Universidad de las Ciencias Informáticas

"Facultad 15"

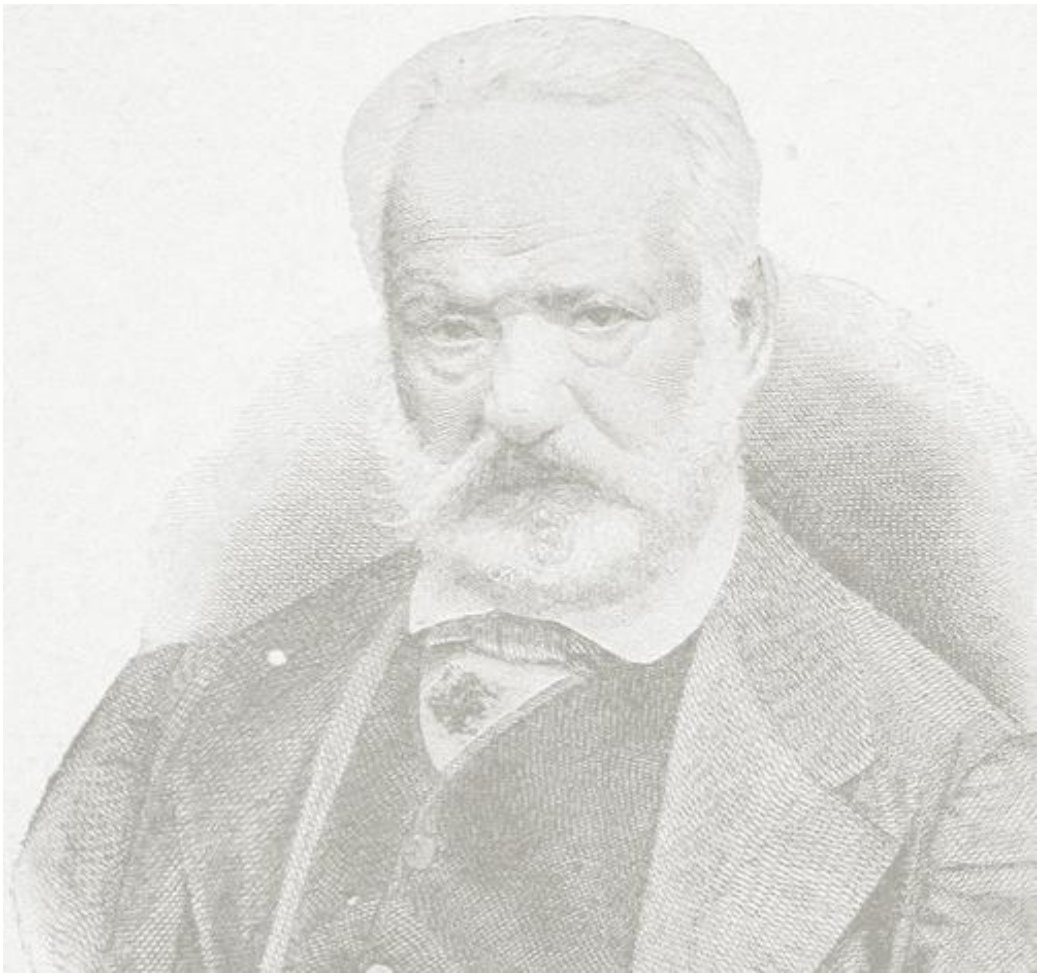
Título: "Implementación del Sistema de Gestión y Control para la evaluación del Sistema de Ciencia e Innovación Tecnológica de la Dirección de Deportes en la UCI"

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

Autores: Mario Luis del Pozo Hernández
Jorge Alejandro Serrano Pacheco
Tutora: Msc. Cecilia Gutiérrez Guerra
Co-Tutor: Ing. Jorge Yuniel Jorrín Perdomo

Ciudad de La Habana, Junio 2010

"Año 52 de la Revolución"



"El futuro tiene muchos nombres. Para los débiles es lo inalcanzable. Para los temerosos, lo desconocido. Para los valientes es la oportunidad"

Víctor Hugo

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los ____ días del mes de Junio del 2010.

Nombre completo del primer autor	Nombre completo del segundo autor
Firma de la Tutora	Firma del Co-Tutor

DATOS DE CONTACTO

Tutora: Msc. Cecilia Gutiérrez Guerra.
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Email: Cecilia@uci.cu.

Co-Tutor: Ing. Jorge Yuniel Jorrín Perdomo.
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Email: jyjorrin@uci.cu.

DEDICATORIA

Mario Luis del Pozo Hernández

Dedico este trabajo a mis padres Clara Julia Hernández Roja y Mario Benito del Pozo Villegas por ser un gran ejemplo de respeto, responsabilidad y amor, por educarme sin descanso, por su preocupación constante, por darme todo a cambio de nada, gracias por existir.

A mi hermano Julio Guillermo por ser lo más hermoso que tengo en la vida. Espero que sigas mi ejemplo y te hagas una persona de bien.

A mi abuelo Luis Maria Hernández Espinosa por estar siempre ahí, por brindarme su apoyo, su amor, su comprensión y sobre todo por ser mi guía...mi ejemplo a seguir.

A mi abuela Buenviaje Roja Duquernez y a mi abuelo Mario Ramón Del Pozo Martínez que aunque hoy no están conmigo fueron fuente de amor, sabiduría y respeto, los llevo siempre presentes.

A mi tío Luis Hernández Roja y a mi tía Maria del Pilar Del Pozo Villegas por ayudarme y apoyarme, ustedes representan un papel importante en mi vida.

A mi abuela Esther Maria Villegas García, a mis hermanos Mario Igor Del Pozo Komaskova, Marek Del Pozo Komaskova y Milada Esther Del Pozo Komaskova, por su atención conmigo, por estar siempre presentes, por brindarme toda su fuerza, amor y apoyo.

A Teresa Hernández Espinosa y Juanito Hernández, gracias por siempre estar ahí, por su incondicional apoyo, por transmitirme siempre mucho afecto y cariño, con ustedes pasé la mayor parte de mi infancia, gracias por haberla hecho tan especial.

A mis primos y a toda mi familia en general por su apoyo y preocupación a los largo de estos cinco años.

A todas mis amistades en la UCI y fuera de ella que siempre me han brindado su afecto, por estar en los buenos y malos momentos.

A mis hermanos del WOW y el Dota que fueron cómplices de tantas noches de desvelo, no caben en este espacio tan reducido pero si en mi corazón, con ustedes pasé los mejores momentos en la UCI, gracias por estar presentes, son parte de una etapa de mi vida que hoy termina para dar comienzo a otra, nunca los olvidaré.

Hay muchas personas que se involucran en la vida de uno, algunas quedan, otros simplemente pasan. Gracias a las que pasaron, igual fueron buenas en su momento y me ayudaron a superarme como persona. Pero, a las que están, mil gracias, esos son mis amigos, los verdaderos, esto también es de ustedes.

A todos, gracias.

Jorge Alejandro Serrano Pacheco

A mis padres por tanto amor y confianza depositados en mí, por enseñarme a luchar por mis sueños por apoyarme en todo momento de mi vida, sin su apoyo, educación y guía no estaría donde estoy. ¡Mami, papi los Quiero Mucho!

A mi hermana Laura, la cual quiero con todas las fuerzas de mi corazón, por ser una persona tan buena y tan especial conmigo.

A mis abuelos por ser tan atentos, tan preocupados, tan llenos de amor y dedicación, por darme fuerza cuando más la he necesitado.

A mis tíos por siempre apoyarme en todo y por el amor y cariño que siempre me han dado.

A toda mi familia en general por brindarme toda su fuerza, amor y apoyo.

A todos mis amigos por estar siempre presente, por compartir buenos y malos momentos, nunca los olvidare.

A todos los que hicieron posible este sueño que ya hoy es realidad.

Muchas Gracias.

AGRADECIMIENTOS

Agradecemos a nuestros padres por ser nuestra máxima fuente de inspiración, por darnos la vida y todo su amor, por formarnos como la persona que hoy somos.

A nuestros lindos abuelos…hermanos, familiares y amigos.

A nuestra tutora Cecilia Gutiérrez Guerra y al co-tutor Jorge Yuniel Jorrín Perdomo por su asesoría constante y dedicación durante todo este tiempo.

A todos aquellos que de una forma u otra estuvieron presentes en la realización de este trabajo.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI), en su estructura organizativa, cuenta con una Dirección de Deportes, que tiene dentro de sus funciones la gestión de la actividad científica.

En la actualidad se cuenta con dos aplicaciones para calcular la actividad científica, SIndiCIT y SysMo cada una de ellas tiene características particulares y no cumplen totalmente con las exigencias de la Dirección de Deportes. Luego de un estudio realizado a los mecanismos de procesamiento y almacenamiento de la información, se identificaron los procesos fundamentales que se llevan a cabo en dicho lugar. Con el objetivo de informatizar el manejo de esta información y contribuir al control de la misma se está desarrollando un sistema de gestión y control para la evaluación del Sistema de Ciencia e Innovación Tecnológica de la Dirección de Deportes en la UCI.

Este trabajo realiza el análisis, diseño e implementación del sistema de gestión y control para la evaluación del Sistema de Ciencia e Innovación Tecnológica de la Dirección de Deportes en la UCI

Se identifican las funcionalidades y características que deberá cumplir el sistema a desarrollar, se diseñan las clases y se implementan los componentes.

ÍNDICE

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) provocan continuas transformaciones en las estructuras económicas, sociales y culturales, incidiendo en casi todos los aspectos de la vida.

Debido al impacto que ha tenido la implantación y utilización de las TIC en todo el mundo, cada vez se torna más difícil actuar eficientemente prescindiendo de ellas.

En los últimos años, nuestro país se ha emprendido el reto de la informatización de la sociedad, este proyecto se ha realizado de manera acelerada auspiciado por la dirección del país y alcanzando resultados satisfactorios en áreas tan esenciales como la educación, el deporte, la salud y la investigación.

Los efectos de esta tarea han llegado a centros importantes del país, entre ellos a La Universidad de las Ciencias Informáticas (UCI), nacida como un proyecto de la Revolución Cubana con la misión de formar a especialistas de primer nivel en ciencias informáticas. Este centro, además de generar productos y servicios informáticos, así como soluciones tecnológicas integrales para la economía nacional y la exportación a través de un amplio grupo de programas, también cubre sus propias necesidades tecnológicas.

La Universidad de las Ciencias Informáticas (UCI), en su estructura organizativa, cuenta con una Dirección de Deportes, que tiene dentro de sus funciones la gestión de la actividad científica.

En sus objetivos estratégicos para la organización del Sistema de Ciencia e Innovación Tecnológica, está propuesta la Línea de investigación relacionada con el uso de las Tecnologías de la Informática y las Comunicaciones en los Proyectos de Deportes y todo lo relacionado con la Cultura Física.

En la actualidad se cuenta con dos aplicaciones para calcular la actividad científica, cada una de ellas tiene características particulares y no cumplen totalmente con las exigencias de la Dirección de Deportes, primeramente el sistema SIndiCIT es capaz de calcular los indicadores que son necesarios para realizar la actividad científica en dicha entidad pero sin embargo no permite hacer un seguimiento personal de los eventos en los que ha participado cada trabajador de la institución, por lo que para realizar este proceso hay que realizar una búsqueda engorrosa en cada una de las copias legales archivadas, lo que conlleva a una pérdida considerable de tiempo. Desde el punto de vista de los resultados SIndiCIT es capaz de mostrar también los resultados de una entidad pero no puede hacer referencia a la persona que los obtuvo aspecto importantísimo para la Dirección de Deportes actualmente; el segundo software SysMo no se utiliza debido a que al igual que el anterior no permite hacer un seguimiento personal de los eventos y además los indicadores de medición de la actividad de

ciencia con los que se implementó la aplicación anterior fueron modificados y uno fue eliminado. Luego de un estudio realizado a los mecanismos de procesamiento y almacenamiento de la información, se identifica como problema científico a resolver ¿Cómo contribuir a la gestión, control y almacenamiento de la información para la evaluación de la actividad científica del Sistema de Ciencia e Innovación Tecnológica de la Dirección de Deportes en la UCI?

El problema planteado tiene como **objeto de estudio**: Proceso de desarrollo de los Sistemas de Gestión de la Información para la evaluación científica.

El objeto delimita el **campo de acción**: Análisis, Diseño e Implementación de los Sistemas de Gestión de la Información para la evaluación científica.

En la búsqueda de la solución al problema planteado el **objetivo** que se desea alcanzar es: Desarrollar el Análisis, Diseño e Implementación de un sistema de gestión y control para la evaluación del Sistema de Ciencia e Innovación Tecnológica de la Dirección de Deportes en la UCI.

Resultando como **objetivos específicos**:

- Analizar las aplicaciones similares existentes.
- Seleccionar la metodología y las herramientas óptimas para desarrollar el sistema.
- Modelar el negocio.
- Obtener las funcionalidades y las características del sistema.
- Modelar el sistema a desarrollar.
- Implementar el sistema a desarrollar.
- Probar el sistema implementado.

El trabajo consta de Introducción, 4 Capítulos, Bibliografía y Anexos.

En el Capítulo 1 Fundamentación Teórica: en este capítulo se se abordarán temas que fueron necesarios estudiar para el desarrollo de la investigación, como fueron el estudio del arte sobre sistemas que existen en el mundo y en la UCI en busca de mejoras para el sistema en construcción. Además se aborda sobre las principales características del Proceso Unificado del Rational (RUP), los roles a desempeñar, los artefactos y las herramientas a utilizar.

En el Capítulo 2 Características del Sistema: se explica detalladamente cuáles son los objetivos estratégicos de la Dirección de Deportes de la Universidad de las Ciencias Informáticas y se realiza la modelación del negocio, para comprender cuáles son los procesos que involucran la mayor cantidad

de actividades a automatizar. Se brindan las descripciones de los casos de uso, actores y trabajadores de negocio, así como el diagrama de casos de uso de negocio, el diagrama de actividades y el modelo de objetos. Se identifican requerimientos funcionales y no funcionales, los actores del sistema, casos de uso del sistema, se realizan los diagramas de casos de uso del sistema y se describen los casos de uso del sistema.

En el Capítulo 3 Diseño del Sistema: se representan los diagramas de las clases, los diagramas de interacción y se hace referencia a la arquitectura del sistema a través de la Vista Lógica y de Despliegue.

En el Capítulo 4 Implementación y prueba: en este capítulo se estructura el modelo de implementación, se realiza el diagrama de componentes, se implementa el sistema y se describen en detalle las pruebas que se le han realizado al sistema.

Capítulo 1

Fundamentación Teórica

Introducción

En este capítulo, para lograr una mejor comprensión de la necesidad de este trabajo se aborda acerca de los Sistemas de Gestión de Información de forma general, brinda una descripción general sobre el comportamiento de aplicaciones informáticas similares a nivel nacional, además se realiza una descripción de los procesos de desarrollo de software más conocidos, de la metodología, herramientas y los patrones a utilizar.

1.1. Gestión de la Información

“En la era de la información, de la explosión de sus tecnologías, se vive la etapa en la que la humanidad ha alcanzado un desarrollo imprevisible; cada día son mayores las diferencias sociales, políticas y económicas. Se habla constantemente sobre la sociedad de la información, es visible el paso de las sociedades industriales a las posindustriales y del conocimiento, donde el factor esencial de progreso es el conocimiento. Esta nueva sociedad, con organizaciones basadas en el aprendizaje, cuyo capital más preciado es el ser humano, se sustenta en un desarrollo tecnológico sin precedentes, es el punto en el cual las grandes compañías planifican sus productos en función de la gestión de la información y de la viabilidad para su obtención” (Quiroga, 2002).

“La gestión de la información es el proceso de analizar y utilizar la información que se ha recabado y registrado para permitir a los administradores (de todos los niveles) tomar decisiones documentadas" (2009).

“Algunas de las funciones de la Gestión de Información:

- Determinar necesidades internas de información, relativas a las funciones, actividades

y procesos administrativos de la organización y a su satisfacción.

- Optimizar el flujo organizacional de la información y el nivel de la comunicación.
- Manejar eficientemente los recursos organizacionales de información.
- Entrenar a los miembros de la organización en el manejo o la utilización de los recursos informacionales.

- Contribuir a modernizar u optimizar las actividades organizativas y los procesos administrativos relacionados con los mismos.
- Garantizar la calidad de los productos de la organización y asegurar su diseminación efectiva.
- Determinar las necesidades de información externa de la organización y satisfacerlas" (Amador, 2007).

