Universidad de las Ciencias Informáticas
Facultad 1

Título:

Aplicación web para la gestión de eventos científicos en la Universidad de las Ciencias Informáticas.

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

Autores: Danay Guará Faure.
Adriana Verdecia Rondón.

Tutor: Ing. Susana Gonce Fernández.

Ciudad de la Habana, junio 2010
"Año 52 de la Revolución"

PENSAMIENTO

"El conocimiento nos hace responsables." Che.

Declaración de autoría

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de la presente tesis “Aplicación web para la gestión de eventos científicos en la Universidad de las Ciencias Informáticas” y autorizamos a la Facultad 1 y a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio.

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

Danay Guará Faure

_____

Firma del autor

Adriana Verdecia Rondón

_____

Firma del autor

Ing. Susana Gonce Fernández

_____

Firma del tutor

DEDICATORIA

Adriana

Dedico mi tesis a mis padres, las personas que más amo en mi vida para que se sientan orgullosos de mí.

Danay

A mis padres, las personas más especiales de mi vida. Ellos han apoyado toda mi trayectoria estudiantil así como todas las decisiones que he tomado en mi vida. A ustedes va dedicado este trabajo, para que se sientan orgullosos de su hija y para que hagan realidad el sueño de tener una hija ingeniera.

AGRADECIMIENTOS

Adriana

A la Revolución por ofrecerme la posibilidad de formarme como profesional.

A mis padres por haberme dado siempre el apoyo incondicional de padre a hijo, por estar dispuestos a ayudarme, algo que solo se paga con el cariño, el respeto, la consideración y el amor. Les agradezco por ser el tesoro más grande que la vida me ha dado.

A toda mi familia en especial mi tía Marcia , a mis abuelos , mis primos , mis tíos de Casa Blanca que me han apoyado siempre.

A mi hermana que es lo más lindo que tengo ….y a mi prima Sarita que siempre ha estado a mi lado.

A mi novio Evelio por todo el apoyo y amor que me ha brindado, dándome fuerzas para salir adelante.

A mi suegra Maité por ser como una madre para mí y a toda la familia de mi novio por apoyarme y quererme.

A todos mis amigos, en especial Tatiana y Katerina que han estado siempre conmigo.

Danay

A mi madre por ser la persona más comprensiva del mundo, por su amor inagotable, su confianza y su apoyo incondicional.

A mi padre por su ayuda durante toda mi vida, por ser un guía, un ejemplo a seguir por mí.

A toda mi familia, a mis abuelas, mis hermanos, a todos mis tíos, en especial a mi tía Ileana por su gran apoyo.

A las personas que me han brindado su amistad, Nati, Cori, Yaimy, Julia, Arelis, Gertrudis, Lisandra, Yureisy, Arianny, Alba, Nadieska, Gabriela, Cuza, Orlando, Ramón Usatorres, Raisel, Hector, Adita, por haber podido contar con ellos sin pena siempre que lo necesité.

A todos mis compañeros de grupo de ese 1109 que tanto amé.

A la gente del barrio por sus buenos deseos para mí: Yanet, Yander, Elsy, Nelsi, Tomás, Beba, Zoraida, Gisela, a Ariel por cuidar y ayudar a mi mamá mientras estoy ausente.

A las personas que me ayudaron en mi preparación durante la carrera, a los profesores y a Ernesto por ayudarme en la preparación para la PNP.

Agradecimientos

Adriana y Danay

A nuestra tutora Susana y a Yunaysy por su generosidad al brindarnos la oportunidad de recurrir a su capacidad y experiencia en un marco de confianza, afecto y amistad.

A nuestros compañeros del proyecto por su apoyo y colaboración.

En general a todas y cada una de las personas que han vivido la realización de este trabajo por habernos brindado su apoyo, colaboración, ánimo y sobre todo cariño y amistad.

RESUMEN

Desde el momento en que surge la Universidad de las Ciencias Informáticas (UCI), se vienen desarrollando diferentes eventos relacionados con la actividad científica como parte de la formación productiva de los estudiantes y profesores, donde la Dirección de Investigaciones (DI), entidad principal en la organización de dichos eventos, realiza todo el proceso de control manualmente y en el mejor de los casos en documentos *Microsoft Word* o tablas *Microsoft Excel*. La presente investigación analiza la propuesta de un sistema para informatizar estos procesos.

Este documento expone los resultados de todo el trabajo investigativo realizado. Se identifican y describen los procesos que desarrolla la DI, específicamente aquellos que están relacionados con la organización de los eventos científicos que se realizan dentro y fuera de la universidad; argumentando y demostrando que la situación problémica requiere de un sistema que cumpla con los requisitos propuestos para satisfacer las necesidades identificadas. Seguidamente se exponen y argumentan las herramientas y tecnologías a utilizar.

Esta investigación se desarrolla siguiendo la metodología Proceso Unificado de Desarrollo (RUP), presentando todo el proceso de desarrollo del *software* hasta la fase de implementación donde se muestran los resultados del diseño de la propuesta del sistema. Una vez culminado el trabajo se contará con la informatización de las principales actividades que se ejecutan en la DI, las cuales se realizan de forma un tanto engorrosas, por el gran cúmulo de información que allí se maneja.

Índice

ÍNDICE DE CONTENIDO

Índice

Índice de tablas

ÍNDICE DE TABLAS

Índice de tablas

Índice de figuras

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad la ciencia, la tecnología e innovación se han convertido en elementos fundamentales para el desarrollo de cualquier país, donde la tecnología es el conjunto de habilidades que permiten construir objetos y máquinas para adaptar el medio y satisfacer necesidades de la ciencia como conocimiento sistematizado (1), elaborado mediante observaciones, razonamientos y pruebas metódicamente organizadas**;** la ciencia y la técnica no serán capaces por sí solas de garantizar el desarrollo, pues necesitan de una sociedad innovadora que proporcione contextos económicos, políticos, educacionales, valorativos y culturales favorables con la intención de ser útiles para el incremento de la productividad.

Debido a esto, el país ha enfocado sus esfuerzos al desarrollo científico, técnico e innovador y con él, surge la Universidad de las Ciencias Informáticas (UCI) como una fuerte base tecnológica y con un amplio perfil productivo, donde la formación de los estudiantes desde la producción, garantiza una amplia capacidad investigativa que facilita resultados científico-técnicos de impacto.

La presencia de la investigación científica en las actividades de la Universidad de las Ciencias Informáticas contribuye significativamente a la superación profesional y formación científica de su joven claustro, mediante su incorporación al trabajo científico vinculado al proceso productivo dentro de su perfil profesional.

La Dirección de Investigaciones, entre otras funciones, tiene la misión de llevar un control de la participación de sus profesores y estudiantes en los eventos científicos, tanto nacionales como internacionales, así como los organizados por la propia Universidad: Jornada Científica Estudiantil (JCE), Fórum de Ciencia y Técnica (FCT) y Conferencia Científica Uciencia, por citar algunos.

Actualmente todo este proceso de gestión de eventos científicos se realiza de forma manual, archivándose toda la información en papel y en el mejor de los casos en documentos *Microsoft Word* o tablas *Microsoft Excel*. Las informaciones son enviadas hacia los niveles superiores a través de correo electrónico. Todo esto trae como consecuencia un difícil y engorroso procesamiento de los datos, debido a la gran cantidad de información que se maneja, entre los que se encuentran: convocatoria de eventos (fecha del evento, comisiones y sus responsables, período de recepción de los trabajos), trabajos (autores, coautores, título y resumen); además de que se introduzcan errores humanos con mayor frecuencia, retrasos en la información, que exista inconsistencia de la misma en las diferentes estructuras de la Universidad y que no se cuente en todo momento con la información más actualizada.

Teniendo en cuenta esta **situación problémica**, se puede identificar el siguiente **problema científico**:

Introducción

¿Cómo viabilizar el proceso de gestión de la información de los eventos científicos en la Universidad de las Ciencias Informáticas?

La presente investigación tiene como **objeto de estudio**: el proceso de gestión de la información relacionado con la actividad de ciencia, tecnología e innovación.

Se incidirá específicamente en el proceso de gestión de la información de los eventos científicos en la Universidad de las Ciencias Informáticas, como **campo de acción**.

El **objetivo general** está encaminado a: desarrollar un sistema que viabilice el proceso de gestión de la información de los eventos científicos la Universidad de las Ciencias Informáticas.

Como **objetivos específicos** se plantean los siguientes:

- ✓ Realizar una fundamentación teórica del tema.
- ✓ Describir las características del negocio y del sistema.
- ✓ Realizar el diseño del sistema a utilizar.
- ✓ Implementar la propuesta de solución.
- ✓ Realizar las pruebas para verificar el correcto funcionamiento del sistema.

La investigación se sustenta en la siguiente **hipótesis**: El desarrollo de un sistema que gestione la información de los procesos vinculados con los eventos científicos en la Universidad de las Ciencias Informáticas, aumentará la calidad de dichos procesos.

Tomando en cuenta la hipótesis anterior se define como variable independiente, calidad de los procesos, y como variable dependiente, sistema, en el *Anexo 1* queda reflejada la operacionalización de las variables.

Para lograr el cumplimiento de los objetivos se proponen las siguientes **tareas de la Investigación**:

- ✓ Valoración de las técnicas de programación, lenguaje y marco de trabajo propuestos por el proyecto Investigaciones de la universidad.
- ✓ Estudio del estado del arte relacionado con la gestión de eventos científicos.
- ✓ Modelado de las condiciones actuales en que se desarrolla el control de la participación en eventos científicos en la UCI.
- ✓ Identificación de las funcionalidades que tendrá el sistema.
- ✓ Diseño de la aplicación web.
- ✓ Implementación del sistema.
- ✓ Realización de las pruebas necesarias para el correcto funcionamiento del sistema.

Para llevar a cabo estas tareas se emplearon **métodos teóricos y empíricos** de la investigación científica.

Los **métodos teóricos** utilizados para cumplir con las tareas a desarrollar son:

- ✓ Análisis histórico-lógico: Se usa este método para estudiar las formas de solución dadas de años anteriores a problemas similares al planteado sobre la gestión de la información de eventos científicos en la universidad desde su surgimiento hasta la actualidad.
- ✓ Analítico-sintético: Este método se usa para analizar la bibliografía encontrada referente al tema en cuestión y se sintetizan los aspectos más importantes para la investigación.
- ✓ Método de la modelación: Este método se usa porque permite crear abstracciones con el objetivo de investigar la realidad y además posibilita conocer la respuesta de los procesos sin tener que ejecutar los mismos en el mundo real.

Los **métodos empíricos** utilizados para obtener información sobre el objeto de estudio son:

- ✓ Entrevista: Se usa a través de la realización de entrevistas al personal de la Dirección de Investigaciones, con el objetivo de profundizar en el problema y entender sus especificidades para obtener la solución óptima.
- ✓ Observación: Este método se usa porque permite investigar los procesos externamente sin tener que llegar a la esencia de los mismos, lo cual sirve de ayuda para realizar un seguimiento del comportamiento de estos procesos, sirviendo como guía para conocer si se retrocede o se avanza hacia el objetivo final.

El presente documento está estructurado por 4 capítulos, a continuación se expone una breve descripción de los mismos.

Capítulo 1: Se aborda todo lo relacionado con la fundamentación teórica que sustenta la presente investigación, acerca del estudio del estado del arte así como de las herramientas, lenguajes y gestor de base de datos a utilizar y de igual forma el lenguaje de modelado y metodología a emplear.

Capítulo 2: Se identifican y describen los procesos del negocio específicamente aquellos que se van a automatizar lo cual incluye el modelo de los procesos del negocio también las características del sistema a través de los requisitos funcionales, no funcionales y la descripción de los casos de uso del sistema.

Capítulo 3: Se expondrá el diseño de la solución propuesta, lo cual incluye la definición del modelo de clases del diseño al igual que los diagramas de secuencia y el diagrama de clases persistentes.

Capítulo 4: Muestra cómo va estar estructurada la implementación del sistema, contiene los diagramas de componentes y de despliegue así como las pruebas correspondientes al sistema.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se hace un estudio del estado del arte relacionado con soluciones informáticas existentes para la gestión de eventos científicos, se abordan conceptos que están relacionados con los procesos que integran el campo de acción, así como un análisis de las tendencias y tecnologías propuestas por el proyecto Investigaciones de la universidad.

1.2 Conceptos básicos relacionados con el dominio del problema

Eventos científicos: El evento científico es parte de un proceso que se inicia en la organización de las investigaciones, pasa por una producción científica y continua con un proceso de divulgación como una vía para introducir los resultados en la práctica social. (2)

El proceso de gestión de eventos es un conjunto de actividades o procesos con un orden lógico, para compartir el conocimiento asociado a resultados obtenidos por el potencial científico del centro, que se inicia con la solicitud de participación, se evalúa la propuesta, su factibilidad, el presupuesto y al final se determina si se aprueba o no el evento por la Comisión Científica. Una vez aprobado el evento se procede a crear el equipo de trabajo que se corresponde con el Comité Organizador, dirigido por el director del evento, su prioridad, el Comité de Compras para el control de los gastos. Una vez que se desarrolla el evento, concluye con la premiación y la elaboración del informe final del evento donde se recogen los resultados del mismo para dar paso al cierre del evento.

1.3 Sistemas automatizados similares vinculados al campo de acción

1.3.1 Soluciones informáticas relacionadas con la gestión de la información de eventos científicos

Se realizó una búsqueda de soluciones informáticas que respondieran, de manera parcial o completa a la gestión de eventos científicos, y en este sentido se encontraron, sobre todo, sitios web que de alguna manera cubren algunas de las funcionalidades previstas para el futuro sistema informático. En este caso se encontró el sitio web de la XIII edición de la Convención y Feria Internacional Informática 2009, el cual contó, durante la recepción de los trabajos, con funcionalidades para la inscripción de los mismos de forma *online*. Aunque es un sitio web que no responde a los objetivos del proyecto, sí posibilitó adoptar la concepción de la inscripción de trabajos en la propuesta de solución.

Se encontraron otros sitios web que se limitaban solo a la búsqueda teniendo en cuenta determinados criterios y al listado de eventos de una determinada institución, como la sección de “Eventos” del portal

de salud en Cuba: Infomed, no es un sitio que revela los objetivos del proyecto, pero si permitió adoptar la concepción de listar los eventos que se estén desarrollando y aquellos que ya se hayan realizado, así como registrar un nuevo evento.(3) (4)

1.3.2 Sistema para la gestión de eventos científicos en la Universidad de las Ciencias Informáticas

En el curso 2008 - 2009 se realizó por parte de la facultad 8 de la Universidad de las Ciencias Informáticas un trabajo de diploma relacionado con el análisis y diseño de un módulo para gestionar los eventos científicos que se realizan dentro de la UCI, y fuera, los nacionales e internacionales. Ese sistema se realizó lo más genérico posible para que pudiera ser usado por la Dirección de Investigaciones. Por tal razón en el mismo se gestiona un conjunto de información que puede ser usada por este departamento, por tanto, hay muchas funcionalidades que sirven de base para la especificación de requisitos que se necesita en la presente investigación; pero aún así no resuelve el problema planteado, pues sobre esa idea se necesita ser más específico y agregar otras funcionalidades que no están contempladas en dicho diseño. Por todos los elementos planteados la investigación se sustenta en la idea de partir de un modelo de negocio para conocer los procesos existentes en el departamento y proponer mejoras al mismo, gestionando información similar a la del diseño estudiado pero haciendo una especificación de requisitos más detallada e incorporando nuevas funcionalidades, por ejemplo en la Dirección de Investigaciones se necesita tener el control de la inscripción de trabajos, así como registrar las convocatorias de eventos, dar la posibilidad de visualizar las convocatorias y modificarlas. (5)

1.4 Tendencias y tecnologías actuales

Las herramientas, tecnologías y metodologías son componentes fundamentales para la realización de cualquier *software*, los cuales forman parte de la política del proyecto Investigaciones y están explícitos en la elaboración de la aplicación.

1.4.1 Aplicaciones web

Las aplicaciones web son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. En otras palabras, es una aplicación de *software* que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, etc.) en la que se confía la ejecución al navegador.

Ventajas de las aplicaciones web

La creciente popularidad de las aplicaciones web se debe a sus múltiples ventajas, entre las cuales se pueden citar:

- ✓ **Multiplataforma:** Con un solo programa, un único ejecutable, las aplicaciones pueden ser utilizadas a través de múltiples plataformas, tanto de *hardware* como de *software*.
- ✓ **Actualización instantánea:** Debido que todos los usuarios de la aplicación hacen uso de un sólo programa que radica en el servidor, los usuarios siempre utilizarán la versión más actualizada del sistema.
- ✓ **Fácil de integrar con otros sistemas:** Debido a que se basa en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.
- ✓ **Acceso móvil:** El usuario puede acceder a la aplicación con la única restricción de que cuente con un acceso a la red privada de la organización o a internet, dependiendo de las políticas de dicha organización; puede hacerlo desde una computadora de escritorio, una computadora portátil o desde una agenda electrónica; desde su oficina, hogar u otra parte del mundo**.** (6) (7)

1.4.2 Lenguajes de programación

1.4.2.1 Lenguajes de programación del lado del servidor

Los lenguajes del lado del servidor son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Estos son empleados para el diseño de páginas web dinámicas donde los más utilizados para el desarrollo son: ASP (Página de Servidor Activo), PHP y PERL (Lenguaje Práctico para la Extracción e Informe). (8)

PHP

Sus siglas vienen de Hypertext Pre-Processor (Pre-Procesador de Hipertexto) es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. Generalmente se ejecuta en un servidor web, tomando el código PHP como entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es un lenguaje de alta potencia, fácil de usar e incluye la programación orientada a objetos.

