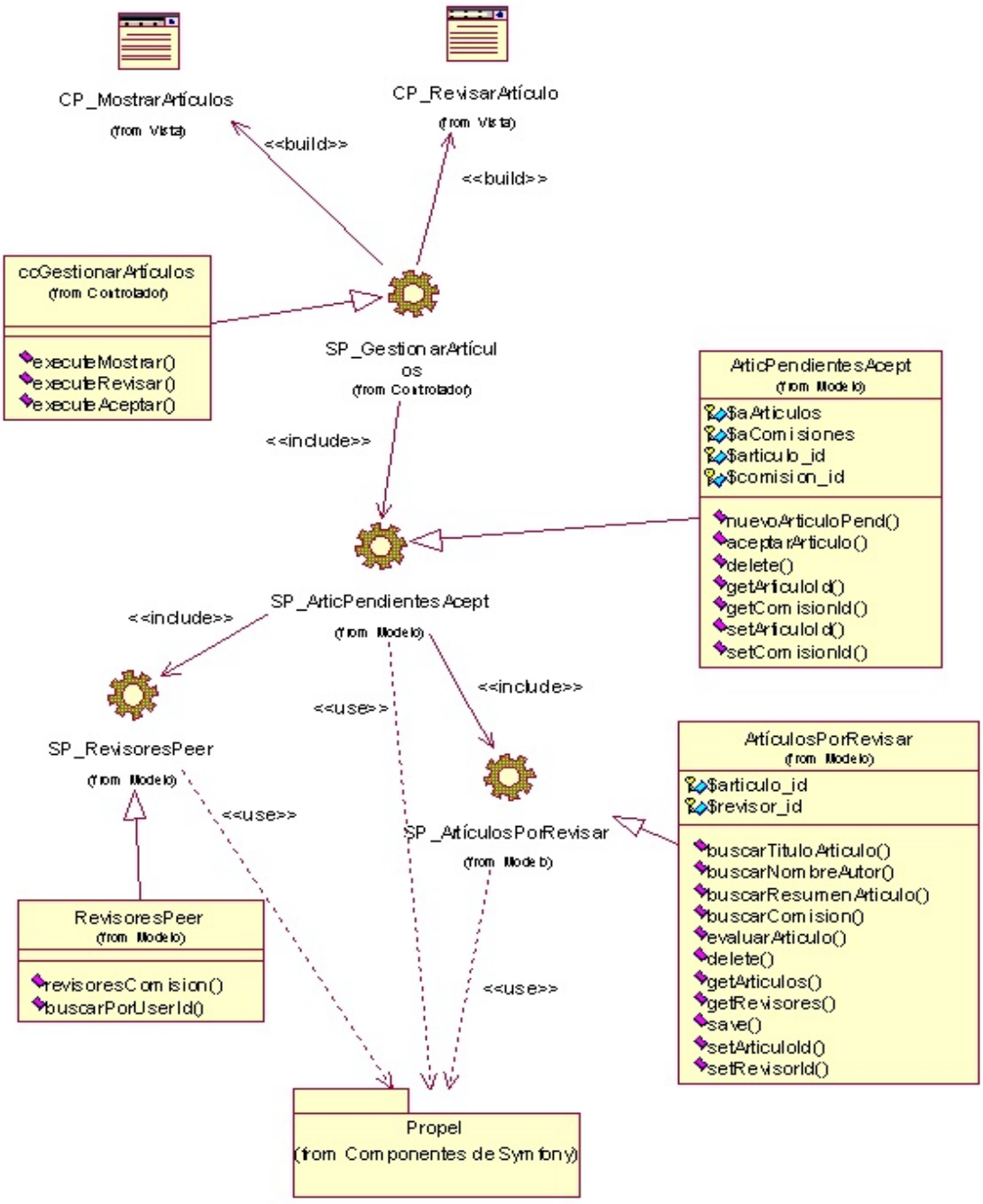
Anexo 1: Diagrama de Clases: Gestionar Comisiones.