1.1.1. Sistemas de Gestión de la Información

Los sistemas de gestión de Información constituyen hoy una alternativa de imprescindible presencia en cada organización. Son posiblemente la herramienta integral de gerencia más cotizada y necesaria para alcanzar con éxito los resultados propuestos por una organización.

"Un Sistema de Información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.

Entrada de Información: Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información.

Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, porque a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos.

Procesamiento de Información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones.

Salida de Información: La salida es la capacidad de un Sistema de Información para sacar al exterior la información procesada o bien datos de entrada. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información" (Briceño, 2005)

1.1.2. Sistemas similares a nivel internacional

En las modernas economías, los objetivos prioritarios de la política científica de los países son el desarrollo de la ciencia, la tecnología y la innovación. Para poder planificar, ejecutar y evaluar la

actividad científica, se requiere necesariamente un trabajo estadístico previo de toma de datos básicos y posterior análisis de los mismos, para llegar a construir indicadores para evaluar dicha actividad, estos datos e indicadores se adaptan para cada país en específico.

"El **SNCTI** es el Sistema Nacional de Ciencia, Tecnología e Innovación de Venezuela, es un sistema abierto del cual forman parte las políticas, estrategias, programas, metodologías y mecanismos para la gestión, promoción, financiación, protección y divulgación de la investigación científica y la innovación tecnológica, así como las organizaciones públicas, privadas o mixtas que realicen o promuevan el desarrollo de actividades científicas, tecnológicas y de innovación" (Arias, 2009).

"Una revisión de los principales aspectos que conforman el SNCTI venezolano debe incluir: condiciones macroeconómicas, marco legal y de incentivos, organismos públicos que administran y coordinan a las instituciones de ciencia y tecnología, capacidades en comunicaciones y tecnologías de información y en investigación y desarrollo, instituciones de formación de personal, principalmente universidades y postgrados, vinculaciones entre las instituciones académicas y científicas con el sector empresarial, servicios tecnológicos (metrología, normalización, información y asistencia técnica), régimen de protección a la propiedad industrial, sistemas de financiamiento y caracterización del sector empresarial" (Marianela Lafuente, 2004).

SNCTI no se puede utilizar debido a que los indicadores que utiliza para medir la evaluación científica no coinciden con los indicadores que se emplean en la Dirección de Deportes de la UCI. Los indicadores empleados en SNCTI responden a las necesidades localmente identificadas.

"COLCIENCIAS es el Departamento Administrativo de Ciencia, Tecnología e Innovación en Colombia y crea el Sistema Nacional de Ciencia, Tecnología e Innovación. COLCIENCIAS es el rector que formula, orienta, dirige, coordina, ejecuta e implementa la política del Estado en el sector de ciencia, tecnología e innovación. Busca llevar al país a niveles más elevados de competitividad mediante el desarrollo tecnológico innovador y la integración de actividades de los diversos sectores para impulsar áreas del conocimiento estratégicas de clase mundial. COLCIENCIAS es una organización líder en generación de políticas y capacidades que permiten incorporar la Ciencia la Tecnología y la Innovación, a la cultura del país y convierten el conocimiento en motor de desarrollo local, regional y nacional" (Dadak, 2010).

COLCIENCIAS no puede ser utilizado porque al igual que SNCTI los indicadores y él índice de ponderación que se le asigna a cada uno de ellos para medir la evaluación científica no coinciden con los empleados en la Dirección de Deportes de la UCI.

“ANEP (Agencia Nacional de Evaluación y Prospectiva), uno de los países que más ha trabajado en el establecimiento de criterios de calidad en la evaluación científica es España, donde a través de la Fundación Española para la Ciencia y la Tecnología y la Agencia Nacional de Evaluación y Prospectiva, constituyó un grupo de trabajo con el objetivo de establecer criterios de valoración que incentivaran a los investigadores a mejorar la calidad y visibilidad de sus contribuciones científicas. Dichos criterios fueron elaborados a partir del consenso de un conjunto de investigadores españoles y extranjeros, confeccionando una serie de informes a partir de los comentarios y sugerencias aportados por otros científicos consultados y por entidades relevantes para la evaluación” (Piñeiro, 2009). Se utilizan principalmente indicadores de tipo cuantitativo. Para la ciencia y la tecnología, se mide la eficiencia de los recursos humanos y financieros otorgados en la investigación científica y tecnología a nivel macro.

El sistema científico español utiliza los "indicadores bibliométricos", que son datos estadísticos basados en el análisis de las publicaciones científicas, y sirven para evaluar la ciencia y a los científicos. Estos indicadores no coinciden con los que necesita la Dirección de Deportes, por lo que se hace imposible la utilización de este sistema.

Luego de un estudio realizado sobre los Sistemas de Gestión de la Información existentes a nivel internacional se llegó a la conclusión de que no existe ninguno que cumpla con las características que requiere la Dirección de Deportes de la UCI.

El sistema debe estar básicamente guiado por un conjunto de indicadores generales, que junto con un grupo de fórmulas permiten medir la actividad científica en dicha dirección.

A continuación se muestran los indicadores generales de medición de la actividad de ciencia y técnica, necesarios para la implementación del sistema.

- Premios obtenidos.
- Publicaciones científicas.
- Patentes y registros.
- Participación en proyectos.
- Resultados introducidos.
- Trabajos presentados en eventos.
- Capacitación ofertada o recibida.

- Uso de estudiantes.

1.1.3. Sistemas similares a nivel nacional

La empresa moderna requiere un Sistema de Gestión y Control que le sirva para la evaluación del Sistema de Ciencia e Innovación Tecnológica que desarrolla, este sistema debe servir como herramienta organizativa a los equipos de dirección.

En Cuba, específicamente en la Universidad de las Ciencias Informáticas (UCI) se cuenta con dos aplicaciones (SIndiCIT, SysMo) para calcular la actividad científica, cada una de ellas tiene características particulares y no cumplen totalmente con las exigencias de la Dirección de Deportes.

SIndiCIT brinda la posibilidad de llevar un control de los indicadores relacionados con la ciencia, tecnología e innovación que se realiza en la Universidad de las Ciencias Informáticas, aunque también puede ser utilizado con el mismo propósito en otros centros. Es un sistema de indicadores que permite evaluar la producción científica de los profesores, investigadores y estudiantes de la Universidad, que potencia los resultados científicos y de innovación, que premia el trabajo en equipo y se adapta a las características de la Universidad. Este sistema contribuye a ponderar de manera diferenciada aquellas investigaciones científicas de ciclo completo (esto es, Investigación + Desarrollo + Producción +Comercialización) con relación a aquellas investigaciones puramente académicas.

Deficiencia: no permite hacer un seguimiento personal de los eventos en los que ha participado cada trabajador de la institución. Desde el punto de vista de los resultados SIndiCIT es capaz de mostrar también los resultados de una entidad pero no puede hacer referencia a la persona que los obtuvo aspecto importantísimo para la Dirección de Deportes actualmente.

SysMo permite evaluar la actividad científica para la Dirección de Deportes de la UCI, esta aplicación brinda la posibilidad de calcular una serie de indicadores y parámetros preestablecidos, los cuales pueden dar una idea de cómo se encuentra la Dirección de Deportes y a partir del resultado de los cálculos de dichos indicadores poder trazar las nuevas líneas de trabajo a seguir con el fin de mejorar la actividad científica en la entidad.

Deficiencia: SysMo no se utiliza debido a que no permite hacer un seguimiento personal de los eventos en los que ha participado cada trabajador de la institución y además los indicadores de medición de la actividad de ciencia con los que se implementó la aplicación fueron modificados y uno fue eliminado.

1.2. Proceso de Desarrollo de Software

“El proceso de desarrollo de software es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo. Concretamente define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Zavala-Ruiz, 2008).

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. Entre los procesos de desarrollo más conocidos se encuentran:

- Programación Extrema (Extreme Programming, XP).
- Proceso Unificado de Desarrollo (Rational Unified Process, RUP).

1.2.1. Programación Extrema (Extreme Programming, XP)

“La programación extrema es una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas de buenas maneras que tiene como objetivo aumentar la productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación“ (Daniel Gomez, 2008). XP, está basado específicamente en el trabajo orientado al objetivo, basándose para esto en las relaciones interpersonales y en la velocidad de reacción para la implementación y para los cambios que puedan surgir durante el desarrollo del proceso. XP es el proceso de desarrollo de software más exitoso en la actualidad, utilizado para proyectos a corto plazo y equipos de trabajo reducido. Las principales características de XP son:

- Pruebas Unitarias: se basa en las pruebas realizadas a los procesos principales, de manera tal que adelantándose en algo hacia el futuro se puedan hacer pruebas de las fallas que pudieran suceder.
- Refabricación: consiste en la reutilización de código, para lo cual se crean patrones o modelos estándares permitiendo ser más flexible al cambio.
- Programación en pares: consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo con el propósito de lograr un mismo objetivo.

En este proceso de desarrollo es fundamental lograr la comunicación entre los usuarios y los desarrolladores, la simplicidad al desarrollar, codificar los módulos del sistema y la retroalimentación del equipo de desarrollo, el cliente y los usuarios finales.

1.2.2. Proceso Unificado de Desarrollo (Rational Unified Process, RUP)

Este es uno de los procesos más generales que existe, está enfocado a cualquier tipo de proyecto así no sea de software, se basa en la documentación generada en cada una de sus fases. "Es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable" (Angela, 2010). "Este proceso de desarrollo de software junto con el lenguaje unificado de modelado constituyen la metodología estándar más utilizada para el análisis, implementación y documentación de los sistemas orientados a objetos" (Cruz, 2007). La definición de este proceso está dada por tres características fundamentales:

- Dirigido por casos de uso: el proceso de desarrollo sigue una trayectoria a través de flujos de trabajos generados por casos de uso, que describen la funcionalidad del sistema, en términos de su importancia para el usuario.
- Centrado en la arquitectura: describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. Documenta de la mejor manera, basándose en UML (Unified Modeling Language - Lenguaje de Modelado Unificado).
- Iterativo e incremental: se encarga de dividir el trabajo en partes más pequeñas o en mini proyectos, permitiendo el equilibrio entre casos de uso y arquitectura durante cada mini proyecto. Cada mini proyecto se puede ver como una iteración de la cual se obtiene un incremento, provocando un aumento del producto. **(Ver Anexo 1)**

RUP incluye cuatro fases, donde una fase es el intervalo de tiempo entre dos hitos importantes del proceso durante el que se cumple un conjunto bien definido de objetivos, se completan artefactos y se toman decisiones sobre si pasar o no a la siguiente fase. Las cuatro fases de RUP son las siguientes: **(Ver Anexo 2)**

- Inicio: el objetivo es determinar el alcance del proyecto y entender qué es lo que se va a construir.
- Elaboración: el objetivo es determinar la arquitectura óptima, definir, analizar y diseñar el proyecto.

- Construcción: el objetivo es llegar a obtener la capacidad operacional inicial, implementar.
- Transición: el objetivo es garantizar que el usuario aprenda a trabajar y mantener el sistema.

En cada fase del ciclo de vida del proyecto se ejecutarán una o varias iteraciones, y dentro de cada una de ellas, se seguirá, un modelo de cascada en los flujos de trabajo que lo requieran. RUP es un proceso, en su modelación define como sus principales elementos:

- Trabajadores (“quién”): define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades (“cómo”): es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos (”qué”): productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Flujo de actividades (“cuándo”): secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

En RUP se han agrupado las actividades en grupos lógicos definiéndose nuevos flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

- Modelamiento del negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

- Prueba (Testeo): busca los defectos a los largo del ciclo de vida.
- Instalación o despliegue: produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- Administración del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- Ambiente: contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Analizando lo referente a estas tecnologías y determinando el alcance de este software, la metodología que se ha escogido es RUP. Esta metodología aplica varias de las mejores prácticas en el desarrollo moderno de software y cuenta además con diferentes elementos de planificación para el control del desarrollo de software y gestión de riesgos permitiendo prevenir y corregir problemas y fallos conocidos. Está respaldada por buenos resultados en proyectos en el mundo y en la UCI. Otra de las razones por la que ha sido escogida es que, provee un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software. Permite además que todos los integrantes de un equipo de trabajo, conozcan y compartan el proceso de desarrollo utilizando un lenguaje de modelado común: UML. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. Con ella se asegura la producción de software de alta calidad, que cumpla las necesidades del usuario final. Además las metodologías ágiles dan mayor valor al individuo, a la colaboración constante con el cliente, algo que se dificultaba en gran medida en la realización de este trabajo ya que el cliente no podía ser parte del equipo de desarrollo.

En el presente trabajo se realizarán fundamentalmente los siguientes flujos de trabajo: Modelamiento del negocio, Requerimientos, Análisis y Diseño, Implementación y Prueba.

1.3. Lenguaje de Modelado

Una exigencia de la gran mayoría de instituciones, es que los desarrollos de software se formalicen con un lenguaje estándar y unificado.

Es decir, se requiere que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con un lenguaje común.
"Los lenguajes de modelado utilizan modelos visuales y diagramas que realizan esa representación de manera precisa, entendible y económica, lo que facilita su uso en las herramientas CASE convencionales" (Zapata, 2009).

1.3.1. Lenguaje de Modelado: BPMN

"BPMN (Business Process Modeling Notación, Notación de Modelado de Procesos de Negocio) es el estándar más reciente para modelado de procesos del negocio y servicios web. BPMN es una notación necesaria para expresar los procesos de negocio en un único diagrama de proceso de negocio (Business Process Diagram – BPD) BPMN permite hacer un mejor uso de la gestión de procesos del negocio (BPM), ya que normaliza el método de notación que sirve como ayuda en la automatización de los procesos" (Fajardo, 2008).

BPMN está dirigido a gerentes, directores, dueños de empresas, ingenieros de procesos, analistas de negocios, analistas de sistemas, administradores de proyectos, responsables de calidad y todo aquel que necesita definir, documentar y hacer más eficientes sus procesos de negocio con el estándar más avanzado y aceptado a nivel internacional.

UML (El lenguaje de modelado unificado) toma un perfil orientado a objetos en el modelado de aplicaciones, mientras que BPMN toma un perfil orientado a procesos en el modelado de sistemas. BPMN tiene un enfoque en procesos de negocio, UML se enfoca al diseño de software y por lo tanto ambas notaciones son totalmente compatibles entre sí.

1.3.2. Lenguaje Unificado de Modelado (UML)

"UML (por sus siglas en inglés, Unified Modeling Language) lenguaje de modelado unificado, es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra cierta cantidad de software, permite la modelación de sistemas con tecnología orientada a objetos y recomienda la utilización de diagramas para representar las distintas vistas de un sistema. Los diagramas de UML son los siguientes" (Ramos, 2008):

- Diagrama de Clases: muestra las clases (descripciones de objetos que comparten características comunes) que componen el sistema y cómo se relacionan entre sí.
- Diagrama de Objetos: muestra una serie de objetos (instancias de las clases) y sus relaciones.

- Diagrama de Casos de Uso: modela la funcionalidad del sistema agrupándola en descripciones de acciones ejecutadas por un sistema para obtener un resultado.
- Diagrama de Estados: modela el comportamiento de acuerdo con eventos.
- Diagrama de Secuencia: enfatiza la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos.
- Diagrama de Actividades: simplifica el diagrama de estados modelando el comportamiento mediante flujos de actividades.
- Diagrama de Colaboración: muestra la interacción entre los objetos resaltando la organización estructural de los objetos en lugar del orden de los mensajes intercambiados.
- Diagrama de Distribución: muestra la arquitectura física de un sistema informático, pudiendo representar los equipos y dispositivos, mostrar sus interconexiones y el software que estaría presente en cada máquina.

Luego del estudio realizado se decide escoger a UML como lenguaje de modelado, por sus características y beneficios, además hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

Este lenguaje se ha convertido en la actualidad en uno de los más utilizados por lo expresivo, claro y uniforme que resulta para el diseño orientado a objeto y además por permitir una fuerte integración entre las herramientas, los procesos y los dominios. Desde finales de 1997 es un lenguaje de modelado orientado a objetos estándar. Posibilita el intercambio de modelos entre las distintas herramientas CASE. UML mejora el tiempo total de desarrollo, además presenta una alta reutilización y minimización de costos.

1.4. Herramientas CASE

En las últimas décadas se ha trabajado en el área de desarrollo de sistemas para encontrar técnicas que permitan incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software. CASE es una sigla que corresponde a las iniciales de: Computer Aided Software Engineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación.

“Las Herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y

de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar el diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación y detección de errores" (Miranda, 2010).