Principales usos de PHP

- ✓ Programación de páginas web dinámicas, habitualmente en combinación con los motores de bases de datos PostgreSQL y MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC (Estándar de Acceso a Bases de Datos), lo que amplía en gran medida sus posibilidades de conexión.
- ✓ Programación en consola, al estilo de PERL o Shell scripting.
- ✓ Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas de PHP

- ✓ Es un lenguaje multiplataforma.
- ✓ Capacidad de conexión con la mayoría de los manejadores de bases de datos que se utilizan en la actualidad, destaca su conectividad con PostgreSQL y MySQL.
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- ✓ Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- ✓ Permite crear los formularios para la web.
- ✓ Permite las técnicas de programación orientada a objetos. (9)

1.4.2.2 Lenguajes de programación del lado del cliente

Los lenguajes de lado cliente son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pre tratamiento. Entre ellos no sólo se encuentra el HTML, sino también el Java y el JavaScript, que actualmente son simplemente incluidos en el código HTML. (8)

XHTML (Lenguaje de Marcado de Hipertexto Extensible)

Es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML (Lenguaje de Marcas Extensible). XHTML extiende HTML 4.0 combinando la sintaxis del mismo y diseñado para mostrar datos, mientras que con la de XML fue diseñado para describir los datos. Su objetivo fundamental es avanzar en el desarrollo del proyecto del *World Wide Web Consortium*, más conocido como W3C, con la idea de lograr una web semántica, donde la información y la forma de presentarla estén claramente separadas. Este lenguaje presenta ventajas claras sobre el HTML, entre ellas:

- ✓ Un navegador no necesita implementar heurísticas para detectar qué quiso poner el autor, por lo que la conversión de lenguajes puede ser mucho más sencillo.
- ✓ Como es también una implementación de XML se puede utilizar fácilmente herramientas creadas para procesamiento de documentos XML genéricas (editores, XSLT).

JavaScript

Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los que soportan la carga de procesamiento. JavaScript es un lenguaje interpretado, basado en prototipos con el que se pueden crear pequeños programas que luego son insertados en una página web y en programas más grandes. Además, es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera y Mozilla Firefox. Este lenguaje posee varias características, entre ellas podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del *mouse*, aperturas, utilización de teclas, cargas de páginas entre otros.

CSS

Las Hojas de Estilo en Cascada o Cascading Style Sheets (CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado, escrito en HTML o XML (Lenguaje de Marcas Extensible). Este estilo propone las ventajas:

- ✓ Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- ✓ Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.
- ✓ El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea). (10) (11)

Ajax

Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una tecnología de desarrollo web para crear aplicaciones interactivas o RIA sus siglas vienen de *Rich Internet Applications* (Aplicaciones de Internet Enriquecidas). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de

recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona. JavaScript es el lenguaje en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante una petición en segundo plano que viene con formato XML. El objeto encargado de realiza la misma está disponible en los navegadores actuales. Ajax no es solo una tecnología. Es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas.

Ajax incorpora:

- ✓ Presentación basada en estándares usando XHTML y CSS.
- ✓ Exhibición e interacción dinámica usando el DOM (Objetos para la representación de Documentos).
- ✓ Intercambio y manipulación de datos usando XML y XSLT.
- ✓ JavaScript.

1.4.3 Servidor web

Un servidor web es un programa que implementa el protocolo HTTP sus siglas vienen de Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto) o el protocolo HTTPS (la versión cifrada y autenticada). El protocolo HTTP pertenece a la capa de aplicación del modelo OSI sus siglas vienen de Open Systems Interconnection (Interconexión de Sistemas Abiertos) y está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. (12)

Servidor web Apache

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. La licencia Apache es una descendiente de la licencias BSD (Distribución de Software Berkeley), no es GPL (Licencia Pública General). Esta licencia permite hacer lo que se desee con el código fuente (incluso productos propietarios). Algunas razones por las que este *software* libre es grandemente reconocido en muchos ámbitos empresariales y tecnológicos son:

- ✓ Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- ✓ Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este *software* de manera que si se quiere ver que es lo que se está instalando como servidor se puede saber, sin ningún secreto, sin ninguna puerta trasera.

- ✓ Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario.
- ✓ Trabaja con PERL, PHP (Pre-Procesador de Hipertexto) y otros lenguajes de script. También trabaja con Java y páginas JSP (*Java Server Pages* o Páginas de Servidor Java). Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- ✓ Apache permite personalizar la respuesta ante los posibles errores que puedan ocurrir en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto. (13)

1.4.4 Sistemas Gestores de Base de Datos (SGBD)

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones:

- ✓ Definición de los datos.
- ✓ Mantenimiento de la integridad de los datos dentro de la base de datos.
- ✓ Control de la seguridad y privacidad de los datos.
- ✓ Manipulación de los datos.

Existen distintos objetivos que deben cumplir los SGBD:

- ✓ **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- ✓ **Consistencia**: En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa aspectos de la realidad que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.
- ✓ **Seguridad**: La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero desorientado. Normalmente, los SGBD disponen de un complejo sistema de

permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- ✓ **Integridad**: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de *hardware*, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada. Los SGBD proveen mecanismos para garantizar la recuperación de la base de datos hasta un estado consistente conocido en forma automática.
- ✓ **Control de la concurrencia:** En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, o para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- ✓ **Manejo de transacciones.** Una transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que el estado luego de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.
- ✓ **Respaldo**: Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- ✓ **Tiempo de respuesta.** Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados. (14)

PostgreSQL

El gestor de base de datos PostgreSQL es un sistema muy potente de código abierto. Soporta gran parte del estándar SQL y en algunos aspectos, está diseñado para que sea extensible por los usuarios. Posee interfaces gráficas de usuario y enlazadores para algunos lenguajes de programación. Algunas de sus principales características son:

- ✓ **Bloqueos consultivos:** Permiten el control de objetos de base de datos a nivel de aplicación usando el motor rápido de bloqueos de PostgreSQL.
- ✓ **Sentencias preparadas:** Tiene nuevas interfaces administrativas y mejoras de rendimiento en sentencias preparadas.

Ventajas:

- ✓ Alta concurrencia.
- ✓ Amplia variedad de tipos nativos.
- ✓ Multiplataforma.
- ✓ Estabilidad y confiabilidad.
- ✓ Instalación ilimitada. (15)

1.4.5 Arquitectura de software

La arquitectura de *software*, denominada además como arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del *software* para un sistema de información. Establece los fundamentos para que analistas, diseñadores, programadores, entre otros, trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. Además, define de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe poderse implementar en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea. La arquitectura de *software*, tiene que ver con el diseño y la implementación de estructuras de *software* de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema; así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad y disponibilidad. Por lo general no es necesario inventar una nueva arquitectura de *software* para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto.

1.4.6 Arquitectura cliente-servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. En términos generales, esta arquitectura se forma a partir de la agrupación de conjuntos de elementos que efectúan procesos distribuidos y cómputo cooperativo.

¿Qué es un cliente?

Se denomina cliente al proceso que inicia el diálogo, requerimiento o solicitud de recursos. Esta solicitud puede convertirse en múltiples solicitudes de trabajo a través de redes LAN (Red de Área Local) o WAN (Red de Área Amplia). La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

¿Qué es un servidor?

Se denomina servidor a cualquier recurso de cómputo dedicado a responder las solicitudes del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN o WAN, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

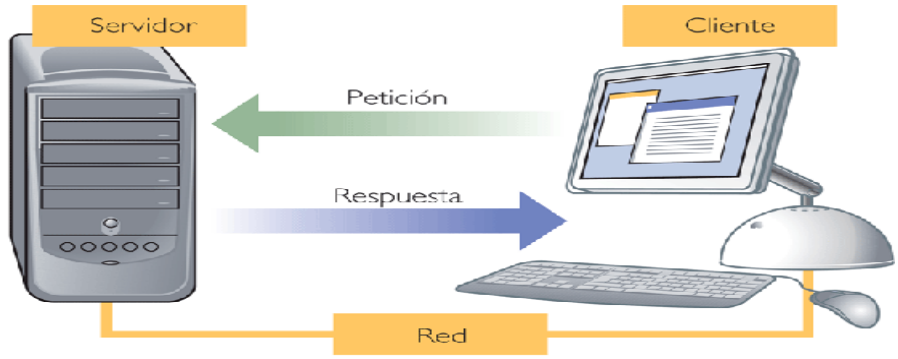

Figura 1 Arquitectura cliente-servidor.

Ventajas de la arquitectura cliente-servidor:

- ✓ El servidor no necesita tanta potencia de procesamiento, parte del proceso se reparte con los clientes.
- ✓ Se reduce el tráfico de red considerablemente.
- ✓ **Centralización del control:** los accesos, recursos y la integridad de los datos son controlados por el servidor, de forma que un programa cliente defectuoso o no autorizado, no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos.
- ✓ **Escalabilidad:** se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado o mejorado en cualquier momento, o se pueden añadir nuevos nodos a la red clientes y/o servidores.
- ✓ **Fácil mantenimiento**: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio o se afectarán mínimamente. Esta independencia de los cambios también se conoce como encapsulamiento.

1.4.6.1 Patrón arquitectónico MVC (Modelo Vista Controlador)

El MVC es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en

aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

Modelo: El modelo representa información persistente del mundo real, con la cual trabajará la aplicación. Encapsula los datos, las funcionalidades y debe preservar la integridad de los datos.

Vista: Muestra la información al usuario. Generalmente consiste de toda la información obtenida del modelo que se le muestra al usuario de la aplicación con elementos de diseño que la hacen amigable e interactiva. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador: Responde a eventos e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista. Es el que debe controlar los eventos.

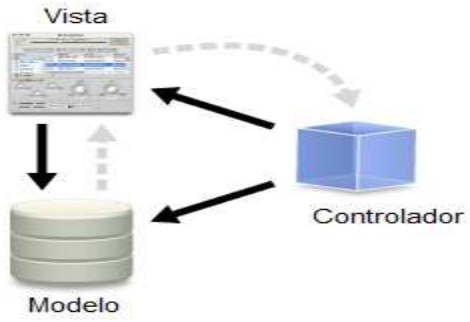

Figura 2 Patrón Modelo Vista Controlador (MVC).

Definidos los conceptos de Modelo, Vista y Controlador, a continuación se definirán las responsabilidades de cada parte:

El modelo deberá:

Acceder a los datos persistentes, a la capa de almacenamiento de datos. Algo ideal es que éste sea independiente de la base de datos utilizada.

- ✓ Definir las reglas del negocio.
- ✓ Notificar los cambios que se hayan realizado.

El controlador deberá:

- ✓ Recibir los eventos de entrada.
- ✓ Realizar la acción correspondiente al evento que se ha disparado.

Las vistas deberán:

- ✓ Conseguir los datos que aporta el modelo y mostrárselos al usuario.

- ✓ Saber cuál es su controlador asociado, el cual puede pedirle algunos servicios típicos como el de actualizar.

Ventajas:

- ✓ Se consiguen múltiples vistas del modelo.
- ✓ Todas las vistas están sincronizadas.
- ✓ No acoplamiento, y facilidad de evolución, para cambiar las vistas y los controladores.
- ✓ La aplicación puede soportar un tipo de interfaz para cada usuario (rol). (22)

1.4.7 Marco de trabajo

Un *framework* (marco de trabajo) es un esquema para el desarrollo y/o la implementación de una aplicación. Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes.

¿Qué ventajas tiene utilizar un marco de trabajo?

Las que se derivan de utilizar un estándar, entre otras:

- ✓ El programador no necesita plantearse una estructura global de la aplicación, sino que el *framework* le proporciona un esqueleto que hay que "rellenar".
- ✓ Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (o incluso con el propio, pasado algún tiempo) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.
- ✓ Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al *framework* concreto para facilitar el desarrollo.

Symfony

Symfony es un marco de trabajo basado en PHP que permite el desarrollo de las aplicaciones web basadas en este lenguaje, es un conjunto de herramientas y utilidades que simplifican el desarrollo de las mismas. Utiliza el patrón de diseño MVC (Modelo Vista Controlador) separando la lógica de negocio y la presentación consiguiendo un mantenimiento más sencillo de las aplicaciones. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Está desarrollado completamente con PHP5, incluye un (ORM) Mapeo Objeto Relacional que constituye una de las mejores soluciones de acceso a datos para PHP. Ha sido probado en numerosos proyectos y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de los gestores de bases de datos, como MySQL,

PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar en plataformas Unix, Linux y Windows. Symfony tiene las siguientes características:

- ✓ Fácil de instalar y configurar en la mayoría de las plataformas y con la garantía de que funciona correctamente en los sistemas Unix, Linux y Windows.
- ✓ Independiente del Sistema Gestor de Base de Datos (SGBD).
- ✓ Incluye las mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo. (16)

Estructura del marco de trabajo Symfony

Symfony está basado en el patrón arquitectónico Modelo Vista Controlador (MVC), implementando todas las ventajas de dicho patrón, que como ya se ha expuesto anteriormente separa la vista (interfaz) y el modelo (base de datos) mediante el controlador que es el encargado de procesar las interacciones del usuario y realizar los cambios apropiados en el modelo o en la vista, pero además Symfony hace otra división en el modelo.

- ✓ **Modelo**

  Symfony divide el modelo en una capa de acceso a datos y otra de abstracción de datos. La abstracción indica que quiere de la base de datos, y la capa de acceso hace las consultas necesarias para obtener esa información, de esta forma, si se cambia de base de datos, solamente se cambiaría la capa de acceso, y la capa de abstracción podría seguir haciendo las mismas operaciones.

- ✓ **Vista**

  En la presentación de la mayoría de las páginas existen varios elementos comunes como son: la cabecera, la navegación, el pie de página y la plantilla global conocido como *layouts*, cambiando tan solo interior o contenido de la página. Así estos 3 elementos quedan separados.

- ✓ **Controlador**

  El trabajo del controlador se repite para muchas acciones. Symfony crea un controlador frontal, único en la aplicación, que está encargado de realizar labores comunes, como son: el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares.

La siguiente figura muestra el flujo de trabajo de Symfony.

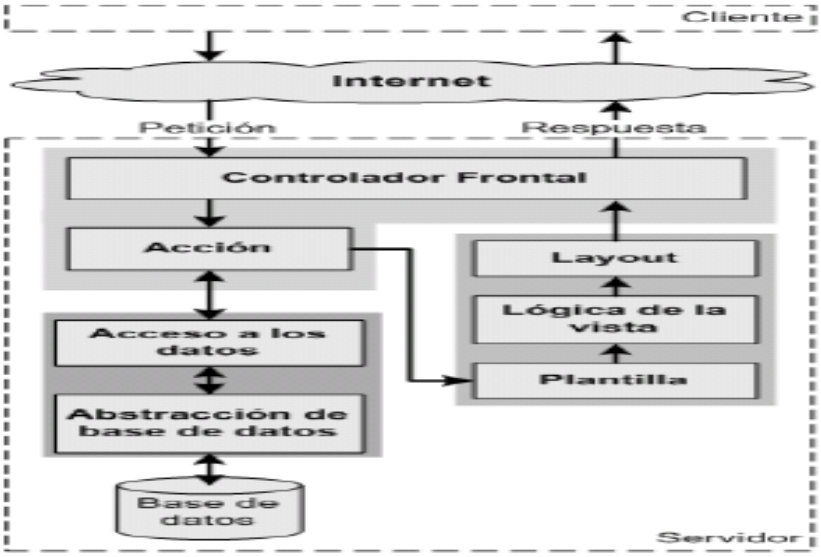

Figura 3 Flujo de trabajo de Symfony.

Symfony, como tantos otros marcos de trabajos, utiliza un Mapeador Objeto Relacional (ORM) para gestionar el acceso a la base de datos. Esto significa que se puede guardar, obtener, modificar y borrar información en una base de datos sin crear sentencias SQL a mano y sin tener que descender hasta los detalles más técnicos de cada base de datos. Además, la otra gran ventaja de los ORM es que se puede cambiar de una base de datos a otra simplemente cambiando una opción en un archivo de configuración. (17)

1.4.8 Metodología de desarrollo de software

Metodología

En un proyecto de desarrollo de *software*, la metodología define "quién" debe hacer "qué", "cuándo" y "cómo" debe hacerlo. Una metodología es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

En la actualidad existen varias metodologías OO (Orientada a Objetos) basadas en UML Unified Model Language (Lenguaje Unificado de Modelado) por ejemplo: RUP Rational Unified Process (Proceso Unificado de Desarrollo de Software), XP Extreme Programing (Programación Extrema), Métrica 3, entre otras.

RUP (Proceso Unificado de Desarrollo)

La metodología RUP sus siglas vienen de *Rational Unified Process* (Proceso Unificado de Desarrollo) propone un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema *software*. Sin embargo, el Proceso Unificado de Desarrollo es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de *software*, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyectos. El Proceso Unificado de Desarrollo está basado en componentes, lo cual quiere decir que el sistema *software* en construcción está formado por componentes *software* interconectados a través de interfaces bien definidas. (18)

La metodología RUP se distingue por ser: dirigida por casos de usos, centrada en la arquitectura, además de iterativo e incremental.

Proceso dirigido por casos de uso: Los casos de uso son como un fragmento de funcionalidad en el sistema que brinda al usuario un valor observable, son los requisitos funcionales del sistema que guían su diseño, implementación y prueba. No solo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

Proceso centrado en la arquitectura: La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además, la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. RUP presta una especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Proceso iterativo e incremental: El equilibrio correcto entre los casos de uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone RUP es tener un proceso iterativo e incremental donde el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo que el equilibrio entre casos de uso y arquitectura se vaya logrando durante cada mini proyecto. Se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. (19)

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. A continuación se exponen las fases y los flujos de trabajos del ciclo de vida de RUP:

- ✓ **Inicio:** Las iteraciones hacen mayor énfasis en actividades del modelado de negocio y requerimiento.
- ✓ **Elaboración:** Las iteraciones se orientan al desarrollo de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocio (refinamiento), análisis, diseño y una parte de la implementación orientada a la arquitectura.
- ✓ **Construcción:** Se lleva a cabo la construcción del producto por medio de una serie de iteraciones.
- ✓ **Transición:** Se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

RUP usa el lenguaje UML (Lenguaje Unificado de Modelado) en la preparación de todos los planos del sistema. De hecho, UML es una parte integral del Proceso Unificado de Desarrollo, se desarrollaron a la par y además utiliza el paradigma orientado a objeto para su descripción. RUP comprende 6 flujos de trabajo principales y 3 flujos de trabajo que son de apoyo:

Flujos ingenieriles:

- ✓ Modelado de negocio.
- ✓ Requisitos.
- ✓ Análisis y diseño.
- ✓ Implementación.
- ✓ Pruebas.
- ✓ Despliegue.