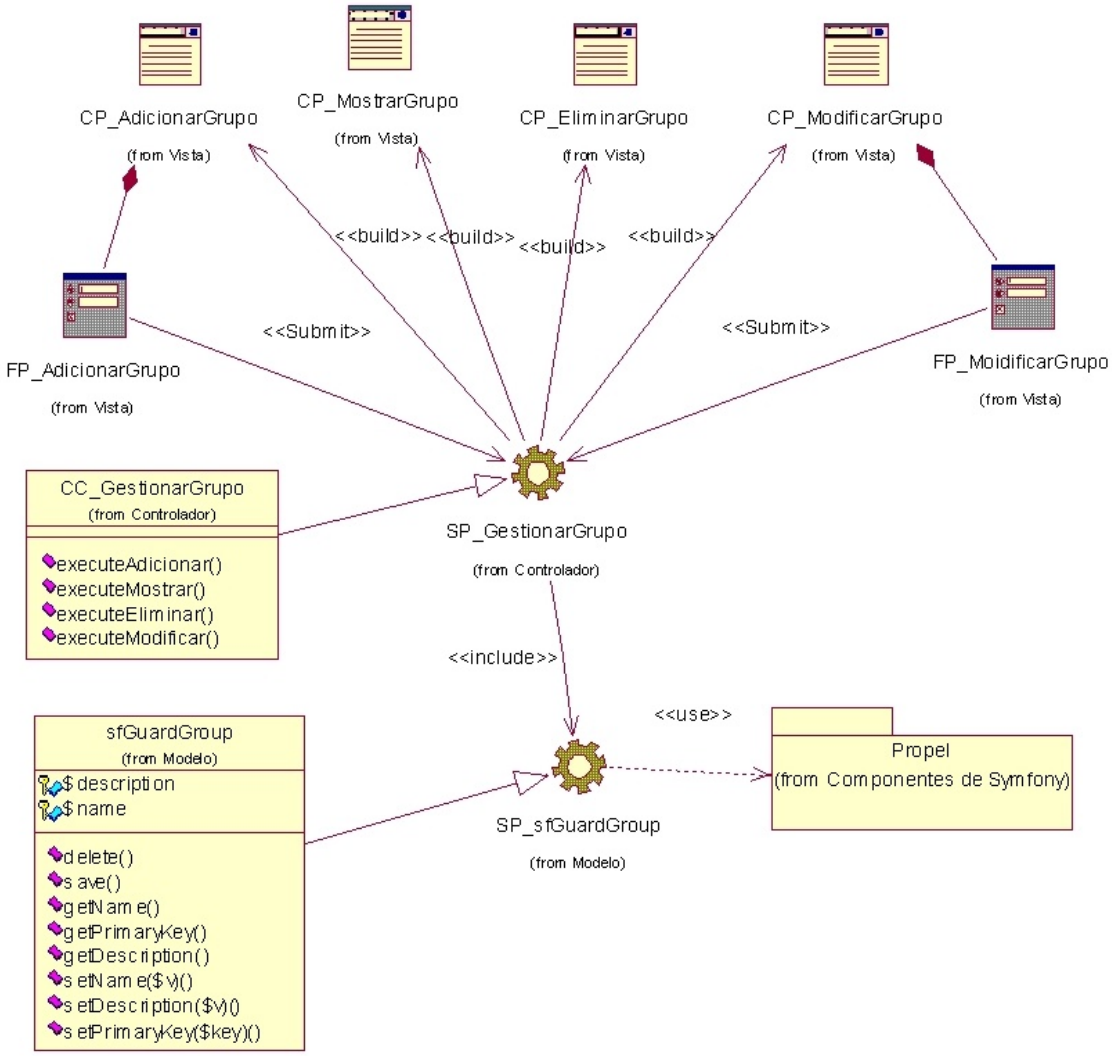
Anexo 2: Diagrama de Clases: Gestionar Promoción