Entre las herramientas CASE para el modelado de los artefactos se encuentran Rational Rose Enterprise y Visual Paradigm. A continuación se verán algunas de las características más importantes de estas herramientas que ayudarán en la selección de la misma para el desarrollo de la aplicación que se desea desarrollar.

1.4.1. Rational Rose

Es una de las más poderosas herramientas de modelado visual basado en UML para el análisis y diseño de sistemas basados en objetos. Es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas del Rational Rose es que utiliza la notación estándar en la arquitectura de Software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común. Los diseñadores pueden modelar sus componentes e interfaces de forma individual y luego unirlos con otros componentes del proyecto, ayuda a los desarrolladores de software a construir mejores productos en menor tiempo, da un excelente soporte en el manejo de cambios durante el ciclo de vida del proyecto y mejora la comunicación entre los miembros del equipo.

Rational Rose presenta desventajas, una es que necesita de mucha memoria para poder de alguna forma ser manejado de forma rápida y eficiente, además el costo de sus licencias es elevado.

1.4.2. Visual Paradigm

"Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML" (Jams, 2007).

Principales características de Visual Paradigm:

- Entorno de creación de diagramas para UML.

- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.
- Generación de bases de datos. Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Importación y exportación de ficheros XMI.

Analizando lo referente a estas herramientas, para un mayor nivel de conocimiento se escoge como herramienta de modelado Visual Paradigm, que sustenta el lenguaje de modelado UML con una amplia documentación y todos sus diagramas de diseño. Además de ser la herramienta aplicada por el movimiento de software libre. Es considerada la herramienta Case de preferencia para el desarrollo de este trabajo pues se integra con muchas bases de datos entre las que se encuentra PostgreSQL, lenguajes de programación como PHP y varios IDEs, entre ellos NetBeans. Entre las ventajas de esta herramienta se encuentra que es portable y muy rápida en su tiempo de ejecución, no es gratuita pero la UCI cuenta con una licencia para su uso. Visual Paradigm también proporciona características tales como generación de código, ingeniería inversa y generación de informes. Tiene la capacidad de crear el esquema de clases a partir de una base de datos, y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente. Además cubre todos los flujos de trabajo que se requieren en el desarrollo de este sistema.

1.5. Herramientas IDE (Integrated Development Environment)

“Los IDEs (Integrated Development Environment) o entorno integrado de desarrollo son un conjunto de herramientas para el programador, que suelen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores y debuggers e integración con sistemas controladores de versiones o repositorios” (Luciano, 2009).

Dentro de las herramientas utilizadas para el desarrollo de aplicaciones web en PHP se encuentran Eclipse y NetBeans, en el presente trabajo se hace un resumen de las características fundamentales de estas herramientas para un mayor nivel de conocimiento.

1.5.1. Eclipse

Eclipse es un IDE (Integrated Development Environment, entorno integrado de desarrollo) principalmente para Java, muy potente. Es libre, de código abierto multiplataforma y fue creado originalmente por IBM. Eclipse no es tan sólo un IDE, se trata de un marco de trabajo modular ampliable mediante complementos (plugins). Eclipse permite la realización tanto de aplicaciones web como de aplicaciones de escritorios, existen complementos que nos permiten usar Eclipse para programar en PHP, Perl, java, Python y C/C++.

1.5.2. NetBeans 6.0 como entorno integrado de desarrollo (IDE)

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. NetBeans es un producto libre y gratuito sin restricciones de uso.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Estos a su vez pueden ser desarrollados independientemente, razón por la que las aplicaciones basadas en NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software

"¿Que se puede hacer con NetBeans?

- Desarrollar aplicaciones:

✓ De escritorio
✓ **Web**
✓ Mobile
✓ Enterprise

- Con:

✓ Java
✓ C/C++
✓ Ruby on Rails
✓ **PHP**, Groovy, Python, **Javascript**" (Cerda, 2009).

Luego del estudio realizado referente a las características fundamentales de estas dos herramientas se escoge como entorno integrado de desarrollo NetBeans por todas las características antes mencionadas. Además NetBeans trae muchos plugins configurados lo cual es muy útil y el depurado de las aplicaciones es más sencillo. Con NetBeans no solo es posible elaborar potentes aplicaciones de escritorio, también aplicaciones Web y para dispositivos portátiles, como móviles o Pocket PC, sin que cambie la forma de programar. A la hora de debuguear es muy sencillo y se puede ver todas las variables del programa en tiempo de ejecución. Los errores son bastante descriptivos, ayuda mucho las indicaciones que da.

1.6. Lenguaje de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras.

1.6.1. Lenguaje de programación: Java

"Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java. Con Java se puede programar páginas web dinámicas, con accesos a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que se desee hacer con acceso a través web se puede hacer utilizando Java" (Alvarez, 2001).

1.6.2. Lenguaje de programación: PHP

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

"Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente.

Este lenguaje fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red.
Algunas de las más importantes capacidades de PHP son:

- Compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, ODBC y PostgreSQL.
- Incluye funciones para el envío de correo electrónico, upload de archivos.
- Crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.
- Ofrece la integración con las varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en PDF hasta analizar código XML" (Alvarez, 2001).

Se realiza un estudio de las características fundamentales de estos dos lenguajes de programación para un mayor nivel de conocimiento y se escoge PHP para el desarrollo de la aplicación por todas las características antes mencionadas y porque además PHP es un lenguaje que tiene una gran cantidad de librerías y de documentación y resuelve cualquier problema práctico que se pueda tener. Además PHP es el lenguaje de scripting más famoso del mundo debido a muchas razones, pero principalmente por su amplia flexibilidad y simpleza.

1.7. Librería ExtJS 3.0

Para programar la capa de presentación se usará el Ext JS pues permite crear interfaces de usuarios muy amigables y funcionales de una manera intuitiva.

ExtJS es una librería de componentes que facilita las herramientas necesarias para la creación de aplicaciones Web con excelentes gráficos; ya que posee una considerable colección de elementos para el diseño de interfaces, ventanas, pestañas, menús, tablas, etc.

Actualmente ExtJS es considerado un FrameWork independiente; ya que a principios del 2007 se creó una compañía para comercializar y dar soporte al mismo, dicha compañía proporciona los servicios de consultoría necesarios para ayudar a los clientes en el aprovechamiento máximo de las ventajas de

ExtJS. Es importante señalar que la ExtJS 3.0 tiene dos tipos de licencias, LGPL (Open Source) y la comercial, esta última es obligatoria si se desea obtener soporte.

1.8. Framework

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, porque encapsula operaciones complejas en instrucciones sencillas.

Entre los Framework más conocidos se encuentran: Prado, CodeIgniter y Symfony.

1.8.1. Framework: Prado

"Prado es un framework de desarrollo de aplicaciones web en PHP basado en componentes y en eventos. Inicialmente inspirado en Apache Tapestry, la primera versión se realizó para PHP4, pero se reescribió completamente para PHP5. Entre las características que ofrece se encuentran: la separación entre la presentación y la lógica, su arquitectura modular configurable, componentes web, internalización y localización, manejo de errores, logs, caché, ACL y prevención de XSS" (2007).

1.8.2. Framework: CodeIgniter

"CodeIgniter es un framework, utilizado por una gran comunidad de usuarios. Construido para codificadores PHP que necesitan una herramienta de desarrollo fácil para crear aplicaciones web simples y elegantes.

Entre sus características se puede encontrar su compatibilidad con PHP 4 y PHP 5, incorpora el modelo MVC, soporte para múltiples bases de datos, plantillas, validaciones, no requiere instalación, se puede encontrar una librería con un gran número de clases" (Sanando, 2009).

1.8.3. Framework: Symfony (Versión: 1.0.11)

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación Web.

Está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows. Symfony está basado en el patrón clásico del diseño Web conocido como arquitectura MVC. A continuación se muestran algunas de sus características.

2.3.1.1. Características de Symfony

"Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de 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.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo" (Zaninotto, 2007).
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Luego del estudio de las características fundamentales de estos Framework para un mayor nivel de conocimiento se escoge Symfony por todas las características antes mencionadas. Además Symfony es un Framework maduro, rápido, configurable y ampliable. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Entre otra de las ventajas por las que se escoge Symfony como framework es que la capa de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.

Los frameworks para el desarrollo de aplicaciones web están en pleno apogeo. Symfony es una solución práctica y sencilla para el desarrollo de estas aplicaciones. La calidad de su código fuente y la gran cantidad de documentación disponible, son dos ventajas muy importantes sobre otros frameworks disponibles.

1.9. Sistemas de Gestión de Bases de Datos a utilizar

"Los Sistemas de Gestión de Base de Datos o SGBD (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta" (Ríos, 2009).

Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Entre los principales gestores de Bases de Datos a nivel mundial se encuentran Oracle, SQL Server, MySQL, PostgreSQL entre otros.

1.9.1. Sistema Gestor de Bases de Datos Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation.

"Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de aplicaciones web pasa lo mismo, como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, Access, MySQL, SQL Server y PostgreSQL" (Masip, 2002).

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

- Soporte de transacciones
- Estabilidad
- Escalabilidad
- Multiplataforma

1.9.2. Sistema Gestor de Bases de Datos PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales.

"Características de PostGreSQL:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos" (Pecos, 2002).

PostgreSQL es un motor de bases de datos: Avanzado y de código abierto.

Soporta:

- Querys complejos, incluyendo subselects
- Integridad referencial (Foreign Keys)
- Triggers
- Vistas (Views)
- Integridad Transaccional (ACID)
- Control de versionado concurrente (MVCC)
- Tiene licencia de tipo BSD

- Posee interfaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby.

Teniendo en cuenta la necesidad de utilizar una herramienta libre y por las características antes mencionadas se seleccionó como sistema gestor PostgreSQL 8.2, por ser multiplataforma, confiable, estable, con gran escalabilidad, control de concurrencia y funcionalidades que lo destacan como uno de los SGBD más potentes en la actualidad. PostgreSQL es un magnífico gestor de bases de datos, capaz de competir con muchos gestores comerciales.

Conclusiones

En este capítulo se detallaron las condiciones que rodean al objeto de estudio mediante los diferentes conceptos y definiciones planteadas, esclareciendo la lógica de negocio.

Para el desarrollo del software se utilizan en su mayoría herramientas libres y portables, algunas hasta de código abierto. Luego de las investigaciones realizadas durante este capítulo se llegó a la conclusión de que la propuesta consiste en desarrollar una aplicación Web usando la metodología robusta RUP, UML como lenguaje para describir todo el proceso, Visual Paradigm como herramienta CASE para modelar la aplicación, NetBeans como entorno integrado de desarrollo, como Sistema Gestor de Bases de Datos se seleccionó PostgreSQL, por ser uno de los más potentes en la actualidad, PHP como lenguaje de programación y Symfony como framework.

Capítulo 2
Características del Sistema

Introducción

Este capítulo tiene como objetivo principal realizar la modelación del negocio para lograr una mayor comprensión del trabajo que se realiza en la Dirección de Deportes de la UCI. Se identifican los actores, trabajadores y procesos del negocio actual así como la relación existente entre ellos. Además se brinda la descripción textual del caso de uso del negocio identificado, así como el diagrama de casos de uso, el diagrama de actividades y el modelo de objetos. Se definen además en este capítulo los requerimientos funcionales y no funcionales, los actores del sistema y los diagramas de casos de uso del sistema. Se realiza la descripción textual de los casos de uso del sistema definidos.

2.1. Objeto de Estudio

2.1.1. Objetivos estratégicos de la Dirección de Deportes

La Universidad de las Ciencias Informáticas en su estructura organizativa, cuenta con una Dirección de Deportes, que tiene dentro de sus funciones la gestión de la actividad científica.

En sus objetivos estratégicos para la organización del Sistema de Ciencia e Innovación Tecnológica, está propuesta la Línea de investigación relacionada con el uso de las Tecnologías de la Informática y las Comunicaciones en los Proyectos de Deportes y todo lo relacionado con la Cultura Física.

Para la gestión de la actividad científica se necesita evaluar cada uno de los indicadores con sus respectivos subindicadores de los eventos en los que los usuarios van participando, para así poder elaborar la información de la entidad.

2.1.2. Descripción General del negocio

La Gestión de la Actividad Científica es el proceso de mayor envergadura que se desarrolla en la Dirección de Deportes, tiene como propósito controlar la realización efectiva de la actividad científica de cada trabajador de la entidad de forma individual y de la entidad en sí, además analizar el comportamiento y evolución de todos los indicadores de medición de la actividad de ciencia y técnica. Actualmente esto no se realiza de forma eficiente, debido a varios factores, uno de ellos es la incapacidad del software SIndiCIT para brindar un seguimiento personal de los eventos en los que ha participado cada trabajador de la entidad, lo que trae consigo que para conocer estos datos de los

trabajadores de forma individual haya que hacer una búsqueda engorrosa en cada una de las copias legales archivadas, otro aspecto significativo es la ocurrencia de pérdidas de los documentos legales, debido al sistema de manipulación y almacenamiento de la información. Estos problemas planteados originan inconformidades por parte de los directivos y trabajadores de la Dirección de Deportes de la UCI, por lo que se hace necesario buscarle una solución inmediata y eficiente.

Dentro del proceso de Gestionar la Actividad Científica que se lleva a cabo en la Dirección de Deportes de la UCI, se realizan actividades como:

- Revisar el documento presentado por el usuario de la entidad: este documento no es más que una copia del certificado de participación en el evento.
- Archivar eventos: consiste en archivar la copia del certificado de participación en los eventos científicos presentada por el usuario y revisada por el Asesor Técnico.
- Calcular indicadores: esta actividad consiste en realizar el cálculo de los indicadores que aparecen en el capítulo anterior, con sus respectivos subindicadores. Este cálculo actualmente se realiza con el software SIndiCIT.
- Realizar informe sobre la actividad científica: consiste en realizar un informe que se presenta en la reunión que se lleva acabo en la entidad para analizar la evaluación científica de la Dirección de Deportes.
- Direccionar la actividad científica de la entidad: esta actividad consiste en, luego de realizado el análisis de la entidad, trazar nuevas metas y nuevas líneas de trabajo con el objetivo de mejorar la evaluación científica de la Dirección de Deportes.

En general para realizar la Gestión de la Actividad Científica es necesario consultar, crear o almacenar los documentos:

- Copia del documento legal: este documento no es más que una copia del certificado de participación en el evento.
- Informe actividad científica: este documento es un informe, que resume toda la evaluación científica de la Dirección de Deportes.

2.2. Modelo del negocio

“El negocio es el conjunto de servicios que una entidad, organización o empresa brinda a un conjunto de clientes o usuarios con el propósito de satisfacer las necesidades de estos. Al emplear la metodología basada en RUP, la modelación del negocio plantea la identificación de los procesos del

negocio y su completo análisis, el cual servirá como base para la identificación de probables candidatos a sistemas informáticos que soporten el negocio total o parcialmente" (Ivar, 2004).

El primer paso del modelado del negocio consiste en capturar y definir los procesos, lo cual constituye la base fundamental para el posterior modelado. Cuando se hable de procesos de negocio se puede decir que son un grupo de tareas relacionadas lógicamente que se llevan a cabo en una determinada secuencia y forma, que emplean los recursos de la organización para dar resultados que apoyen sus objetivos de comprender la estructura y la dinámica de la organización en la cual se va a implantar el sistema, en este caso la Dirección de Deportes, teniendo conocimientos de los problemas actuales de la misma, identificando las mejoras potenciales, asegurando que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.

2.2.1. Reglas del negocio

"Una Regla del Negocio es la declaración de políticas y restricciones de negocio de una organización. Este artefacto consiste en definir una exigencia específica o invariable que debe satisfacerse por el negocio. Las Reglas del Negocio pueden aplicarse siempre o sólo bajo una condición específica" ().
A continuación se enumeran las reglas del negocio identificadas.

1. Para que el Asesor Técnico pueda realizar el proceso de revisión del documento legal, el usuario tiene que presentar una copia del documento donde aparezcan claramente los siguientes datos: nombre y los apellidos del usuario, carnet de identidad y categoría docente.
2. Para que el Administrador General pueda realizar el proceso de archivar un evento, este debe ser aprobado antes por el Asesor Técnico.

2.2.2. Actores y trabajadores del negocio

A continuación se muestra el actor que se identificó en e negocio:

Nombre del Actor	Descripción
Usuario de la entidad	Representa un trabajador que inicia las actividades que se desarrollan con cierta periodicidad dentro de la Dirección de Deportes de la UCI.

Tabla 2.1 Descripción del actor del negocio.