Flujos de apoyo:

- ✓ Administración de configuración y cambio.
- ✓ Administración de proyecto.
- ✓ Ambiente.

1.4.9 Lenguaje Unificado de Modelado (UML)

UML de sus siglas en inglés Unified Modeling Language en su traducción al español Lenguaje Unificado de Modelado es uno de los más conocidos y utilizados en la actualidad; aún cuando no es un estándar oficial, está respaldado por el Grupo Administrativo de Objetos (OMG) de sus siglas en inglés Object Management Group. Ofrece un estándar para describir un plano del sistema, incluyendo

aspectos conceptuales tales como procesos de negocios y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de base de datos y componentes de *software* reutilizables.

Es importante resaltar que UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de *software* Orientado a Objetos (OO) y no para describir métodos o procesos. Es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos**.** (9)

1.4.10 Herramienta de modelado

Las herramientas CASE de sus siglas en inglés *Computer Aided Software Engineering* en español Ingeniería de *Software* Asistida por Ordenador, son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras. Estas herramientas tienen como objetivos:

- ✓ Mejorar la productividad en el desarrollo y mantenimiento del *software*.
- ✓ Aumentar la calidad del *software*.
- ✓ Mejorar el tiempo, coste de desarrollo y mantenimiento de los sistemas informáticos.
- ✓ Mejorar la planificación de un proyecto.
- ✓ Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- ✓ Garantizar una correcta documentación, generación de código, pruebas de errores y gestión del proyecto.
- ✓ Ayuda a la reutilización del *software*, portabilidad y estandarización de la documentación.
- ✓ Gestión global en todas las fases de desarrollo de *software* con una misma herramienta.
- ✓ Facilitar el uso de las distintas metodologías propias de la ingeniería del *software*.
- ✓ Dentro de la lista de aplicaciones CASE se encuentra Visual Paradigm para UML, que cuenta con una versión gratuita denominada *Community Edition,* traducida al español Edición Comunitaria.

Visual Paradigm

Visual Paradigm para UML (VP-UML) es una de las herramientas CASE, que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones, considerada como: muy completa y fácil de

usar y con soporte multiplataforma. Visual Paradigm para UML está diseñado para una amplia gama de usuarios, incluidos los ingenieros de *software*, analistas de sistemas, analistas de negocios, sistema de arquitectos, al igual que para aquellas personas interesadas en la construcción de sistemas de *software* de forma fiable a través de la utilización del enfoque orientado a objetos. Esta herramienta es muy fácil de instalar y actualizar. También proporciona características tales como: generación del código; permite crear una ingeniería directa como inversa; permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación; está diseñada para usuarios interesados en sistemas de *software* de gran escala con el uso del acercamiento orientado a objeto; incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios realizados por sus compañeros. (20)

1.4.11 Herramienta de desarrollo

NetBeans

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las Apis de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de *software*.

Características:

- ✓ Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- ✓ Administración de las configuraciones del usuario.
- ✓ Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- ✓ Administración de ventanas.
- ✓ Framework basado en asistentes (diálogo paso a paso). (21)

1.5 Conclusiones

A pesar de la existencia de ciertas soluciones informáticas que cubren parcialmente algunas de las funcionalidades prevista para el sistema, no se encontró ninguna que respondiera a todas las necesidades del cliente, y que pudiera ser adoptada por el mismo; por lo que se decide llevar a cabo el desarrollo de una aplicación web para la gestión de eventos científicos en la Universidad de las

Ciencias Informáticas. Se realizó un análisis de herramientas, tecnologías y lenguajes propuestas por políticas del proyecto Investigaciones de la universidad, seleccionando para el desarrollo del trabajo como metodología de desarrollo de *software* RUP, como lenguaje de modelado UML, como herramienta CASE Visual Paradigm, como marco de trabajo Symfony, como servidor web Apache y como lenguaje de programación PHP.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se hace una propuesta general del sistema a desarrollar después de haber realizado un estudio detallado de los procesos de negocio que tienen lugar en la Dirección de Investigaciones de la Universidad. Se obtienen el modelo de negocio, se identifican los requerimientos funcionales y no funcionales que debe tener el sistema propuesto y el modelo del sistema, como artefactos fundamentales en esta etapa, aplicando la metodología RUP y haciendo uso de la herramienta Visual Paradigm y UML como lenguaje de modelado.

2.2 Flujo actual de los procesos del negocio

El negocio que requiere la solución informática a desarrollar en esta investigación comprende el desempeño de procesos relacionados con la organización de eventos científicos en la Universidad de las Ciencias Informáticas. En este sentido, la Dirección de Investigaciones (DI) es la estructura encargada de promover las políticas científicas de la Universidad, apoyada por los Vicedecanos de Investigación y Postgrado (VDIP), quienes se encargan de organizar, promover y controlar la actividad científica y postgraduada en las facultades, incluyendo las facultades regionales. Los VDIP deben mantener actualizado el estado de la participación en eventos en la DI.

Básicamente el desarrollo del negocio involucra tres procesos fundamentales: Registrar eventos UCI, registrar eventos nacionales e internacionales en Cuba y registrar eventos internacionales en el extranjero. Seguidamente se especifica con más detalles la secuencia de actividades que se realizan en cada uno de ellos.

Registrar eventos UCI

La organización de los eventos científicos desarrollados por la Universidad, queda a cargo de los VDIP de las diferentes facultades o de la DI, en dependencia del nivel en el que se desarrolle. Durante este proceso se envían los trabajos que son arbitrados posteriormente por una Comisión Científica, quienes valoran su calidad y deciden si van a ser presentados en el evento. Existe además un Comité Organizador, que como su nombre lo indica, es el grupo encargado de la organización, la acreditación y el control de todos los ponentes, conferencistas y otros invitados; en el caso de tener el evento invitados externos a la UCI, es el responsable de garantizar alojamiento, almuerzo y transporte en caso que las direcciones que responden a dichas áreas lo posibiliten. Una vez que se desarrolla el evento, se dan las premiaciones, en caso que el evento lo conciba, y se hacen las conclusiones del mismo.

Registrar eventos nacionales o internacionales en Cuba

Para la participación en eventos científicos nacionales o internacionales en Cuba el miembro de la comunidad universitaria, profesor o estudiante, debe solicitar la participación en el evento presentando

en la DI la documentación necesaria: carta de aceptación del Comité Organizador del evento, aval del Jefe administrativo superior (decanos, directores, vicerrectores) y prefactura. Posteriormente la DI hace una valoración de la utilidad que podría tener la participación en el evento para el desarrollo de la labor científica y tecnológica que se desarrolla en la Universidad; en caso de que sea útil se realiza un dictamen sobre el evento en el que se da una valoración de su importancia y se reflejan datos como las temáticas, las áreas de interés de la universidad, datos de especialistas de alto nivel que participen en el mismo y con los que la UCI podría establecer colaboraciones. Finalmente la DI realiza las verificaciones necesarias para la aprobación o no del participante y procede a tramitar el pago de inscripción en el evento presentando el caso al Comité de Compras, quien es encargado de valorar las propuestas, en cuanto a presupuesto disponible para ello y de acuerdo con su valoración se materializará la participación en dicho evento.

Registrar eventos internacionales en el extranjero

Para la participación en eventos científicos internacionales desarrollados en el extranjero, la propuesta la debe realizar el Jefe administrativo de la persona interesada (decanos, directores, vicerrectores) presentando la documentación necesaria en la DI: carta de aceptación del Comité Organizador del evento, aval del Jefe administrativo superior y resumen del trabajo. En caso de que propicie un intercambio de ideas y conocimiento que pudieran ser útiles para el desarrollo de la labor científica y tecnológica que se desarrolla en la Universidad, se realiza el dictamen sobre el evento y se entrega el expediente con toda la documentación requerida a la Dirección de Cooperación Internacional, quien recepciona la información, que luego será valorada en Rectoría. Cuando la DI recibe la decisión tomada, procede a tramitar el pago de la inscripción del evento presentando el caso al Comité de Compras.

Una vez que se materializa la participación en el evento, la DI debe dejar constancia de dicha participación y de alguna manera retroalimentar al área a la cual pertenece el participante.

2.3. Modelo de negocio

Modelar un negocio es una labor de los analistas de procesos de negocio, quienes tienen la misión de entender cómo funciona el negocio; modelando el mismo a través de diagramas de actividades donde se refleja la secuencia de pasos que se llevan a cabo, las personas beneficiadas con las acciones realizadas y las que realizan las actividades, que son los que, en ocasiones, crean, modifican o acceden a contenedores de información llamados entidades.

Los objetivos del modelamiento del negocio son:

- ✓ Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.

- ✓ Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- ✓ Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- ✓ Derivar los requerimientos del sistema que va a soportar la organización. (23)

2.3.1 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externa; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. (23)

A continuación se muestran los actores del negocio en la siguiente tabla:

Actor	Descripción
Miembro de la comunidad universitaria	Puede ser cualquier persona de la comunidad universitaria (profesores o estudiantes, etc.).
Jefe administrativo	Es la persona que debe presentar la solicitud de participación en eventos internacionales. Pueden ser decanos, directores, vicerrectores y cualquier otro jefe administrativo con facultad para ello.

Tabla 2.1 Actores del negocio.

2.3.2 Trabajadores del negocio

Un trabajador del negocio representa un rol que juega una persona o grupo de personas, una máquina o un sistema automatizado; actuando en el negocio. Son los que realizan las actividades, interactuando con otros trabajadores del negocio y manipulando entidades. (23)

A continuación se muestran los trabajadores del negocio en la siguiente tabla:

Trabajador	Descripción
Dirección de Investigaciones(DI)	Es el especialista de la DI facultado para desarrollar determinados procesos en la Dir.
Comité Organizador	Es el grupo de personas encargadas de organizar un evento desarrollado en la universidad.
Comisión Científica del evento	Es el grupo encargado de realizar el arbitraje a los trabajos que se presentan en los eventos que se desarrollan en la universidad.

Comité de Compras	Es el grupo encargado de valorar las propuestas de participación en eventos nacionales e internacionales en cuanto a presupuesto disponible para ello.
Dirección de Cooperación Internacional	Es el grupo encargado de recepcionar los expedientes con la documentación necesaria de las propuestas de participación en eventos internacionales en el extranjero.

Tabla 2.2 Trabajadores del negocio.

2.3.3 Diagrama de casos de uso del negocio

El diagrama de casos de uso del negocio es un modelo que describe los procesos de un negocio (casos de uso) y su interacción con los elementos externos (actores). Su objetivo fundamental es describir cómo el negocio es utilizado por sus clientes y socios. (24)

A continuación se muestra la figura 4 correspondiente al diagrama de casos de uso del negocio:

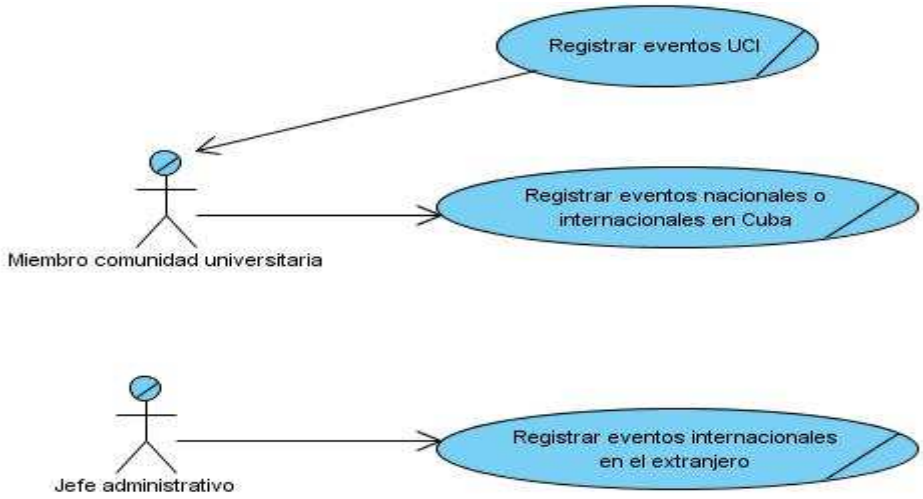

Figura 4 Diagrama de casos de uso del negocio.

2.3.4 Reglas del negocio a considerar

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, puesto que regulan algún aspecto del negocio. El proceso de especificación implica que hay que "identificarlas" dentro del negocio, "evaluar" si son relevantes dentro del campo de acción que se está modelando e "implementarlas" en la propuesta de solución. (23)

A continuación se especifican las reglas del negocio correspondientes a la gestión de eventos:

- ✓ Si el evento UCI (organizado por la universidad) tiene niveles, primero debe lanzarse la

convocatoria del nivel superior, luego los niveles inferiores lanzarán las suyas.

- ✓ El nivel superior realizará su evento, solo cuando todas sus áreas hayan realizado el suyo.
- ✓ En los eventos que no tienen niveles y en los de nivel UCI los trabajos deben ser inscritos antes que se cumpla la fecha de aceptación de los mismos.
- ✓ El Comité Organizador del evento y la Comisión Científica, se acreditan en el evento.
- ✓ En el nivel base se podrá lanzar una convocatoria por cada área, mientras que en los demás niveles de lanzará solo una.

2.3.5 Descripción de los casos de uso del negocio

La descripción de un caso de uso del negocio muestra cómo colaboran los trabajadores y las entidades del negocio para ejecutar determinados procesos.

A continuación se muestra la tabla perteneciente a la descripción del caso de uso del negocio: Registrar eventos UCI, los demás quedarán expuestos en los anexos. *Ver Anexo 2.*

Caso de uso del negocio	**Registrar eventos UCI**
Actores	Miembro de la comunidad universitaria.
Resumen	El caso de uso del negocio comienza cuando el Comité Organizador lanza la convocatoria de un evento que se desarrollará en la universidad. Mientras la convocatoria esté abierta, los participantes enviarán sus trabajos. Se desarrolla el evento y este finaliza cuando se dan los resultados del mismo.
Casos de uso asociados	
Acción del actor	**Respuesta del proceso de negocio**
	1. El Comité Organizador lanza la convocatoria.
2. La comunidad universitaria envía sus trabajos.	3. El Comité Organizador desarrolla el evento.
4. La comunidad universitaria expone sus trabajos.	5. El caso de uso del negocio termina cuando el Comité Organizador da los resultados del evento.
Otras secciones	
Mejoras propuestas	

Tabla 2.3 Descripción literal del caso de uso del negocio: Registrar eventos UCI.

2.3.6 Diagrama de actividad

Un diagrama de actividad describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio.(23)

Los diagramas de actividades correspondientes a cada caso de uso del negocio tienen representados con un color más fuerte las actividades a automatizar. A continuación se muestra la figura perteneciente al diagrama de actividades del caso de uso del negocio: Registrar eventos UCI, los demás quedarán expuestos en los anexos. *Ver Anexo 3.*

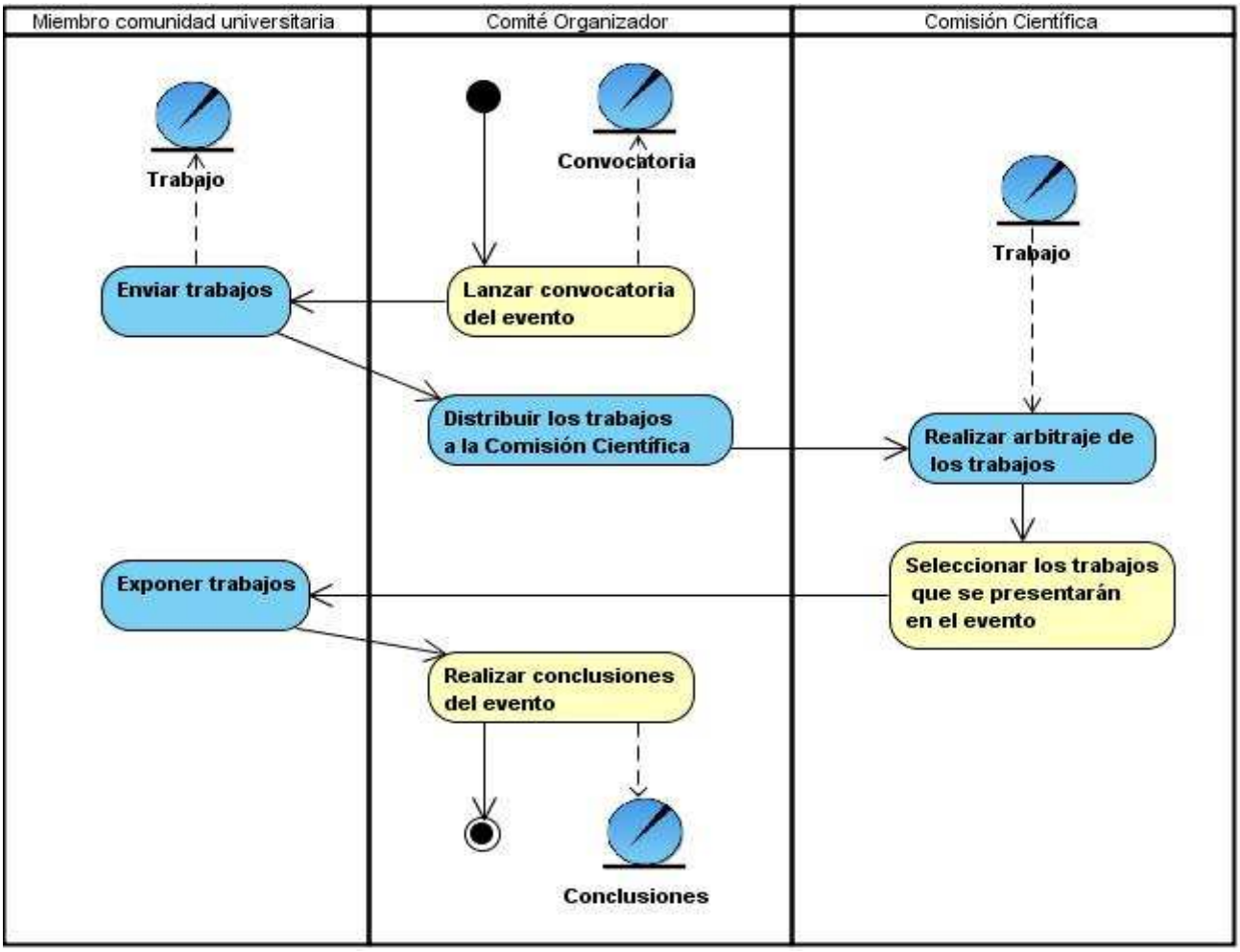

Figura 5 Diagrama de actividades del caso de uso: Registrar eventos UCI.