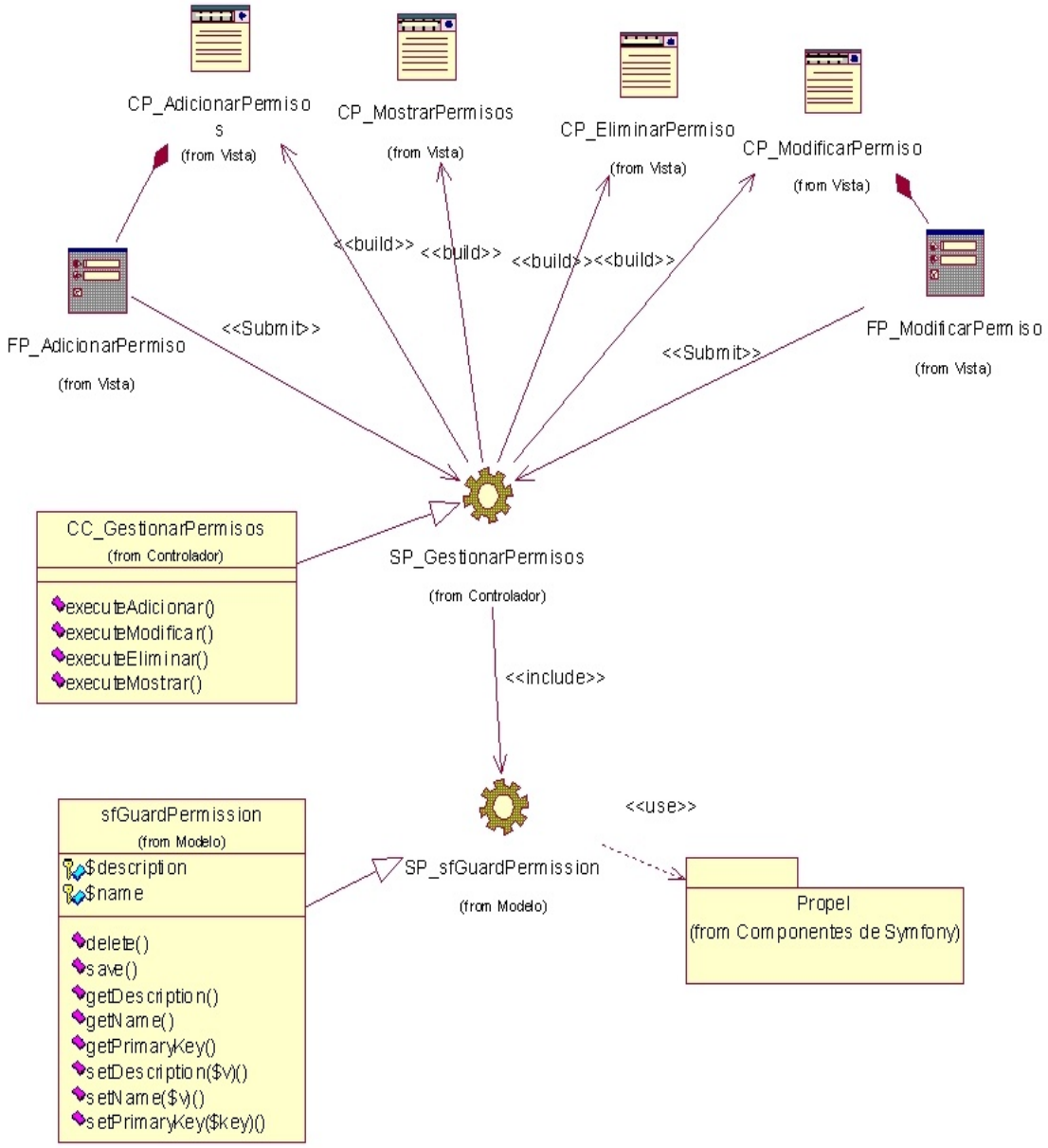
Anexo 3: Diagrama de Clases: Gestionar Artículos