A continuación se muestra los trabajadores que se identificaron en el negocio:

Nombre del Trabajador	Descripción
Asesor Técnico	Es el encargado de revisar el documento legal que presenta el usuario y verificar que éste cumpla con las especificaciones necesarias, se encarga además de dar la autorización para que el nuevo evento sea archivado.
Administrador General	Es el encargado de archivar el nuevo evento en el cual participó el usuario.
SIndiCIT	Se encarga de realizar los cálculos de los indicadores
Especialista	Analiza el resultado final de los cálculos realizados, realiza un informe sobre la actividad científica de la entidad y a partir de ahí direcciona el trabajo a seguir en la entidad con el fin de mejorar la evaluación científica de la misma.

Tabla 2.2 Descripción de los trabajadores del negocio.

2.2.3. Diagrama de Casos de Uso de Negocio (CUN)

Un diagrama de CUN representa gráficamente a los procesos del negocio y su interacción con los actores del negocio, además el tipo y el orden en que los elementos interactúan.

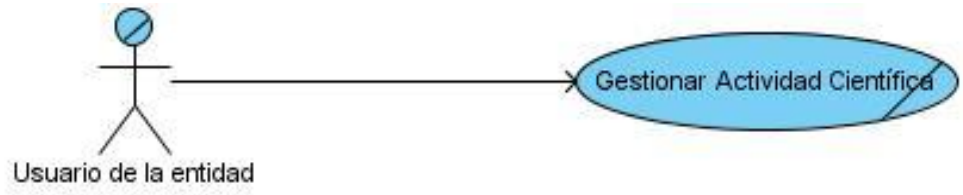


Figura. 2.1 Diagrama de CU del negocio *"Gestionar Actividad Científica"*

2.2.4. Descripción textual del Caso de Uso del Negocio

A continuación se presenta la descripción textual resumida del CU del negocio, para ver la descripción textual ampliada, donde se definieron todas las acciones a ejecutarse remitirse al Expediente de Proyecto.

Caso de Uso:	Gestionar Actividad Científica.
Actores:	Usuario de la Entidad (Inicia).
Trabajadores:	Asesor Técnico, Administrador General, Especialista, SIndiCIT.
Resumen:	El caso de uso se inicia cuando el usuario de la entidad se presenta con una copia del documento legal, con los datos requeridos y se la entrega al Asesor Técnico. El Asesor Técnico revisa el documento, si está en orden el Administrador General la archiva. Luego SIndiCIT calcula los indicadores y el Especialista analiza el resultado de los cálculos. Posteriormente el Especialista realiza un informe sobre la actividad científica y a partir de ahí, direcciona el trabajo a seguir en la entidad, con el fin de mejorar la evaluación científica de la misma.

Tabla 2.3 Descripción textual resumida del CU del negocio *"Gestionar Actividad Científica".*

2.2.5. Diagrama de actividades

A continuación se muestra el diagrama de actividades que describe todo el proceso del caso de uso del negocio y el orden en que va realizando las tareas para lograr los objetivos.

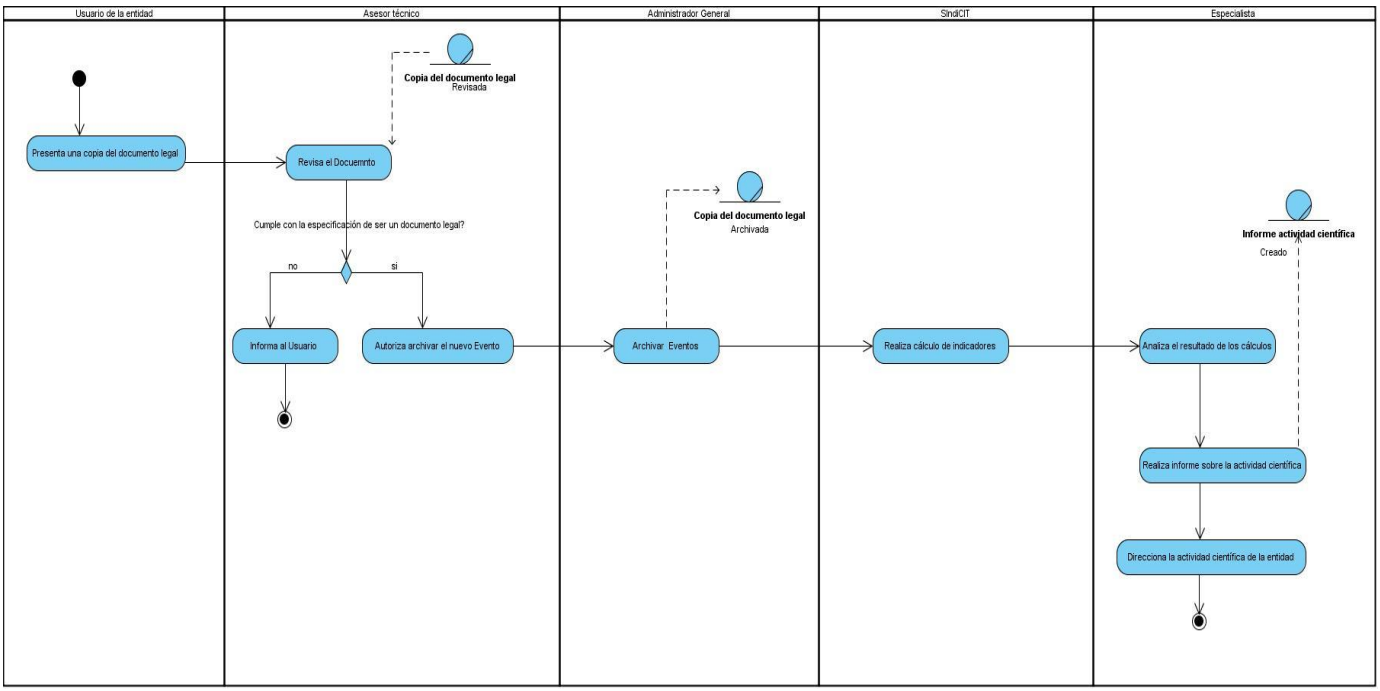


Figura. 2.2 Diagrama de actividades del CU del negocio *"Gestionar Actividad Científica".*

2.2.6. Modelo de objetos

El objetivo que persigue el diagrama del modelo de objetos es el de representar los trabajadores del negocio y su interacción con las entidades del mismo.

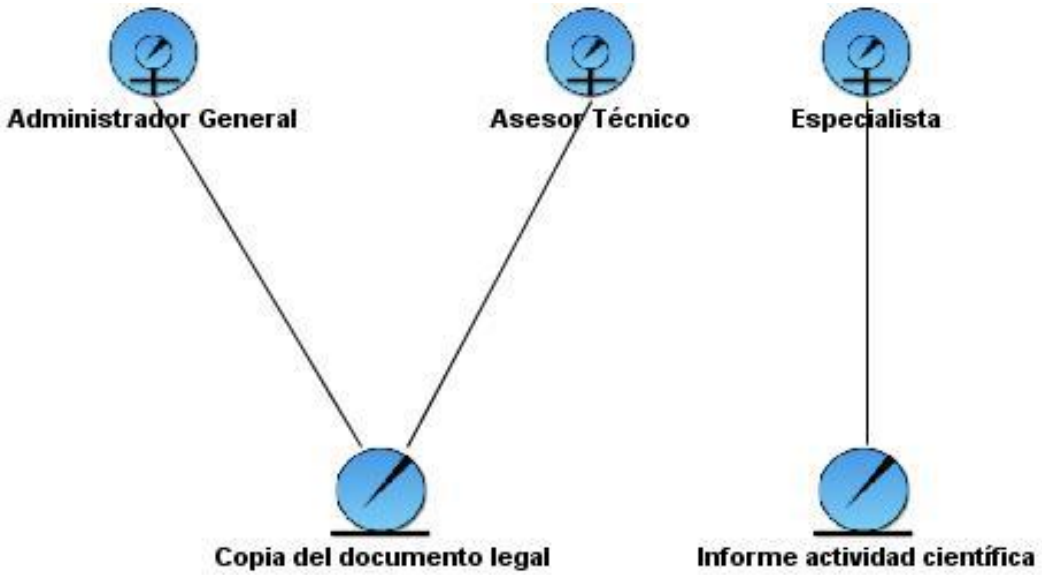


Figura. 2.3 Diagrama del modelo de objeto.

Una vez modelado el proceso del negocio actual se procede al Modelado del Sistema con el objetivo de definir las funcionalidades y características del sistema.

2.3. Requerimientos

El flujo de trabajo Requerimientos del Proceso Unificado de Desarrollo (RUP, Rational Unified Process) tiene su mayor peso en la fase de Inicio, aunque se continúan refinando los requisitos a lo largo de todo el ciclo de vida. Además es muy importante para la elaboración de un software porque en el los desarrolladores identifican y clasifican los requerimientos, encuentran actores y casos de uso, los priorizan y los detallan, se encargan de realizar los prototipos de interfaz de usuario, y estructurar el modelo de casos de uso del sistema.

2.3.1. Requerimientos funcionales y no funcionales del sistema

Existen varios tipos de requerimientos, en el presente trabajo se identificaron requisitos funcionales y no funcionales, estos deben ser descritos y en la fase de construcción deben ser posibles de probar o verificar.

2.3.1.1. Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requerimientos funcionales que se identificaron son:

RF 1- Adicionar administrador.

RF 2- Buscar administrador.

RF 3- Mostrar administrador.

RF 4- Modificar administrador.

RF 5- Eliminar administrador.

RF 6- Autenticar usuario.

RF 7- Adicionar resultados actividad científica.

RF 8- Buscar resultados actividad científica

RF 9- Mostrar resultados actividad científica.

RF 10- Modificar resultados actividad científica.

RF 11- Eliminar resultados actividad científica.

RF 12- Adicionar usuario.

RF 13- Buscar usuario.

RF 14- Mostrar usuario.

RF 15- Modificar usuario.

RF 16- Eliminar usuario.

RF 17- Calcular indicador premios obtenidos.

RF 18- Calcular indicador publicaciones científicas.

RF 19- Calcular indicador patentes y registros.

RF 20- Calcular indicador participación en proyectos.

RF 21- Calcular indicador resultados introducidos.

RF 22- Calcular indicador trabajos presentados en eventos.

RF 23- Calcular indicador capacitación ofertada o recibida.

RF 24- Calcular indicador uso de estudiantes.

RF 25- Visualizar resultado.

RF 26- Ordenar resultado.

2.3.1.2. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades se ven como las características que hacen al producto atractivo, usable, rápido o confiable. Los requerimientos no funcionales que se identificaron son:

- **Apariencia o interfaz externa**

El sistema, tendrá los colores correspondientes al logo de la UCI. Las páginas de la aplicación no se cargarán con mucha información y contendrán solo las imágenes necesarias.

- **Usabilidad**

El sistema debe permitir a los usuarios un acceso fácil y rápido. Podrá ser usado por las personas con acceso permitido, deben poseer conocimientos básicos en el manejo de una computadora y de un ambiente Web.

- **Rendimiento**

Los tiempos de respuestas deben ser rápidos, al igual que la velocidad de procesamiento de la información.

- **Soporte**

Se requiere de la instalación de un servidor Web que interprete código PHP 5 o superior.

Las computadoras clientes requerirán de un navegador capaz de interpretar código Java Script.

- **Portabilidad**

El sistema podrá ser usado sobre los sistemas operativos Windows y Linux.

- **Seguridad**

Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren de acuerdo al usuario que esté activo.

El sistema debe contar con protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

- **Políticos-culturales**

El idioma que se empleará en la aplicación será el español.

El sistema tendrá logotipos e imágenes en correspondencia con el carácter científico y profesional de la UCI.

Algún cambio que se quiera realizar en la aplicación, será tramitado por la Dirección de Deportes de la UCI.

- **Legales**

El sistema se dará a conocer a todos los trabajadores de la Dirección de Deportes como la herramienta para gestionar la información.

Se estará usando para el desarrollo de la aplicación herramientas de software libre con licencia GNU/GPL.

- **Confiabilidad**

El sistema será usado y administrado solamente por trabajadores de la Dirección de Deportes de la UCI, por lo tanto la información que fluirá en el mismo, será la emitida por dicha entidad.

Podrán acceder a visualizar ciertas informaciones, directivos de otras áreas, trabajadores de la entidad.

El sistema validará la entrada de datos para evitar entradas inadecuadas.

- **Software**

Se deberá disponer, para instalar la aplicación, del Sistema Operativo Windows 98 o superior, o cualquier distribución de Linux. Las computadoras clientes de los usuarios accederán al sistema usando uno de los siguientes navegadores: Internet Explorer 5.5 o superior, Netscape, Mozilla 1.7 o superior o FireFox 0.9.3 o superior.

Para el servidor de la aplicación y para el Servidor de Bases de Dato el Sistema Operativo recomendado es Windows Server 2003 o superior o Linux.

Se debe instalar un servidor Web Apache 1.3 o superior.

- **Hardware**

El servidor debe tener las siguientes características: capacidad de disco duro superior a 80.0 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

- **Restricciones en el diseño y la implementación**

La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones de los datos de entrada.

Se utilizarán herramientas de desarrollo que garanticen la calidad de todo el ciclo de desarrollo del producto.

Se usará el lenguaje de programación PHP 5, como framework de desarrollo Symfony (Versión 1.0.11), NetBeans como entorno de desarrollo y como gestor de bases de datos PostgreSQL (Versión 8.2).

2.3.2. Actores del sistema a automatizar

Solo se identificó como actor del sistema al Administrador General, los demás candidatos no realizan ninguna función dentro del sistema que no sea autenticarse.

Nombre del Actor	Descripción
Administrador General	Es el encargado de crear los usuarios del sistema y asignar los

	privilegios de cada uno dentro del mismo. Es el encargado además de gestionar la actividad científica, realizar los cálculos de los indicadores y publicar el resultado de dichos cálculos.

Tabla 2.4 Descripción de los actores del sistema.

2.3.3. Casos de Uso del Sistema

Se muestran a continuación los Casos de Uso del sistema que se identificaron basados en los requerimientos expuestos con anterioridad:

1. Gestionar administrador.
2. Autenticar usuario.
3. Gestionar actividad científica.
4. Gestionar usuario común.
5. Calcular indicadores.
6. Visualizar resultado.
7. Ordenar resultado.

2.3.4. Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso del sistema ayuda a comprender gráficamente los procesos del sistema y su interacción con los actores. A continuación se muestra el diagrama de Casos de Uso estructurado, donde todos son significativos, por lo que el diagrama de Casos de Uso del sistema coincide con la Vista de Casos de Uso.

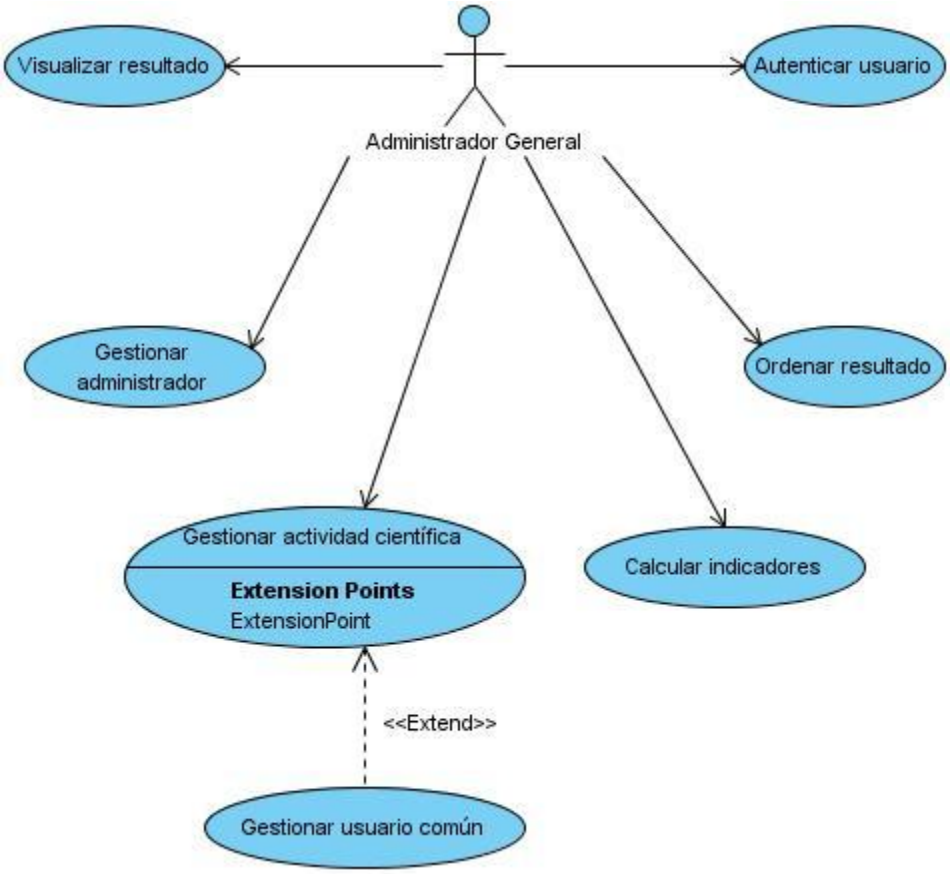


Figura 2.4 Diagrama de Casos de Uso del Sistema.