2.3.7 Modelo de objetos del negocio

Es un modelo de objetos que describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo de trabajo del proceso de negocio.

A continuación se muestra la figura perteneciente al diagrama clases del modelo de objetos del caso de uso del negocio: Registrar eventos UCI, los demás quedarán expuestos en los anexos *Ver Anexo 4*.

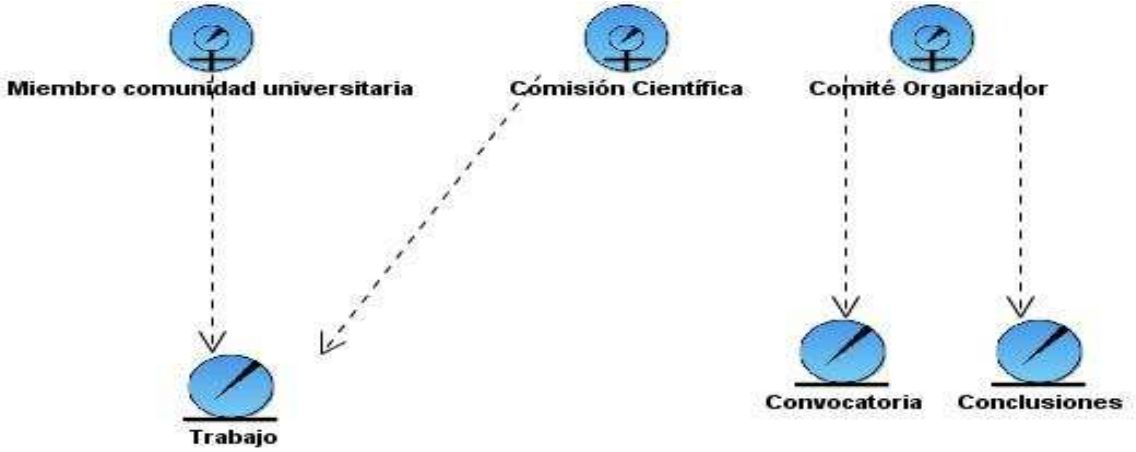

Figura 6 Diagrama de clases del modelo de objetos del caso de uso del negocio: Registrar eventos UCI.

2.4 Modelado del sistema

Con el conocimiento adquirido hasta el momento sobre los conceptos que rodean al objeto de estudio, se pueden analizar las características que debe tener el sistema para que se cumplan los objetivos planteados al inicio. Para ello se identifican los requisitos funcionales y no funcionales, modelando los requisitos funcionales en términos de casos de uso.

2.4.1 Requerimientos funcionales

Los requerimientos funcionales no alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin tener en cuenta con qué propiedades o cualidades se relacionen.

"Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir". (24)

2.4.1.1 Paquete: eventos UCI

En este paquete se definirán todos los requisitos funcionales relacionados con la gestión de los eventos internos que son organizados por la universidad.

Gestionar eventos UCI

RF1 -Crear evento (registrar).

RF2 -Modificar evento.

RF3 -Listar eventos internos.

RF4-Eliminar eventos internos.

RF5-Visualizar datos del evento.

Gestionar edición eventos UCI

RF6-Crear edición (insertar).

RF7-Modificar edición.

RF8-Visualizar datos de la edición.

RF9-Cerrar edición (no se podrán lanzar más convocatorias de ese evento en esa edición).

Gestionar convocatoria general eventos UCI

RF10-Registrar convocatoria general.

RF11-Modificar convocatoria general.

RF12-Visualizar convocatoria general.

RF13-Eliminar convocatoria general.

Gestionar inscripción de trabajos en eventos UCI

RF14-Registrar datos del trabajo.

RF15-Modificar datos del trabajo.

RF16-Visualizar datos del trabajo.

RF17-Listar trabajos registrados.

2.4.1.2 Paquete: eventos externos

En este paquete se definirán todos los requisitos funcionales relacionados con la gestión de los eventos que no son organizados por la universidad, o sea, los eventos externos en los que la comunidad universitaria participa.

Gestionar eventos externos

RF1- Listar eventos externos.

RF2- Registrar eventos externos.

RF3- Visualizar datos de eventos externos.

RF4- Modificar datos de eventos externos.

RF5- Aprobar o no evento externo.

RF6- Eliminar evento externo.

Gestionar edición de evento externo

RF7-Crear edición (insertar).

RF8-Modificar edición.

RF9-Visualizar datos de la edición.

RF10-Cerrar edición (no se podrán lanzar más convocatorias de ese evento en esa edición).

Gestionar convocatoria de evento externo

RF11-Registrar convocatoria.

RF12-Visualizar datos de la convocatoria.

RF13-Modificar datos de la convocatoria.

RF14-Eliminar convocatoria.

Gestionar inscripción de trabajos en eventos externos

RF15-Registrar datos del trabajo.

RF16-Modificar datos del trabajo.

RF17-Visualizar datos del trabajo.

RF18-Listar trabajos registrados.

2.4.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (24)

A continuación se presentan las cualidades que debe cumplir el sistema:

Usabilidad

RNF1-Facilidad de uso por parte de los usuarios: El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Además, debe ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.

RNF2-Especificación de la terminología utilizada: El sistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.

RNF3-Emplear perfiles de usuario: Diferenciar las interfaces y opciones para los usuarios que accedan al sistema con diferentes roles (Dirección de Investigaciones, Jefe administrativo, profesores o estudiantes).

RNF4-Menús: El sistema debe presentar una serie de menús tanto superiores como laterales que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.

Seguridad

RNF5-Seguridad de la base de datos: La base de datos deberá estar fraccionada en esquemas que permitan un mejor uso de la información y la división de forma lógica de las funcionalidades del sistema, trayendo consigo además la protección de la información al ocurrir un incidente sobre una

parte de la base de datos. El sistema gestor de base de datos escogido debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.

RNF6-Servicios web restringidos: Los servicios web que brinde el sistema deben estar restringidos a grupos de usuarios definidos y aprobados previamente.

RNF7-Políticas de seguridad por usuarios y roles: El sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.

RNF8-Registro sistemáticos de incidencias: El sistema debe ser capaz de registrar el accionar del usuario sobre el mismo, así como permitir auditorias y exámenes de las trazas tanto en tiempo real como en históricos. Se precisa un monitor de incidencia para la visualización y tratamiento de las mismas.

RNF9-Alta protección de los datos: Al estar trabajando con información sensible, se hace necesario una alta protección de los datos a nivel de aplicación y de tráfico por la red, para tal fin se ha definido además la seguridad en varios niveles dentro de la aplicación (nivel de interfaz, nivel de acceso a datos y nivel de base de datos), así como el uso del protocolo seguro HTTP para el tráfico de la información por la red.

Eficiencia

RNF10-El sistema debe soportar un tiempo de respuesta menor o igual a 5 segundos.

RNF11-El sistema debe soportar una conexión simultánea de más de 10 000 usuarios.

Soporte

RNF12-Grupo de soporte y asesoría: el sistema contará con un grupo de soporte y asesoría al cliente del producto destinado a brindar asesoría y soporte técnico al mismo.

Restricciones de diseño y la implementación

RNF13-Herramientas y lenguajes de programación:

Lenguaje de programación será: PHP 5.0

El framework base de desarrollo que se utilizará es: Symfony 1.4

Como IDE se empleará NetBeans 6.8.

Como servidor web se explotará Apache 2.2.11.

El SGDB deberá ser PostgreSQL 8.4.

El modelado UML se hará con Visual Paradigm 3.1

El sistema operativo a utilizar en el entorno de desarrollo deberá ser: Windows XP SP 2.

Interfaz

RNF14-Interfaz web: la interfaz deberá ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo de su empleo.

RNF15-Interfaz interna: la interfaz interna estará determinada por los desarrolladores, construyendo así una vista escalable de las clases o agrupaciones de clases que permitirán un mejor encapsulamiento de las funcionalidades y una mayor abstracción modular del sistema.

Interfaces hardware

Para el desarrollo

RNF16- Ordenador con las siguientes características: Intel Pentium 4 o superior, CPU 3GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.

Para explotación del cliente

RNF17- Ordenador con las siguientes características**:** Pentium 3 o superior, CPU 133 MHZ o superior, 128 RAM mínimo, 512 RAM recomendada o superior.

Para explotación del servidor

RNF18- Ordenador con las siguientes características: CPU: Dual Core 2.0 GHZ o superior, 4 GB RAM, 250 GB HDD. (25)

2.4.3 Actores del sistema

Los actores del sistema definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que interactúan con el mismo, intercambiando información con este. (24)

A continuación se muestran los actores del sistema en la siguiente tabla.

Actor	Descripción
Dirección de Investigaciones(DI).	Es el usuario encargado de la configuración y administración del sistema. Tiene acceso a todas las funcionalidades del mismo.
Comité Organizador del evento	Es el encargado de gestionar toda la información referente a la organización de los eventos en su nivel.
Usuario simple	Son todos los usuarios de la comunidad universitaria, que solo podrán visualizar determinada información del sistema, aparte de poder inscribir sus trabajos en los eventos UCI y en los eventos nacionales e internacionales en Cuba.
Responsable del área	Son las personas que podrán registrar eventos científicos externos (nacionales e internacionales). Entre ellas se pueden encontrar el asesor de investigación, vicedecano de producción-investigación, líderes científicos, jefes de polos y otros

	directivos.

Tabla 2.4 Actores del sistema.

2.4.4 Diagrama de casos de uso del sistema

Los Casos de Uso del Sistema (CUS) son un conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor. (24)

A continuación se muestra la figura 7 correspondiente al diagrama de casos de uso del sistema del paquete eventos científicos UCI y el listado de todos los casos de uso:

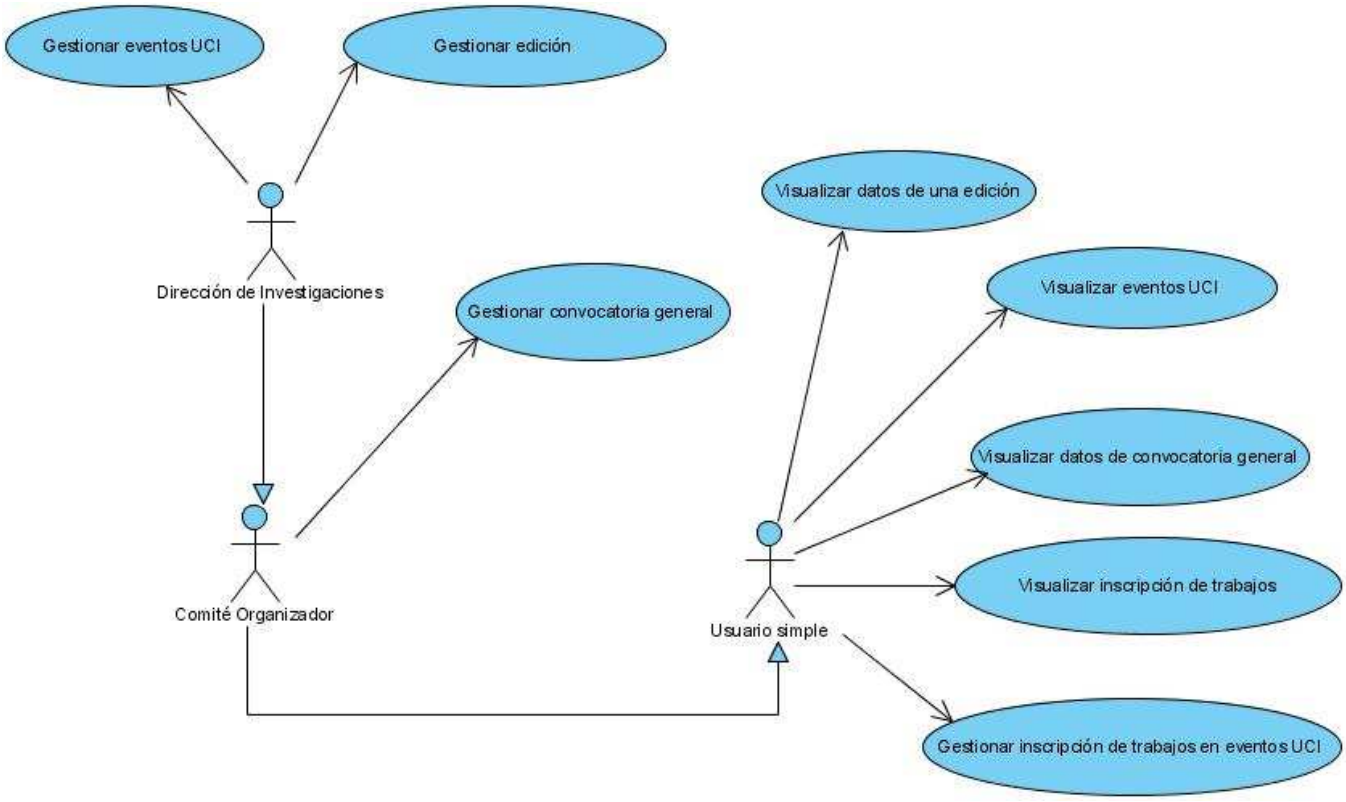

Figura 7 Diagrama de casos de uso del sistema del paquete: eventos científicos UCI.

Caso de uso:	Gestionar eventos UCI
Actores:	Dirección de Investigaciones
Resumen:	El CUS se inicia cuando un usuario accede a registrar un evento interno, modificar sus datos o eliminar el evento. El CUS finaliza cuando se ejecuta cualquiera de estas

	acciones.
Precondiciones:	• Para realizar cualquier acción sobre un evento, primeramente se deben listar los eventos internos que hayan registrados. • Para modificar, debe acceder a los datos en detalle del evento.
Referencias	RF1, RF2, RF4

Tabla 2.5 CUS_Gestionar eventos UCI.

Caso de uso:	Visualizar eventos UCI
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar los eventos internos que estén registrados. También se podrá ver toda la información de los eventos. El CUS finaliza cuando se listan dichos eventos.
Precondiciones:	
Referencias	RF3, RF5

Tabla 2.6 CUS_Visualizar eventos UCI.

Caso de uso:	Gestionar edición
Actores:	Dirección de Investigaciones
Resumen:	El CUS se inicia cuando el usuario accede a los eventos internos para crear, modificar, o cerrar una edición. Crear una edición a un evento es la manera de indicar que a partir de ese momento se puede lanzar la convocatoria de ese evento. Cerrar la edición significa que para ese evento no se podrán lanzar más convocatorias hasta tanto no se abra una nueva. Cerrar la edición significa que para ese evento no se podrán lanzar más convocatorias hasta tanto no se abra una nueva. El caso de uso finaliza cuando se crea, se modifica o se cierra una edición.
Precondiciones:	• Para crear una edición debe primero listar los eventos y acceder a los datos del evento. • Para crear una edición, no puede tener ninguna edición abierta. • Para modificar una edición, debe acceder a los datos de la edición.

Referencias	RF6, RF7, RF9

Tabla 2.7 CUS_Gestionar edición.

Caso de uso:	Visualizar datos de una edición
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar los datos de una edición de un evento interno. El CUS finaliza cuando se muestra dicha información.
Precondiciones:	• Debe primeramente listar los eventos internos. • El evento seleccionado debe tener una edición abierta.
Referencias	RF8

Tabla 2.8 CUS_Visualizar datos de una edición.

Caso de uso:	Gestionar convocatoria general
Actores:	Comité Organizador del evento
Resumen:	El CUS se inicia cuando el usuario accede a los eventos internos que tengan ediciones. Para poder registrar la convocatoria, deberá seleccionar el evento. Cuando esto ocurre se muestran los datos de la edición del evento, y el usuario tendrá la posibilidad de registrar la convocatoria para ese evento. Una vez registrada la convocatoria, se podrán visualizar, modificar sus datos y eliminar dicha convocatoria. El caso de uso finaliza cuando se registra, se modifica o se elimina la convocatoria.
Precondiciones:	• El evento debe tener edición para poder lanzar una convocatoria. • La convocatoria debe estar abierta para poder modificar y eliminar convocatoria
Referencias	RF10, RF11, RF13

Tabla 2.9 CUS_Gestionar convocatoria general.

Caso de uso:	Visualizar datos de convocatoria general
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar la información de una convocatoria general de un evento interno. El CUS finaliza cuando se muestra dicha

	información.
Precondiciones:	• Debe primeramente listar los eventos internos. • El evento seleccionado debe tener una convocatoria abierta.
Referencias	RF12

Tabla 2.10 CUS_Visualizar datos de convocatoria general.

Caso de uso:	Gestionar inscripción de trabajos en eventos UCI
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario desea subir un trabajo a una convocatoria determinada o cuando desea ver los trabajos inscritos en dicha convocatoria. Una vez registrado un trabajo, se podrán modificar sus datos. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	La convocatoria del evento general debe estar abierta.
Referencias	RF14, RF15

Tabla 2.11 CUS_Gestionar inscripción de trabajos en eventos UCI.

Caso de uso:	Visualizar inscripción de trabajos en eventos UCI
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar los trabajos inscritos en la convocatoria de un evento interno. También se podrá ver toda la información de los trabajos. El CUS finaliza cuando se listan dichos trabajos.
Precondiciones:	
Referencias	RF16, RF17

Tabla 2.12 CUS_ Visualizar inscripción de trabajos en eventos UCI.