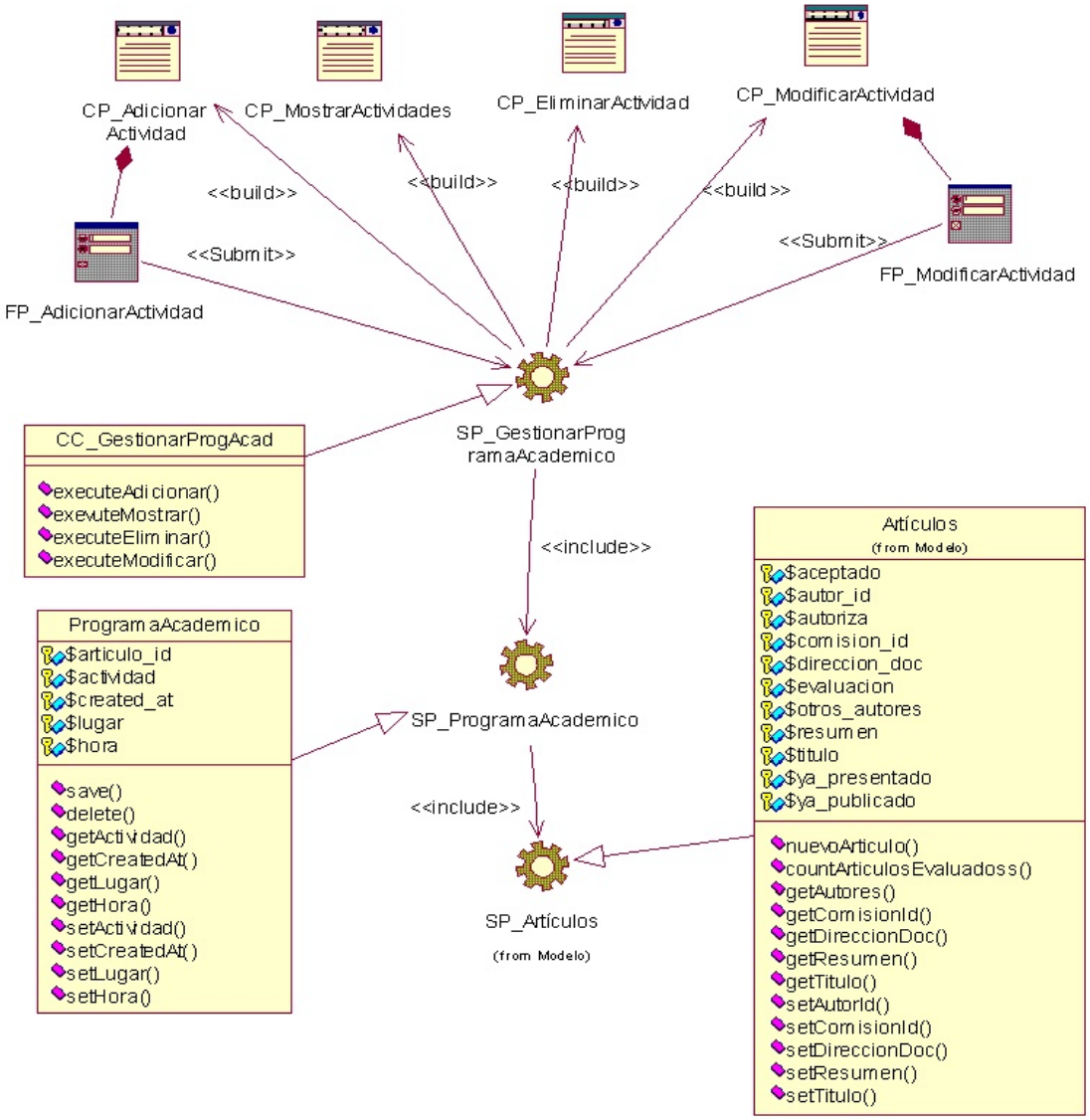
Anexo 4: Diagrama de Clases: Gestionar Grupos de Usuarios



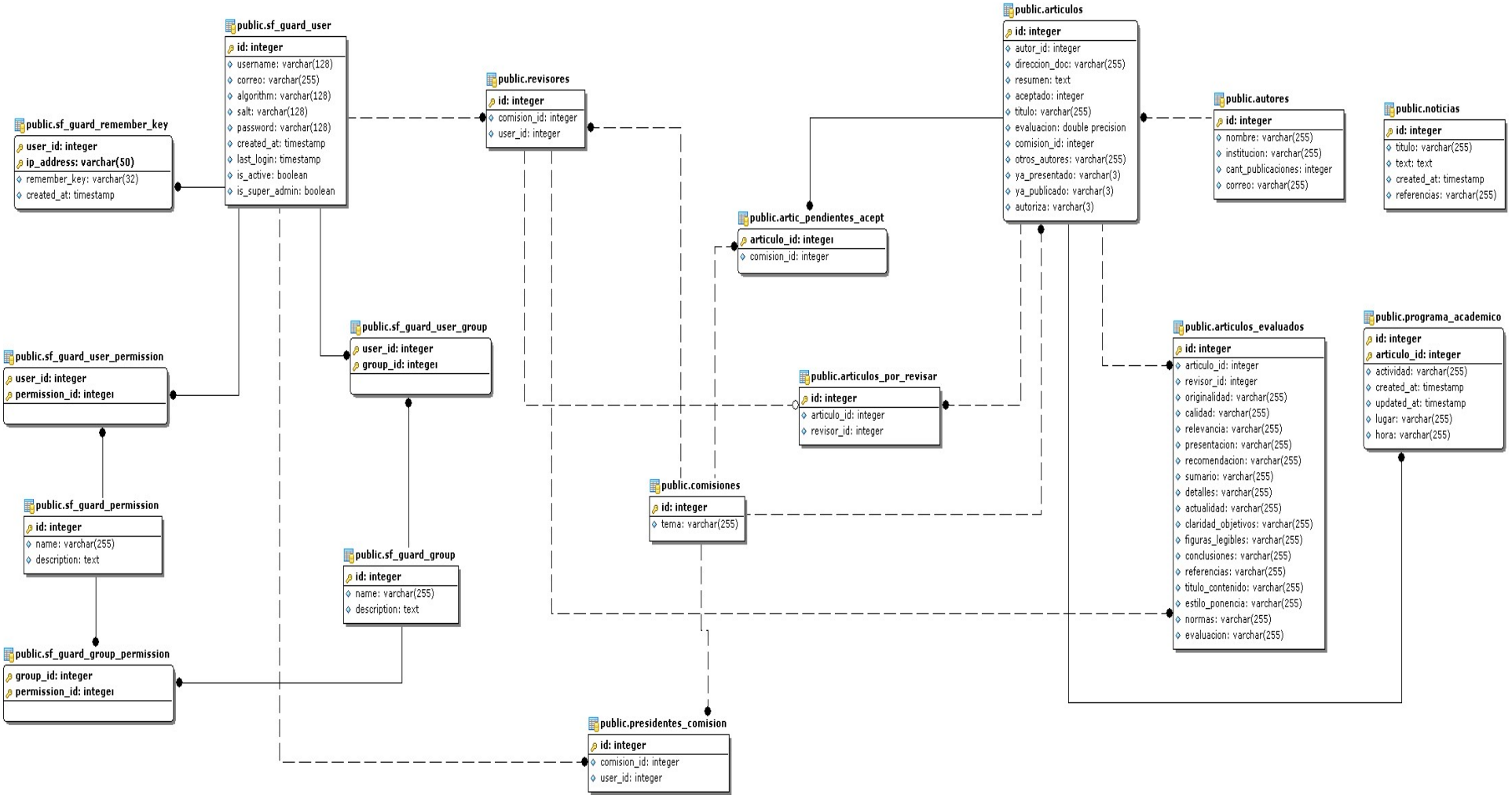
Anexo 5: Diagrama de clases: Gestionar Permisos.