2.3.5. Patrones de Casos de Usos aplicados

Para el desarrollo del presente modelo de casos de uso, se aplicó el patrón CRUD que permitió reflejar con más precisión los requerimientos existentes y lograr que los resultados se obtengan de forma más rápida, haciendo más fácil el trabajo.

Creating, Reading, Updating and Deleting (Patrón CRUD): propone identificar un CU, llamado "Información CRUD" o "Administrar Información", se usó para modelar todas las operaciones que se pueden realizar sobre una parte de información de cierto tipo (o sea una misma entidad), tal como crearla, leerla, actualizarla y eliminarla.

Aplicación: debe ser usado cuando todos los flujos contribuyen al mismo valor de negocio, son cortos y sencillos.

Con la aplicación de este patrón se permitió resolver los problemas que se presentaron en la modelación del sistema con mayor calidad y de forma más rápida.

2.3.6. Descripción textual resumida de los Casos de Uso (CU)

Para entender la funcionalidad asociada a cada caso de uso no sólo basta con la representación gráfica de los Casos de Uso, es por ello que se han descrito textualmente cada uno. A continuación se presentan las descripciones textuales resumidas para cada CU, para ver las descripciones ampliadas donde se definieron cada una de las acciones a ejecutarse con sus respectivos flujos alternos remitirse al Expediente de Proyecto.

2.3.6.1. Descripción textual resumida del CU "Gestionar administrador"

Nombre del Caso de Uso	Gestionar administrador.
Actores	Administrador General (Inicia).
Propósito	Gestionar información relativa a los diferentes administradores almacenados en el sistema.
Resumen	El Caso de Uso se inicia cuando el actor solicita realizar las siguientes actividades: adicionar un nuevo administrador, buscar administrador, mostrar administrador, modificar un administrador o eliminar un administrador. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	RF 1; RF 2; RF 3; RF 4

Tabla 2.5 Descripción textual resumida del CU *"Gestionar administrador".*

2.3.6.2. Descripción textual resumida del CU "Gestionar actividad científica"

Nombre del Caso de Uso	Gestionar actividad científica.
Actores	Administrador General (Inicia).
Propósito	Gestionar información relativa a los eventos que son insertados en el sistema.
Resumen	El Caso de Uso se inicia cuando el actor solicita realizar las siguientes actividades: adicionar resultados actividad científica, modificar resultados actividad científica, eliminar resultados actividad científica, buscar resultados actividad científica o mostrar resultados actividad científica.El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	RF 6; RF 7; RF 8; RF 9;

Tabla 2.6 Descripción textual resumida del CU *"Gestionar actividad científica".*

2.3.6.3. Descripción textual resumida del CU "Autenticar usuario"

Nombre del Caso de Uso	Autenticar usuario.
Actores	Administrador (Inicia).
Propósito	Permitir que el administrador se autentique.
Resumen	El caso de uso inicia cuando el actor introduce los datos que se requieren para acceder a la aplicación, éstos se verifican y finaliza el caso de uso otorgándole los privilegios y habilitándole la entrada según su rol.
Referencias	RF 5

Tabla 2.7 Descripción textual resumida del CU *"Autenticar usuario".*

2.3.6.4. Descripción textual resumida del CU "Gestionar y usuario común"

Nombre del Caso de Uso	Gestionar usuario común.
Actores	Administrador General (Inicia).
Propósito	Gestionar información relativa a los diferentes usuarios.
Resumen	El Caso de Uso se inicia cuando el actor solicita realizar las siguientes actividades: adicionar un usuario, buscar y mostrar usuario, modificar un usuario o eliminar un usuario. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	RF 10; RF 11; RF 12; RF 13

Tabla 2.8 Descripción textual resumida del CU *"Gestionar usuario común".*

2.3.6.5. Descripción textual resumida del CU "Calcular indicadores"

Nombre del Caso de Uso	Calcular indicadores.
Actores	Administrador General (Inicia).
Propósito	Realizar los cálculos de los indicadores generales y subindicadores para la medición de la actividad científica.
Resumen	El Caso de Uso se inicia cuando el actor solicita calcular la actividad científica a través de los indicadores generales para esto solicita realizar las siguientes actividades: calcular indicador premios obtenidos, calcular indicador publicaciones científicas, calcular indicador patentes y registros, calcular indicador participación en

	proyectos, calcular indicador resultados introducidos, calcular indicador trabajos presentados en eventos, calcular indicador capacitación recibida y calcular indicador uso de estudiantes El sistema ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite y se publica el resultado de la operación solicitada para que pueda ser consultada.
Referencias	RF 14; RF 15; RF 16; RF 17; RF 18; RF 19; RF 20; RF 21

Tabla 2.9 Descripción textual resumida del CU *"Calcular indicadores".*

2.3.6.6. Descripción textual resumida del CU "Visualizar resultado"

Nombre del Caso de Uso	Visualizar resultado.
Actores	Administrador General (Inicia).
Propósito	Mostrar toda la actividad científica del departamento.
Resumen	El Caso de Uso se inicia cuando el actor solicita realizar la siguiente actividad: visualizar resultados del departamento. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.
Referencias	RF 22

Tabla 2.10 Descripción textual resumida del CU *"Visualizar resultado".*

Conclusiones

En este capítulo se cumplió el objetivo propuesto, se obtuvieron los artefactos principales correspondientes a este flujo de trabajo, se inició el desarrollo de la propuesta de solución, y a través de los procesos de negocio y el levantamiento de requisitos, se pudo obtener un listado con las funcionalidades que debe tener el sistema y se describieron paso a paso todas las acciones del actor del sistema con los casos de uso con los que interactúa. A partir de este capítulo, se está en condiciones de comenzar la elaboración del sistema, específicamente, de realizar el análisis y diseño del mismo, partiendo siempre, por supuesto, de las funcionalidades consideradas en este capítulo.

Capítulo 3
Diseño del Sistema

Introducción

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. El diseño es el núcleo técnico de la ingeniería de software, contribuye a una arquitectura estable y sólida, y crea un plano del modelo de implementación. En el presente capítulo se hace referencia a la arquitectura del sistema a través de la Vista Lógica y de Despliegue, se presentan los prototipos de interfaz de usuario, los diagramas de secuencia y los diagramas de clases del diseño. Se confecciona el mapa de navegación.

3.1. Propósitos del diseño

“Concretamente se puede definir como propósitos del diseño:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cual es muy útil cuando se utilizan interfaces como elementos de sincronización entre diferentes equipos de desarrollo.
- Para modelar el diseño se utilizan las clases del diseño que son una construcción similar en la implementación del sistema.
- El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Las operaciones, atributos, tipos, visibilidad, se pueden especifican con la sintaxis del lenguaje elegido.
- Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases” (Conallen, 1999).

3.2. Principios de diseño

“UML posee una extensión para el modelado de aplicaciones Web, propuesta por Conallen, dicha extensión es usada para el diseño de las clases. Los estereotipos que usa son:

<<Server Page>> Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.

Esta clase sólo puede tener relaciones con objetos en el servidor, una relación 1:1 con un fichero en el servidor. En las aplicaciones en PHP se corresponde con un fichero .php

<<Client Page>> Una instancia de Página Cliente es una página Web, con formato HTML; mezcla de datos, presentación y lógica. Son interpretadas por el browser. Cada página cliente solo puede ser construida por una página servidor.

<<Form>> Grupo de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y Hidden Fields)” (Conallen, 1999).

3.3. Referencia a la Arquitectura del Sistema

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución" (Reynoso, 2008)

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida. Las arquitecturas más conocidas son:

- Monolítica. Donde el software se estructura en grupos funcionales muy acoplados.
- Cliente-servidor. Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.

- Arquitectura de tres niveles. Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

En el presente trabajo se utilizó la arquitectura de tres niveles, esta arquitectura tiene como ventaja principal que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que ocurra algún cambio, sólo se afecta al nivel requerido, a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Uno de los modelos más conocidos es el 4+1" de Philippe Kruchten, vinculado al Rational Unified Process (RUP), que define cuatro vistas diferentes: la Vista de Casos de Uso, la Vista Lógica, la Vista de Despliegue, Vista de Procesos y la Vista de Implementación. En este trabajo se desarrollará la Vista Lógica, la Vista de Despliegue y la Vista de Casos de Uso que coincide con el diagrama de Casos de Uso del sistema ya que todos son arquitectónicamente significativos.

3.3.1. Vista Lógica

La Vista Lógica es un subconjunto del artefacto Modelo de diseño, la cual representa los elementos de diseño más importantes para la arquitectura del sistema. Describe las clases más importantes, su organización en paquetes y subsistemas, éstos en capas y las realizaciones de casos de uso más importantes.

Para estructurar la vista del sistema, no se identificaron subsistemas funcionales, ni se organizó en paquetes, se representaron los casos de uso arquitectónicamente significativos, que constituyen la totalidad de los casos de uso identificados y la relación entre ellos. A continuación se muestra la Vista Lógica en la **Figura 3.1**.

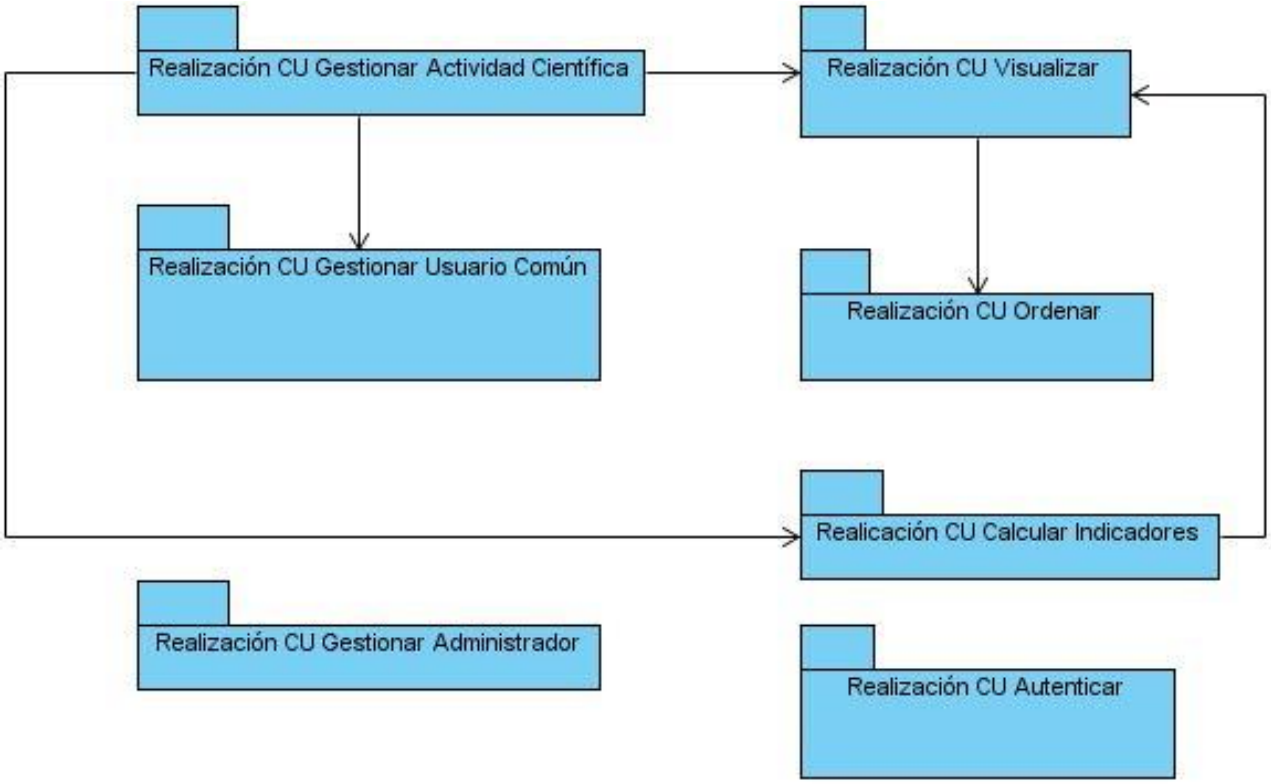


Figura 3.1 Vista Lógica.

A continuación en la Figura 3.2, se muestra la realización del CU Gestionar Actividad Científica, la cual está conformada por su diagrama de clases del diseño y de interacción, en cada uno de estos se evidencia el patrón de arquitectura (Modelo-Vista-Controlador) a partir del uso de Symfony como framework para desarrollar la aplicación web.

Figura 3.2 Realización del CU *"Gestionar actividad científica".*

3.3.2. Vista de Despliegue

Vista de Despliegue: Muestra la distribución física del sistema (ordenadores, dispositivos) y sus conexiones, y se plasma en el diagrama de Despliegue.

La vista se encuentra representada a través del diagrama de despliegue que se muestra a continuación, que se encuentra estructurado de la siguiente manera:
Los nodos representan un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar, se cuenta con un Servidor de Aplicaciones conectado a la PC Cliente, a través del protocolo HTTPS.
Para el almacenamiento de los datos de la aplicación se utiliza un servidor de Base de Datos representado como un nodo y para lograr la conexión del sistema con la base de datos se utiliza TCP / IP + SSL como protocolo de comunicación.

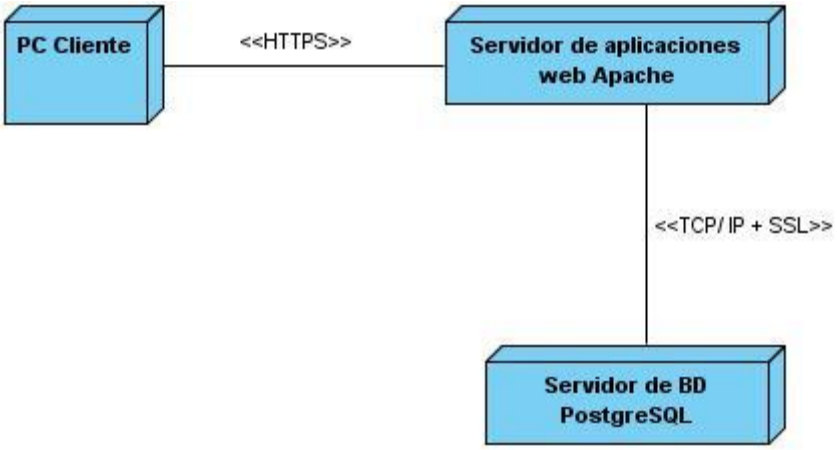


Figura 3.3 Vista de Despliegue.

3.4. Prototipos de interfaz de usuario

Los prototipos de interfaz de usuario son una representación de la funcionalidad contenida en los casos de uso; de manera que permita que el usuario verifique que el sistema va a satisfacer sus necesidades. De esta manera se válida que se esté cumpliendo con los requerimientos exigidos antes de que se realice todo el esfuerzo necesario para el desarrollo.
A continuación se muestra uno de los prototipos de interfaz de usuario que se desarrollaron, para ver el resto remitirse al Expediente de Proyecto.

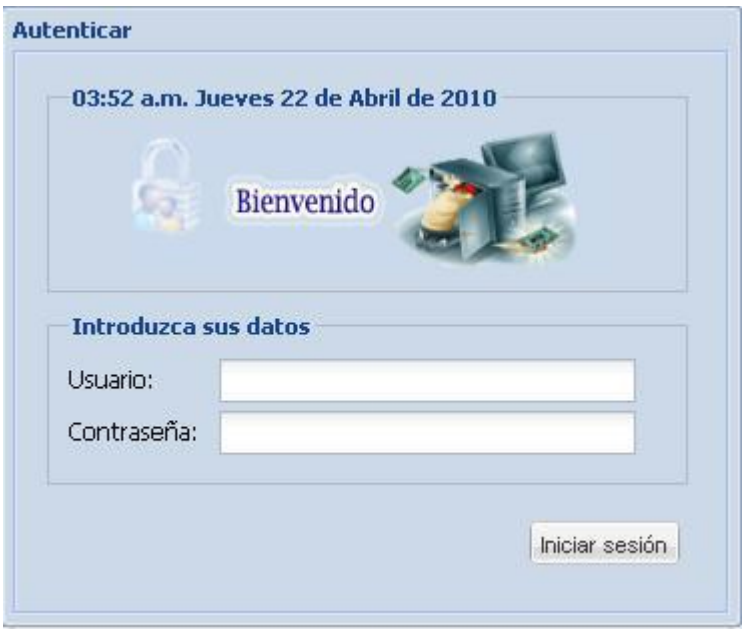


Figura 3.4 Prototipo de Interfaz de Usuario *"Autenticar usuario".*

3.5. Mapa de Navegación

Un mapa de navegación es la representación del sitio a través de una secuencia de interfaces. Para la construcción del mapa de navegación se utilizó UML, la navegabilidad se estructuró de la siguiente manera: a partir de la página principal (Index) se podrá acceder a la página inicio, una vez en ella se puede acceder a la Gestión de los resultados y a Administración, estos módulos contiene todos los procesos que se llevan a cabo en la aplicación. A través del módulo Administración se tiene acceso a Gestionar administrador, permitiendo así adicionar, eliminar o editar administrador. Del módulo Gestión de resultados se puede tener acceso a la pagina buscar, ver resultados del departamento y Gestionar resultados, de ésta última se podrá acceder a las páginas Adicionar, Eliminar, Editar y Ver evaluaciones. Como parte de la navegabilidad del sitio se podrá volver a la página principal (Index) desde cualquiera de las páginas de la aplicación.

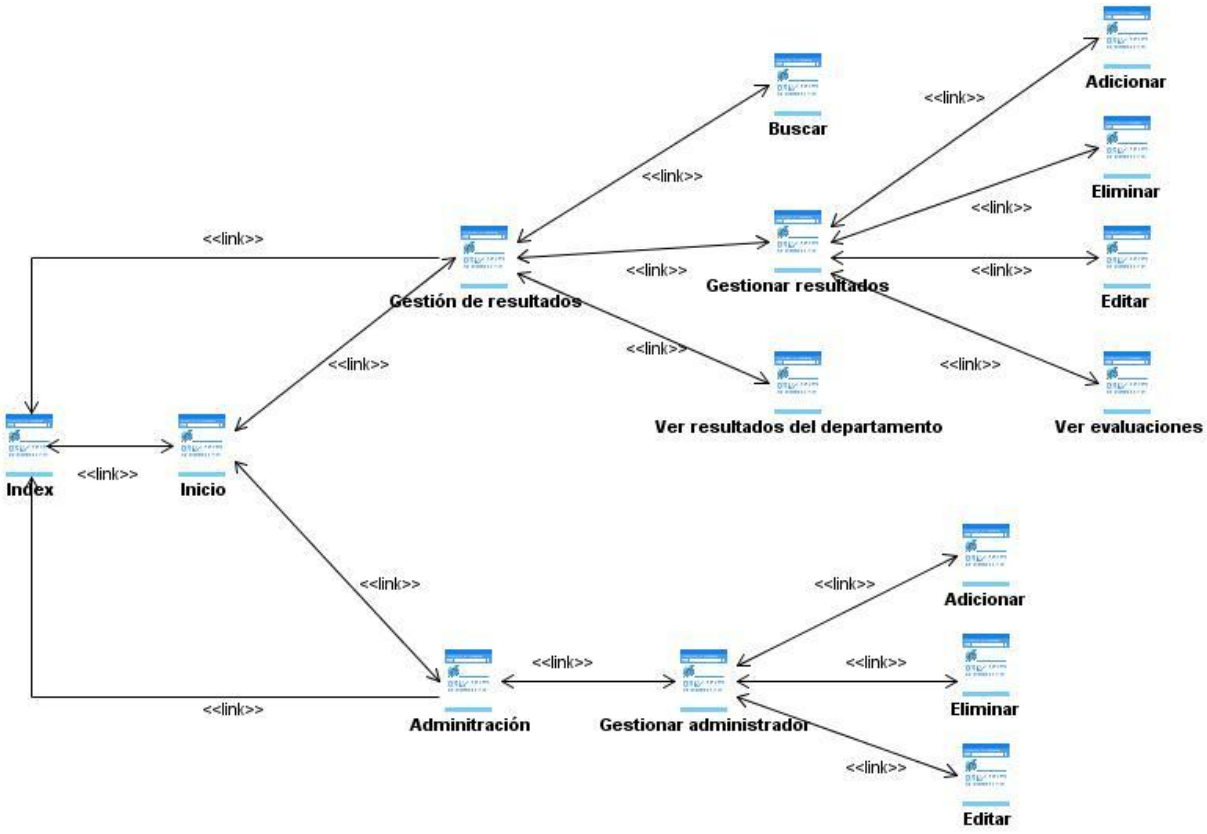


Figura 3.5 Mapa de Navegación.

3.6. Diagramas de Clases del Diseño

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Se dice que son diagramas estáticos porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases “conocen” a qué otras clases o qué clases “son parte” de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas.

En la Vista se encuentran: las páginas clientes, los formularios, páginas servidoras y el layout, aquí se representa la información con la que trabaja la aplicación y la lógica de negocio. El Controlador se encuentra conformado por el controlador frontal que es el único punto de entrada a la aplicación, carga la configuración y determina la acción a ejecutarse, y las acciones: verifican la integridad de las peticiones, preparan los datos requeridos por la capa de presentación y contienen la lógica de la aplicación. La lógica de negocio de las aplicaciones Web depende de su modelo de datos, en el se encuentran todas las clases con sus atributos y las relaciones existentes entre ellas y las clases que genera Symfony: Objeto, ObjetoPeer, BaseObjeto y BaseObjetoPeer.

- Clases Base: Son las que se generan directamente a partir del esquema, no se deben modificar esas clases, porque cada vez que se genera el modelo, estas se borran.
- Clases Objeto: Heredan de las clases con nombre Base, estas no se modifican cuando se genera el modelo, por lo que en las mismas se añaden los métodos propios.
- Clases Peer: tienen métodos estáticos para trabajar con las tablas de la base de datos, proporcionan los medios necesarios para obtener los registros de las tablas y sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada.

Para ver los diagramas de clases del diseño desarrollados remitirse al Expediente de Proyecto.

3.7. Patrones de Diseño utilizados

El framework Symfony cumple con una gran gama de patrones de diseño que lo hacen muy robusto tanto en su código fuente como en las aplicaciones que se implementan con él. A continuación se presentan una serie de patrones utilizados en el diseño de la aplicación.
Dentro de los patrones de diseño que se usan para la arquitectura del framework Symfony:

"Modelo-Vista-Controlador **(MVC):** que está formado por tres niveles:

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

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación" (Zaninotto, 2007). **(Ver Anexo 3)**

3.7.1. Patrones GRASP utilizados

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, el MVC hace uso de dos de los patrones básicos de GRASP bajo acoplamiento y alta cohesión, que constituyen dos de sus principales beneficios:

Bajo acoplamiento:

La clase Actions hereda solamente de sfActions para lograr un bajo acoplamiento de clases

- Desacopla las vistas de los modelos.
- Desacopla los modelos de la forma en que se muestran e ingresan los datos.

Alta cohesión:

- Cada elemento del patrón está altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).

3.7.2. Patrones GOF utilizados

El grupo de GoF clasificaron los patrones en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Dentro de los patrones estructurales se pone de manifiesto:

Decorator: es otro de los patrones de diseño utilizados por Symfony, es implementado por el layout, archivo denominado también plantilla global que 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 otro punto de vista, el layout decora la plantilla.

También conocido como Wrapper, a veces se desea adicionar responsabilidades a un objeto, pero no a toda la clase. Las responsabilidades se pueden adicionar por medio de los mecanismos de herencia, pero este mecanismo no es flexible porque la responsabilidad es adicionada estáticamente. La solución flexible es la de rodear el objeto con otro objeto que es el que adiciona la nueva responsabilidad. Este nuevo objeto es el Decorator, este patrón se relaciona con otros patrones los cuales son: Adapter, Composite y el Strategy.

Aplicación:

El Decorator se debe usar para:

- Adicionar responsabilidades a objetos individuales dinámicamente sin afectar otros objetos.
- Para agregar responsabilidades que pueden ser retiradas.
- Cuando no es práctico adicionar responsabilidades por medio de la herencia.

"Ventajas del patrón Decorator

- Más flexible que la herencia: responsabilidades pueden añadirse y eliminarse en tiempo de ejecución.
- Diferentes decoradores pueden ser conectados a un mismo objeto.
- Reduce el número de propiedades en las clases de la parte alta de la jerarquía.
- Es simple añadir nuevos decoradores de forma independiente a las clases que extienden.
- Un objeto decorador tiene diferente OID (identificador único que proporciona el sistema) que el objeto que decora" (Mario Rodríguez Martín, 2008).

3.8. Diagramas de secuencia

El diagrama de secuencia de un sistema es una representación que muestra, en determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del sistema. A continuación está expuesto el Escenario Adiciona que corresponden al CU "Gestionar Actividad Científica", para ver el resto de los diagramas de secuencia remitirse al Expediente de Proyecto.

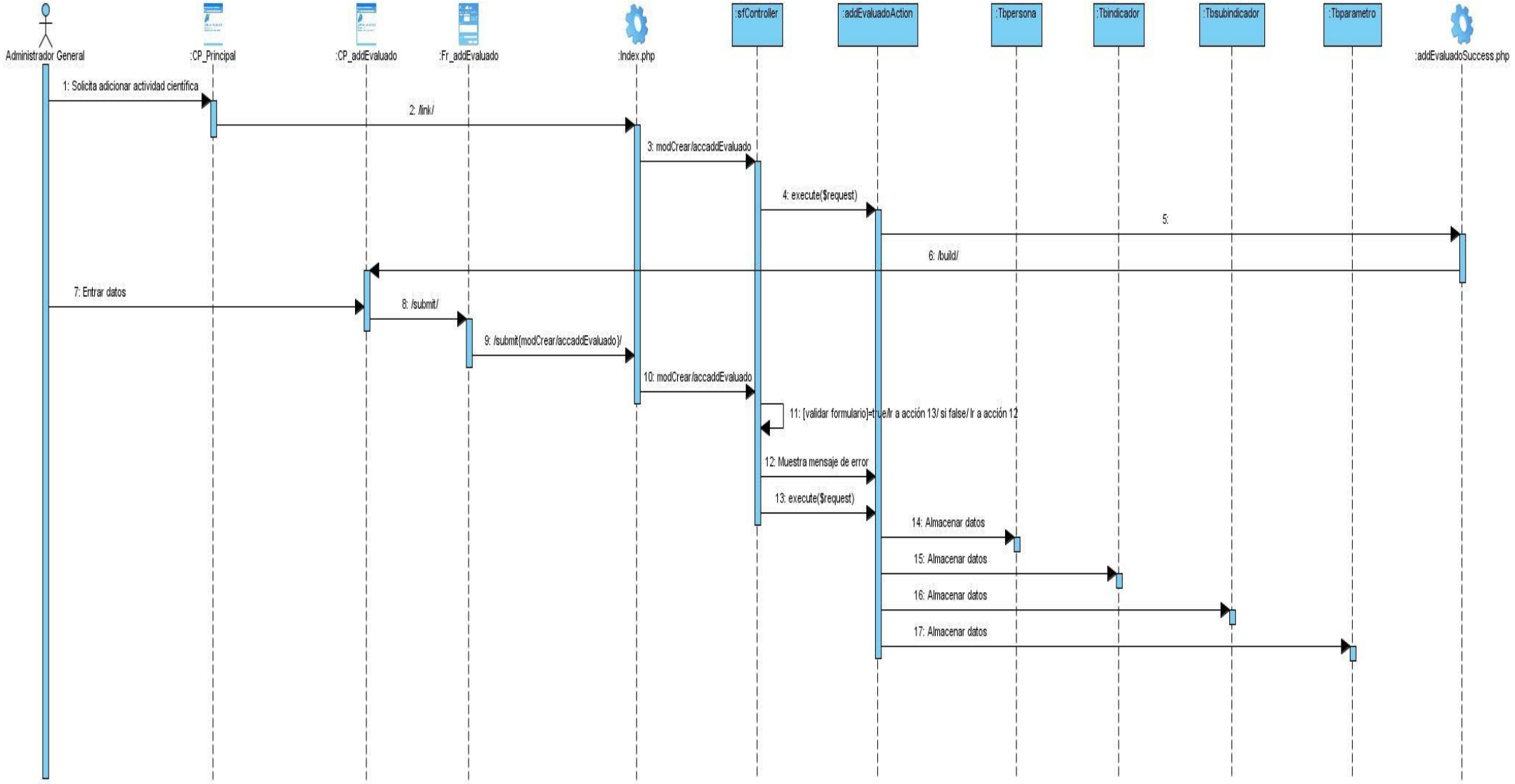


Figura 3.6 Diagrama de Secuencia (Escenario Adicionar): CU *"Gestionar actividad científica".*

3.9. Diseño de la Base de Datos

Para realizar el diseño de la base de datos de la aplicación se confeccionaron el diagrama de clases persistentes y el modelo de datos. Las clases persistentes son clases entidades que pueden mantener su valor en el espacio y el tiempo. En contrapartida las clases temporales son aquellas de las cuales el sistema se encarga de manejar y almacenar en tiempo de ejecución, razón por la que dejan de existir cuando termina la ejecución del programa.

A continuación se presenta el diagrama de clases persistentes que está compuesto por dichas clases y las correspondientes relaciones entre ellas y el modelo físico de datos que es una representación de las tablas de la base de datos y sus relaciones.

3.9.1. Modelo lógico de datos

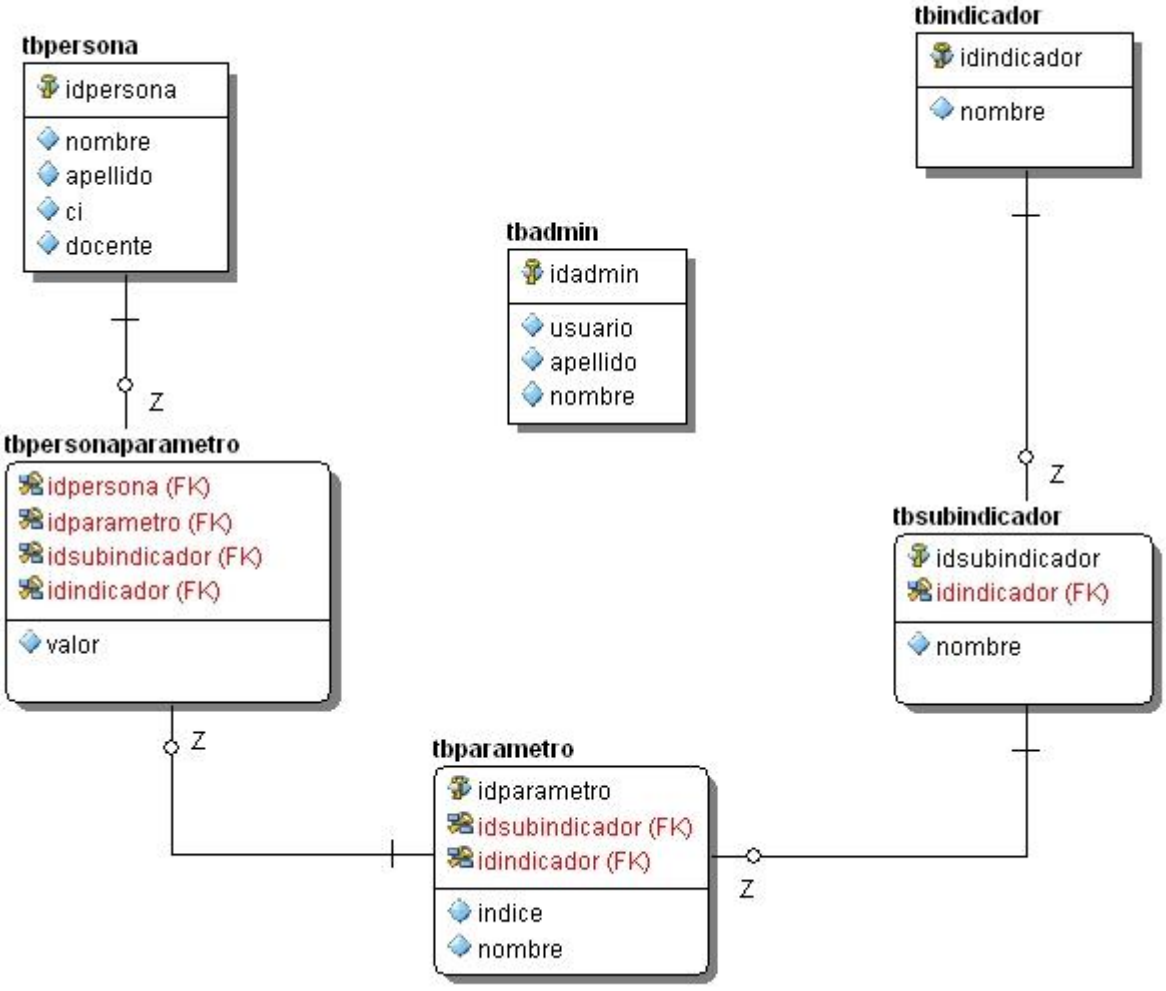


Figura 3.7 Modelo lógico de datos.

3.9.2. Modelo físico de datos

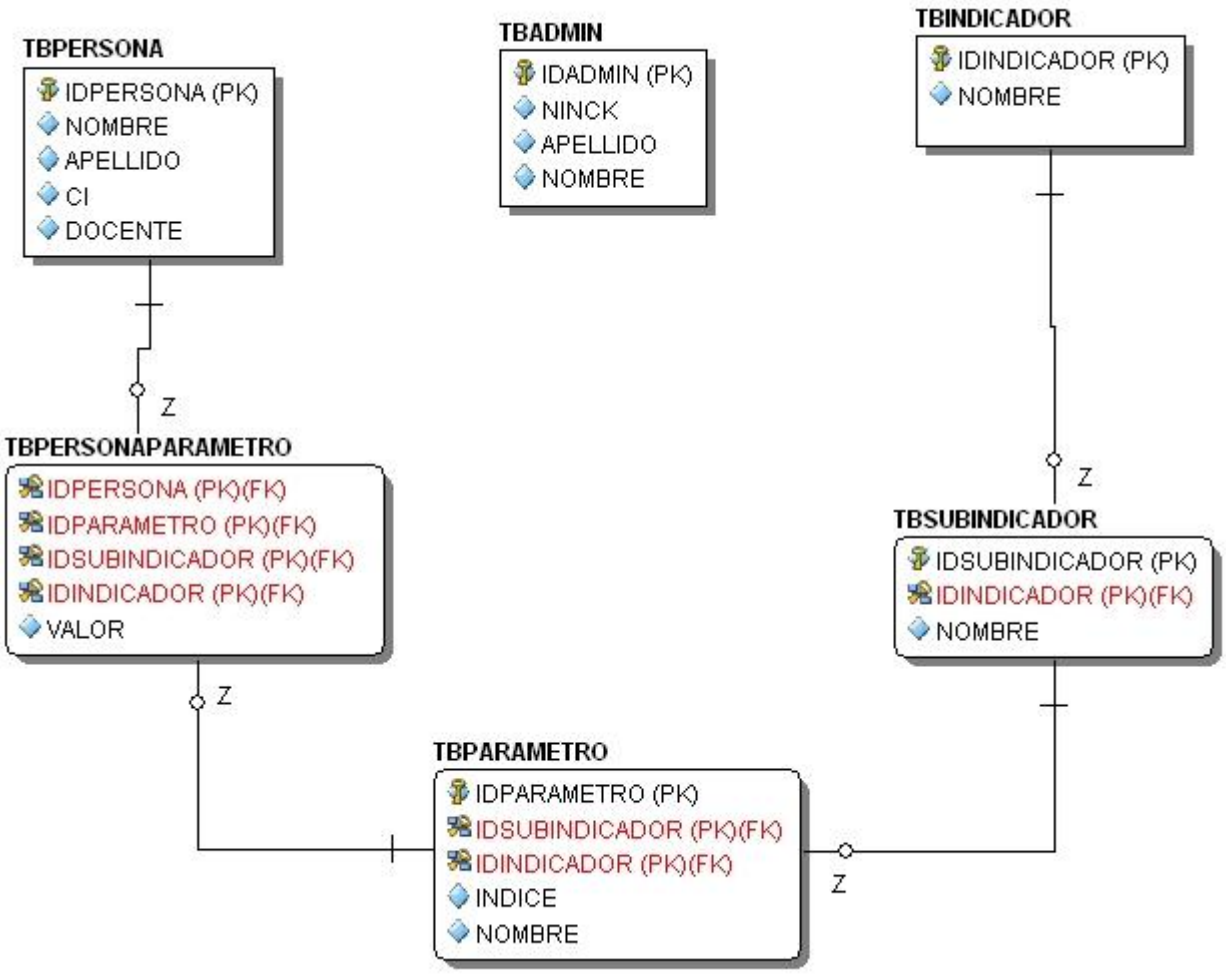


Figura 3.8 Modelo físico de datos.

3.10. Descripción de las Tablas de la Base de Datos

Se realizaron las descripciones de las tablas de la BD para un mayor entendimiento de la misma. Para ver las descripciones remitirse al Expediente de Proyecto.

A continuación se hace mención a las tablas que componen la BD y se da una breve descripción de las mismas.

Tbpersona: En esta tabla se almacenan las informaciones referentes a los usuarios.

Tbindicador: En esta tabla se almacenan las informaciones referentes a los indicadores.

Tbsubindicador: En esta tabla se almacenan las informaciones referentes a los subindicadores.

Tbparametro: En esta tabla se almacenan las informaciones referentes a los parámetros de los subindicadores.

Tbpersonaparametro: En esta tabla se almacenan las informaciones referentes al valor obtenido de acuerdo a la actividad científica de cada persona.

Tbadmin: En esta tabla se almacenan las informaciones referentes a los administradores

Conclusiones

Se finaliza la etapa de análisis y diseño del sistema obteniendo un modelo más detallado de la solución propuesta.

En este capítulo se definieron las clases del sistema y sus relaciones describiendo las mismas en términos de diagramas de clases. También se representó el diseño de la base de datos a través del modelo lógico y físico de la misma, describiéndose algunas de las tablas de dicha base de datos que utiliza este sistema.

Capítulo 4
Implementación y Prueba

Introducción

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes. El desarrollo del software implica una serie de actividades de producción en las que las posibilidades de que aparezca la falibilidad humana son comunes. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. En este capítulo se estructura el modelo de implementación, se realiza el diagrama de componentes, se implementa el sistema y se describen en detalle las pruebas que se le han realizado al sistema.

4.1. Modelo Implementación

Los componentes son la parte física y reemplazable de un sistema, compuestos por un conjunto de interfaces que proporcionan la realización de los mismos. Se usan para modelar los elementos físicos que pueden hallarse en un nodo, por lo que empaquetan elementos como clases, colaboraciones o interfaces. Son independientes entre ellos y tienen sus propias estructuras e implementación. Tienen relaciones de traza con los elementos del modelo de diseño que implementa.

4.1.1. Diagrama de componentes

Un diagrama de componentes es la representación de la forma en que los componentes físicos de un sistema serán separados (pueden ser: archivos, cabeceras, módulos, paquetes). Muestra las dependencias entre estos componentes. Son utilizados para modelar la vista estática de un sistema. Muestra la organización y las dependencias que existen entre un conjunto de componentes.

A continuación se muestran el diagrama de componentes del CU Gestionar actividad científica, dicho diagrama está formado por interfaces, componentes de Symfony y modelos, los cuales están formados por las capas de presentación, lógica del negocio y de datos. Para ver los diagramas de componentes definidos para el resto de los casos de uso del sistema remitirse al Expediente de Proyecto.

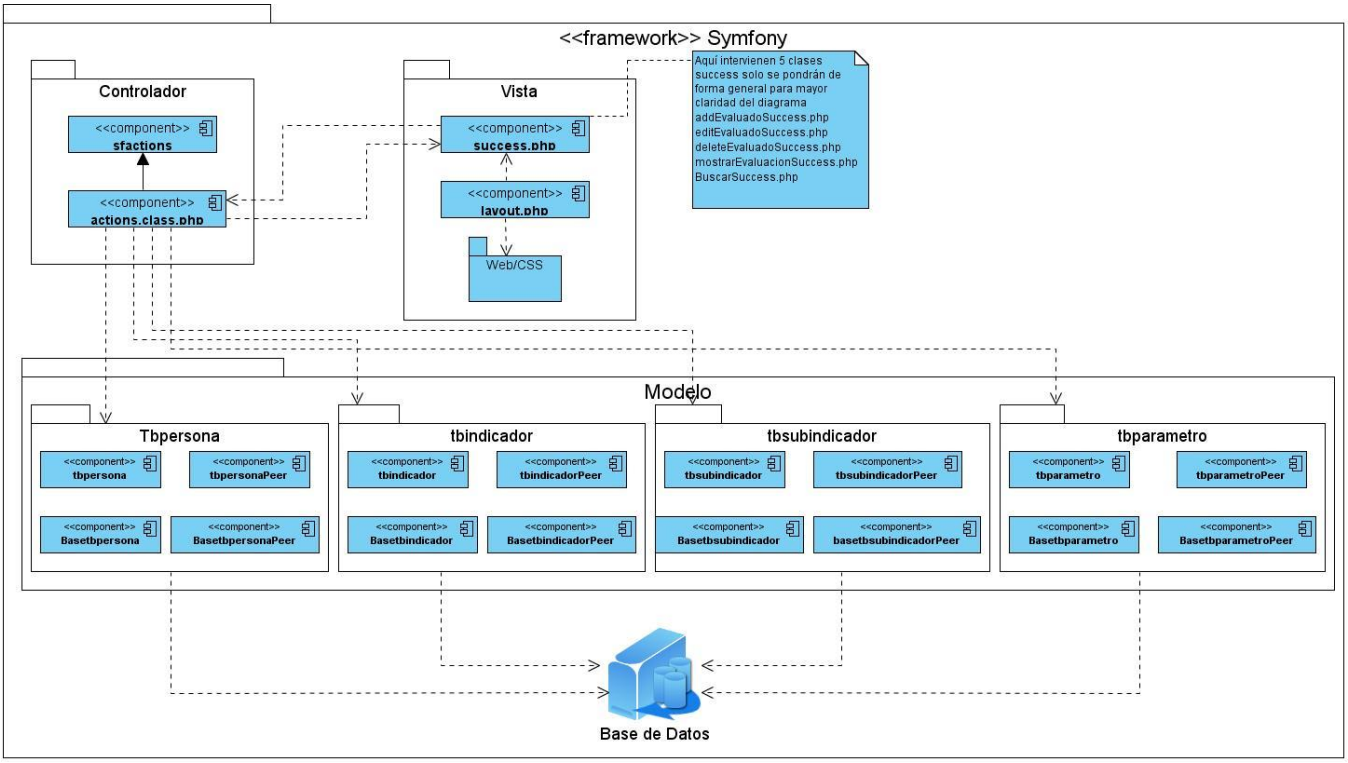


Figura 4.1 Diagrama de Componentes CU *"Gestionar actividad científica".*

Prueba

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores. Constituye una actividad en la cual un sistema o componente de este, es ejecutado bajo ciertas condiciones o requerimientos específicos, en el que los resultados obtenidos son observados y registrados, para la realización posterior de alguna evaluación de dicho componente o sistema.

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas:

- Prueba de desarrollador
- Prueba independiente
- Prueba de Unidad
- Prueba de Integración
- Prueba de sistema
- Prueba de aceptación

El sistema fue probado mediante las pruebas de caja negra y la prueba de aceptación.

Pruebas de Caja Negra: Este tipo de pruebas se pueden llevar a cabo conociendo la funcionalidad específica para la cual fue diseñado el producto, para demostrar que cada función es completamente operativa.

Prueba de aceptación: Prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

4.2. Modelo de prueba

4.2.1. Casos de prueba de caja negra

El sistema se encuentra probado bajo las Pruebas de Funcionalidad (CN), para esto fueron definidos diferentes casos de prueba por cada caso de uso del sistema, chequeando el complimiento de requisitos funcionales que debe tener el software.

Las pruebas de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba tienen como objetivo demostrar que las funcionalidades del software son operativas, que los datos de entrada se aceptan de forma adecuada y que se produce una salida correcta garantizando la integridad de la información que se almacena y procesa.

Pruebas de Caja Negra para el caso de uso Autenticar

Condición de Entrada	Casos Válidos	Casos No Válidos
Usuario	letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Números.
Contraseña	letras, números, caracteres	Dejar el campo vacío.

Caso de Uso:	Autenticar.
Caso de prueba:	Autenticarse introduciendo algún dato incorrecto.
Entrada:	El usuario introduce incorrectamente algún dato de los requeridos para autenticarse. Ejemplo: Usuario: “M@rio”. Contraseña:”campo vacío”.

Resultado:	El sistema muestra un mensaje de error indicando la acción realizada incorrectamente (“Credenciales incorrectas”).
Condiciones:	Dato introducido incorrectamente. Campo vacío.

Tabla 4.1 Caso de prueba de caja negra CU *"Gestionar actividad científica".*

Al aplicarse los casos de pruebas se arrojaron resultados satisfactorios, donde se demostró lo siguiente:

- El sistema responde a las funcionalidades descritas en la descripción de los CU.
- Se logra el objetivo de cada caso de uso en su flujo básico y alternos, demostrándose que están correctamente estructurados.

Para ver el resto de los casos de prueba de caja negra realizados remitirse al Expediente de Proyecto.

4.2.2. Pruebas de Aceptación

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado.

Es imposible que un desarrollador de software pueda prever como utilizara el usuario realmente el programa. Se pueden malinterpretar las instrucciones de uso, se pueden utilizar habitualmente extrañas combinaciones de datos, y una salida que puede parecer clara para el responsable de las pruebas y puede ser ininteligible para el usuario, hay errores que sólo el cliente puede detectar:

Estas pruebas son muy importantes, ya que definen el paso de nuevas fases del proyecto como el despliegue y mantenimiento.

Se emplean dos técnicas para las pruebas de aceptación:

La prueba alfa: Se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un entorno controlado. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.

La prueba beta: se lleva a cabo por los usuarios finales del software en Los lugares de trabajo de Los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación en vivo del software en un entorno que no puede ser controlado por el

desarrollador. El cliente registra todos Los problemas que encuentra durante la prueba beta e informa al desarrollador. Como resultado de los problemas informados durante la prueba beta, el desarrollador del software lleva a cabo modificaciones en el mismo.

El sistema fue probado bajo la prueba alfa y beta, dando como resultado lo siguiente:

Casos de prueba	Solicitudes de cambio	Días de prueba	Revisión
4	1	1	3

Tabla 4.2 Resultados de la prueba de aceptación alfa aplicada al software.

Casos de prueba	Solicitudes de cambio	Días de prueba	Revisión
4	0	1	2

Tabla 4.3 Resultados de la prueba de aceptación beta aplicada al software.

Conclusiones

Durante el proceso de implementación y prueba se deben poner en práctica todas las estrategias posibles para garantizar que el usuario del sistema se encuentre libre de problemas.
En este se obtuvo como uno de los artefactos de este flujo de trabajo los diagramas de componentes, ilustrando en el mismo su distribución en las diferentes capas de la arquitectura definida. A medida que el sistema se va creando es necesario ir realizándole las pruebas para poder prevenir y corregir con tiempo los errores y defectos que este presente. Se realizaron las pruebas al sistema y se encontraron algunas inconformidades en cuanto a la concepción del sistema, las cuales fueron rectificadas.

CONCLUSIONES GENERALES

Al término del presente Trabajo de Diploma quedó desarrollado completamente el Sistema de Gestión y Control para la evaluación del Sistema de Ciencia e Innovación Tecnológica de la Dirección de Deportes en la UCI, alcanzando de esta manera los objetivos propuestos y dándole solución al problema planteado.

Los resultados de este trabajo y la utilización de este sistema serán de gran utilidad para Dirección de Deportes de la universidad, teniendo en cuenta que se eliminará el trabajo manual que hasta el momento se estaba realizando. Proveerá una buena gestión de la evaluación científica dando así un paso para el mejoramiento de la ciencia y la tecnología en la Dirección de Deportes.

RECOMENDACIONES

Luego de dar cumplimiento a los objetivos planteados en este trabajo y teniendo en cuenta las experiencias adquiridas durante el desarrollo del mismo, se recomienda:

- Generalizar la propuesta del desarrollo del Sistema de Gestión y Control para la evaluación del Sistema de Ciencia e Innovación Tecnológica a otros centros del país.
- Realizar una encuesta que permita recoger las preferencias, criterios, sugerencias y satisfacciones de los trabajadores de la Dirección de Deportes de la UCI.
- Dar continuidad al sistema de acuerdo a necesidades futuras que puedan presentarse.

REFERENCIAS BIBLIOGRÁFICAS

Alvarez, Miguel Angel. 2001. desarrolloweb. [En línea] 18 de 7 de 2001. [Citado el: 11 de 1 de 2010.] http://www.desarrolloweb.com/articulos/497.php.

—. 2001. desarrolloweb. [En línea] 9 de 5 de 2001. [Citado el: 11 de 1 de 2010.] http://www.desarrolloweb.com/articulos/392.php.

Amador, Lic. Yimian de Lyz Contreras Díaz y Lic. Soleydi Rivero. 2007. gestiopolis. [En línea] 4 de 10 de 2007. [Citado el: 8 de 1 de 2010.] http://www.gestiopolis.com/administracion-estrategia/sistemas-de-gestion-de-informacion-en-estudio-de-medio-ambiente.htm.

Angela. 2010. gerenciadeproyectoap. [En línea] 14 de 4 de 2010. [Citado el: 9 de 1 de 2010.] http://gerenciadeproyectoap.blogspot.com/2010/04/rup.html.

Arias, Francisco Javier. 2009. edutecno. [En línea] 7 de 2 de 2009. [Citado el: 8 de 1 de 2010.] http://edutecno.org/2009/02/ley_1286de2009/.

Briceño, Edgar Armando Vega. 2005. gestiopolis. [En línea] 6 de 2005. [Citado el: 8 de 1 de 2010.] http://www.gestiopolis.com/Canales4/mkt/simparalas.htm.

Cerda, Felipe. 2009. techbloog. [En línea] 12 de 2 de 2009. http://www.techbloog.com/talks/netbeans65es_cl.pdf.

Conallen, Jim. 1999. *UML Extension for Web Applications v0.* 1999.

Cruz, Jhonatan Alexander Ruiz Zelaya Elmer Josué Ruiz ZelayaLuis Orlando. 2007. slideshare. [En línea] 17 de 6 de 2007. [Citado el: 9 de 1 de 2010.] http://www.slideshare.net/jhonatanalex/modelos-y-capas-de-la-ingenieria-de-software.

Dadak. 2010. buenastareas. [En línea] 28 de 2 de 2010. [Citado el: 8 de 1 de 2010.] http://www.buenastareas.com/ensayos/Colciencias/142424.html.

Daniel Gomez, Elisabet Aranda y Jordi Fabrega. 2008. ecured. [En línea] 20 de 8 de 2008. [Citado el: 9 de 1 de 2010.] http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf.

Fajardo, Jorge Ugarte. 2008. slideshare. [En línea] 2008. [Citado el: 9 de 1 de 2010.] http://www.slideshare.net/gugarte/bpmn-estandar-para-modelamiento-de-procesos-presentation.

Ivar, Jacobson. 2004. *El Proceso Unificado de Desarrollo de Software.* España : s.n., 2004.

Jams. 2007. freedownloadmanager. [En línea] 7 de 3 de 2007. [Citado el: 10 de 1 de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/..

Luciano. 2009. uauf. [En línea] 9 de 2 de 2009. [Citado el: 10 de 1 de 2010.] http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/.

Marianela Lafuente, Carlos Genatios. 2004. voltairenet. [En línea] 1 de 3 de 2004. [Citado el: 8 de 1 de 2010.] http://www.voltairenet.org/article120763.html.

Mario Rodríguez Martín, David Sánchez Fernández, Roberto Santa Escolástica Villoria. 2008. *Patrones de Diseño: Decorator.* 2008.

Masip, David. 2002. desarrolloweb. [En línea] 19 de 7 de 2002. [Citado el: 12 de 1 de 2010.] http://www.desarrolloweb.com/articulos/840.php.

merinde. [En línea] [Citado el: 12 de 1 de 2010.] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=117&Itemid=294.

Miranda, Cristina Lazalde. 2010. centrocamaleon. [En línea] 21 de 1 de 2010. [Citado el: 10 de 1 de 2010.] http://www.centrocamaleon.com/v1/?p=233.

Pecos, Daniel. 2002. danielpecos. [En línea] 7 de 6 de 2002. [Citado el: 12 de 1 de 2010.] http://danielpecos.com/docs/mysql_postgres/x15.html.

Piñeiro, González Guitián y Molina. 2009. eumed. [En línea] 11 de 2009. [Citado el: 8 de 1 de 2010.] http://www.eumed.net/rev/cccss/06/ggmp.htm.

Quiroga, Lourdes Aja. 2002. bvs. [En línea] 10 de 5 de 2002. [Citado el: 8 de 1 de 2010.] http://www.bvs.sld.cu/revistas/aci/vol10_5_02/aci04502.htm.

Ramos, Anay Carrillo. 2008. eumed. [En línea] 2008. [Citado el: 9 de 1 de 2010.] http://www.eumed.net/libros/2009c/587/Lenguaje%20de%20Modelado%20Unificado.htm.

Reynoso, Billy. 2008. microsoft. [En línea] 2008. [Citado el: 11 de 3 de 2010.] http://download.microsoft.com/download/4/F/F/4FF88340-43CC-4C5B-8E50-09002969D0DD/20051122-ARC-BA.ppt..

Ríos, Deysi Yuri Hernández. 2009. dysihdez. [En línea] 15 de 12 de 2009. http://dysihdez.blogspot.com/.

Sanando, Carlos. 2009. dokeoslatinoamerica. [En línea] 9 de 6 de 2009. [Citado el: 11 de 1 de 2010.] http://dokeoslatinoamerica.wordpress.com/2009/01/14/algunos-frameworks-para-php-mas-usados/.

2009. scn. [En línea] 9 de 9 de 2009. [Citado el: 8 de 1 de 2010.] http://www.scn.org/mpfc/modules/mon-miss.htm.

2007. sentidoweb. [En línea] 28 de 8 de 2007. [Citado el: 11 de 1 de 2010.] http://sentidoweb.com/2007/08/28/prado-framework-para-php5.php.

Zaninotto, Fabien Potencier y Francois. 2007. librosweb. [En línea] 20 de 12 de 2007. [Citado el: 11 de 1 de 2010.] http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html.

—. 2007. librosweb. [En línea] 2007. [Citado el: 10 de 1 de 2010.] http://librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html.

Zapata, Carlos Mario. 2009. scielo. [En línea] 23 de 2 de 2009. [Citado el: 9 de 1 de 2010.] http://www.scielo.org.co/pdf/ring/n29/n29a3.pdf.

Zavala-Ruiz. 2008. angelfire. [En línea] 31 de 3 de 2008. [Citado el: 9 de 1 de 2010.] http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html.

BIBLIOGRAFÍA

- Yalina Lamas García y Yadira S Tamarit Quintana. Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis y Diseño del módulo Análisis Químico. 2008. (1 8, 2010)
- Adairis Galván Rodriguez y yanelys Olivera Osorio. Sistema de gestión integral de la planificación y control del servicio de comedores UCI: análisis y diseño del módulo "Administración, análisis económico y gestión de aseguramientos" 2008. (1 8, 2010)
- http://www.colciencias.gov.co/web/guest/sncti. (1 8, 2010)
- http://www.gestiopolis.com/Canales4/mkt/simparalas.htm. (1 8, 2010)
- Grupo Soluciones Innova. http://www.rational.com.ar/herramientas/herramientas.html

(1 8, 2010)

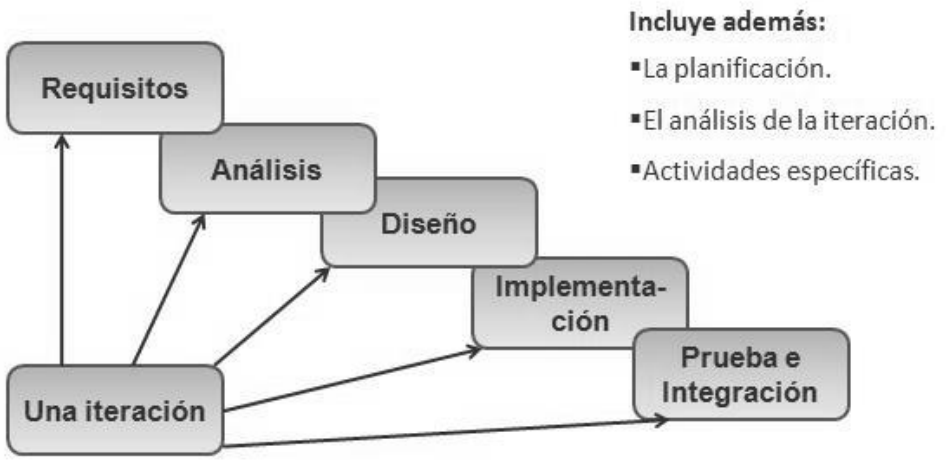
- Jacobson, I., Booch, G. y Rumbaugh, J. El Proceso Unificado de Desarrollo de Software. (1 8, 2010)
- Jacobson, I.; Booch, G. y Rumbaugh, J. El Proceso Unificado de Desarrollo de Software. 2000, [disponible en: http://bibliodoc.uci.cu/pdf/reg00060.pdf] (1 8, 2010)
- http://www.docirs.cl/uml.htm (1 8, 2010)
- http://java.ciberaula.com/articulo/diseno_patrones_j2ee]. (1 8, 2010)
- Ayuda extendida del Rational Rose Enterprise Edition 2003 (1 8, 2010)
- http://www.visual-paradigm.com/product/vpuml. (1 9, 2010)
- Universidad de Chile Departamento de Ciencias de la Computación CC61J - Taller de UML, tema: UML - Diagramas de interacción / auth. Guerrero Luis A. (1 9, 2010)
- http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm (1 9, 2010)
- http://www.sparxsystems.com.ar/products/ea.html (1 9, 2010)
- http://metodologiaxpvsmetodologiarup.blogspot.com/ (1 9, 2010)
- http://www.techbloog.com/talks/netbeans65es_cl.pdf (1 9, 2010)
- http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/ (1 9, 2010)
- http://plataformaeclipse.com/ (1 9, 2010)

- http://www.guia-ubuntu.org/index.php?title=Eclipse (1 9, 2010)
- http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm (1 9, 2010)
- http://www.docirs.cl/uml.htm (1 10, 2010)
- http://danielpecos.com/docs/mysql_postgres/x15.html (1 10, 2010)
- http://www.logisticaonline.org/programas/Curso_Informacion.pdf (1 8, 2010)
- http://www.mitecnologico.com/Main/ElementosDeSistemaDeInformacion (1 8, 2010)
- http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html (1 8, 2010)
- http://kmels.net/files/2009/uvg/cc2003/Resources/Contenidos/XP/xp.pdf (1 8, 2010)
- http://hanzcocchi.net/el-rol-del-analista-en-rup/. (1 9, 2010)
- [http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm (1 9, 2010)
- http://kybele.escet.urjc.es/documentos/HC/Exposiciones/MagicDrawUML.pdf (1 9, 2010)
- http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/. (1 9, 2010)
- http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/ (1 9, 2010)
- http://www.techbloog.com/talks/netbeans65es_cl.pdf (1 10, 2010)
- http://sentidoweb.com/2007/08/28/prado-framework-para-php5.php (1 10, 2010)
- http://dokeoslatinoamerica.wordpress.com/2009/01/14/algunos-frameworks-para-php-mas-usados/ (1 10, 2010)
- http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html (1 10, 2010)
- http://librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html (1 10, 2010)
- Mario Rodríguez Martín, David Sánchez Fernández, Roberto Santa Escolástica Villoria. Patrones de Diseño: Decorator. (1 10, 2010)
- http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/ (1 10, 2010)
- http://www.desarrolloweb.com/articulos/840.php (1 11, 2010)
- http://www.dbrunas.com.ar/postgres/migrapg.pdf (1 11, 2010)

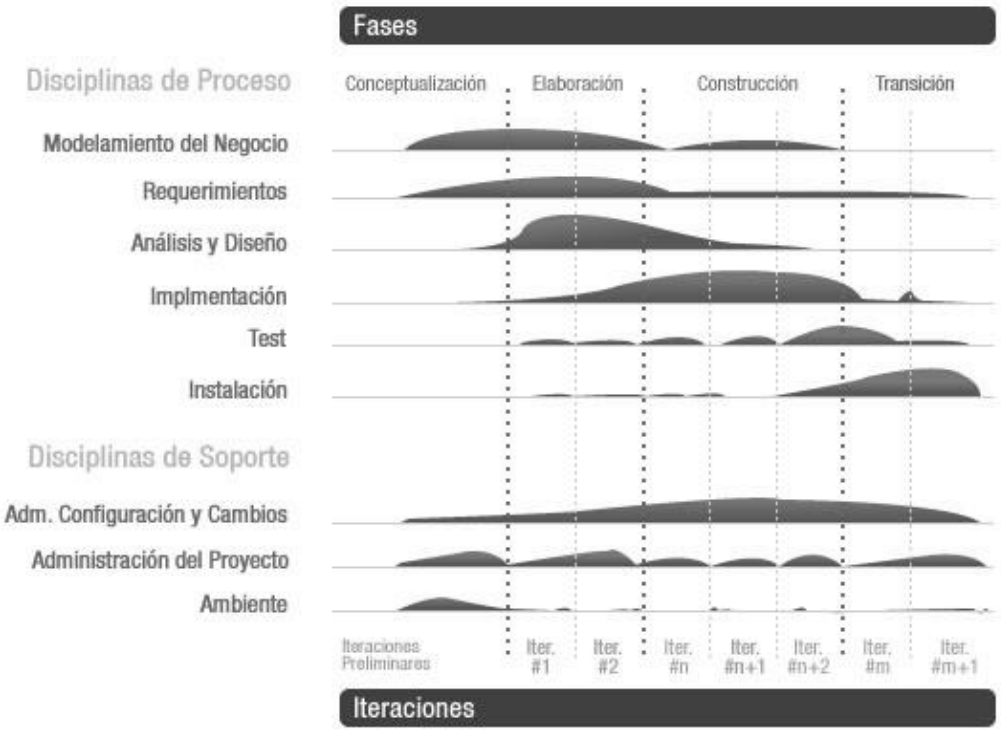
- http://www.desarrolloweb.com/articulos/497.php (1 11, 2010)
- http://www.desarrolloweb.com/articulos/392.php (1 11, 2010)

ANEXOS

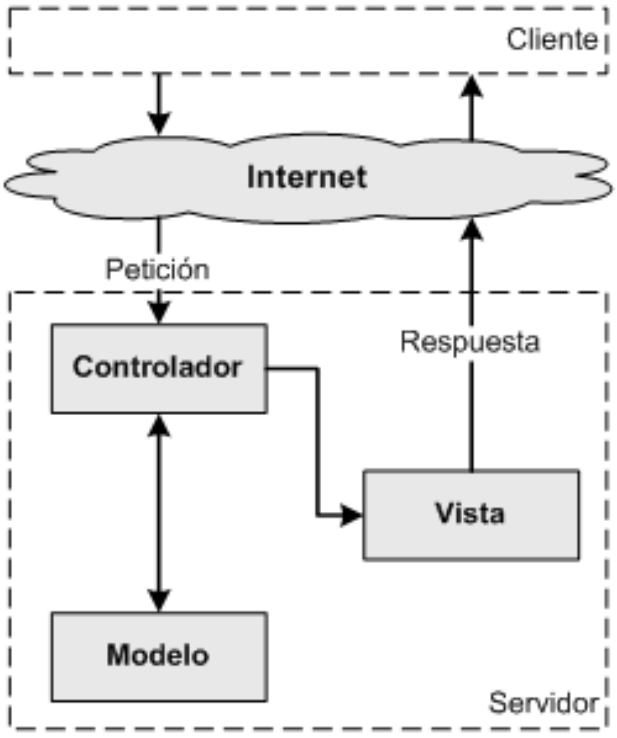
Anexo 1. Una iteración de RUP



Anexo 2. Fases e Iteraciones de RUP



Anexo 3. Funcionamiento del patrón MVC



GLOSARIO DE TÉRMINOS

UCI: Universidad de las Ciencias Informáticas.

Artefacto: pieza de información utilizada o producida por un proceso de desarrollo de software, como un documento externo o el producto de un trabajo. Un artefacto puede ser un modelo, una descripción o un software.

Caso de uso (CU): especificación de las secuencias de acciones, incluyendo secuencias variantes y una descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

GPL: (del inglés General Public License) Licencia pública general. Es una licencia que está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Hardware: componentes físicos que constituyen las computadoras y demás dispositivos periféricos.

Interfaz de usuario: una colección de operaciones que son utilizadas para especificar un servicio de una clase o de un componente.

PHP: (Del inglés Hyper Text Pre-Prosessor / Personal Home Pages). Es un lenguaje de programación usado frecuentemente para la creación de contenido para sitios web con los cuales se puede programar las páginas html y los códigos de fuente.