A continuación se muestra la figura 8 correspondiente al diagrama de casos de uso del sistema del paquete eventos externos y el listado de todos los casos de uso:

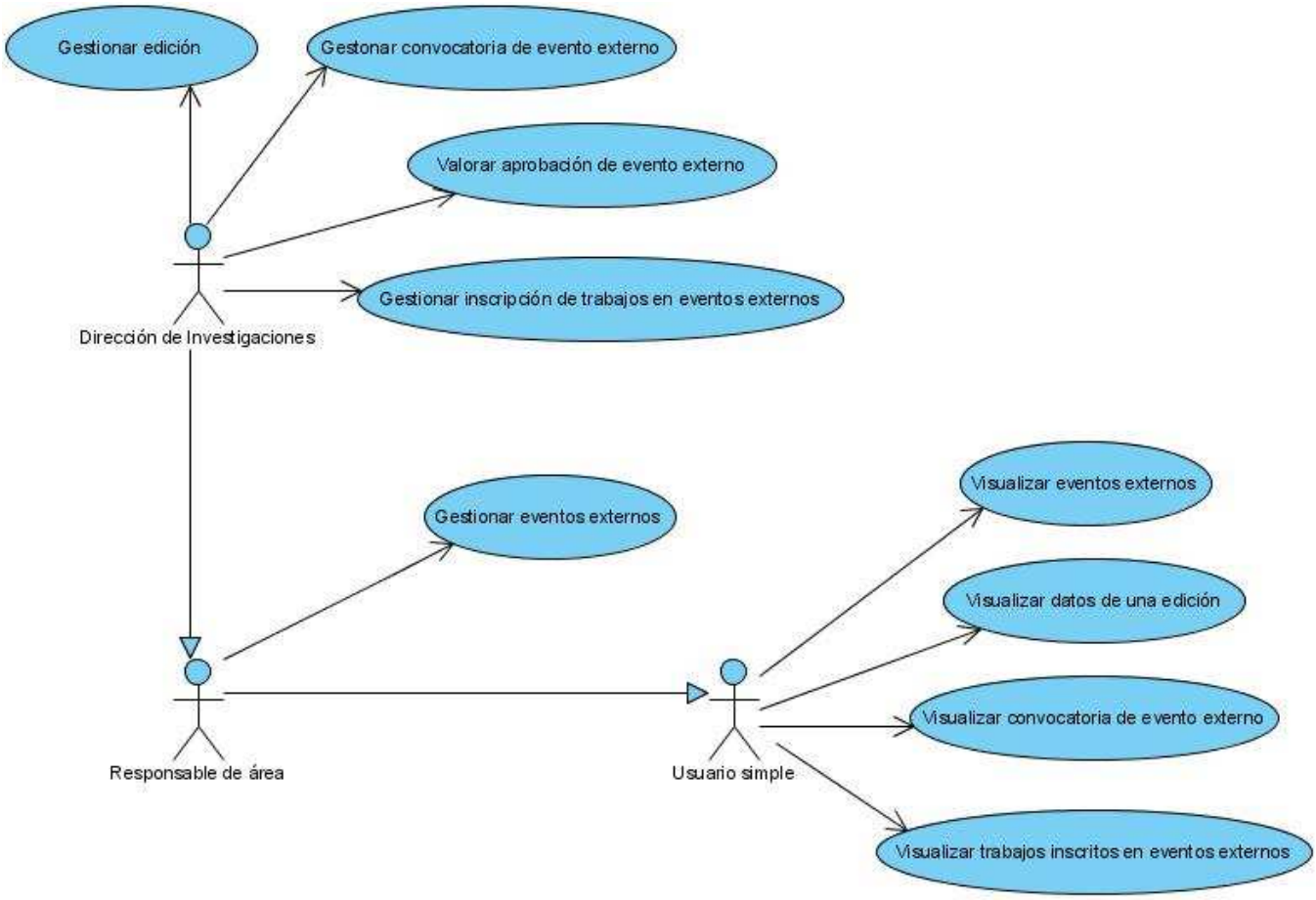

Figura 8 Diagrama de casos de uso del sistema del paquete: eventos externos.

Caso de uso:	Gestionar eventos externos
Actores:	Responsable de área
Resumen:	El CUS se inicia cuando un usuario accede a registrar un evento externo, eliminar o modificar sus datos. El CUS finaliza cuando se ejecuta cualquiera de estas acciones.
Precondiciones:	• Para realizar cualquier acción sobre un evento, primeramente se deben listar los eventos externos que hayan registrados. • Para modificar, debe acceder a los datos en detalle del evento.
Referencias	RF2, RF4, RF6.

Tabla 2.13 CUS_Gestionar eventos externos.

Caso de uso:	Valorar aprobación de evento externo

Actores:	Dirección de Investigaciones.
Resumen:	El caso de uso se inicia cuando el usuario accede a los datos de un evento externo para modificar su estado al ser aprobado o no. El caso de uso finaliza cuando se modifica el estado.
Precondiciones:	• Debe listar los eventos externos. • Debe acceder a los datos en detalles del evento.
Referencias	RF5

Tabla 2.14 CUS_Valorar aprobación de evento externo.

Caso de uso:	Visualizar eventos externos
Actores:	Usuario simple
Resumen:	El caso de uso se inicia cuando el usuario accede a visualizar los eventos externos aprobados en la universidad. El caso de uso finaliza cuando se muestran los mimos.
Precondiciones:	
Referencias	RF1, RF3

Tabla 2.15 CUS_Visualizar eventos externos.

Caso de uso:	Gestionar edición
Actores:	Dirección de Investigaciones
Resumen:	El CUS se inicia cuando el usuario accede a los eventos externos para crear, modificar, o cerrar una edición. Crear una edición a un evento es la manera de indicar que a partir de ese momento se puede lanzar la convocatoria de ese evento. Cerrar la edición significa que para ese evento no se podrán lanzar más convocatorias hasta tanto no se abra una nueva. Cerrar la edición significa que para ese evento no se podrán lanzar más convocatorias hasta tanto no se abra una nueva. El caso de uso finaliza cuando se crea, se modifica o se cierra una edición.
Precondiciones:	• Para crear una edición debe primero listar los eventos y acceder a los datos del evento. • Para crear una edición, no puede tener ninguna edición abierta.

	• Para modificar una edición, debe acceder a los datos de la edición.
Referencias	RF7, RF8, RF10

Tabla 2.16 CUS_Gestionar edición.

Caso de uso:	Visualizar datos de una edición
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar los datos de una edición de un evento externo. El CUS finaliza cuando se muestra dicha información.
Precondiciones:	• Debe primeramente listar los eventos externos. • El evento seleccionado debe tener una edición abierta.
Referencias	RF9

Tabla 2.17 CUS_Visualizar datos de una edición.

Caso de uso:	Gestionar convocatoria de evento externo
Actores:	Dirección de Investigaciones
Resumen:	El CUS se inicia cuando el usuario accede a los eventos externos que tengan ediciones. Para poder registrar la convocatoria, deberá seleccionar el evento. Cuando esto ocurre se muestran los datos de la edición del evento, y el usuario tendrá la posibilidad de registrar la convocatoria para ese evento. Una vez registrada la convocatoria, se podrán visualizar, modificar sus datos o eliminar la convocatoria. El caso de uso finaliza cuando se registra, se modifica o se elimina la convocatoria.
Precondiciones:	• El evento debe tener edición para poder lanzar una convocatoria. • La convocatoria debe estar abierta para poder modificar y eliminar convocatoria.
Referencias	RF11, RF13, RF14

Tabla 2.18 CUS_Gestionar convocatoria de evento externo.

Caso de uso:	Visualizar convocatoria de evento externo
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar la información de una

	convocatoria de un evento externo. El CUS finaliza cuando se muestra dicha información.
Precondiciones:	• Debe primeramente listar los eventos externos. • El evento seleccionado debe tener una convocatoria abierta.
Referencias	RF12

Tabla 2.19 CUS_Visualizar convocatoria de evento externo.

Caso de uso:	Gestionar inscripción de trabajos en eventos externos
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario desea subir un trabajo a una convocatoria determinada o cuando desea ver los trabajos inscritos en dicha convocatoria. Una vez registrado un trabajo, se podrán modificar sus datos. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	La convocatoria del evento debe estar abierta.
Referencias	RF15, RF16

Tabla 2.20 CUS_ Gestionar inscripción de trabajos en eventos externos.

Caso de uso:	Visualizar inscripción de trabajos en eventos externos
Actores:	Usuario simple
Resumen:	El CUS se inicia cuando el usuario accede a visualizar los trabajos inscritos en la convocatoria de un evento externo. También se podrá ver toda la información de los trabajos. El CUS finaliza cuando se listan dichos trabajos.
Precondiciones:	
Referencias	RF17, RF18

Tabla 2.21 CUS_ Visualizar inscripción de trabajos en eventos externos.

2.4.5 Expansión de casos de uso

La expansión de los casos uso quedará expuesta en los anexos. *Ver Anexo 5*

2.5 Conclusiones

Se arribó a la conclusión que en las fases de negocio y de requerimientos se logra una mejor comprensión del problema a revolver, donde fueron expuestas las características del negocio tomando este como apoyo para definir los requisitos funcionales, no funcionales seguido el modelado de los casos de uso que reflejarán las características del sistema. Además, el hecho de que tales requisitos surjan de la descripción de los procesos del negocio y que éstos sean el resultado del análisis de los objetivos del negocio, posibilita que los requisitos del sistema sean validados y verificados. Las descripciones textuales correspondientes a los casos de uso, facilitarán una visión general de lo que el sistema debe hacer, por lo que se está en condiciones de pasar a ver el diseño que es como el mismo va a realizar las operaciones antes descritas y con ello, darle continuidad para la solución del problema planteado.

CAPÍTULO 3. DISEÑO DEL SISTEMA

3.1 Introducción

Este capítulo se centrará en el desarrollo del sistema propuesto anteriormente. Para ello se construirá el modelo de diseño, que constituye un plano bastante cercano a la implementación, y contribuirá a una arquitectura estable y sólida. Como parte del modelo del diseño se hará una propuesta de los diagramas de clases del diseño, basándose en la estructura del marco de trabajo Symfony. Además, se presenta el diagrama de clases persistentes para el diseño de la base de datos.

3.2 Modelo del diseño

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar, constituyendo una entrada principal en la actividad de implementación. (27)

3.2.1 Diagramas de clases del diseño

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones del sistema con sus relaciones estructurales dando así una vista del diseño estático. En el caso de las aplicaciones web, el diagrama de clases representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase. Los diagramas de clases del diseño son utilizados con el objetivo de representar las relaciones que existen entre los distintos tipos de clases.

Para la realización de los diagramas se tuvo en cuenta la estructura del marco de trabajo Symfony. Es válido resaltar que los componentes de Symfony, como son el controlador frontal, *layouts* y el Doctrine, encargado de realizar el mapeo relacional de los objetos, al ser transparentes al programador se decidió no modelarlos, es decir, se modelaron tratando siempre de administrar la complejidad y por tanto se eliminaron todos los elementos que por su relevancia no juegan en el diagrama un papel fundamental. A continuación se mostrará una representación del diagrama de clases del diseño web, del caso de uso gestionar inscripción de trabajos en eventos UCI, con una breve explicación de sus componentes fundamentales, los demás diagramas se exponen en los anexos: *Ver Anexo 6.*

Nombre de la clase	**TrabajosActions**
Nombre del método:	New()
Descripción:	Este método se encarga de registrar los datos de cada nuevo trabajo que se desea inscribir.

Nombre del método:	Show()
Descripción:	Este método se encarga de visualizar los datos de los trabajos inscritos.
Nombre del método:	Edit()
Descripción:	Este método se encarga de modificar los datos deseados en determinado trabajo.
Nombre del método:	Delete()
Descripción:	Este método es el encargado de eliminar el trabajo deseado de la base de datos.
Nombre del método:	Update()
Descripción:	Este método es el encargado de actualizar los datos que se modifiquen.

Tabla 3.1 Descripción del diagrama de clases del diseño web del caso de uso: Gestionar inscripción de trabajos en eventos UCI.

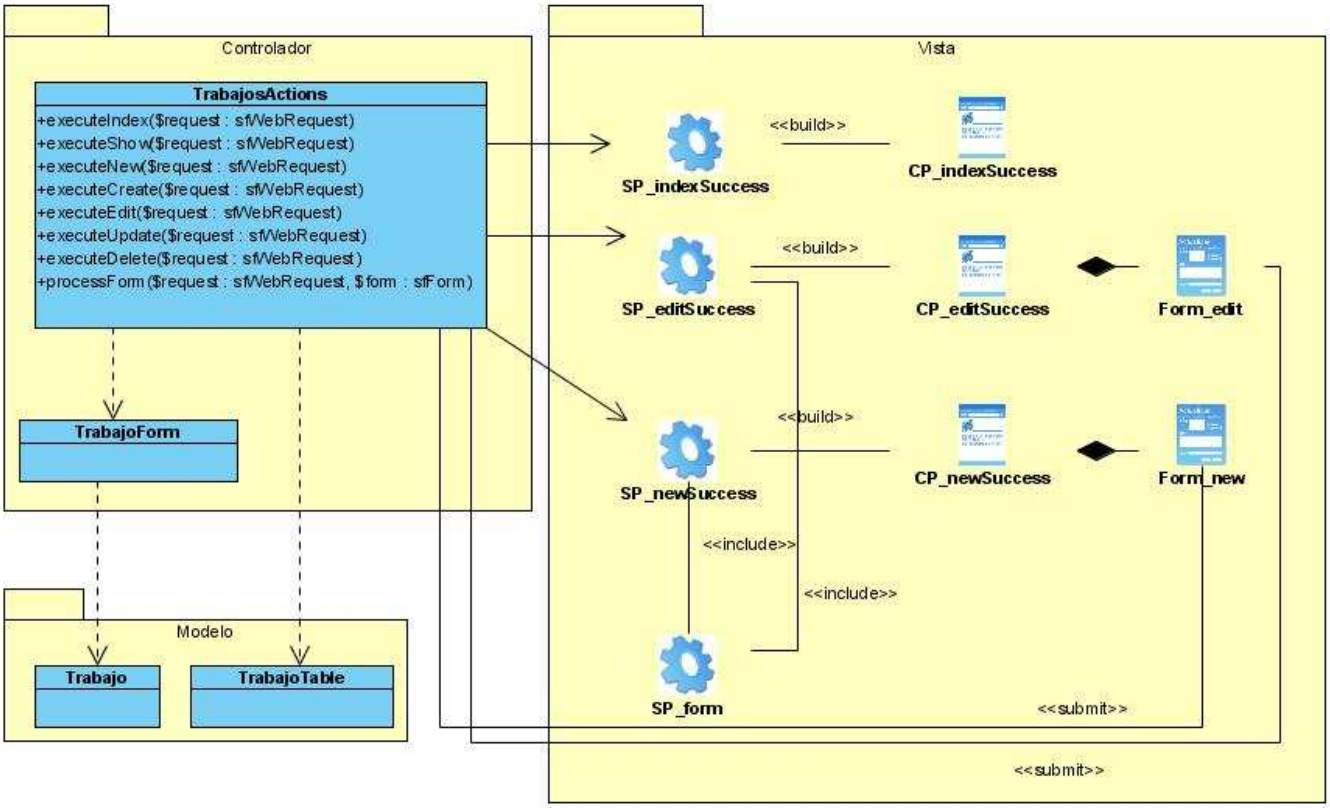

Figura 9 Diagrama de clases del diseño web: Gestionar inscripción de trabajos en eventos UCI.

3.2.2 Diagramas de interacción. Diagramas de secuencia

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el escenario que ilustra un comportamiento determinado. En el contexto de las clases se describe la forma en que grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión externa del sistema, la funcionalidad de dichos casos de uso se recoge como un flujo de eventos utilizando para ello interacciones entre sociedades de objetos. Cada caso de uso es una red de escenarios primarios (flujo normal del caso de uso) y secundarios (flujos excepcionales y alternativos). Por tanto, para un caso de uso podemos definir diferentes instancias (escenarios) que nos ayudan a la identificación de objetos, clases e interacciones entre objetos necesarios para llevar a cabo la parte de funcionalidad que especifica el caso de uso. Los escenarios documentan el reparto de las responsabilidades que se especifican en el caso de uso.

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes pasados entre los objetos.(27)

A continuación se muestran los diagramas de secuencia pertenecientes al caso de uso gestionar inscripción de trabajos en eventos UCI, los demás diagramas se exponen en los anexos. *Ver Anexo7*

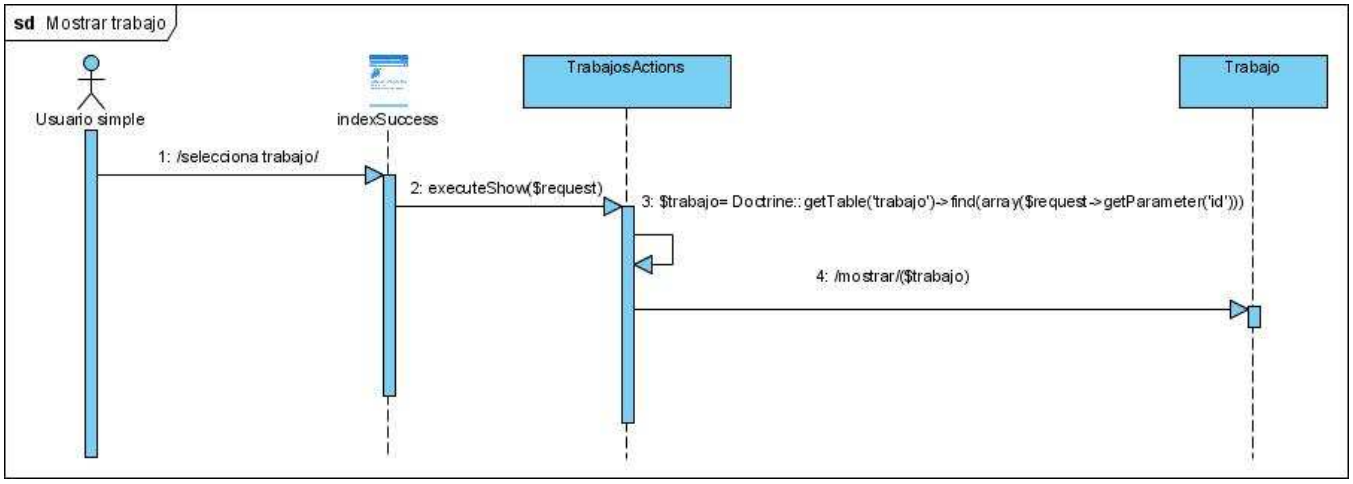

Figura 10 Gestionar inscripción de trabajos en eventos UCI: Ver trabajo.