Anexo 6: Diagrama de Clases: Gestionar Comité Académico



Anexo 7: Diagrama del Modelo de Datos



Glosario de Términos:

OOP (Programación Orientada a Objetos): “La idea de la programación orientada a objetos es que una aplicación se puede considerar como una colección de unidades individuales, llamadas objetos, que interactúan entre sí. Los programas tradicionales pueden considerarse como una colección de funciones o como una lista de instrucciones de programación.”

ORM (Mapeo de Objetos a Bases de datos): Método poderoso para diseñar y consultar modelos de bases de datos a nivel conceptual, dónde la aplicación está descrita en términos fácilmente comprendidos por usuarios poco especializados. (Halpin, 2009)

Propel: Servicio de objeto persistente y de consulta lo que significa que provee un sistema para almacenar objetos en una base de datos y un sistema para búsqueda y restauración de objetos desde una base de datos.

DRY (Don't Repeat Yourself): Principio de refactorización de código llamado “No te repitas”, por la traducción al español. Aplicando este principio los bloques de código repetidos se refactorizan en un único lugar.

Refactorización de código: Es cuando debe reescribir parte del código existente, cuando se modifican los requisitos o se añade una nueva funcionalidad.

TDD (test-driven development): Enfoque evolutivo del desarrollo que involucra otras dos prácticas: *Escribir las pruebas primero (Test First Development)* y *Refactorización (Refactoring)*.

PEAR: Es un "framework y sistema de distribución para componentes PHP reutilizables". Permite descargar, instalar, actualizar y desinstalar scripts de PHP.

OMG (Object Management Group): Asociación sin fines de lucro formada por grandes corporaciones, como por ejemplo: IBM, Apple Computer, Sun Microsystems Inc y Hewlett-Packard Esta asociación se encarga de la definición y mantenimiento de estándares para aplicaciones de la industria de la computación.

Multiplataforma: Poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Negocio: Cualquier ambiente o entorno en cual está enmarcado el problema.

Patrones: Describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que se puede reutilizar continuamente.

Sistema de Gestión: Conjunto de políticas y normas relacionadas entre sí que se establecen para el acceso y tratamiento de los recursos de información. Incluye los registros administrativos y los archivos, el soporte tecnológico de los recursos y el público a que se destina.

Software libre: Aplicaciones informáticas que pueden ser libremente copiadas, distribuidas, estudiadas y modificadas por el usuario, según los parámetros establecidos por su creador.

Análisis: Investigación de un dominio, la cual da origen a modelos que describen sus características estáticas y dinámicas. Se centra en cuestiones de “qué” más que de “cómo”.

Caso de Uso: Descripción narrativa textual de la secuencia de eventos y acciones que ocurren cuando un usuario parte o divide en un diálogo con un sistema durante un proceso significativo.

Metodología: Un sistema de principios y normas generales de organización y estructuración teórico-práctica de actividades.

Modelo: Descripción de las características estáticas, dinámicas o ambas de un tema, presentada en varias vistas (generalmente diagramáticas o textuales).

CamelCase: Notación que consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra. La variante "UpperCamelCase" también pone en mayúscula la primera letra de todas.