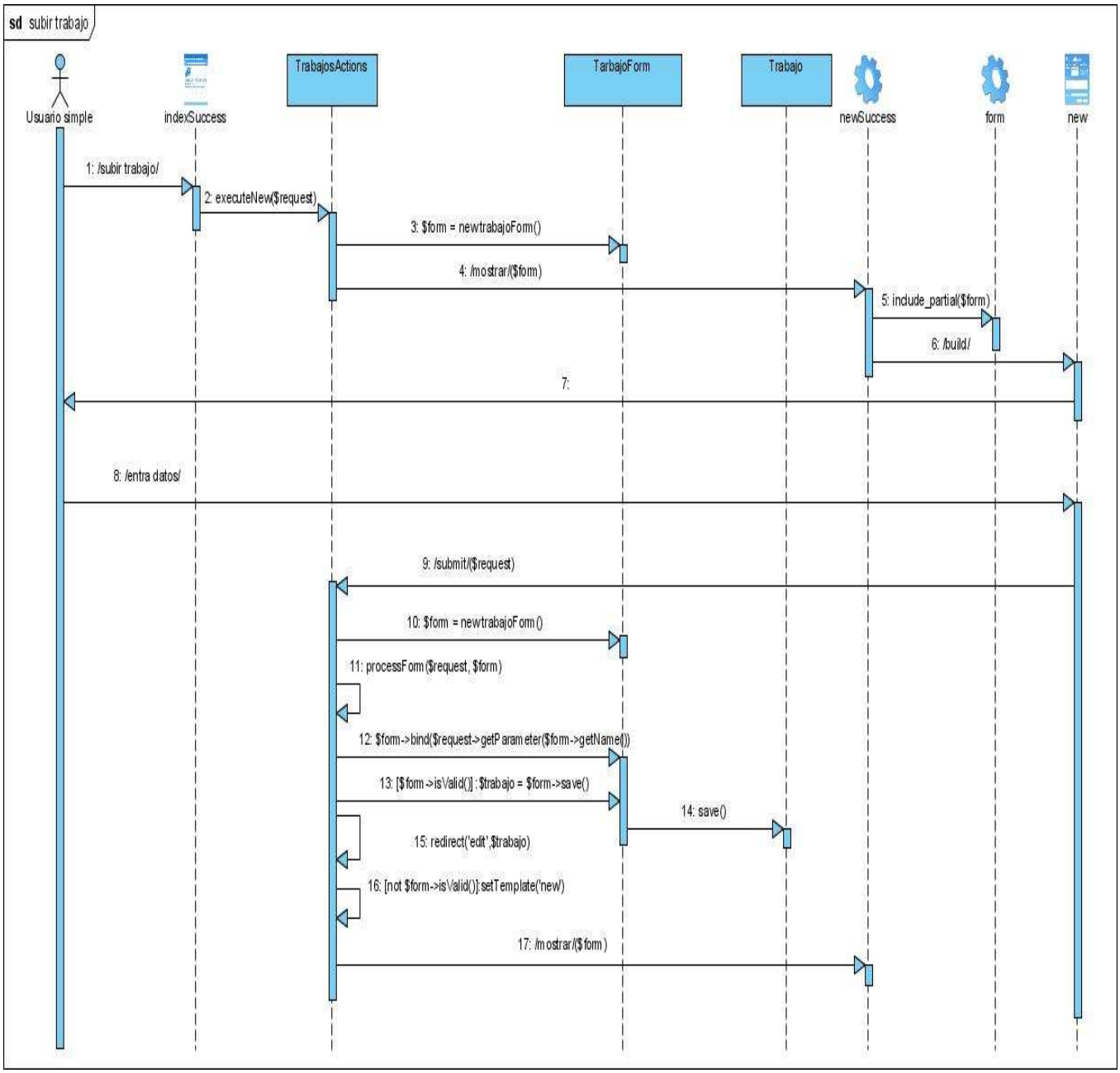

Figura 11 Gestionar inscripción de trabajos en eventos UCI: Subir trabajo.

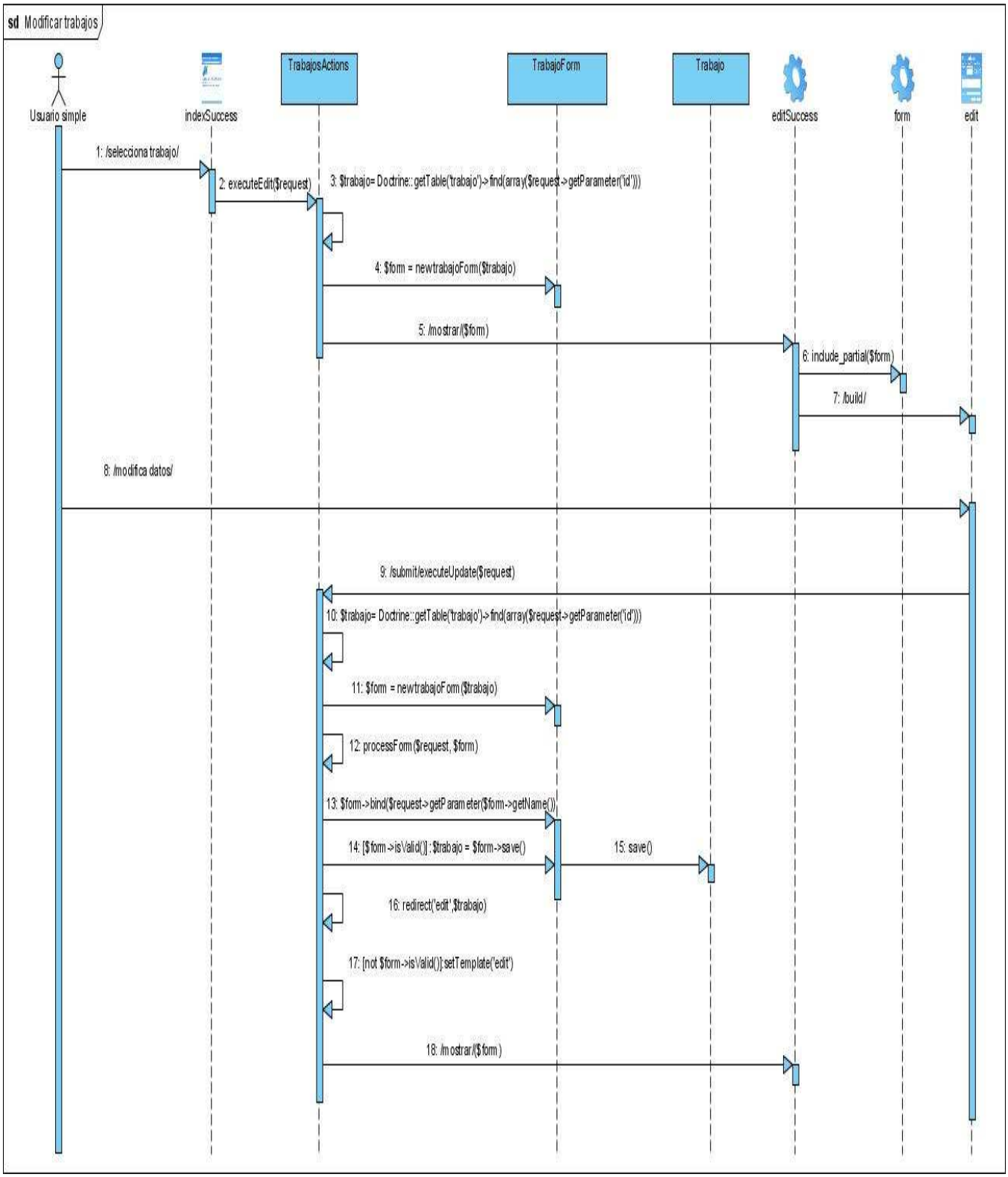

Figura 12 Gestionar inscripción de trabajos en eventos UCI: Modificar trabajo.

3.3 Diseño de la base de datos

La base de datos necesita de una definición de su estructura, de manera que permita almacenar datos, reconocer el contenido, y recuperar la información. Para diseñar una base de datos es necesario seguir un conjunto de pasos que comienzan con definir las clases persistentes, luego refinarlas y clasificarlas junto con sus atributos, para más tarde realizar el diagrama de clases persistentes. Realizar el diagrama de transición de estado es el siguiente paso para diseñar la base de datos, y el último es la conversión de las clases al medio de almacenamiento.

3.3.1 Diagrama de clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Por lo general las clases persistentes tienen como origen las clases clasificadas como entidad porque ellas modelan la información del sistema y el comportamiento asociado de algún fenómeno o concepto. El diagrama de clases persistentes describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. (28)

A continuación se muestra en la figura 13 el diagrama de clases persistentes:

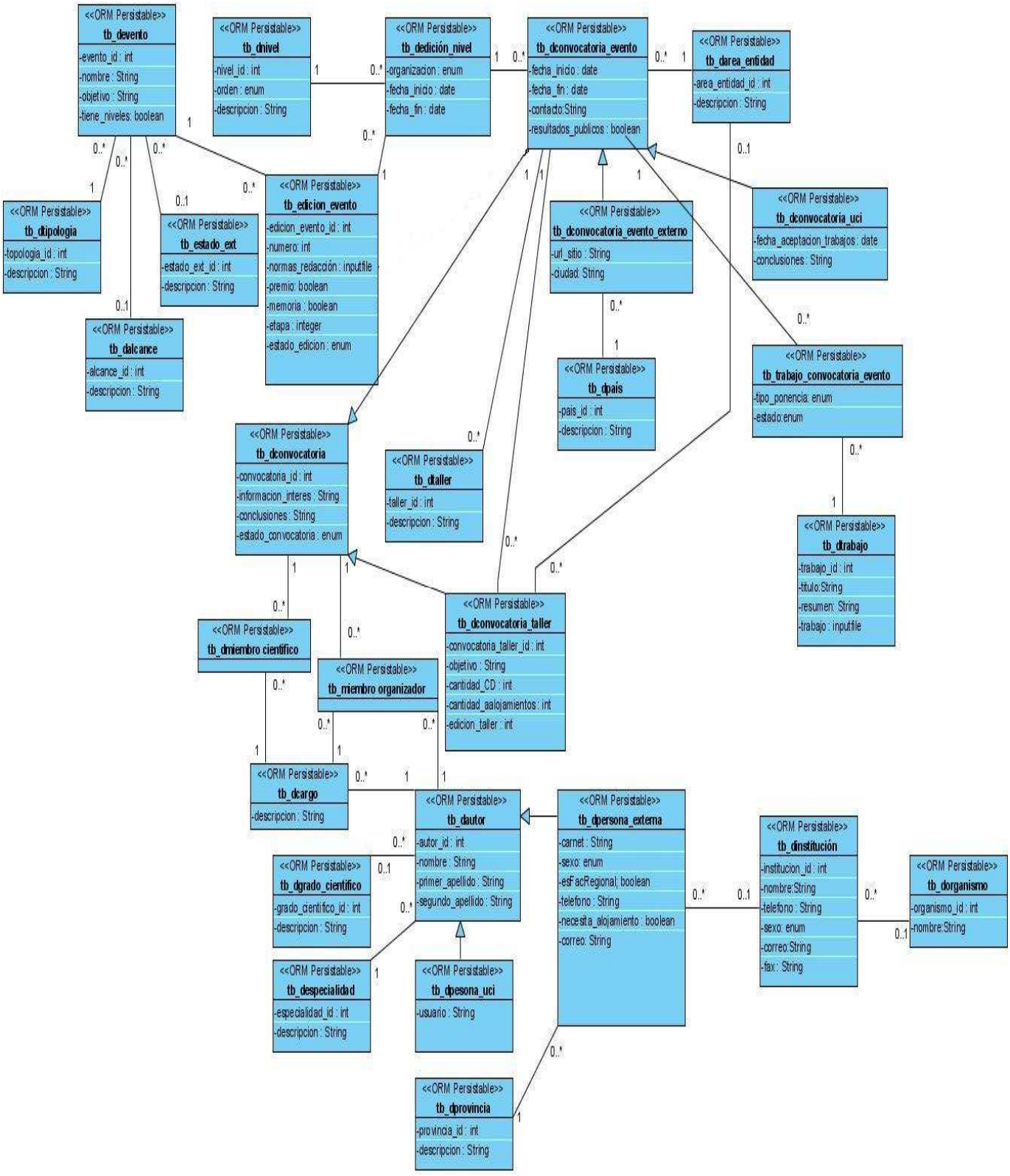

Figura 13 Diagrama de clases persistentes.

3.3.1.1 Descripción de las tablas de la base de datos

Nombre: tb_dtipología		
Descripción: En esta tabla se almacenan las tipologías de eventos.		
Atributo	**Tipo**	**Descripción**
descripción	string	Tipología del evento.
tipología_id	integer	Identificador de la tabla dtipología.

Nombre: tb_dalcance		
Descripción: En esta tabla se almacenan los alcances de los eventos.		
Atributo	**Tipo**	**Descripción**
descripción	string	Alcance del evento.
alcance_id	integer	Identificador de la tabla dalcance.

Nombre: tb_destado_ext		
Descripción: En esta tabla se almacenan los estados de eventos externos.		
Atributo	**Tipo**	**Descripción**
descripción	string	Estado del evento externo.
estado_ext_id	integer	Identificador de la tabla destado_ext.

Nombre: tb_dnivel		
Descripción: En esta tabla se almacenan los niveles de ediciones de eventos.		
Atributo	**Tipo**	**Descripción**
orden	enum	El orden del nivel de la edición (0, 1, 2, 3, 4 o 5.)
descripción	string	Nivel del evento según el orden.
nivel_id	integer	Identificador de la tabla dnivel.

Nombre: tb_dárea_entidad		
Descripción: En esta tabla se almacenan los datos de las diferentes áreas.		
Atributo	**Tipo**	**Descripción**
descripción	string	Breve descripción del área.
área_entidad_id	integer	Identificador de la tabla dárea_entidad.

Nombre: tb_dcargo		
Descripción: En esta tabla se almacenan los datos de los cargos que puede tener una persona.		
Atributo	**Tipo**	**Descripción**
descripción	string	Cargo de la persona.

Nombre: tb_dpaís		
Descripción: En esta tabla se almacenan los datos de los diferentes países.		
Atributo	**Tipo**	**Descripción**
descripción	string	Nombre del país.
país_id	integer	Identificador de la tabla dpaís.

Nombre: tb_dprovincia
Descripción: En esta tabla se almacenan los datos de las provincias a la que pertenecen las personas

externas.

Atributo	Tipo	Descripción
descripción	string	Nombre de la provincia.
provincia_id	integer	Identificador de la tabla dprovincia.

Nombre: tb_dgrado_científico		
Descripción: En esta tabla se almacenan los datos de los grados científicos que puede tener una persona.		
Atributo	**Tipo**	**Descripción**
descripción	string	Grado científico de la persona.
grado_científico_id	integer	Identificador de la tabla dgrado_científico.

Nombre: tb_despecialidad		
Descripción: En esta tabla se almacenan los datos de las especialidades que puede tener una persona.		
Atributo	**Tipo**	**Descripción**
descripción	string	Especialidad de la persona.
especialidad_id	integer	Identificador de la tabla despecialidad.

Nombre: tb_dorganismo		
Descripción: En esta tabla se almacenan los datos del organismo al que pertenece una institución.		
Atributo	**Tipo**	**Descripción**
descripción	string	Nombre del organismo.
organismo _id	integer	Identificador de la tabla dorganismo.

Nombre: tb_devento		
Descripción: En esta tabla se almacenan los datos de los eventos.		
Atributo	**Tipo**	**Descripción**
evento_id	integer	Identificador de la tabla devento.
objetivo	string	Objetivo del evento.
tiene_niveles	boolean	Si el evento tiene o no tiene niveles.
nombre	string	Nombre del evento.

Nombre: tb_dedición_evento		
Descripción: En esta tabla se almacenan los datos de la edición de los eventos.		
Atributo	**Tipo**	**Descripción**
número	integer	Número de la edición del evento.
etapa	integer	Etapa de la edición.
estado_edición	enum	Estado de la edición del evento (abierta o cerrada).
premio	boolean	Si tiene o no tiene premio.
normas de redacción	inputfile	Documento que contiene las normas de redacción de los trabajos que se inscriben en las convocatorias de la edición.
memoria	boolean	Si tiene o no tiene memoria.
edición_evento_id	integer	Identificador de la tabla dedición_evento.

Nombre: tb_dedición_nivel
Descripción: En esta tabla se almacenan las ediciones de los eventos en un nivel.

Atributo	Tipo	Descripción
organización	enum	Tipo de organización (externo o interno).
fecha_inicio	date	Fecha de inicio de la edición de un evento en un nivel.
fecha_fin	date	Fecha de fin de la edición de un evento en un nivel.

Nombre: tb_dconvocatoria		
Descripción: En esta tabla se almacenan los datos de las convocatorias de eventos. Constituye una generalización de la tabla: dconvocatoria_evento.		
Atributo	**Tipo**	**Descripción**
información_interés	string	Información que la persona que lanza la convocatoria desee publicar.
conclusiones	integer	Conclusiones de la convocatoria.
estado_convocatoria	enum	Estado de la convocatoria (abierta o cerrada).
convocatoria_id	integer	Identificador de la tabla dconvocatoria.

Nombre: tb_rtrabajo_convocatoria_evento		
Descripción: En esta tabla se almacenan los datos de los trabajos inscritos en convocatorias de los eventos.		
Atributo	**Tipo**	**Descripción**
tipo_ponencia	enum	Tipo de ponencia (virtual o presencial).
estado	enum	Tipo de estado (propuesto, aprobado o rechazado).

Nombre: tb_dtrabajo		
Descripción: En esta tabla se almacenan los datos de los trabajos.		
Atributo	**Tipo**	**Descripción**
título	string	Título del trabajo.
resumen	string	Resumen del trabajo.
trabajo	inputfile	Fichero del trabajo.
trabajo_id	integer	Identificador de la tabla dtrabajo.

Nombre: tb_dconvocatoria_evento		
Descripción: En esta tabla se almacenan los datos de las convocatorias de los eventos. Constituye una generalización de las tablas: dconvocatoria_evento_externo y dconvocatoria_uci.		
Atributo	**Tipo**	**Descripción**
fecha_inicio	date	Fecha de inicio de la convocatoria del evento
fecha_fin	date	Fecha de fin de la convocatoria del evento.
resultados_públicos	boolean	Si los resultados están o no publicados.
contacto	string	Datos de contacto.

Nombre: tb_ dconvocatoria_evento_externo		
Descripción: En esta tabla se almacenan los datos de las convocatorias de eventos externos.		
Atributo	**Tipo**	**Descripción**
url_sitio	string	Dirección del sitio de la convocatoria del evento externo.
ciudad	string	Ciudad donde se realiza la convocatoria del evento.

Nombre: tb_ dconvocatoria_uci		
Descripción: En esta tabla se almacenan los datos de la convocatoria de eventos internos.		
Atributo	**Tipo**	**Descripción**
fecha_aceptación_trabajos	date	Fecha de aceptación de los trabajos de la convocatoria del evento interno.
conclusiones	string	Conclusiones de la convocatoria del evento interno.

Nombre: tb_dtaller		
Descripción: En esta tabla se almacenan los datos de los talleres.		
Atributo	**Tipo**	**Descripción**
taller_id	integer	Identificador de la tabla dtaller.
descripción	string	Nombre del taller.

Nombre: tb_dautor		
Descripción: Almacena todos los contenidos de las personas. Constituye una generalización de las tablas: dpersona_uci y dpersona_externa.		
Atributo	**Tipo**	**Descripción**
nombre	string	Nombre de la persona
primer_apellido	string	Primer apellido de la persona.
segundo _apellido	string	Segundo apellido de la persona.
autor_id	integer	Identificador de la tabla dautor.

Nombre: tb_dpersona_uci		
Descripción: Almacena todos los contenidos de la personas internas.		
Atributo	**Tipo**	**Descripción**
usuario	string	Usuario de la persona UCI.

Nombre: tb_dpersona_externa		
Descripción: Almacena todos los contenidos de la personas externas.		
Atributo	**Tipo**	**Descripción**
carnet	string	Carnet de identidad de la persona externa.
teléfono	string	Teléfono de la persona externa.
correo	string	Correo electrónico de la persona externa.
es_facultad_regional	boolean	Si la persona es de alguna facultad regional.
necesita_alojamiento	boolean	Si la persona externa necesita alojamiento.
sexo	enum	Sexo de la persona externa (masculino o femenino).

Nombre: tb_dinstitución		
Descripción: En esta tabla se almacenan los datos de las instituciones.		
Atributo	**Tipo**	**Descripción**
institución_id	integer	Identificador de la tabla dinstitución.
nombre	string	Nombre de la institución.
teléfono	string	Teléfono de la institución.
fax	string	Fax de la institución.
correo	string	Correo electrónico de la institución.

3.4 Patrones usados en el diseño

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Symfony en su implementación usa una serie de patrones para su funcionamiento como son los que se mencionan a continuación:

3.4.1 Patrones GRASP

Creador

En la clase Acción de cada uno de los módulos se encuentran definidas las acciones del sistema 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 Acción es "creador" de dichas entidades.

Experto

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

Alta Cohesión

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

Controlador

Todas las peticiones son manejadas por un solo controlador frontal (sfAcción), 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

Este patrón se manifiesta en cada uno de los módulos del sistema, la clase Acción hereda solamente de sfAcción para lograr un bajo acoplamiento de clases.

3.4.2 Patrones GOF

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, para no tener que 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. La siguiente imagen ilustra lo anteriormente descrito.

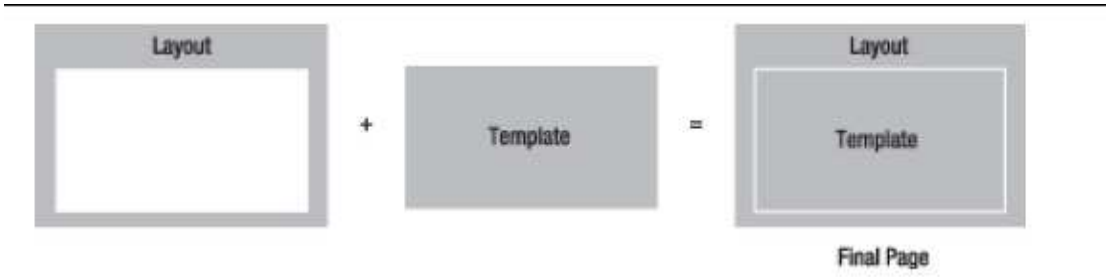

Figura 14 Ejemplo del patrón Envoltorio.

- ✓ **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.

En la categoría comportamiento:

- ✓ **Command (Acción):** El cual permite que diferentes objetos puedan ejecutar la misma acción sin necesidad de repetir su declaración e implementación. (29)

3.5 Conclusiones

En este capítulo quedaron expuestos una serie de elementos y diagramas que muestran como está construido el sistema en términos de clases del diseño. Donde se arribó a la conclusión de que haciendo un diseño detallado de la estructura de la futura aplicación dará la posibilidad de comprender mejor la lógica del sistema en general para el lector, y optimizará el tiempo de implementación.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA

4.1 Introducción

En este capítulo quedan contenidos todos los detalles referentes a la implementación del sistema, mostrando de esta forma los diagramas de componentes dando una vista detallada de cada uno de los paquetes en que se ha dividido y los múltiples elementos físicos y archivos que conforman la aplicación con el objetivo de lograr una mayor claridad y comprensión del modelo, seguidamente el diagrama de despliegue que muestra la distribución del sistema en los diferentes elementos de *hardware* que le darán soporte.

4.2 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Un componente es una parte física y reemplazable de un sistema que se conforma con un conjunto de interfaces y proporciona la realización de dicho conjunto. Se usan para modelar los elementos físicos que pueden hallarse en un nodo por lo que empaquetan elementos como clases, colaboraciones e interfaces.

Algunos estereotipos estándar de componentes son los siguientes:

- ✓ Ejecutable: Es un programa que se puede ejecutar en un nodo.
- ✓ Biblioteca: Es una biblioteca de objetos estática o dinámica.
- ✓ Tabla: Es una tabla de una base de datos.
- ✓ Archivo: Es un fichero que contiene código fuente o datos.
- ✓ Documento: Es un documento.
- ✓ Página web: Es una página que se obtiene de la ejecución del sistema.

Los componentes tienen las siguientes características:

- ✓ Tienen relaciones de traza con los elementos del modelo que implementan.
- ✓ Pueden implementar varios elementos. Por ejemplo, varias clases. Sin embargo, la forma exacta en que se crea esta traza depende de cómo van a ser estructurados y modularizados los ficheros de código fuente, dado el lenguaje de programación que se esté usando. (30)

En los diagramas realizados, se muestran como están distribuidos los componentes según el patrón arquitectónico Modelo Vista Controlador que utiliza Symfony, paradigma en su organización interna. El componente *sfFrontWebController* o Controlador Frontal maneja todas las peticiones web, siendo el punto de entrada 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 escrita o solicitada por el usuario. El controlador frontal se encarga de despachar las peticiones, lo que implica algo más que detectar la acción que se ejecuta. De hecho, ejecuta el código común a todas las acciones. En pocas palabras el controlador frontal es el encargado de determinar qué combinación de módulo-acción se ejecutará. El paquete vista se encarga de producir las páginas que se muestran como resultado de las acciones que se soliciten, las cuales se integran con el *layout*. En el paquete controlador se representan todas las acciones, estas actions.php están relacionadas con todos los archivos Success.php de la vista que contiene el código que liga la lógica de negocio con la presentación. Al acceder a las diferentes acciones lo primero que se hace es verificar si el usuario está autenticado y tiene los permisos correspondientes a la acción que desea realizar.

El modelo es la capa que contiene las clases: entidad y entidadTable. Estas clases son construidas por el subsistema Doctrine para el acceso a datos, permitiendo el acceso a la base de datos uci_eventos mediante el mapeo de objetos.

A continuación se muestra de forma general, como quedan distribuidos los componentes en el diagrama de componentes perteneciente a la aplicación:

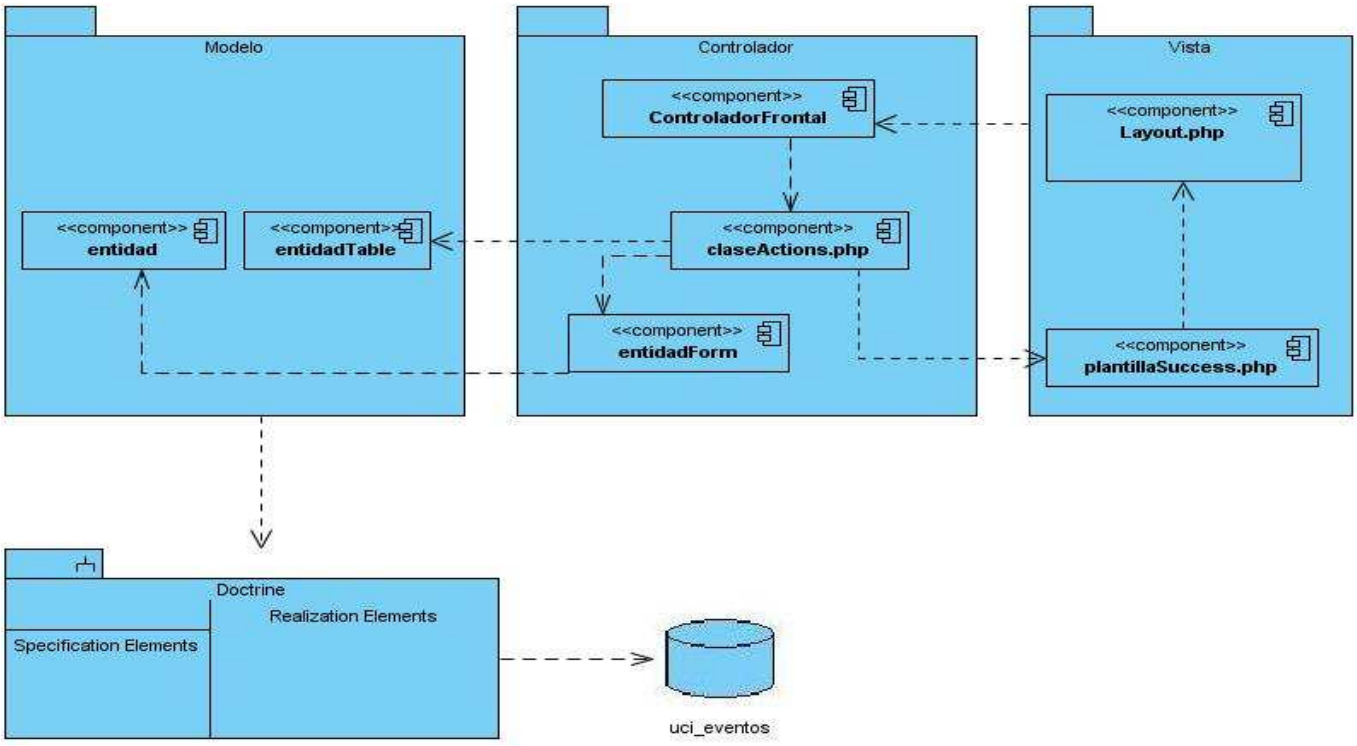

Figura 15 Diagrama de componentes de la aplicación.

4.3 Diagrama de despliegue

El diagrama de despliegue es utilizado para modelar el *hardware* empleado en las implementaciones de sistemas y las relaciones entre sus componentes. El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.(30) A continuación se explican los recursos presentes en cada nodo del diagrama:

- ✓ PC Cliente: Estación de trabajo desde donde interactúa el cliente con la aplicación.
- ✓ Servidor web Apache + PHP: Contiene la aplicación web.
- ✓ Servidor de base de datos PostgreSQL: Servidor donde se almacenan los datos de la aplicación.
- ✓ Impresora: Dispositivo que permite imprimir las informaciones generadas.
- ✓ Servidor de autenticación con el dominio uci.cu.

A continuación, en la figura 16 se muestra el diagrama de despliegue:

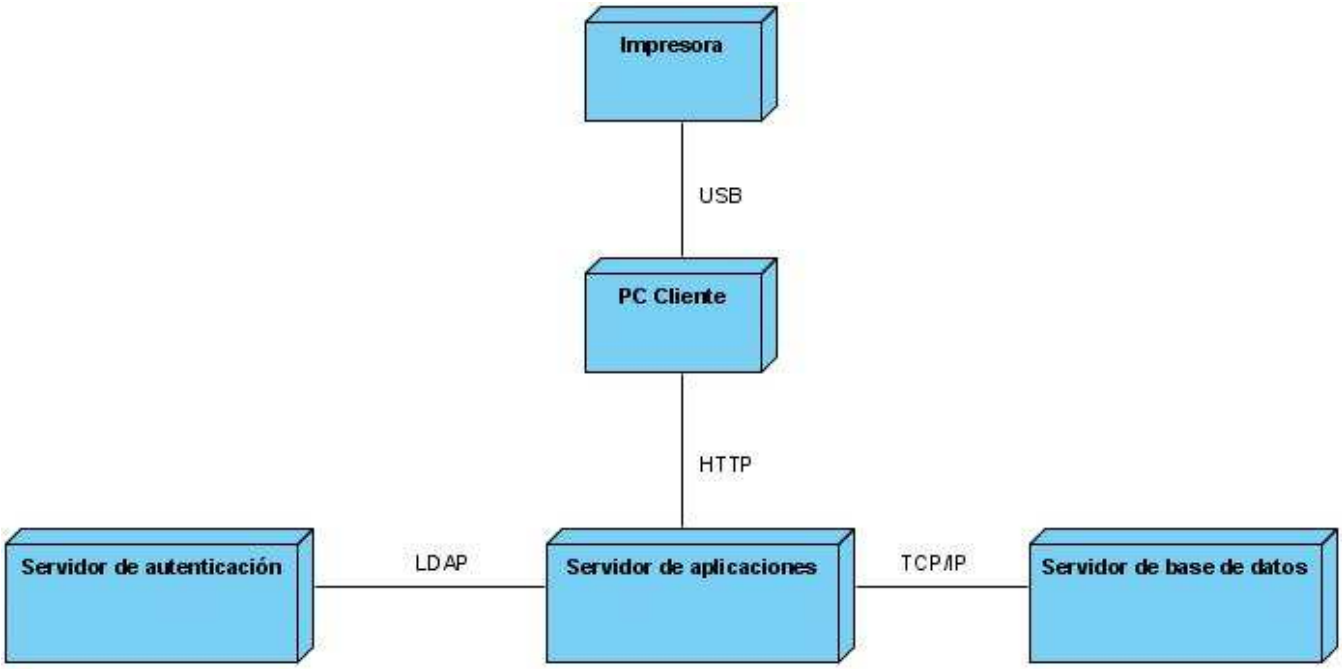

Figura 16 Diagrama de despliegue.

4.4 Prueba del sistema

4.4.1 Casos de prueba

Un caso de prueba especifica una forma de probar el sistema incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse, generalmente lo que se prueba puede venir dado por un requisito o por una colección de ellos cuya implementación justifica una prueba que es posible realizar, siendo muy comunes en este entorno las pruebas de caja negra.

4.4.2 Prueba de caja negra

Una prueba de caja negra verifica el comportamiento observable externamente del sistema, o sea, se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*. Pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Especifica como probar un caso de uso o un escenario específico de un caso de uso. Dentro de las pruebas de caja negra se encuentran las pruebas de sistema, a continuación se muestran el resultado de este tipo de prueba realizada a la aplicación.

Se realizaron pruebas de caja negra basadas en un método de tres pasos para obtener un conjunto de casos de prueba del sistema a partir de los CU, los pasos son los siguientes:

Paso	Descripción	Resultado
1	Generar escenarios de casos de uso	Obtener los posibles caminos de ejecución de cada caso de uso. Cada camino es un escenario de caso de uso.
2	Identificar casos de prueba	Conjunto de casos de prueba a partir de los escenarios anteriores.
3	Identificar los valores a probar	Valores de prueba asociados a cada caso de prueba anterior.

Tabla 4.1 Pasos para obtener los casos de pruebas.

Para generar los casos de prueba primero se generan los posibles escenarios, o caminos de ejecución de cada caso de uso. Después, se identifican los casos de prueba a partir de esos escenarios y por último se identifican los valores a probar de cada caso de prueba.

Lo más importante de los casos de uso (CU) para generar los casos de prueba son los caminos de ejecución. Este camino se divide en dos: en el camino principal o secuencia normal y en el camino alterno. El camino principal son los pasos que sigue el sistema siempre y cuando no surja algún imprevisto, mientras que el camino alterno son las variaciones que van surgiendo durante el camino principal a causa de los errores. Cada uno de estos caminos constituye el escenario de casos de uso.

Un caso de prueba es el conjunto de entradas de datos, las condiciones para la ejecución y los resultados esperados. Esto se hace con el propósito de identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de prueba son necesarios para verificar el éxito de la aplicación y la aceptabilidad del producto.

Para generar los casos de prueba se seguirán los siguientes pasos:

- ✓ Para cada caso de uso (CU), generar un sistema completo de los escenarios.
- ✓ Para cada escenario, identificar los casos de pruebas y las condiciones que hagan que se ejecute.
- ✓ Para cada caso de prueba, identificar los valores de los datos con los cuales se harán las pruebas.

Las celdas de la tabla contienen V o I. V indica válido, I indica inválido.

CP: casos de pruebas

4.4.2.1 Caso de uso: Gestionar eventos UCI

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Registrar evento	Flujo normal	Flujo alterno 1
2	Modificar evento	Flujo normal	Flujo alterno 2
3	Eliminar evento	Flujo normal	
4	Ver detalles	Flujo normal	

Tabla 4.2 Escenarios CU_Gestionar eventos UCI.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema los datos del evento.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Se elimina de la base de datos el evento seleccionado.
CP6	4	V	Se muestran los datos del evento y de la edición.

Tabla 4.3 Casos de pruebas CU_Gestionar eventos UCI.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos del evento correctamente.	Se guarda en la base de datos del sistema los datos del evento.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de	Se actualiza en la base de datos del

		algún evento.	sistema los datos que fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se elimina el evento registrado.	Se elimina de la base de datos el evento seleccionado.
CP6	4	Se muestran los detalles del evento.	Se muestran los datos del evento y de la edición.

Tabla 4.4 Casos de prueba con los valores de los datos.

4.4.2.2 Caso de uso: Gestionar edición.

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Crear edición	Flujo normal	Flujo alterno 1
2	Modificar edición	Flujo normal	Flujo alterno 2
3	Cerrar edición	Flujo normal	

Tabla 4.5 Escenarios CU_Gestionar edición.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema que el evento tiene una edición creada.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Registra que la edición del evento se cierra.

Tabla 4.6 Casos de pruebas CU_Gestionar edición.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos de la edición del evento correctamente.	Se guarda en la base de datos del sistema que el evento tiene una edición creada.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de la edición de algún evento.	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se cierra la edición del evento.	Registra que la edición del evento se cierra.

Tabla 4.7 Casos de prueba con los valores de los datos.

4.4.2.3 Caso de uso: Gestionar convocatoria general

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Abrir convocatoria	Flujo normal	Flujo alterno 1
2	Modificar convocatoria	Flujo normal	Flujo alterno 2
3	Eliminar convocatoria	Flujo normal	

Tabla 4.8 Escenarios CU_Gestionar convocatoria general.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema que el evento tiene una convocatoria abierta.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Se elimina de la base de datos la convocatoria del evento seleccionado.

Tabla 4.9 Casos de pruebas CU_Gestionar convocatoria general.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos de la convocatoria del evento correctamente.	Se guarda en la base de datos del sistema que el evento tiene una convocatoria abierta.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de la convocatoria de algún evento.	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se elimina la convocatoria de un evento.	Se elimina de la base de datos la convocatoria del evento seleccionado.

Tabla 4.10 Casos de prueba con los valores de los datos.

4.4.2.4 Caso de uso: Gestionar inscripción de trabajos

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Subir trabajo	Flujo normal	Flujo alterno 1
2	Modificar datos del trabajo	Flujo normal	Flujo alterno 2
3	Ver trabajo	Flujo normal	

Tabla 4.11 Escenarios CU_Gestionar inscripción de trabajos.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del

			sistema los datos del trabajo.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Se muestran los datos del trabajo.

Tabla 4.12 Casos de pruebas CU_Gestionar inscripción de trabajos.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos del trabajo correctamente.	Se guarda en la base de datos del sistema los datos del evento.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de algún trabajo.	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se muestran los detalles del trabajo.	Se muestran los datos del trabajo.

Tabla 4.13 Casos de prueba con los valores de los datos.

4.4.2.5 Caso de uso: Gestionar eventos externos

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Registrar evento externo	Flujo normal	Flujo alterno 1
2	Modificar datos del evento externo	Flujo normal	Flujo alterno 2
3	Eliminar evento externo	Flujo normal	
4	Ver detalles	Flujo normal	

Tabla 4.14 Escenarios CU_ Gestionar eventos externos.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema los datos del evento externo.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Se elimina de la base de datos el evento externo seleccionado.

Tabla 4.15 Casos de pruebas CU_ Gestionar eventos externos.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos del evento externo correctamente.	Se guarda en la base de datos del sistema los datos del evento externo.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de algún evento externo.	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se elimina el evento externo registrado.	Se elimina de la base de datos el evento externo seleccionado.
CP6	4	Se muestran los detalles del evento.	Se muestran los datos del evento externo y de la edición.

Tabla 4.16 Casos de prueba con los valores de los datos.

4.4.2.6 Caso de uso: Gestionar edición

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Crear edición	Flujo normal	Flujo alterno 1
2	Modificar edición	Flujo normal	Flujo alterno 2
3	Cerrar edición	Flujo normal	

Tabla 4.17 Escenarios CU_Gestionar edición.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema que el evento externo tiene una edición creada.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Registra que la edición del evento externo se cierra.

Tabla 4.18 Casos de pruebas CU_Gestionar edición.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos de la edición del evento externo correctamente.	Se guarda en la base de datos del sistema que el evento externo tiene una edición creada.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de la edición de algún evento	Se actualiza en la base de datos del sistema que los datos fueron

		externo.	cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se cierra la edición del evento externo.	Registra que la edición del evento externo se cierra.

Tabla 4.19 Casos de prueba con los valores de los datos.

4.4.2.7 Caso de uso: Gestionar convocatoria de evento externo

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Abrir convocatoria	Flujo normal	Flujo alterno 1
2	Modificar convocatoria	Flujo normal	Flujo alterno 2
3	Eliminar convocatoria	Flujo normal	

Tabla 4.20 Escenarios CU_Gestionar convocatoria de evento externo.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema que el evento externo tiene una convocatoria abierta.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Se elimina de la base de datos la convocatoria del evento externo seleccionado.

Tabla 4.21 Casos de pruebas CU_Gestionar convocatoria de evento externo.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos de la convocatoria del evento externo correctamente.	Se guarda en la base de datos del sistema que el evento externo tiene una convocatoria abierta.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de la convocatoria de algún evento externo.	Se actualiza en la base de datos del sistema que los datos fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se elimina la convocatoria de un evento externo.	Se elimina de la base de datos la convocatoria del evento externo seleccionado.

Tabla 4.22 Casos de prueba con los valores de los datos.

4.4.2.8 Caso de uso: Gestionar inscripción de trabajos en eventos externos

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Subir trabajo	Flujo normal	Flujo alterno 1
2	Modificar datos del trabajo	Flujo normal	Flujo alterno 2
3	Ver trabajo	Flujo normal	

Tabla 4.23 Escenarios CU_Gestionar inscripción de trabajos en eventos externos.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se guarda en la base de datos del sistema los datos del trabajo.
CP2	1	I	Mensaje de error.
CP3	2	V	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	I	Mensaje de error.
CP5	3	V	Se muestran los datos del trabajo.

Tabla 4.24 Casos de pruebas CU_Gestionar inscripción de trabajos en eventos externos.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se entran los datos del trabajo correctamente.	Se guarda en la base de datos del sistema los datos del evento.
CP2	1	Que algún dato entrado esté incorrecto.	Mensaje de error.
CP3	2	Se cambian los datos de algún trabajo.	Se actualiza en la base de datos del sistema los datos que fueron cambiados.
CP4	2	Los datos que se modifican son incorrectos.	Mensaje de error.
CP5	3	Se muestran los detalles del trabajo.	Se muestran los datos del trabajo.

Tabla 4.25 Casos de prueba con los valores de los datos.

4.5 Conclusiones

En este capítulo queda plasmando cómo está construido el sistema a través del diagrama de componentes de la aplicación en general. También fue mostrado el diagrama de despliegue, el cual ilustra cuáles serán los nodos que serán usados para la implantación de la aplicación. Como parte del flujo de prueba se realizaron las pruebas correspondientes a la solución para verificar que el sistema cumple con las funcionalidades previstas.

CONCLUSIONES

Los procesos de gestión de eventos científicos dentro de la Dirección de Investigaciones (DI) no contaban con una herramienta capaz de automatizar el trabajo, de manera que el primer gran aporte de esta investigación es la creación de un sistema completamente funcional capaz de informatizar el flujo de trabajo de la DI. El desarrollo de esta aplicación se ha basado en el estándar y la arquitectura diseñada por el proyecto Investigaciones de la universidad, de manera que un gran aporte de esta investigación es su integración al grupo de aplicaciones desarrolladas dentro del mismo.

El uso de este sistema informático ahorra tiempo de trabajo a la vez que contribuye a la disminución de errores en los procesos de registro y control de la información vinculada a la gestión de eventos científicos por parte de los trabajadores de la DI, de manera que aumenta la confiabilidad y la eficiencia en el trabajo. Esta herramienta se ha desarrollado como un complemento que aporta valor agregado a una aplicación más grande que abarca todos los procesos de la Dirección de Investigaciones. Sin embargo, la filosofía de creación de la misma permite usar este complemento como una aplicación enteramente funcional por sí sola, y adaptable a cualquier institución de carácter científico, en la cual se gestionen eventos científicos. Durante todo el proceso de creación de esta aplicación se ha generado una gran cantidad de documentación relacionada con los procesos de negocio, implementación y pruebas de la aplicación, de manera que todo esto constituye una fuente de conocimiento importante, que puede servir de material de consulta en futuras investigaciones de carácter científico.

RECOMENDACIONES

- ✓ Añadirle otras funcionalidades que le aporten al sistema valor agregado.
- ✓ Valorar la integración de esta aplicación como un subsistema más del conjunto de aplicaciones que integran el proyecto Sistema de Gestión Universitaria.

BIBLIOFRAFÍA

1. Tecnología y ciencia. *Tecnología y ciencia.* [En línea] [Citado el: 2 de febrero de 2010.] http://www.blogger.com/feeds/9212076838237210203/posts/default.

2. Monografías. *Monografías.* [En línea] [Citado el: 2 de febrero de 2010.] 2. http://www.monografias.com/trabajos-pdf/direccion-integrada-de-proyectos/direccion-integrada-de-proyectos.shtml .

3. Ministerio de la Informática y las Comunicaciones. *XIII Convención y Feria informática 2009.* [En línea] [Citado el: 10 de enero de 2010.] http://www.informaticahabana.com/.

4. Infomed. *Red Telemática de Salud en Cuba.* [En línea] 1999-2000. [Citado el: 9 de diciembre de 2009.] http://www.sld.cu/.

5. **Rojas, Nidia y Hechavarría, Yordanka.** *Sistema para la gestión de los eventos científicos en la Universidad de las Ciencias Informáticas.* 2008-2009.

6. Inducción literatura. *Inducción literatura.* [En línea] 2010. [Citado el: 5 de febrero de 2010.] http://induccionliteratura.wikispaces.com/Aplicaciones+Web+en+la+educacion.

7. Solcre. *Solcre.* [En línea] 2009. [Citado el: 5 de febrero de 2010.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.

8. Anibal de la Torre. *Lenguajes del lado servidor o cliente.* [En línea] 2006. [Citado el: 10 de febrero de 2010.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

9. Taringa. *Inteligencia colectiva.* [En línea] 7 de julio de 2009. [Citado el: 15 de febrero de 2010.] http://www.taringa.net/posts/ebooks-tutoriales/3598122/Ultra-Pack-Para--Aprender-a-Programar-en-Todos-Los-Lenguaje.htm.

10. Maestros del web. *¿Qué es Javascript?* [En línea] 3 de julio de 2007. [Citado el: 15 de febrero de 2010.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript.

11. Miotragus. *Miotragus.* [En línea] [Citado el: 15 de febrero de 2010.] http://foro.miotragus.org/44-terminos/css-hojas-de-estilo-cascada-8300/.

12. Ciberetia. *Ciberetia.* [En línea] [Citado el: 16 de febrero de 2010.] http://www.cibernetia.com/manuales/.

13. Ciberaula. *Ciberaula.* [En línea] 2010. [Citado el: 16 de febrero de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

14. Desarrolloweb.com . *Juan Antonio Breña Moral.* [En línea] [Citado el: 16 de febrero de 2010.] http://www.desarrolloweb.com.

15. PostgreSQL-Administration. *Peter Eisentraut,Bernd Helmle.* [En línea] octubre de 2008. [Citado el: 16 de febrero de 2010.] //www.postgresql-es.org/libros.

Bibliografía

16. **Fabien Potencier.** [En línea] 30 de diciembre de 2008. [Citado el: 17 de febrero de 2010.] // librosweb.es .

17. Libros web. *Libros web.* [En línea] [Citado el: 17 de febrero de 2010.] http://www.librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html.

18. **Rumbaugh, Ivar Jacobson y James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Education S.A, 2000.

19. Wapedia. *Proceso Unificado de Rational.* [En línea] [Citado el: 18 de febrero de 2010.] http://wapedia.mobi/es/RUP.

20. Blog de Eduardo León. *Diseño de Styleshout .* [En línea] [Citado el: 18 de febrero de 2010.] http://slion2000.blogspot.com/2007/04/visual-paradigm-una-herramienta-de-lo.html.

21. Dos Ideas. *Dos Ideas.* [En línea] [Citado el: 18 de febrero de 2010.] http://www.dosideas.com/wiki/NetBeans.

22. *Programación en capas* . [En línea]. [Citado el: 18 de febrero de 2010.] http://*oasis.cisc-ug.org/letzhune/cisc/.../Programacion%20por%20capas.doc*

23. *Entorno virtual de aprendizaje,Ingeniería de Software I. Conferencia # 5.* s.l. : UCI, 2008_2009.

24. **James Rumbaugh, Ivar Jacobson, Grady Booch.** El Lenguaje Unificado de Modelado. *Manual de referencia.* [En línea] 2000. [Citado el: 20 de febrero de 2010.] http://bibliodoc.uci.cu/pdf/reg00060.pdf.

25. *Requerimientos No funcionales. Plantilla Lista de Reserva del producto proyecto Investigaciones.* 2010.

26. Libros web. *Libros web.* [En línea] [Citado el: 10 de marzo de 2010.] http://www.librosweb.es/symfony/capitulo6/seguridad_de_la_accion.html.

27. *Entorno virtual de aprendizaje. Ingenieria de software 2. Conferencia # 1. .* UCI : s.n., 2008-2009.

28. Un Método para el Diseño de la Base de Datos a partir del Modelo Orientado a Objetos. *E-journal.* [En línea] 18 de mayo de 2004. [Citado el: 9 de abril de 2010.] http://www.ejournal.unam.mx/cys/vol07-04/CYS07402.pdf.

29. Symfony-user. *Patrones de diseño que utiliza Symfony.* [En línea] [Citado el: 13 de abril de 2010.] //www.mail-archive.com/symfony-users@googlegroups.com/msg07161.html.

30. *Entorno virtual de aprendizaje. Ingenieria de software 2. Clase práctica para estudiantes .* UCI : s.n., 2008-2009.

GLOSARIO DE TÉRMINOS

Aplicación: Programa informático que proporciona servicios de alto nivel al usuario, generalmente utilizando otros programas más básicos que se sitúan por debajo.

ASP (Página de Servidor Activo) de sus siglas en inglés A*ctive Server Page* es una tecnología de script que corre del lado de servidor y puede ser usado para crear aplicaciones web dinámicas e interactivas.

BSD (Berkeley Software Distribution): traducido al español, Distribución de Software Berkeley, es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

C: Es un lenguaje de programación orientado a la implementación de sistemas operativos, concretamente Unix.

CASE: ***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering* en español Ingeniería de Software Asistida por Ordenador son aplicaciones informáticas para aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

CSS: *Cascading Style Sheets* en español Hojas de Estilo en Cascada es un lenguaje artificial usado para definir la presentación de un documento estructurado escrito en HTML o XML.

DOM: Modelo en Objetos para la representación de Documentos es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

GNU/Linux: Es un sistema operativo tipo Unix (también conocido como Linux que se distribuye bajo la Licencia Pública General de GNU (GNU GPL), es decir que es software libre.

GPL (*General Public License*): Licencia Pública General que permite el uso y modificación del código para desarrollar software libre, pero no propietario.

HTML (*HyperText Markup Language*): siglas en español, Lenguaje de Marcado de Hipertexto, es el lenguaje de marcado predominante para la construcción de páginas Web. Se usa para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTTP (*HyperText Transfer Protocol*)**:** es un protocolo de transferencia de hipertexto es el usado en cada transacción de la web.

Informatización: Aplicación de sistemas y equipos informáticos al tratamiento de información.

Java: Lenguaje de programación orientado a objetos.

Glosario de términos

LAN (Red de Área Local): Una LAN es una red que conecta los ordenadores en un área relativamente pequeña y predeterminada (como una habitación, un edificio, o un conjunto de edificios).

Marco de trabajo: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

MySQL: Sistema de gestión de base de datos relacional (SGBD).

ORM: Mapeo objeto-relacional es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

PERL (Lenguaje Práctico para la Extracción e Informe) es un lenguaje de programación creado por Larry Wall en 1987. PERL toma características del C, del lenguaje interpretado shell (sh), AWK, sed, Lisp y, en un grado inferior, muchos otros lenguajes de programación.

POO: Paradigma de programación que utiliza objetos, estructuras de datos compuestos por datos y métodos, además de sus interacciones para diseñar aplicaciones.

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre.

Unix: Es un sistema operativo portable, multitarea y multiusuario.

WAN (Red de Área Amplia): Es una red de comunicaciones de datos que cubre un área geográfica relativamente amplia y que utiliza a menudo las instalaciones de transmisión proporcionadas por los portadores comunes, tales como compañías del teléfono. Las tecnologías WAN funcionan generalmente en las tres capas más bajas del Modelo de referencia OSI: la capa física, la capa de enlace de datos y la capa de red.

Web: Sistema para presentar información en internet basado en hipertexto.

XML (*Extensible Markup Language*)**:** siglas en español, Lenguaje de Marcas Extensible. Es un formato estándar para el intercambio de datos.

XSLT o **Transformaciones XSL** (lenguaje extensible de hojas de estilo) es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. Las hojas de estilo **XSLT** aunque el término de hojas de estilo no se aplica sobre la función directa del XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla.