Universidad de las Ciencias Informáticas
Facultad 15.

Observatorio Tecnológico para la gestión de la investigación de la Facultad 15.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor(es): Alianny Pérez Santoyo.

Orlanyer Outeriño Solis.

Tutor(es): MSc. Yalice Gámez Batista.

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

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

_____ _____

Autor: Alianny Pérez Santoyo. Autor: Orlanyer Outeriño Solis.

_____

Tutor: MSc. Yalice Gámez Batista.

RESUMEN

En el presente trabajo se lleva a cabo el análisis, diseño e implementación de un observatorio tecnológico para la Facultad 15 de la Universidad de las Ciencias Informáticas. El mismo persigue el objetivo de facilitarle a los departamentos y centros de investigación de la facultad la posibilidad de gestionar toda la información necesaria referente a los principales eventos científicos y técnicos, así como cursos de postgrados que se convocan a nivel nacional e internacional de temas relacionados con las líneas de investigación de la facultad.

El resultado de este trabajo un sistema que se pondrá a disposición de todos los estudiantes y trabajadores de la facultad para ofrecerle una vasta documentación referente a los centros de investigación y proyectos previstos. Además, archivará información relacionada con los profesores proporcionándole un currículum en línea para cada uno.

PALABRAS CLAVE

Observatorio Tecnológico, Aplicación Web, Gestor de Contenidos, Vigilancia Tecnológica.

Índice

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

En la actualidad existen grandes desafíos para lograr un mayor desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC). Las TIC hacen referencia a la utilización de medios informáticos para almacenar, procesar y difundir todo tipo de información. Dichas tecnologías constituyen una herramienta fundamental para favorecer a los profesionales y técnicos, y por su intermedio, mejorar la calidad de su servicio y su eficiencia a todas las esferas de la sociedad. Para lograr un desarrollo tecnológico avanzado en todas estas esferas, las estrategias de informatización deben centrarse en los problemas que afectan a la sociedad, logrando elevar la calidad de vida de los ciudadanos.

En nuestros días nos enfrentamos a un enorme desarrollo tecnológico y una gran cantidad de información. Lo que trae consigo que las organizaciones inviertan mucho tiempo en obtener la información que verdaderamente supla sus necesidades, y por ende la toma de decisiones en las mismas se dificulta. Por otro lado, este avance ha permitido que se potencie más la innovación y el desarrollo en las empresas y entidades en general, por lo que se trata de desarrollar, cada vez más, sistemas que optimicen y gestionen la información más actualizada y confiable de vital interés para las instituciones.

Actualmente la creación de Observatorios Tecnológicos en las organizaciones es muy recurrido, pues posibilitan a las mismas anticiparse a las oportunidades, prevenir las amenazas, facilitar la toma de decisiones, potenciar una percepción dinámica del cambio y por ende facilitar la innovación y el desarrollo. Estos servicios operan con éxito en países de Europa, Latinoamérica y Asia, y lo que hacen es gestionar información, fomentar el desarrollo científico y tecnológico y promover iniciativas empresariales, apoyados en estrategias de la Inteligencia Empresarial y la Gestión de Información de forma general.

En la Universidad de las Ciencias Informáticas (UCI) existe un Observatorio Tecnológico que ofrece información sobre eventos científico-técnicos, doctorados en la UCI, así como documentación de proyectos y temas afines. Dicho observatorio está presentando problemas de actualización por lo que no está manteniendo a toda la comunidad universitaria informada correctamente en lo que respecta a la investigación y producción a nivel central.

Actualmente la facultad 15 no cuenta con una aplicación que les permita a los jefes de departamentos archivar y actualizar toda la información referente a los docentes de dicha facultad de forma eficiente y segura, ya que la única vía de realizar este trabajo es mediante el correo electrónico, haciendo un uso inadecuado de dicha tecnología. Por otro parte los asesores de investigación no cuentan con el medio adecuado para brindar detalles de cada uno de los centros

productivos y a su vez de los proyectos correspondientes. Así mismo no hay implementado un mecanismo mediante el cual se le pueda avisar a los profesores con cierta anticipación de la realización de algún evento científico-técnico, ofrecer información sobre los cursos de postgrado que se planifiquen y dedicar un lugar donde los docentes puedan archivar toda la información importante para actualizar su currículum.

Atendiendo a la situación problémica planteada anteriormente deviene el problema: no existe un medio que permita la divulgación y control de la información asociada a la superación e investigación de los profesores.

Por consiguiente, el **objetivo de la investigación** es: desarrollar una solución informática que permita la gestión y control de la información referente a la investigación y postgrado de la Facultad 15.

Objeto de estudio: Proceso de desarrollo del software.

Campo de acción: Observatorios Tecnológicos para la gestión y control de la investigación.

Hipótesis: Si se desarrolla la solución informática entonces se mejorará la gestión y control de la investigación en la Facultad 15.

Del objetivo general se derivan los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación.
2. Definir características acordes con el Observatorio Tecnológico que se necesita.
3. Obtener artefactos del análisis y diseño.
4. Implementar la solución propuesta.
5. Validar la solución propuesta.

Para dar solución a los objetivos específicos anteriormente planteados se trazaron las siguientes tareas:

- Tarea 1: Revisión y análisis de las bibliografías correspondientes al tema.
- Tarea 2: Selección de la metodología de desarrollo, paradigmas de programación, lenguajes de programación y herramientas que faciliten la creación y garantice la calidad del sistema.
- Tarea 3: Confección de los diagramas de clases del análisis con los diagramas de colaboración correspondientes.
- Tarea 4: Confección de los diagramas de clases del diseño con los diagramas de secuencia correspondientes.
- Tarea 5: Diseño de las bases de datos que soporten las funcionalidades del sistema.
- Tarea 6: Proposición de un proceso que garantice los principios básicos de la seguridad de la información.

- Tarea 7: Desarrollo de los casos de prueba que permitan validar la solución propuesta.

Métodos de Investigación:

Métodos Teóricos

- *Analítico-Sintético:* Se analizan las teorías planteadas, documentos, planteamientos, entre otros, para luego extraer los elementos más importantes que se relacionan con el objeto de estudio y adaptarlo a la situación.
- *Inductivo-Deductivo:* Se hace uso de deducciones para llegar a tener una visión clara de lo que se quiere hacer y adquirir así nuevos conocimientos. Este método se aplica en inducción y deducción de los lenguajes de programación que se va a escoger.
- *Histórico Lógico:* Se estudian los distintos sistemas encargados de la gestión de la investigación, detallando su uso, evolución y desarrollo.

Métodos empíricos

- *Entrevista:* Se realizan entrevistas a especialistas en el tema para determinar el trabajo que se genera en los procesos a desarrollar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace una descripción de los conceptos que son utilizados en la investigación y que pueden resultar difíciles de comprender, además de exponer de forma exhaustiva el objeto de estudio con el fin de lograr un mayor conocimiento del problema a resolver. Se realiza un estudio de los diferentes Observatorios Tecnológicos para determinar las características funcionales de los mismos. Además, se hace un análisis de las tecnologías actuales y las principales herramientas que pudieran ser adecuadas para la construcción del sistema que se pretende desarrollar.

1.1.1 Definición de Observatorio Tecnológico

En la actualidad la tecnología informática se ha vuelto muy importante para la sociedad, cada vez, más personas tienen una computadora, acceden a Internet y hacen uso de tecnologías que les faciliten la solución de problemas en la vida cotidiana.

La sociedad conoce muy poco sobre tecnologías que impactan, es por eso que se necesitan lugares que permitan la interacción de usuarios consultando, publicando información y/o realizando encuestas sobre un tema determinado que facilite que investigadores interesados por el avance tecnológico del país puedan tomar esta información y crear una serie de informes para el análisis y la observación de dicha tecnología. Dichos lugares conocidos como Observatorios Tecnológicos se han creado para la divulgación de proyectos que vayan en pro del desarrollo de un país.

¿Qué es un Observatorio Tecnológico?

"Un Observatorio Tecnológico es un instrumento colectivo de potenciación de la capacidad de detección de cambios tecnológicos, de su grado de maduración y de las oportunidades de mercado. Se basa en el empleo de técnicas y procedimientos de vigilancia tecnológica y está dirigido a un colectivo bien definido de usuarios". (AITEX, 2002)

¿Por qué utilizar un Observatorio Tecnológico?

El futuro de la empresa depende de su capacidad de reacción al cambio, lo que supone una gran anticipación a las innovaciones tecnológicas y obliga a las empresas a investigar acerca de las limitaciones y las oportunidades que se tienen a la hora de adaptarse a la evolución de la ciencia y la tecnología.

La vigilancia tecnológica es la herramienta para intentar comprender y explicar la evolución de la tecnología y permitir a las empresas anticiparse a los efectos negativos que sobre su actividad puede tener, y aprovechar las oportunidades que la misma ofrece, ayudando a la identificación de los

escenarios más probables y al estudio del impacto previsible sobre la actividad de la empresa de las tecnologías emergentes y los desarrollos tecnológicos que en el inmediato futuro se deriven de su evolución. (La Coctelera, 2008)

1.1.2 Surgimiento, evolución e importancia de los Observatorios Tecnológicos

El vocablo "Observatorio" se asocia a un lugar (edificio) o posición que sirve para hacer observaciones por medio de instrumentos apropiados y dedicados a observaciones comúnmente astronómicas o meteorológicas; pero hoy las empresas lo han asumido con gran popularidad como herramienta colectiva para obtener y ofertar información relevante, orientado al apoyo de los procesos de innovación tecnológica de las organizaciones empresariales, sociales y/o gubernamentales.

Los observatorios se definen como el producto de la unión de un grupo de personas con una amplia visión en materia de negocios, basada en la experiencia que han acumulado a lo largo del tiempo de trabajo en la empresa y grupos de expertos en la materia a tratar.

Al dar soporte a procesos de innovación abierta, los Observatorios Tecnológicos, integran fuentes de información científico-tecnológicas tradicionales (bases de datos de información científica, web, bases de patentes, normas, etc.) con fuentes derivadas del proceso de creación de nuevas tecnologías por parte del sector académico (eventos científicos, bases de expertos, foros virtuales, reportes de investigación, etc.). Es decir, integran conocimiento explícito documental y conocimiento implícito de expertos del área de conocimiento y sector que atiende el Observatorio Tecnológico, lo que permite a sus beneficiarios (empresas, organizaciones sociales y/o gubernamentales), reducir el riesgo de los procesos de innovación tecnológica y detectar oportunidades del entorno, que optimicen la toma de decisiones estratégicas y mejoren la eficiencia y la productividad de los diversos sectores. Es por eso que las organizaciones empresariales necesitan de la observación científica - tecnológica ya que en la mayoría de las ocasiones los investigadores que trabajan en proyectos de investigación no tienen un claro entendimiento del mundo empresarial porque provienen del mundo académico en el cual no se maneja el concepto cliente. (Solórzano, 2007)

Por otra parte, los desarrolladores de productos y servicios en la empresa tienen un gran conocimiento de cómo satisfacer las necesidades de los clientes, orientados a la obtención de resultados a corto plazo, pero no cuentan con suficiente tiempo para conocer los avances científicos - tecnológicos. Por lo que es importante, para mejorar los resultados de la empresa, la unión de estos dos grupos, lo que constituye el papel fundamental de los observatorios tecnológicos: vincular a un grupo de personas con amplia visión de negocios con grupos de investigadores, para de esta

manera, conocer e influir en la evolución de las tecnologías relevantes como factor diferenciador de la competencia. (García, y otros, 2009)

Evolución de los Observatorios Tecnológicos

La aceleración del ritmo de lanzamiento de productos y de mejora de procesos, necesarios para mantener la competitividad de las empresas, así como la escasez de capital, obligan a optimizar los recursos destinados al desarrollo de productos y servicios. Por esta razón, en el mundo surgen una serie de herramientas para minimizar riesgos y facilitar el desempeño de las empresas. (La Coctelera, 2008)

Una de esas herramientas es la vigilancia tecnológica que contribuye a reducir las decisiones erróneas en el inicio de un proyecto de investigación y desarrollo, en las fases de lanzamiento previo al mercado, etc. La vigilancia tecnológica aparece en 1985 como una de las seis funciones clave para la administración de la tecnología.

Sin embargo, la adopción de ese elemento ha dependido del país y de la empresa en cuestión en dependencia del grado de necesidad que tenga la empresa de innovar para competir. De tal manera que para satisfacer dicha necesidad se obliga a disponer de un sistema organizado y colectivo de vigilancia. La complejidad y abundancia de información dificulta la adopción de sistemas o servicios de vigilancia tecnológica.

En algunos países la vigilancia tecnológica ya forma parte de la cultura empresarial, este es el caso de Instituto Tecnológico Textil de España, que puso en marcha en mayo de 2002 un observatorio tecnológico textil con el objetivo de posibilitar a las empresas usuarias a anticiparse a las oportunidades, prevenir las amenazas y dinamizar su respuesta al cambio. (La Coctelera, 2008)

Un Observatorio cuenta con una interfaz de usuario dónde la página principal es la página de búsquedas y los servicios que ofrece es la generación de mapas de tecnología, de oportunidades y de informes técnicos.

En España, Francia, Japón y Estados Unidos, existen empresas de consultoría en Vigilancia Tecnológica e Inteligencia Competitiva, ofreciendo los servicios de: (La Coctelera, 2008)

- Implantación de Unidades de Vigilancia Tecnológica e Inteligencia Competitiva.
- Formación en Vigilancia Tecnológica e Inteligencia Competitiva.
- Sistemas de Vigilancia en Internet.
- Alertas personalizadas.
- Vigilancia de Patentes en Internet.

Objetivos que persiguen los Observatorios Tecnológicos

Los Observatorios Tecnológicos persiguen los siguientes objetivos fundamentales (Espinosa, 2008):

- Ayudar a decidir a la dirección de las empresas a configurar su estrategia de innovación.
- Permitir incorporar nuevos avances tecnológicos a los propios productos y diversificar la propia cartera de los mismos.
- Identificar oportunidades de inversión y comercialización.
- Identificar amenazas potenciales que pueden suponer pérdida de cuota de mercado.
- Contribuir a abandonar a tiempo un determinado proyecto de innovación tecnológica.
- Identificar posibles socios para colaboraciones.

1.1.3 Observatorios Especializados

En el mundo existen muchas empresas que cuentan con Observatorios Tecnológicos como herramientas que les ayudan a medir y analizar la competitividad de las mismas mediante la vigilancia tecnológica. Dentro de los observatorios reconocidos a nivel mundial se pueden mencionar:

Observatorio de Prospectiva Científica y Tecnológica de Argentina: tiene por misión relevar el desarrollo de la Sociedad de la Información y el impacto de las Tecnologías de la Información y la Comunicación en dicho país.

Observatorio de Competitividad de la República Dominicana: Se trata de un sistema integrado de información para monitorear y dar seguimiento a los avances alcanzados por el país en el marco del Plan Nacional de Competitividad Sistémica. Apoya la formulación de políticas públicas en materia de competitividad, promoviendo la toma de decisiones más acertadas.

Observatorio Aragonés de la Sociedad de la Información: Su objetivo fundamental es servir como instrumento de información y formación sobre el impacto de las nuevas tecnologías en Aragón, su uso y su evolución a lo largo de estos últimos años. También se pretende divulgar el potencial de las TIC en el territorio aragonés mediante elementos que recojan la evolución de la Sociedad de la Información en Aragón, así como atraer y agrupar fuerzas y opiniones en torno a las TIC.

En Cuba también existen varias organizaciones que se apoyan de los Observatorios Tecnológicos para identificar los nuevos avances tecnológicos ayudando a la dirección de estas entidades a configurar su estrategia de innovación. Entre los observatorios reconocidos en el ámbito nacional se encuentran:

Observatorio Cubano de Periodismo: está encaminado a la realización de la vigilancia informacional en la rama del periodismo en Cuba. Creado como instrumento capaz de aprovechar

sistemáticamente la información y el conocimiento para elevar al máximo la capacidad de respuesta respecto al mensaje cubano, tanto en la opinión pública nacional como internacional.

El Observatorio Cubano de Ciencia y Tecnología: tiene como misión analizar y evaluar las perspectivas de los temas estratégicos del desarrollo de la ciencia y la innovación tecnológica en Cuba y su relación con las prioridades del desarrollo económico, social y medioambiental nacional.

El Observatorio Nacional de Ciencia, Tecnología e Innovación (ONCTI): es un programa coordinado por la Dirección General de Prospección y Planificación del Ministerio de Ciencia y Tecnología de Venezuela. Vale destacar que esta Dirección General tiene entre sus competencias, generar los indicadores y estadísticas del Sistema Nacional de Ciencia, Tecnología e Innovación (SNCTI) y promover el monitoreo científico y tecnológico.

La Universidad de las Ciencias Informáticas es otra de las entidades cubanas que hace uso de un Observatorio Tecnológico:

Servicio de Vigilancia Tecnológica (Vigitec): Sirve de instrumento para monitorear el desarrollo científico, tecnológico e innovador en la universidad. Además, de brindar información relacionada con los proyectos más importantes que se desarrollan en el centro.

Teniendo en cuenta los grandes beneficios que ofrecen los Observatorios Tecnológicos, y la situación problémica existente en la Facultad 15, se determinó implementar un observatorio encargado de la vigilancia tecnológica, sobre los temas investigativos, de superación y desarrollo tecnológico de la facultad y a la vez sirviera como mecanismo de retroalimentación a partir de la actividad científica de los profesores.

Partiendo de la necesidad de contar con un instrumento de vigilancia tecnológica para dar solución a la problemática planteada; se analizaron los observatorios que existen actualmente y se decidió no usar ninguno de estos por la siguiente razón:

El diseño e implementación de un observatorio implica determinar y conocer los escenarios de trabajo a los que se enfocará a escala nacional e internacional, establecer las características de los usuarios a los que se dirigen los servicios de alto valor agregado, así como establecer los perfiles que interesan a estos usuarios. Por tanto, estos observatorios no conseguirían satisfacer las carencias de la facultad, los servicios que ofrecen están orientados hacia las necesidades específicas de las propias entidades que los sustenten.

La forma óptima de obtener la aplicación es implementando sus funcionalidades mediante la programación Web y utilizando un Sistema Gestor de Contenidos. Las aplicaciones web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de escritorio,

pues no dependen de ningún sistema operativo o configuración de hardware específica, basta tener un navegador web mínimamente actualizado para utilizarlas. Pueden ser utilizadas por múltiples usuarios al mismo tiempo, ya que el acceso a los servicios se realiza desde varias computadoras permitiendo la compartición de datos por parte de varios usuarios, y a la vez le garantiza alta privacidad mediante el uso de un sistema seguro de autenticación. Además, tienen mucho sentido en aplicaciones en-línea, característica que debe cumplir la aplicación que se necesita. Por su parte los Sistemas Gestores de Contenidos son herramientas que permiten darle un tratamiento más eficiente a la información contenida en las aplicaciones Web. Precisamente, no disponer de una herramienta que facilite la gestión de los contenidos de un portal Web, produce consecuencias negativas, tanto para los usuarios como para la entidad que promociona sus servicios; entre estas se destacan las siguientes:

- El administrador de la aplicación Web y el personal de desarrollo terminan por verse desbordados por el número creciente de peticiones y tareas relacionadas con la publicación de contenidos y la administración de la aplicación.
- Los contenidos no están disponibles cuando se espera que estén, ocurren atrasos que afectan la validez e integridad de la información.
- Resulta muy difícil mantener actualizadas las herramientas de navegación a medida que se añaden y eliminan páginas al Sitio Web, al tratarse de un proceso que requiere la actualización manual de múltiples páginas.
- Resulta muy difícil, si no imposible, reutilizar contenidos.

De esta forma, puede afirmarse que una organización gestiona correctamente sus contenidos si responde de forma ordenada a los problemas anteriormente citados, y si implementa las herramientas y procedimientos necesarios para dar respuesta a dichos problemas. Además, el equipo de desarrollo debe regirse por una metodología de desarrollo de software para proveer un entorno de procesos configurables basado en estándares; permitiendo a cada miembro del equipo fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades de desarrollo. Así mismo, para garantizar que esta base de conocimientos se obtenga en tiempo y forma debe utilizarse alguna herramienta que soporte el ciclo completo de desarrollo de software de acuerdo con la metodología seleccionada.

1.2 Metodología de desarrollo de software.

La metodología en el desarrollo de un software se puede definir como un conjunto de pasos y procedimientos que sirven de apoyo para la guía de realizar un software de calidad. En un proyecto

de software la metodología de desarrollo define ¿Quién debe hacer qué?, y ¿Cuándo debe realizarlo?

1.2.1 Metodología de desarrollo SCRUM.

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación**,** la competitividad y la productividad son fundamentales.

Esta metodología también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado**,** los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto. (Albaladejo, 2009)

Scrum como metodología de desarrollo de software no es recomendable para algunos contextos y proyectos de software, donde el cliente no pueda estar en colaboración permanente con el equipo, condiciones que presenta el proyecto de creación del Observatorio Tecnológico precisamente, por lo que su aplicación al mismo no es recomendable.

1.2.2 Metodología de desarrollo Extreme Programing (XP)

La Programación Extrema surge ideada por Kent Beck, como proceso de creación de software diferente al convencional. En palabras de Beck: "XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software".

XP es una metodología ágil de desarrollo de software en la que se postula que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo. Se caracteriza por la simplicidad,

la comunicación, la realimentación o reutilización del código desarrollado así como las pruebas unitarias. (Welicki, León E)

La Programación Extrema no es la metodología de desarrollo de software adecuada para utilizar en el proyecto de creación del Observatorio Tecnológico puesto que esta concibe el flujo de trabajo de Requisitos con un excesivo dinamismo sin hacer hincapié en el seguimiento al cambio de los mismos. No obstante, es bueno tomar en cuenta de la misma las buenas relaciones y la abierta comunicación que exige entre los miembros del equipo de desarrollo y con el cliente.

1.2.3 Metodología de desarrollo RUP

La metodología RUP (Rational Unified Process) que en su traducción al Español significa Proceso Unificado de Rational, es utilizada para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. Está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas. Utiliza el Lenguaje Unificado de Modelado o UML, por sus siglas en inglés, Unified Modeling Language como soporte a la metodología. Está dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental. (Jacobson y otros, 2000)

RUP se caracteriza por dividir el ciclo de vida de la producción del software en 4 fases:

1. Inicio: es donde se determina la visión del proyecto, o sea, se comprende el entorno y se determina el alcance del producto.
2. Elaboración: en esta etapa se establecen los cimientos de la arquitectura y se analiza el dominio del problema.
3. Construcción: en esta fase se obtiene la capacidad operacional inicial del producto.
4. Transición: se obtiene el release o liberación del producto y se pone en manos de los usuarios finales.

Las fases antes mencionadas se llevan a cabo desarrollando nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo:

1. Modelado de Negocio.
2. Requerimientos.
3. Análisis y Diseño.

4. Implementación.
5. Prueba.
6. Despliegue.
7. Configuración y Control de Cambios.
8. Gestión de Proyectos.
9. Entorno.

Características de RUP (Kruchten, 2000)**:**

- Centrado en los modelos: Los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema.
- Guiado por los casos de uso: Los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar.
- Iterativo e incremental: Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

Beneficios que brinda RUP (Expósito, 2008)**:**

- Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos.
- Permite a la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible.
- Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos sus miembros.
- Optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria.
- Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas.
- Proporciona guías explícitas para áreas tales como modelado de negocios, arquitectura Web, pruebas y calidad.

- Unifica todo el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software.
- No solo garantiza que los proyectos abordados serán ejecutados íntegramente sino que además evita desviaciones importantes respecto a los plazos.

1.2.4 Consideraciones de la Metodología de Desarrollo de Software

En todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software de calidad superior a tiempo. RUP es una plataforma flexible de procesos de desarrollo de software que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto.

Después de realizar el estudio de las principales metodologías de desarrollo de software por parte del equipo de desarrollo se tomó RUP como proceso rector por adaptarse a las características y complejidad de este proyecto de software. Como una plataforma de procesos que abarca todas las prácticas de la industria, configurable para proyectos pequeños, permitiendo seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto.

Esta metodología describe cómo utilizar de forma efectiva, reglas de negocio y procedimientos comerciales probados en el desarrollo de software para equipos de desarrollo de software, conocidos como “mejores prácticas”. Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Una de las mejores prácticas centrales de RUP es la noción de desarrollar iterativamente. Organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada flujo de trabajo variará a través del ciclo de vida. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, con un progreso demostrable y frecuente entregas de ejecutables.

1.3 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un modo estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. No puede compararse con la programación estructurada, pues con UML no se programa, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una

forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos. (Jacobson y otros, 2000).

Como lenguaje de modelado se utilizará UML debido a que la metodología de desarrollo antes seleccionada, RUP, esta soportada por este lenguaje.

1.4 Herramientas CASE

CASE es una sigla, que corresponde a las iníciales 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 permiten a los desarrolladores modelar y documentar sus artefactos, cubriendo el ciclo de vida del proceso de desarrollo de software. Una herramienta CASE puede incluir: diccionario de datos, herramientas de diseño, herramientas de desarrollo de modelo de datos y herramientas para el desarrollo de prototipos. Para el desarrollo de la solución informática es necesario utilizar algunas herramientas para facilitar el trabajo y por consiguiente realizarlo con la calidad requerida.

1.4.1 Herramienta Rational Rose

Es una herramienta desarrollada por los creadores de UML que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los 42 componentes, transición a los usuarios y certificación de las distintas fases. Proporciona un lenguaje común de modelado que facilita la creación de software con calidad.

Características que tiene la herramienta (Rational, 2009):

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++.
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- Soporte de ingeniería directa y/o inversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo-código configurable.
- Soporte Enterprise Java Beans 2.0.
- Capacidad de análisis de calidad de código.

- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web.
- Es de software propietario, el costo de una licencia para un usuario único es de 4,741 euros contando los valores agregados que significa el mantenimiento por 12 meses.

1.4.2 Herramienta Visual Paradigm

Visual Paradigm es una herramienta 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 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.

Características generales (SlideShare, 2009)**:**

- Soporte de UML versión 2.1
- Diagramas de Procesos de Negocio: Proceso, Decisión, Actor de negocio, Documento.
- Interoperabilidad con modelos UML2 (meta modelos UML 2.x para plataforma Eclipse) a través de XMI.
- Ingeniería inversa: Código a modelo, código a diagrama, ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código: Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso: Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Generación de bases de datos: Transformación de diagramas de Entidad-Relación en modelos de base de datos.
- Ingeniería inversa de bases de datos: Desde sistemas gestores de bases de datos (SGBD) existentes a diagramas de entidad-relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas: Reorganización de las figuras y conectores de los diagramas UML. Importación y exportación de ficheros XMI.

1.4.3 Consideraciones de las Herramientas CASE

Partiendo del estudio realizado a las principales herramientas CASE existentes, se decidió utilizar Visual Paradigm. Se hace esta elección, principalmente, por ser una herramienta multiplataforma,

facilidad que pocas herramientas CASE brinda, permite exportar documentos, es robusta y de fácil uso. Además, se sustenta la elección de la misma en el hecho de que la UCI cuenta con la licencia para el uso de la misma.

1.5 Lenguajes de Programación Web

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Se han investigado los lenguajes de programación más utilizados para la programación web: Python, Java, y PHP, para hacer una comparación y seleccionar el más adecuado para implementar la solución informática.

1.5.1 Lenguaje de Programación Python

Python es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. Se compara habitualmente con Perl, Scheme, Java y Ruby. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation. Es multiparadigma, es decir, más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Otros muchos paradigmas más están soportados mediante el uso de extensiones. Usa tipo de dato dinámico y conteo de referencias, para el manejo de memoria. Una característica importante de Python es la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamada ligadura dinámica de métodos). Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de cadenas, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.

Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

Otra característica importante del lenguaje es la facilidad de extensión, pues nuevos módulos se pueden escribir fácilmente en C o C++. Puede utilizarse como un lenguaje de extensión para módulos y aplicaciones que necesitan de una interfaz programable. Aunque su diseño es de alguna

manera contrario a la programación funcional tradicional del Lisp[1], existen bastantes analogías entre Python y los lenguajes de la familia Lisp como puede ser Scheme. La librería estándar de Python es muy amplia, puede ayudar a hacer varias cosas que involucran: expresiones regulares, generación de documentos, evaluación de unidades, pruebas, procesos, bases de datos, navegadores web, CGI, FTP, correo electrónico, XML, XML-RPC, HTML, archivos WAV, criptografía, etc. (Alvarez, 2003)

1.5.2 Lenguaje de Programación PHP

Páginas de Hipertexto Preprocesador (PHP) es un lenguaje interpretado de alto nivel impregnado en páginas HTML, sus instrucciones son ejecutadas del lado del servidor. Como producto de código abierto, PHP es mantenido y actualizado por un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML, XML o WML.

Ventajas de PHP (Martínez, 2007)**:**

- Se caracteriza por ser un lenguaje muy rápido, y fácil de aprender.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (Documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación.

[1] **Lisp** o (LISP, Acrónimo de LISt Processing), es una familia de lenguajes de programación de computadora de tipo funcional con una larga historia y una sintaxis completamente entre paréntesis.

- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Con PHP se puede hacer cualquier cosa que se puede realizar con un script Common Gateway Interface (CGI), como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.

1.5.3 Consideraciones de los Lenguajes de Programación Web

Se ha decidido utilizar PHP, pues en cuanto a la seguridad, es un poderoso lenguaje e intérprete, ya sea incluido como parte de un servidor web en forma de módulo o ejecutado como un binario CGI separado, es capaz de acceder a archivos, ejecutar comandos y abrir conexiones de red en el servidor. Estas propiedades hacen que cualquier cosa que sea ejecutada en un servidor web sea insegura por naturaleza y PHP está diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI que Perl o C, y con la selección correcta de opciones de configuración en tiempos de compilación y ejecución, y siguiendo algunas prácticas correctas de programación. Además, presenta el problema de contar con escasa documentación, ejemplos y número de aplicaciones, no sucediendo esto con el lenguaje PHP.

1.6 Sistemas de Gestión de Contenidos

Un sistema de gestión de contenidos (en inglés Content Management System, abreviado CMS) es un programa que permite crear una estructura de soporte para la creación y administración de contenidos, principalmente en páginas web. Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio, permitiendo manejar de manera independiente el contenido y el diseño. Genera páginas dinámicas interactuando con el servidor para generar la página web a petición del usuario, con el formato predefinido y el contenido extraído de la base de datos del servidor. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Esto permite gestionar, bajo un buen formato, la información del servidor, reduciendo el tamaño de las páginas para descarga y reduciendo el coste de gestión del portal con respecto a una página estática, en la que cada cambio de diseño debe ser realizado en todas las páginas, de la misma forma que cada vez que se agrega contenido tiene que maquetarse una nueva página HTML y subirla al servidor.

Realizar el portal del Observatorio Tecnológico para la facultad con la ayuda de un Sistema Gestor de Contenidos reportaría grandes beneficios tanto a los desarrolladores como a los usuarios finales del sitio. Esto facilita el acceso a la publicación de contenidos a un rango mayor de usuarios. Permite

que sin conocimientos de programación ni maquetación cualquier desarrollador pueda indexar contenido en el portal. Además permite la gestión dinámica de usuarios y permisos, la colaboración de varios usuarios en el mismo trabajo y la interacción mediante herramientas de comunicación. Además, los gestores de contenidos garantizan que la aplicación a construir sea escalable y tenga una interfaz amigable y sencilla.

Tipos de Gestores de Contenidos (CMS) (Franco, 2008)

Los gestores de contenido se pueden segmentar según diferentes criterios:

- Según el lenguaje de programación empleado:
  - Active Server Pages, Java, PHP, Ruby on Rails, Python.
- Según la propiedad del código:
  - Código abierto; permite que se desarrolle sobre el código.
  - Código propietario; sólo su desarrollador puede desarrollar la aplicación.
- Según el tipo de uso o funcionalidades:
  - Genéricos*:* Ofrecen la plataforma necesaria para desarrollar e implementar aplicaciones que den solución a necesidades específicas. Pueden servir para construir soluciones de gestión de contenidos, para soluciones de comercio electrónico, blogs, portales, etc. Ejemplos: Zope, OpenCMS, Typo3, Apache Lenya.
  - Foros**:** sitio que permite la discusión en línea donde los usuarios pueden reunirse y discutir temas en los que están interesados. Ejemplos: phpBB, MyBB.
  - Blogs**:** Publicación de noticias o artículos en orden cronológico con espacio para comentarios y discusión. Ejemplos: Wordpress, Typo.
  - Wikis**:** Sitio web dónde todos los usuarios pueden colaborar en los artículos, aportando información o reescribiéndola. También permite espacio para discusiones. Indicado para material que irá evolucionando con el tiempo. Ejemplos: Mediawiki, Tikiwiki.
  - eCommerce: Son Sitios web para comercio electrónico.
  - Portal**:** Sitio web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad. Ejemplos: PHPNuke, Drupal, Plone.
  - Galería**:** Permite administrar y generar automáticamente un portal o sitio web que muestra contenido audiovisual, generalmente, imágenes. Ejemplo: Gallery.

- e-Learning**:** Sirve para la enseñanza de conocimientos. Los usuarios son los profesores y estudiantes, existen aulas virtuales donde se ponen a disposición el material del curso. La publicación de un contenido por un profesor es la puesta a disposición de los estudiantes, en un aula virtual, de ese contenido. Ejemplo: Moodle.
- Publicaciones digitales**:** son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc. Ejemplo: ePrints.

Se puede hacer una división de los CMS según el tipo de licencia escogido. Por una parte están los CMS comercializados por empresas que consideran mantener el código fuente en propiedad. Por la otra están los de código fuente abierto, desarrollados por individuos, grupos o empresas que permiten el acceso libre y la modificación del código fuente.

Se podría considerar que la herramienta comercial será más estable y coherente al estar desarrollada por un mismo grupo. En la práctica esta ventaja no es tan grande, ya que los CMS de código abierto también están coordinados por un único grupo o por empresas, de forma similar a los comerciales. Utilizar una herramienta de gestión de contenidos de código abierto tiene otras ventajas que hace decidirse a la mayoría de usuarios:

- Habitualmente todo el software de código abierto es de acceso libre, es decir, sin ningún costo en licencias. Sólo en casos aislados se hacen distinciones entre empresas y entidades sin ánimo de lucro o particulares. En comparación, los productos comerciales pueden llegar a tener un coste que sólo una gran empresa puede asumir.
- En cuanto al soporte, los CMS comerciales acostumbran a dar soporte profesional, con un coste elevado en muchos casos, mientras que los de código abierto se basan más en las comunidades de usuarios que comparten información y solución a los problemas.
- En el mercado hay CMS de calidad tanto comerciales como de código abierto. Muchos CMS de código abierto están poco elaborados aunque en plena evolución. En definitiva, un buen CMS de código abierto es mucho más económico que su homólogo comercial, con la ventaja de disponer de todo el código fuente y de una extensa comunidad de usuarios.

En resumen, los CMS basados en código abierto son los más conocidos, por su facilidad y bajo coste en la instalación. Es suficiente en la mayoría de los casos con un servidor Apache que pueda leer código PHP y una base de datos MySQL**.** A continuación aparece una relación de los CMS de código abierto más utilizados.

1.6.1 Sistema de Gestión de Contenidos WordPress

Wordpress es un Sistema de Gestión de Contenidos de gran sencillez de uso hasta el punto donde solo puede usarse para publicar artículos, por lo que es muy utilizado para la creación de Blog. Aclarar que Wordpress (núcleo) no es un CMS propiamente dicho, aunque puede evolucionar hasta llegar a serlo, y también puede ser extensible utilizando plugins adicionales. Entre las funcionalidades que brinda el sistema está la de generar un archivo cronológico automáticamente, y posee un calendario, en los cuales es posible navegar y buscar información publicada en cualquier día, mes o año. Para su desarrollo utiliza PHP y MySQL y es uno de los CMS más populares junto a Joomla y Drupal. Debido a que los blogs son los sistemas más susceptibles de recibir Spam[2], Wordpress posee un plugin llamado "Akismet" que identifica y frena la mayoría de los comentarios y trackbacks que son spam. (Muras, 2009)

Wordpress presenta algunas desventajas, por ejemplo: no puede modificar el código de su sitio fácilmente. Dependiendo de la cantidad y frecuencia de las publicaciones la administración de Blog puede dificultarse, si no se hace revisión periódica. Por ser un espacio de acceso público pueden recibirse comentarios no deseados que no se encuentren relacionados con las temáticas. El acceso a Blog debe hacerse, necesariamente utilizando un navegador de Internet. Si no se conoce con certeza la dirección del Blog, la búsqueda se torna difícil. Carece de muchas de las funciones de redes sociales, comercio electrónico, foros, wikis, etc.

1.6.2 Sistema de Gestión de Contenidos Drupal

Drupal es un sistema de gestión de contenido modular y muy configurable. Es un programa de código abierto, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Se destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la Web, y un énfasis especial en la usabilidad y consistencia de todo el sistema. El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hacen que sea adecuado para realizar diferentes tipos de sitio web.

Drupal ha sido diseñado con una arquitectura modular, y gracias a esta arquitectura se pueden agregar nuevas funciones a módulos ya existentes y crear nuevos módulos sin afectar los ya

[2] Se llama **Spam** a los mensajes no solicitados, no deseados o de remitente desconocido, habitualmente de tipo publicitario, enviados en grandes cantidades (incluso masivas) que perjudican de alguna o varias maneras al receptor.

creados. Estos módulos son los encargados del funcionamiento del sistema y separan la interfaz gráfica de la información. Este gestor de contenido posee una capa de abstracción de base de datos, implementada y soportada para MySQL y PostgresSQL, aunque puede añadir soporte para varias bases de datos. Está estructurado por temas, los cuales se pueden descargar de Internet o simplemente crearlos como plantillas en PHP, HTML y CSS. La desventaja que presenta Drupal es la alta curva de aprendizaje y la gran cantidad de módulos, hace difícil la selección del más eficiente.

1.6.3 Sistema de Gestión de Contenidos Joomla

Joomla es uno de los sistemas gestores de contenidos de código abierto más popular que permite crear sitios web de alta interactividad, profesionalidad y eficiencia. La administración de Joomla está enteramente basada en la gestión online de contenidos. Se dice "gestión online" porque todas las acciones que realizan los administradores de sitios Joomla, ya sea para modificar, agregar, o eliminar contenidos se realizan exclusivamente mediante un navegador web (browser) conectado a Internet, es decir, a través del protocolo HTTP (Protocolo de Transferencia de Hipertexto).

Joomla Tiene una gran comunidad de usuarios y también toda la documentación para crear diferentes aplicaciones. Además, es posible realizar casi cualquier sitio con muy pocos conocimientos: comercio electrónico, revistas online, intranets, redes sociales, etc. Realiza un gran trabajo gestionando el contenido necesario para que un Sitio Web funcione. Para muchas personas, el verdadero potencial de dicho CMS reside en la arquitectura de la aplicación, que posibilita que miles de desarrolladores en el mundo puedan crear potentes add-ons[3] y extensiones. Algunas de las numerosas extensiones disponibles son: gestores de documentos, generadores de formularios dinámicos, galerías de imágenes multimedia, motores de comercio y venta electrónica, directorios de empresas u organizaciones, software de foros y chats, calendarios, software para blogs, servicios de directorio, boletines de noticias, herramientas de registro de datos, sistemas de publicación de anuncios, servicios de suscripción. Está programado en lenguaje PHP y SQL. Utiliza bases de datos relacionales, más específicamente MySQL y al ser Joomla una aplicación Web, funciona obviamente en servidores de páginas web (HTTP Servers).

Características de publicación de páginas web en Joomla (Joomlaos.net., 2009)**:**

- **Automatización en la publicación:** Las páginas y documentos de Joomla pueden programarse con fecha de publicación y fecha de caducidad.

[3] **Add- ons** son programas para enriquecer los sitios web agregando algún componente instalado o conectado a un equipo que expande las capacidades de todo el sistema.

- **Archivo e historial:** Las páginas viejas o publicaciones que hayan perdido vigencia pueden enviarse a un "archivo" de almacenamiento, sin necesidad de tener que borrarlas. Esto permite también dar la posibilidad a los navegantes de consultar artículos viejos o documentos anteriores en un historial.
- **Formatos de lectura:** Cada documento es generado automáticamente por Joomla en formato **PDF** y en **XML**.
- **Envío por E-mail**: Los usuarios del sitio Joomla podrán enviar automáticamente a un amigo por correo cada documento publicado.
- **Valoración de contenidos**: Los visitantes del sitio podrán votar por la calidad de lo publicado.
- **Comentarios**: Los usuarios podrán comentar sus opiniones o expresar sus inquietudes en la misma página de contenidos.

Ventajas de Joomla (Joomlaos.net., 2009)**:**

- **Organización del sitio web:** Joomla está preparado para organizar eficientemente los contenidos de su sitio en secciones y categorías, lo que facilita la navegabilidad para los usuarios y permite crear una estructura sólida, ordenada y sencilla para los administradores. Desde el panel administrador de Joomla usted podrá crear, editar y borrar las secciones y categorías de su sitio de la manera en que más le convenga.
- **Publicación de contenidos:** Con Joomla CMS podrá crear páginas ilimitadas y editarlas desde un sencillo editor que permite formatear los textos con los estilos e imágenes deseados. Los contenidos son totalmente editables y modificables.
- **Escalabilidad e implementación de nuevas funcionalidades:** Joomla ofrece la posibilidad de instalar, desinstalar y administrar componentes y módulos, que agregarán servicios de valor a los visitantes de su sitio web, por ejemplo: galerías de imágenes, foros, boletines, clasificados, etc.
- **Administración de usuarios:** Joomla le permite almacenar datos de usuarios registrados y también la posibilidad de enviar e-mails masivos a todos los usuarios. La administración de usuarios es jerárquica, y los distintos grupos de usuarios poseen diferentes niveles de permisos dentro de la gestión y administración del sitio.
- **Diseño y aspecto estético del sitio:** Es posible cambiar todo el aspecto del sitio web tan solo con un par de clics, gracias al sistema de plantillas que utiliza Joomla.

- **Navegación y menú:** Totalmente editables desde el panel administrador de Joomla.
- **Administrador de imágenes:** Joomla posee una utilidad para subir imágenes al servidor y usarlas en todo el sitio.
- **Disposición de módulos modificable:** En un sitio creado con Joomla, la posición de módulos puede acomodarse como se prefiera.
- **Encuestas:** Joomla posee un sistema de votaciones y encuestas dinámicas con resultados en barras porcentuales.
- **Feed de noticias:** Joomla trae incorporado un sistema de sindicación de noticias por RSS/XMS de generación automática.
- **Publicidad:** Es posible hacer publicidad en el sitio usando el Administrador de Banners.
- **Estadísticas de visitas:** con información de navegador, sistemas operativos y detalles de los documentos (páginas) más vistos.

1.6.4 Consideraciones de los Sistemas de Gestión de Contenidos

El CMS que se vaya a seleccionar tiene que ser de código fuente abierto (o libre), fiable, robusto y permitir la escalabilidad para adecuarse a futuras necesidades con módulos. Es recomendable, que se utilicen hojas de estilo (CSS) y patrones de páginas. La herramienta tiene que tener soporte tanto por parte de los creadores como por otros desarrolladores, y tiene que ser fácil de utilizar y aprender. Una vez aclarado los requerimientos que tendría que poder satisfacer el CMS a escoger, se concluye que:

WordPress no es el sistema adecuado para usar para el desarrollo de la solución informática, pues sus funcionalidades están encaminadas a la construcción de blogs además de presentar varias desventajas, las cuales fueron expuestas anteriormente. Para desarrollar la aplicación en el menor tiempo posible lo ideal es usar Joomla, aunque Drupal brinda tantos beneficios como Joomla, la decisión de rechazar a Drupal está basada en que Joomla es más fácil de configurar y poner en marcha que su competencia. El panel de control de la administración de Drupal es malo; la separación entre el "front-end y el backend"[4] es débil y confusa, Joomla es mucho mejor. En cuanto a las plantillas, Joomla gana por un amplio margen, Drupal todavía tiene una enorme rampa que subir para conseguir que un Web de noticias, una revista, etc. se vea atractiva, característica esta, indispensable para el sistema en construcción.

[4] **Front-end** es la parte del software que interactúa con el usuario responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el **back-end** pueda usar. La conexión del **front-end** y el **back-end** es un tipo de interfaz.

1.7 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado o IDE (acrónimo en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes que proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante plugins se le puede añadir soporte de lenguajes adicionales.

1.7.1 Entorno de Desarrollo Integrado NetBeans 6.8

En su núcleo, NetBeans IDE es una herramienta de desarrollo Java, escrita puramente sobre la base de la tecnología Java. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Este enfoque de bienes comunes creativos ha permitido una mayor capacidad de uso, con cada nueva versión, y ha proporcionado a los desarrolladores mayor flexibilidad, al modificar el IDE, si así lo desean.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación.

NetBeans es un Entorno de Desarrollo Integrado de código abierto y gratuito para desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones profesionales, empresariales, web y móviles con el lenguaje Java, JavaFX, C/C ++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. NetBeans IDE es fácil de instalar y listo para usar y se puede ejecutar tanto en Windows, Linux, Mac OS X como en Solaris. (Sun Microsystems, 2009)

Entre las características de la plataforma están (Dadiskod, 2010)**:**

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

NetBeans IDE 6.8 incluye nuevas funcionalidades (Sun Microsystems, 2009)**:**

- Mayor soporte PHP: amplía el soporte de lenguajes dinámicos con soporte para PHP 5.3 y el Framework Symfony.
- Mayor integración con Proyecto Kenai: un entorno colaborativo para promover el desarrollo de proyectos de código abierto y libre en la red, ofrece soporte para JIRA[5], mensajería instantánea y seguimiento de incidencias mejorados.
- Mejor parametrización C/C ++: Parametrizar y afinar aplicaciones C/C++ con el nuevo indicador Microstate Accounting, monitor de actividad I/O

1.7.2 Entorno de Desarrollo Integrado Zend Studio for Eclipse 6.0

Zend Studio for Eclipse 6.0 es un IDE que soporta varios lenguajes aunque fue construido especialmente para PHP. Los expertos en PHP consideran a Zend Studio como el entorno más maduro y con más características útiles. Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP.El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez, versiones del producto para Windows, Linux y MacOS. Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Además, permite hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. (Álvarez, 2003)

[5] **JIRA** es una aplicación basada en web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos.

1.7.3 Consideraciones de los Entornos de Desarrollo Integrado

Posteriormente de haber analizado los entornos de desarrollo integrado más poderosos en el mundo de la programación web se ha concluido que el más conveniente para utilizar para el desarrollo de la solución informática es el NetBeans IDE 6.8**,** pues además de estar entre los IDE más potentes para el desarrollo de aplicaciones Web, adquirirlo reportaría muchísimos beneficios y con un costo mínimo ya que es de código abierto y gratuito. A pesar de los grandes beneficios que presenta Zend Studio se descarta la posibilidad de usarlo por ser de código propietario.

1.8 Sistemas Gestores de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es un conjunto de programas transparentes al usuario final que se encarga de la integridad y seguridad de los datos así como de la interacción con el Sistema Operativo. Proporciona una interfaz entre los datos, los programas que los manejan y los usuarios finales, controlando todas las operaciones que el usuario hace contra la base de datos.

El gestor almacena una descripción de datos en lo que le llaman: “diccionario de datos”, así como los usuarios permitidos y los permisos. Asimismo, tiene que haber un usuario administrador que se encargue de centralizar todas estas tareas. En cuanto al diccionario de datos, no es más que una base de datos donde se guardan todas las propiedades de la base de datos, descripción de la estructura, relaciones entre los datos, etc.

Los sistemas gestores de bases de datos deben permitir definir (especificar tipos, estructuras y restricciones de datos), construir (guardar los datos en algún medio controlado por el mismo SGBD) y manipular (realizar consultas, actualizarla, generar informes) a una base de datos. Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc.

1.8.2 Sistema Gestor de Base Datos MySQL

MySQL AB es una compañía de software fundada por David Axmark, Allan Larsson y Michael Widenius en 1995, creadora del sistema administrador de bases de datos relacionales MySQL, y una de las más grandes empresas de software libre del mundo. Desde el 16 de enero de 2008 es una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009.

La base de datos de código abierto MySQL es la más popular del mundo, debido a su gran rendimiento, alta fiabilidad, facilidad de uso y ahorro de costes, pues se distribuye bajo doble licencia. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la

empresa una licencia específica que les permita este uso. Este esquema de licenciamiento dual posibilita que la compañía, además de la venta de licencias privativas, ofrezca soporte y servicios.

MySQL 5.1 es el sistema de bases de datos enfocadas a servidores web y es una de las versiones más estables y utilizadas. Es una base de datos relacional, multihilo y multiusuario que se ha estandarizado al ser el soporte de la gran mayoría de CMS junto con el conocido lenguaje PHP.

A pesar de que MySQL 5 es un proyecto básicamente Open Source, la propiedad real es de Sun Microsystems, a su vez propiedad de Oracle. Sin embargo, el núcleo de MySQL 5 es libre y puede ser utilizado en cualquier proyecto sin ningún problema. Estas empresas ofrecen soporte y fomentan el desarrollo de este estupendo sistema de gestión de bases de datos (SGDB).

Características de MySQL(Saha, 2008)**:**

Velocidad: Arquitectura Multi-hilos, múltiples clientes tienen acceso concurrente. Cachea el resultado de las consultas comunes.

Facilidad de uso: Interfaz de línea de comandos y herramientas gráficas – escritorios, basadas en Web.

Soporte Multi-usuario: Múltiples usuarios tienen acceso concurrente a una o más bases de datos simultáneamente. Sistema de privilegios de usuarios potente y flexible. Esquemas de autenticación basados en usuario – máquina.

Portabilidad: Funciona en diferentes plataformas: Linux, Solaris, Windows, etc.

Amplio Soporte de aplicaciones: Base de datos de aplicaciones para Escritorio y la Web. APIs (Application Programming Interface, o Interfaz de Programación de Aplicaciones) para: C/C++, Java, PHP, Perl, Python, Ruby.

Programas almacenados: Procedimientos y funciones almacenadas y disparadores (triggers).

Motor de almacenamiento: Escribe los datos en almacenamiento persistente. Motores de almacenamiento conectable, permite que el motor de almacenamiento sea cargado y/o cambiado dinámicamente en tiempo de ejecución. Algunos de los motores que utiliza son: HEAP, MyISAM, CSV, Falcon, Cluster, y su propio motor.

Arquitectura interna: Sistema de asignación de memoria basado en hilos. Tablas temporales o tablas virtuales, formadas por consultas SQL son implementadas en tablas hash en memoria.

NetBeans simplifica el desarrollo con MySQL:

Genera fragmentos de código PHP para la base de datos MySQL.

El IDE permite registrar u nuevo servidor MySQL. Se puede trabajar con tablas MySQL desde NetBeans utilizando una interfaz gráfica amigable.

1.8.3 Consideraciones de los Sistemas Gestores de Bases de Datos (SGBD).

Se decidió usar MySQL 5.1 como SGBD debido a que el CMS Joomla, anteriormente seleccionado, está programado en SQL por lo que utiliza bases de datos relacionales, específicamente MySQL. Al mismo tiempo MySQL 5.1 GA está disponible en tres modelos para atender a las necesidades exclusivas de los distintos usuarios. Seleccionando MySQL Community Server; la versión de código abierto disponible gratuitamente. Por estar licenciado bajo el GPL que no requiere soporte comercial o servicios adicionales el ahorro en costes sería notable.

1.9 Conclusiones Parciales

Con el estudio realizado sobre el tema, se lograron identificar los distintos observatorios tecnológicos existentes tanto a nivel nacional como internacional. Se han fundamentado las herramientas que se van a utilizar para el desarrollo de la solución informática. Se utilizará RUP, como metodología de desarrollo de software, por ser una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. Se seleccionó UML y Visual Paradigm como lenguaje de modelado y herramienta CASE respectivamente. Para implementar la aplicación se decidió utilizar PHP y el CMS Joomla, por soportar el lenguaje de programación anteriormente mencionado, además de las muchísimas ventajas que presenta: organización del sitio web, publicación de contenidos, administración de usuarios escalabilidad e implementación de nuevas funcionalidades. También se eligió NetBeans 6.8 como Entorno de Desarrollo Integrado por ser de código libre y gratuito, y ofrece todas las herramientas necesarias para crear aplicaciones Web. Como SGBD se escogió MySQL 5.1 específicamente el modelo MySQL Community Server puesto que es de código abierto, con alto rendimiento, fiabilidad, facilidad de uso y garantiza el ahorro en costes.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

En este capítulo se manifiestan y describen los procesos de negocio exponiendo las reglas que estos deben cumplir. Para describir la solución propuesta se realiza el modelado del negocio y del sistema, utilizando como soporte el modelo de casos de uso y el modelo de objetos. Se desarrollan varias actividades y artefactos que permiten darle cumplimiento al problema planteado, como: la especificación de los requisitos funcionales y no funcionales, y la descripción detallada de cada uno de los casos de uso del sistema. También se expone el análisis y diseño realizado como parte de la propuesta de solución, modelándose los artefactos necesarios que contribuyen a la implementación del sistema. Estos artefactos son el Modelo de Análisis y el Modelo de Diseño, los cuales se encargan de describir en términos de clases cómo deben funcionar los casos de uso del sistema. Aquí también se logra representar la colaboración entre estas clases y se exponen los patrones de diseño y arquitectura que se utilizan en la elaboración de las mismas.

2.2 Etapa de Modelado del Negocio

El flujo de trabajo de la etapa de Modelado del Negocio se desarrolla principalmente en la fase de Inicio para comprender los procesos del negocio de la organización donde se crea una primera versión del Modelo de Negocio el cual describe el contexto del sistema a construir. Los propósitos que se persiguen al realizarse el modelado del negocio, son: comprender la estructura y el dinamismo de la organización en cuestión, concebir los problemas actuales e identificar mejoras potenciales, asegurarse de que los clientes, usuarios finales y desarrolladores tengan una idea común de la organización y derivar los requerimientos del sistema a partir del modelo de negocio logrado.

Esta etapa es necesaria ante el hecho de que muchos de los productos software que se desarrollan automatizan algunos o todos los procesos existentes en un negocio. Tanto los desarrolladores como los clientes tienen que tener bien claro cómo funciona el negocio que se desea automatizar para garantizar que el software desarrollado va a cumplir su propósito, y por esto, se hace un estudio en el dominio del negocio y en el dominio del software.

2.2.1 Técnicas usadas para la solución propuesta

Para el modelado del negocio pueden utilizarse técnicas y notaciones que usualmente se emplean en la disciplina de la Ingeniería del Software. Esto permite utilizar un lenguaje y notación común en ambos dominios (negocio y software).

El esclarecimiento de las necesidades del sistema a automatizar es un proceso complicado, pues en él hay que identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los clientes y usuarios finales. La captura de requisitos es, esencialmente, un proceso de comunicación; desarrollar dicho proceso de forma adecuada permite disminuir los costes y retrasos del proyecto, mejorar la calidad del software, la comunicación entre equipos y aumentar la satisfacción de los usuarios finales. Para lograr esto se utilizaron como técnicas la entrevista y la tormenta de ideas.

Luego de especificar todos los requisitos que tiene que cumplir el sistema entonces se define la arquitectura de software a utilizar para dicho producto, que permita implementar de manera adecuada todas las funcionalidades según las exigencias de los usuarios finales.

2.2.2 Procesos del Negocio

Oscar Barros hace una importante distinción, al introducir el concepto de valor agregado en la definición de proceso, señalando que “un proceso es un conjunto de tareas lógicamente relacionadas que existen para conseguir un resultado bien definido dentro de un negocio; por lo tanto, toman una entrada y le agregan valor para producir una salida. Los procesos tienen entonces clientes que pueden ser internos o externos, los cuales reciben a la salida, lo que puede ser un producto físico o un servicio. Estos establecen las condiciones de satisfacción o declaran que el producto o servicio es aceptable o no” (Barros, 1994; pp.56).

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas. Las entradas son requisitos que deben tenerse antes de que una función pueda ser aplicada. Cuando una función es aplicada a las entradas de un método, se obtienen ciertas salidas resultantes. Es mediante el proceso que una organización ofrece sus servicios a sus clientes.

La solución propuesta debe ofrecer ciertas prestaciones para satisfacer las necesidades que tienen los profesores y estudiantes de la facultad 15 en el ámbito docente, productivo e investigativo. Se propone crear un observatorio tecnológico para garantizar una correcta vigilancia tecnológica en la facultad, y para lograr los objetivos que promete dicha solución se precisa de la implementación de varios procesos.

Procesos básicos identificados:

Elaborar Currículum: Proceso que inicia cuando un profesor comienza a desempeñar su vida laboral. Los datos generales de la persona díganse: nombre y apellidos, carnet de identidad, dirección particular, teléfono, año de graduación, institución, formación académica, grado científico,

categoría docente y científica, líneas de investigación y profesión actual tienen que ser archivados por el jefe del departamento en el cual va a comenzar a trabajar. El currículum será actualizado por esta misma persona a medida de que el docente participe en actividades tales como recibir estudios y cursos de postgrado, participar en eventos, realizar investigaciones y publicaciones.

Buscar y Divulgar Eventos: Proceso que inicia cuando los asesores investigativos de la facultad empiezan a hacer búsquedas, según los intereses de los centros de desarrollo de la facultad, hasta obtener una relación de los principales eventos científicos y técnicos en los que puedan participar los profesores y luego enviarle a los mismos un informe con los datos significativos referente a cada uno de dichos eventos, tales como: nombre, temática, clasificación, interés, área, modalidad, fecha de inicio, fecha de fin, lugar, tur operador, sede, periodicidad y comité.

Publicar Cursos de Postgrado: Proceso que comienza cuando un profesor, la dirección de la facultad o del centro decide impartir un curso de postgrado. Estas personas se encargan de hacer llegar esta convocatoria al resto de los profesores de la facultad. Los profesores pueden hacer la solicitud para matricular en este curso, una vez que cierra la convocatoria es que se hace una selección de los profesores que recibirán dicho curso, por tanto, es en este momento que se le comunica a los profesores que habían realizado la solicitud, si están o no matriculados.

Elaborar Balance de Ciencia y Técnica: Proceso que empieza cuando cada profesor entrega su currículum y los documentos que lo avalan al jefe del departamento al que pertenece. El jefe de cada departamento recoge los datos del currículum que generan indicadores necesarios a medir a la hora de realizar el balance. Una vez que se obtiene el balance para el departamento es archivado para luego compararlo con el logrado en el resto de los departamentos de la facultad. Luego se emite un reporte a los profesores con los resultados del balance del presente año así como los obtenidos en años anteriores. Estos datos son pasados a los asesores de investigación de la facultad para que también se realice el balance de ciencia y técnica para cada centro de investigación.

2.2.3 Actores y Trabajadores del Negocio

Un actor del negocio representa un tipo particular de usuarios (cualquier individuo, grupo, entidad, organización, máquina o sistema de información externa) que interactúa con el negocio para beneficiarse de sus resultados.

Un trabajador del negocio es una abstracción de una persona (o grupo de personas), una máquina o un sistema automatizado; que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio.

Tabla 1. Actores del negocio.

Actor	Descripción
Profesor	Este actor representa a las personas que deben entregar los datos generales al jefe del departamento en el cual comenzará a trabajar. Además, es la persona que recibe la información referente a los eventos científicos – técnicos y cursos de postgrado.

Tabla 2. Trabajadores del negocio.

Trabajadores	Descripción
Jefe de Departamento	Este trabajador es el encargado de archivar y actualizar el currículum de cada uno de los profesores de su departamento. Igualmente realiza el balance de ciencia y técnica para dicho departamento.
Asesores de Investigación	Este trabajador representa a los profesores encargados de comunicarles al resto de los docentes de la facultad sobre la realización de algún evento científico-técnico, y de sitios web donde se pueden hacer publicaciones de trabajos investigativos.

2.2.4 Descripción de los Casos de Uso del Negocio

El negocio cuenta con cuatro procesos básicos los cuales van a ser representados a través de casos de uso del negocio (CUN). En las siguientes tablas se describen cada uno estos casos de uso y se enuncian los actores y trabajadores que intervienen en cada uno de ellos.

Tabla 3. Descripción textual del CUN: "Elaborar Currículum".

Caso de Uso:	Elaborar Currículum
Actores:	Profesor (inicia)
Trabajadores:	Jefe de Departamento
Resumen:	El caso de uso empieza cuando un profesor comienza a desempeñar su vida laborar. Los datos generales de la persona, como: nombre y apellidos, carnet de identidad, dirección particular, teléfono, año de graduación, institución, formación académica, grado científico, categoría docente y científica, líneas de investigación y profesión actual tienen que ser archivados por el jefe del departamento en el cual va a comenzar a trabajar.

	Así mismo el currículum es actualizado por esta misma persona a medida de que el docente participe en actividades tales como recibir estudio y cursos de postgrado, realizar investigaciones, participar en eventos y publicaciones.

Tabla 4. Descripción textual del CUN: "Buscar y Divulgar Eventos Científicos y Técnicos".

Caso de Uso:	Buscar y Divulgar Eventos Científicos y Técnicos
Actores:	Profesor
Trabajadores:	Asesor de Investigación (inicia)
Resumen:	El caso de uso inicia cuando los asesores de investigación de la facultad comienzan a hacer búsquedas hasta obtener una relación de los principales eventos científicos y técnicos en los que puedan participar los profesores de la facultad y luego enviarle a los mismos un informe con los datos significativos referente a cada uno de dichos eventos, tales como: nombre, temática, clasificación, interés, área, modalidad, fecha de inicio, fecha de fin, lugar, tur operador, sede, periodicidad y comité.

Tabla 5. Descripción textual del CUN: "Publicar Cursos de Postgrado".

Caso de Uso:	**Publicar** Cursos de Postgrado
Actores:	Profesor
Trabajadores:	Profesor Capacitado (inicia)
Resumen:	EL caso de uso empieza cuando un profesor capacitado (profesor, la dirección de la facultad o del centro) decide impartir un curso de postgrado. Estas personas se encargan de hacer llegar esta convocatoria al resto de los profesores de la facultad. Los profesores pueden hacer la solicitud para matricular en este curso. Después que cierra la convocatoria es que se hace una selección de los profesores que recibirán dicho curso y se le comunica a los profesores que habían realizado la solicitud, si están o no matriculados.

Tabla 6. Descripción textual del CUN: "Elaborar Balance de Ciencia y Técnica".

Caso de Uso:	**Elaborar Balance de Ciencia y Técnica**
Actores:	Profesor (inicia)
Trabajadores:	Jefe de Departamento
	El caso de uso comienza cuando cada profesor entrega su currículum y

Resumen:	los documentos que lo avalan al jefe del departamento al que pertenece. El jefe de cada departamento recoge los datos del currículum que generan indicadores necesarios a medir a la hora de realizar el balance. Después que cada jefe obtiene el balance para el departamento es archivado para luego compararlo con el logrado en el resto de los departamentos de la facultad. Luego se emite un reporte a los profesores con los resultados del balance del presente año así como los obtenidos en años anteriores. Estos datos son pasados a los asesores de investigación de la facultad para que también se realice el balance de ciencia y técnica para cada centro de investigación.

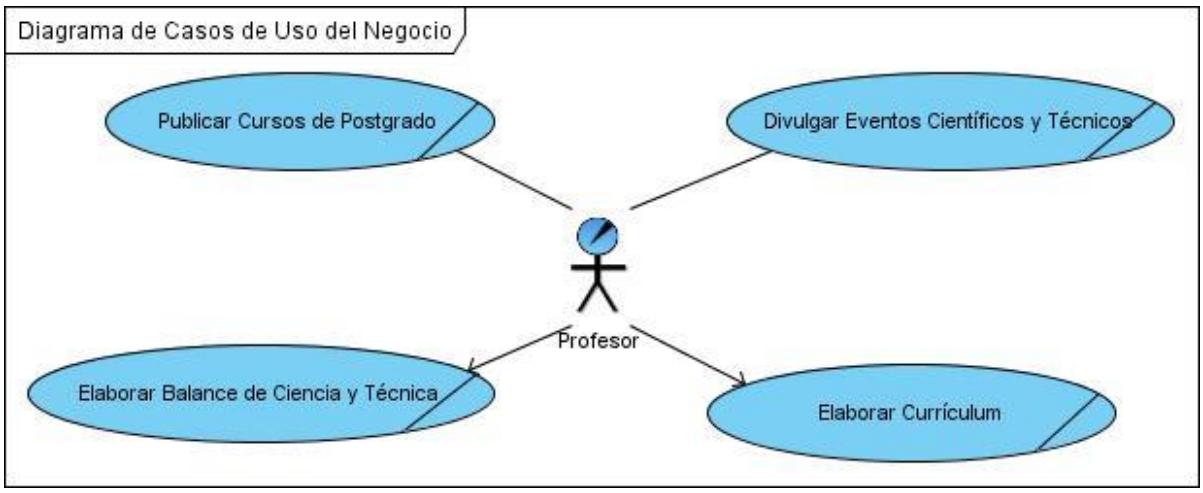


Figura 1. Diagrama de casos de uso del negocio.

2.2.5 Diagramas de Actividades y Modelo de Objetos del Negocio

Un diagrama de actividades describe un proceso que enuncia el orden de las actividades que logran los objetivos del negocio. Es un grafo de acciones que contienen el funcionamiento de cada actividad en un flujo de trabajo o la ejecución de una sentencia de un procedimiento.

Para cada caso de uso del negocio se realiza un diagrama de actividades, de esta forma, se obtiene detalladamente la descripción del proceso que da lugar al caso de uso. Estos diagramas muestran los actores y trabajadores que colaboran dentro del caso de uso. Se utilizan calles para delimitar las actividades que realiza cada persona involucrada, resaltando, con color azul claro, las actividades que serán automatizadas con el desarrollo del sistema. En cada esquema se indica el punto de inicio

y final del proceso que describe, también forman parte de estos esbozos las entidades del negocio, representando la información persistente. Las siguientes imágenes muestran los diagramas de actividades para cada uno de los casos de uso del negocio.

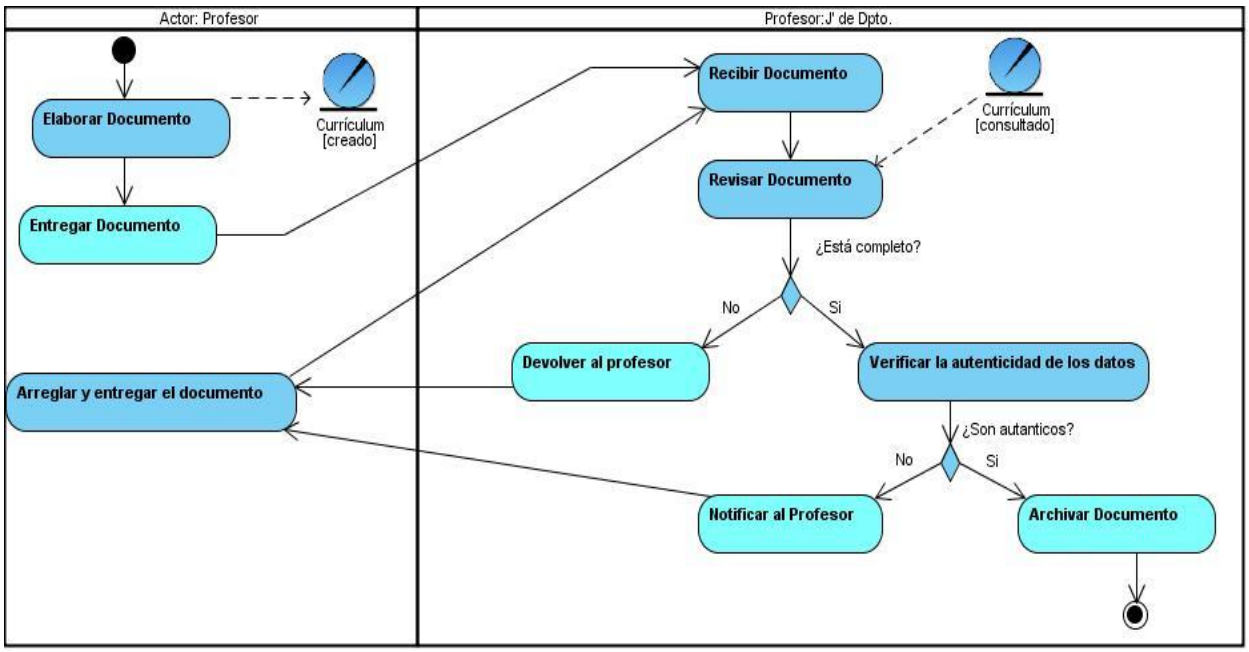


Figura 2. Diagrama de actividades del CUN: "Elaborar Currículum".

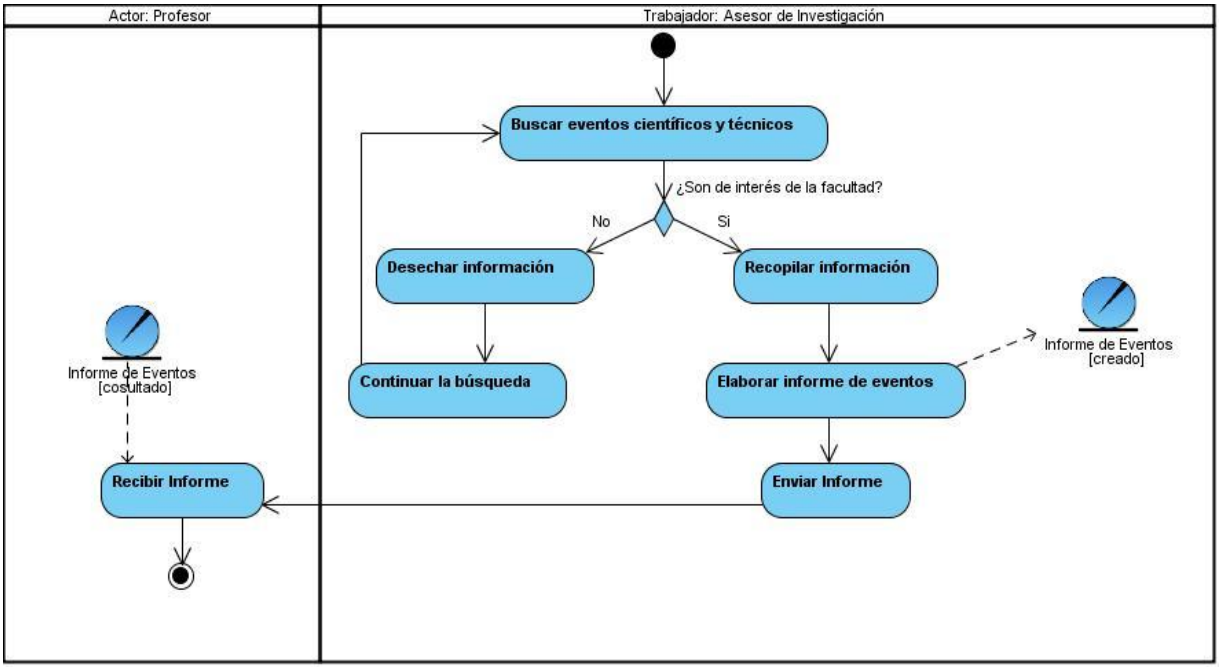


Figura 3. Diagrama de actividades del CUN: "Buscar y Divulgar Eventos Científicos y Técnicos".

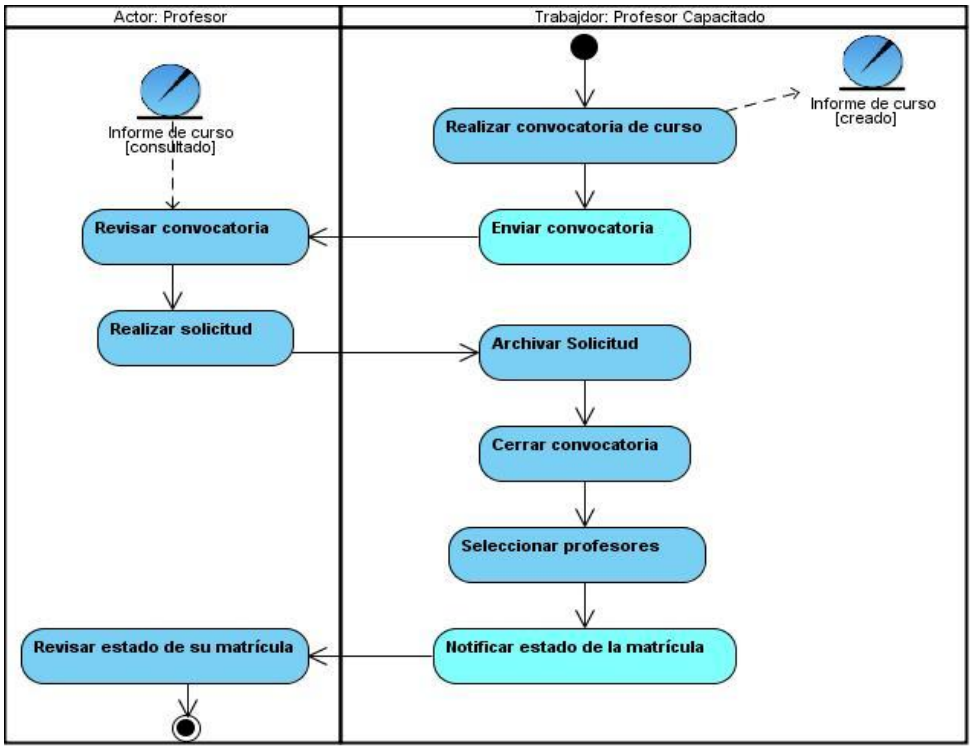


Figura 4. Diagrama de actividades del CUN: "Publicar Cursos de Postgrado".

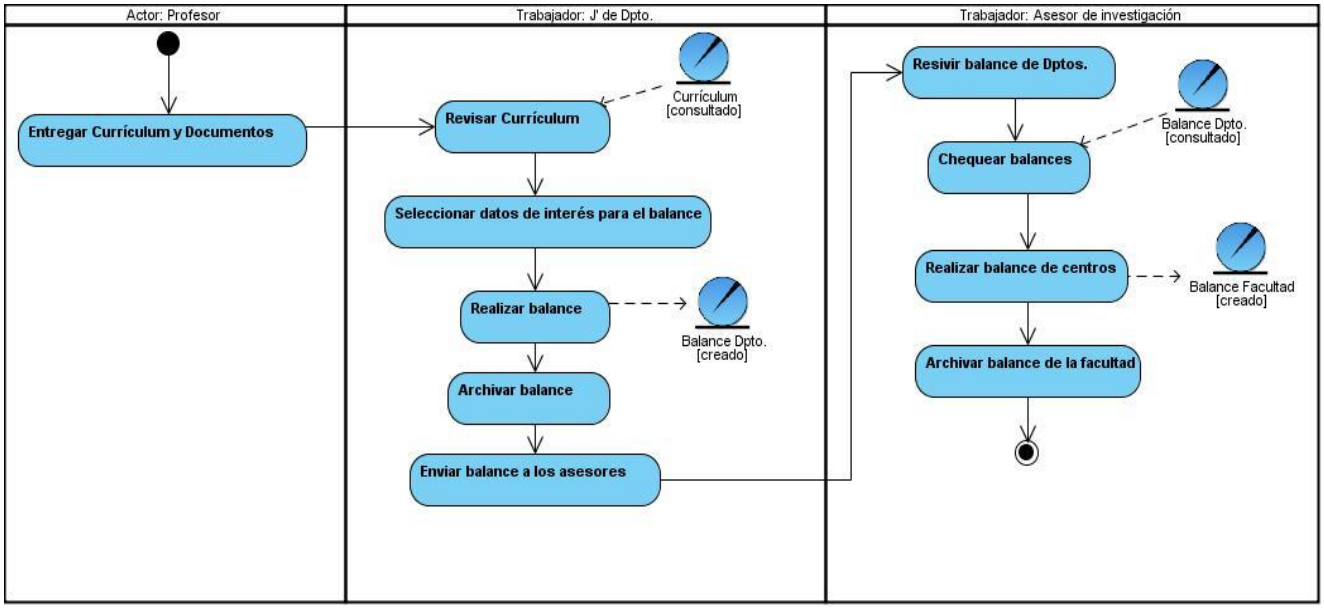


Figura 5. Diagrama de actividades del CUN: "Elaborar Balance de Ciencia y Técnica"

A partir de los diagramas de actividades se logra el modelo de objetos del negocio, está compuesto por diagramas de clases que representan los trabajadores y entidades del negocio, así como las líneas que unen a los trabajadores que se relacionan y a estos con las entidades que manipulan. A continuación se muestran todos los trabajadores y entidades que interviene en el negocio abordado.

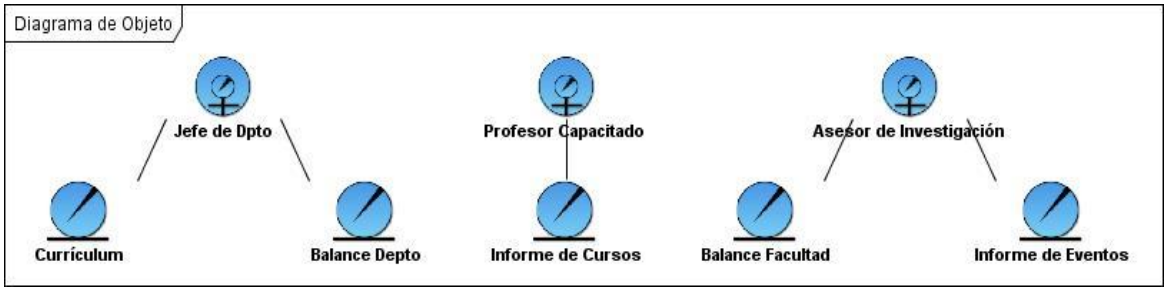


Figura 6. Modelo de objetos del negocio.

2.2.5 Reglas del Negocio

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio.

1. Reglas de estructura:

Cada profesor puede confeccionar únicamente su currículum.

Un profesor tiene que pertenecer a un solo departamento.

Un departamento tiene más de un profesor.

El profesor tiene que entregar su currículum a su jefe de departamento.

2. Reglas de acción:

Un jefe de departamento realiza solamente el balance de ciencia y técnica para su departamento.

El jefe de departamento tiene que archivar el currículum de cada profesor perteneciente a su departamento.

El jefe del departamento tiene que devolver el currículum del profesor en caso de faltar algún dato para que él mismo lo complete y lo vuelva a entregar.

El asesor de investigación se encarga de archivar la información referente a cada departamento de la facultad para realizar el balance de ciencia y técnica para cada centro investigativo.

El asesor de investigación se encarga de buscar la información referente a los eventos científicos y técnicos, y comunicarle a los profesores y estudiantes con cierta anticipación.

El asesor de investigación tiene la misión de buscar las direcciones de las revistas donde se puedan hacer publicaciones de trabajos investigativos y enviarles un informe con toda la información referente a estas revistas a los profesores de la facultad.

2.3 Especificación de Requisitos

La Especificación de Requisitos de Software es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye requisitos funcionales que describen las funciones y servicios que se espera que la aplicación ofrezca a los usuarios finales. También contiene requisitos no funcionales que imponen restricciones en el diseño e implementación de las funcionalidades ofrecidas. En este apartado, se registran los requisitos capturados hasta el momento, y se especifican las capacidades operacionales y funcionales que el sistema deberá tener con el mayor detalle posible.

2.3.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. A continuación se listan los requisitos funcionales que tiene que cumplir la aplicación en desarrollo.

Nota: El requisito gestionar engloban las funciones de insertar, eliminar y modificar.

RF1: Gestionar eventos científicos y técnicos.

RF2: Buscar eventos científicos y técnicos.

RF3: Visualizar eventos científicos y técnicos.

RF4: Visualizar revistas para publicaciones.

RF5: Visualizar nichos investigativos.

RF6: Visualizar información sobre curso de postgrado.

RF7: Autenticar usuario.

RF8: Realizar Balance de ciencia y técnica para cada departamento.

RF9: Confeccionar el currículum de los profesores.

RF10: Realizar solicitud en un curso

RF11: Buscar el currículum de un determinado profesor.

RF12: Actualizar el currículum de los profesores.

RF13: Revisar el currículum de todos los profesores.

RF14: Visualizar información de proyectos.

RF15: Realizar balance de ciencia y técnica para la facultad.

2.3.2 Requisitos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son esas características que posibilitan que el producto sea atractivo, usable, rápido, confiable, etc. Usualmente, están vinculados a requerimientos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Requerimientos de Software:

RNF1: Navegador Internet Explorer v6.0 o superior, Mozilla Firefox v2.x.

RNF2: Servidor Web Apache v2.x o superior con módulo PHP 5.2.3 disponible, este deber estar configurado con las extensiones, “soap”, “ldap”.

RNF3: Como sistema gestor de bases datos: MySQL 5.1.

Requerimientos de Hardware:

RNF4: Para los servidores tanto Web como SGBD: Pentium IV o superior con 256MB de RAM o más.

RNF5: Capacidad de Disco Duro mayor de 10 GB.

Restricciones para el diseño e implementación:

RNF6: Utilizar como lenguaje al lado del servidor PHP v5.2.3 o superior y del lado del cliente HTML y Java Script.

RNF7: Para la programación PHP se recomienda el IDE: NetBeans.

RNF8: El código deberá ser reutilizable.

Requerimientos de Apariencia o Interfaz externa:

RNF9: Diseñado para la resolución de 1024x768, aunque deberá verse en 800x600 o cualquier resolución superior a esta.

RNF10: La interfaz deber ser agradable para el usuario, que combine bien los colores, tipo de letra y tamaño.

Requerimientos de Seguridad:

RNF11: Contar con un sistema de permisos para el acceso a la información.

RNF12: El sistema debe estar disponible las 24 horas del día (Disponibilidad).

Requerimientos de Usabilidad:

RNF13: El sistema debe permitir acceso a los usuarios.

Requerimientos de Rendimiento:

RNF14: El sistema deberá ser capaz de gestionar toda la información y dar respuestas a las solicitudes lo más rápido posible.

2.4 Arquitectura definida para el Sistema

"La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad." (Kruchten, 1995)

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema informático. Lo usual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Precisamente, las arquitecturas más universales 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).

Los sitios web tradicionales que se limitaban a mostrar información se han convertido en aplicaciones capaces de una interacción más o menos sofisticada con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar opciones de diseño nuevas que permitan dar con la arquitectura óptima que facilite la construcción de los mismos. El usuario interacciona con las aplicaciones web a través del navegador. Como consecuencia de la actividad del usuario, se envían peticiones al servidor, donde se aloja la aplicación y que normalmente hace uso de una base de datos que almacena toda la información relacionada con la misma. El servidor procesa la petición y devuelve la respuesta al navegador que la presenta al usuario. Por tanto, el sistema se distribuye en tres componentes: el navegador, que presenta la interfaz al usuario; la aplicación, que se encarga de realizar las operaciones necesarias según las acciones llevadas a cabo por éste y la base de datos, donde la información relacionada con la aplicación se hace persistente. Esta distribución se conoce como el modelo o arquitectura de tres capas.

- **Capa de Presentación*:*** Esta capa es la que ve el usuario, presenta el sistema al usuario. Le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz de gráfica y debe tener la característica de ser amigable para el usuario por lo que generalmente se presenta como formularios.
- **Capa de Negocio:** Es en esta capa donde, se describen las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso la lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.
- **Capa de Datos:** Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de base de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

2.5 Modelado del Sistema

El modelo del sistema está basado en las funcionalidades que este debe cumplir y sirve como acuerdo entre clientes y desarrolladores. Este artefacto contiene actores, casos de uso y sus relaciones y sirve como entrada fundamental para el análisis, diseño y pruebas.

2.5.1 Actores del Sistema

Cada trabajador del negocio es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Tabla 7. Actores del sistema.

Actor	Descripción
Usuario	Es una generalización del actor del sistema representa a todos los profesores y estudiantes, puede acceder al sistema libremente sin necesidad de registrarse pudiendo visualizar la información referente a los proyectos productivos de la facultad, eventos científicos-técnicos, revistas para publicaciones y nichos investigativos.
Profesor	Es una especificación del actor general del sistema “usuario” por tanto además de tener los permisos que tiene dicho actor puede confeccionar y

	actualizar su currículum, así como visualizar la información referente a los cursos de postgrado y si le interesa alguno hacer la solicitud.
Asesor de Investigación	Es una especificación del actor “Profesor”, por lo que puede ejecutar todas las funcionalidades igual que el actor antes mencionado, además de gestionar las revistas para hacer publicaciones de trabajos investigativos, nichos investigativos y eventos científicos - técnicos. La información de los proyectos de la facultad también será actualizada por este mismo actor.
Jefe de Dpto.	Es otra especificación del actor “Profesor”, por tanto además de ejecutar todas las funcionalidades igual que el actor anteriormente citado tiene la responsabilidad de gestionar los cursos de postgrado par los profesores de la facultad, revisar el currículum en línea de cada profesor de su departamento y a partir de ahí realizar el balance de ciencia y técnica. También es el encargado de eliminar el currículum de un profesor determinado.

2.5.2 Descripción de los Casos de Uso del Sistema

Un caso de uso del sistema (CUS) es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. En otras palabras, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los casos de uso son entidades esenciales a la hora de modelar el diagrama de casos de uso de un sistema.

Estos diagramas representan gráficamente a los procesos y su interacción con los actores, especificando la comunicación y el comportamiento del sistema mediante su interacción con los usuarios y/u otros sistemas. Se utilizan para ilustrar los requerimientos del sistema, pues representa las funcionalidades que ofrece el mismo en lo que se refiere a su interacción externa. Los elementos que pueden aparecer en un diagrama de casos de uso son: actores, casos de uso y relaciones entre casos de uso.

A continuación se brinda una breve descripción de un caso de uso crítico del sistema: “Realizar Balance de Ciencia y Técnica de Departamento”. También se expone el diagrama de casos de uso del sistema para ofrecer una visión general de las principales funcionalidades que serán implementadas en el mismo,

destacando, con color azul claro, las que más peso tienen en el sistema. Para conocer las descripciones de los demás casos de uso consultar el documento “Modelo del Sistema”.

Tabla 8. Descripción textual del CUS: “Realizar Balance de Ciencia y Técnica de Departamento”.

Caso de Uso:	Realizar Balance de Ciencia y Técnica del Departamento.
Actores:	Jefe de Dpto.
Resumen:	El CU empieza cuando un jefe de algún departamento accede al sistema para introducir la información necesaria para obtener el balance de ciencia y técnica de su departamento. El jefe de departamento que acceda a esta sección del sistema tiene que registrarse para poder editar la información.
Precondiciones	El sistema debe haberse ejecutado.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del sistema
1. El Jefe de un departamento selecciona la opción: “Sistema de Indicadores de Ciencia y Técnica”.	1.1 Muestra la sección donde se realizara el balance de ciencia y técnica.
2. El usuario introduce los datos precisos para realizar el balance de su departamento.	2.1 Guarda los datos y realiza el balance. 2.2 Genera un resultado y lo archiva.
3. Selecciona ver el resultado	3.1 Muestra el resultado.
4. Selecciona salir	4.1 Cierra la sección.

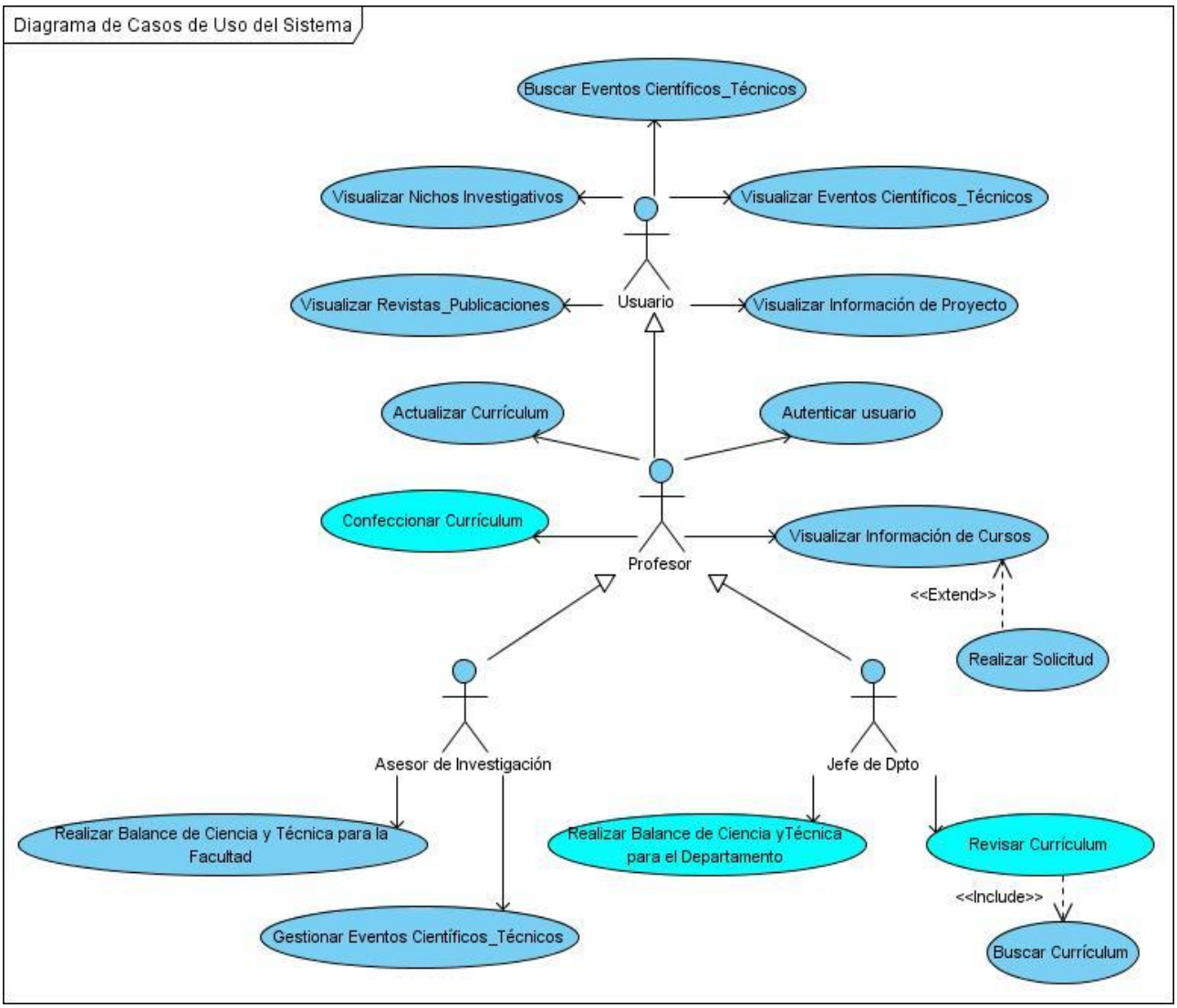


Figura 7. Diagrama de casos de uso del sistema.

2.6 Modelo de Análisis

"Durante el análisis, se analizaron los requisitos que se describen en la captura de requerimientos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura". (Jacobson y otros, 2000)

Aunque en el modelo del análisis hay un refinamiento de los requisitos, no se tiene en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras

características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

El modelo del análisis es un artefacto que usualmente se genera para entender claramente los requisitos y realizar mejor el diseño. Es usado para representar la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del Modelo de Diseño y puede contener: las clases y paquetes de análisis, las realizaciones de los casos de uso, las relaciones y los diagramas.

2.6.1 Diagramas de Clases del Análisis

En el diagrama de clases de análisis se representa los conceptos fundamentales que comprende el dominio del problema. Estos diagramas muestran que clases participan en las realizaciones de los distintos casos de uso, representan las definiciones y relaciones entre las clases. Las clases del análisis se clasifican en Interfaz, de Control o Entidad.

- **Clase Interfaz:** Modela la interacción entre el sistema y sus actores.
- **Clase Control:** Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.
- **Clase Entidad:** Modelan información que posee larga vida y que es a menudo persistente.

La siguiente imagen muestra el diagrama de clases del análisis correspondiente al caso de uso Realizar Balance de Ciencia y Técnica de Departamento. Representa los actores y su interacción con el sistema mediante clases interfaces, así como el resto de las clases que intervienen y sus relaciones. Para consultar los diagramas de los demás casos de uso consultar el documento “Modelo del Análisis”.

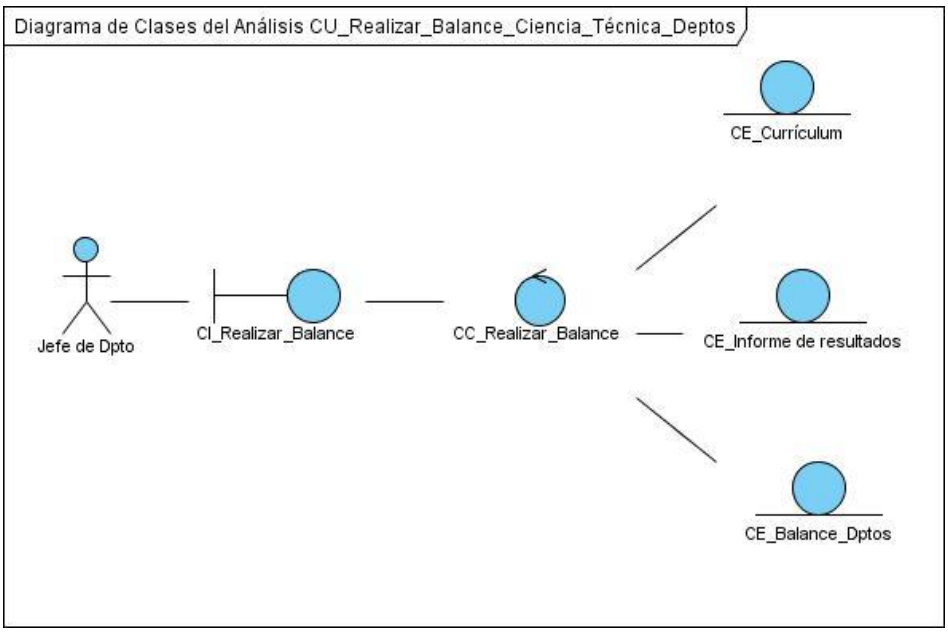


Figura 8. Diagrama de Clases del Análisis del CUS: "Realizar Balance de Ciencia-Técnica para Departamentos."

2.6.2 Diagramas de Interacción

Los diagramas de interacción son diagramas que describen cómo grupos de objetos colaboran para conseguir algún fin. Estos diagramas muestran objetos, así como los mensajes que se pasan entre ellos dentro del caso de uso. Los diagramas de interacción capturan el comportamiento de los casos de uso y tienen dos formas de expresarse: diagramas de secuencia y diagramas de colaboración.

- **Diagramas de secuencia:** Muestran las interacciones expresadas en función de secuencias temporales. Destaca la ordenación temporal de los mensajes.
- **Diagramas de colaboración:** Muestran las relaciones entre los objetos y los mensajes que intercambian. Destaca la organización estructural de los objetos que envían y reciben mensajes

La siguiente imagen expone el diagrama de colaboración del análisis del caso de uso crítico "Realizar Balance de Ciencia y Técnica para Departamento". Este diagrama muestra las relacione entre actor y clases del análisis que intervienen en el caso de uso, así como el flujo de los mensajes que se pasan entre ellos. Para consultar los diagramas de colaboración del análisis correspondientes al resto de los casos de uso del sistema ver el documento "Modelo del Análisis".

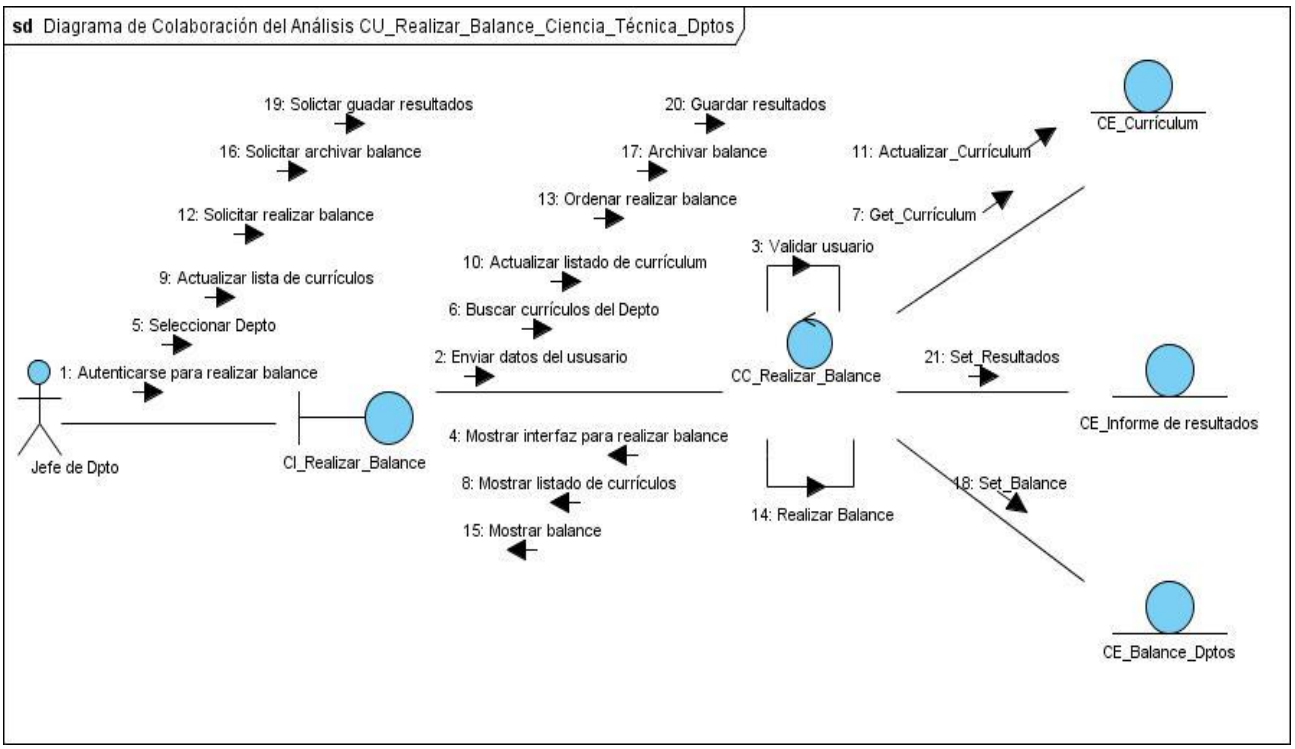


Figura 9. Diagrama de Clases de Colaboración del CU: "Realizar Balance de Ciencia-Técnica para Departamentos."

2.7 Modelo de Diseño

"El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el Modelo de Diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación". (Jacobson y otros, 2000)

El modelo de diseño está muy próximo al de implementación, obviamente para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. En el diseño se modela el sistema y se obtiene su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada fundamental en el diseño es el resultado del análisis, que proporciona una comprensión detallada de los requisitos. Además, impone una estructura del sistema la cual hay que conservar para darle forma al mismo. Este modelo puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas,

protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.

2.7.1 Diagramas de Clases del Diseño

Los diagramas de clases del diseño son los diagramas principales en el flujo de trabajo análisis y diseño para obtener un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.

En la figura siguiente se expone el diagrama de clases del diseño para un caso de uso crítico del sistema: "Realizar Balance de Ciencia y Técnica para Departamentos". En el diagrama se representa las clases que intervienen dentro del caso de uso así como las relaciones entre estas. Para conocer los diagramas de clases del diseño correspondientes al resto los casos de uso del sistema consultar el documento "Modelo de Diseño".

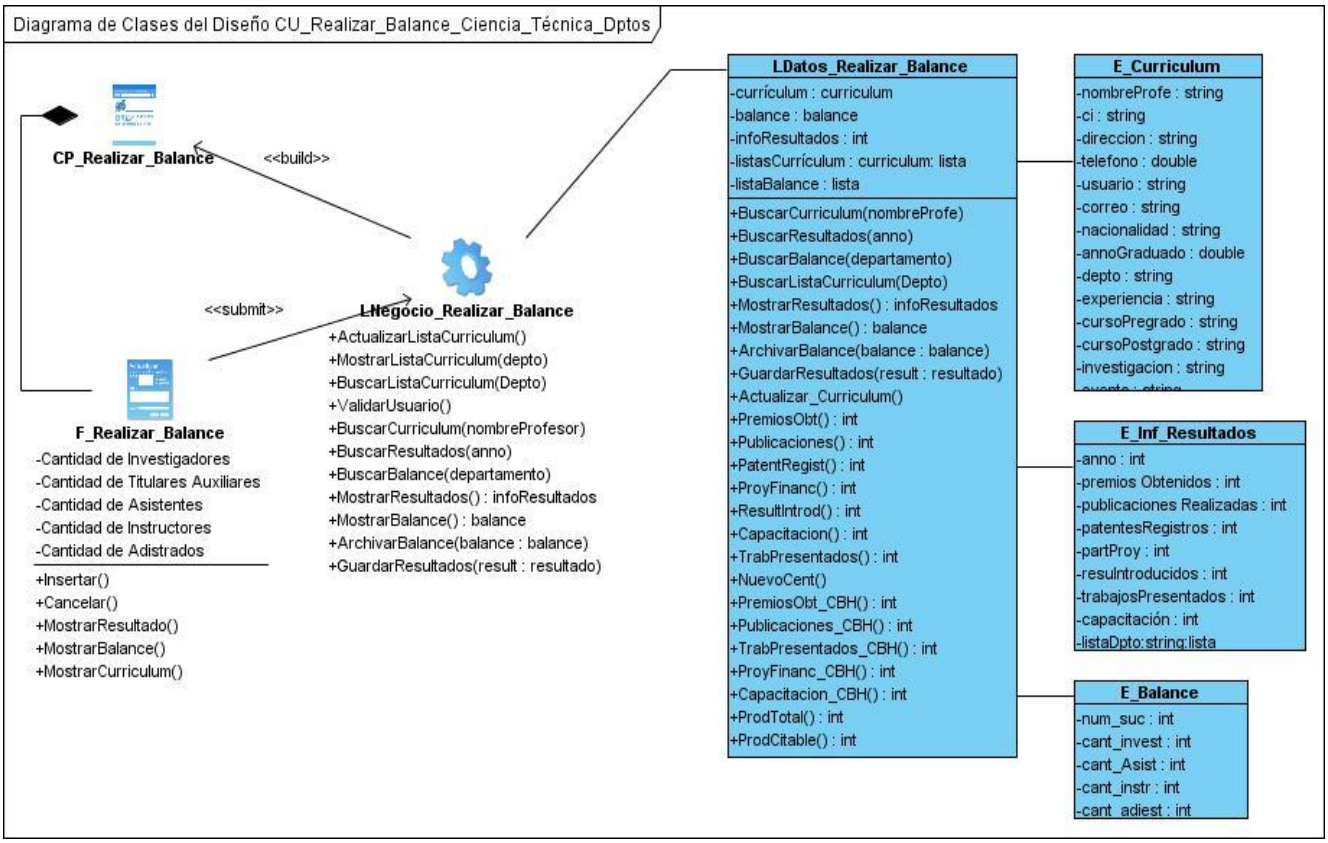


Figura 10. Diagrama de Clases del Diseño del CUS: "Realizar Balance de Ciencia-Técnica para Departamentos."

2.7.2 Diagramas de Interacción del Diseño

Los diagramas de interacción del diseño muestran las relaciones entre actores y las clases del diseño que participan en un caso de uso, además de los mensajes que se pasan entre ellos para lograr un objetivo. Estos diagramas, al igual que los de interacción del análisis, se expresan de dos formas: diagramas de secuencia y diagramas de colaboración.

A continuación se ofrecen la imagen del diagrama de secuencia del diseño para el caso de uso crítico del sistema “Realizar Balance de Ciencia y Técnica de Departamento”. Para chequear los diagramas de secuencia del diseño correspondientes al resto de los casos de uso consultar el documento “Modelo de Diseño”.

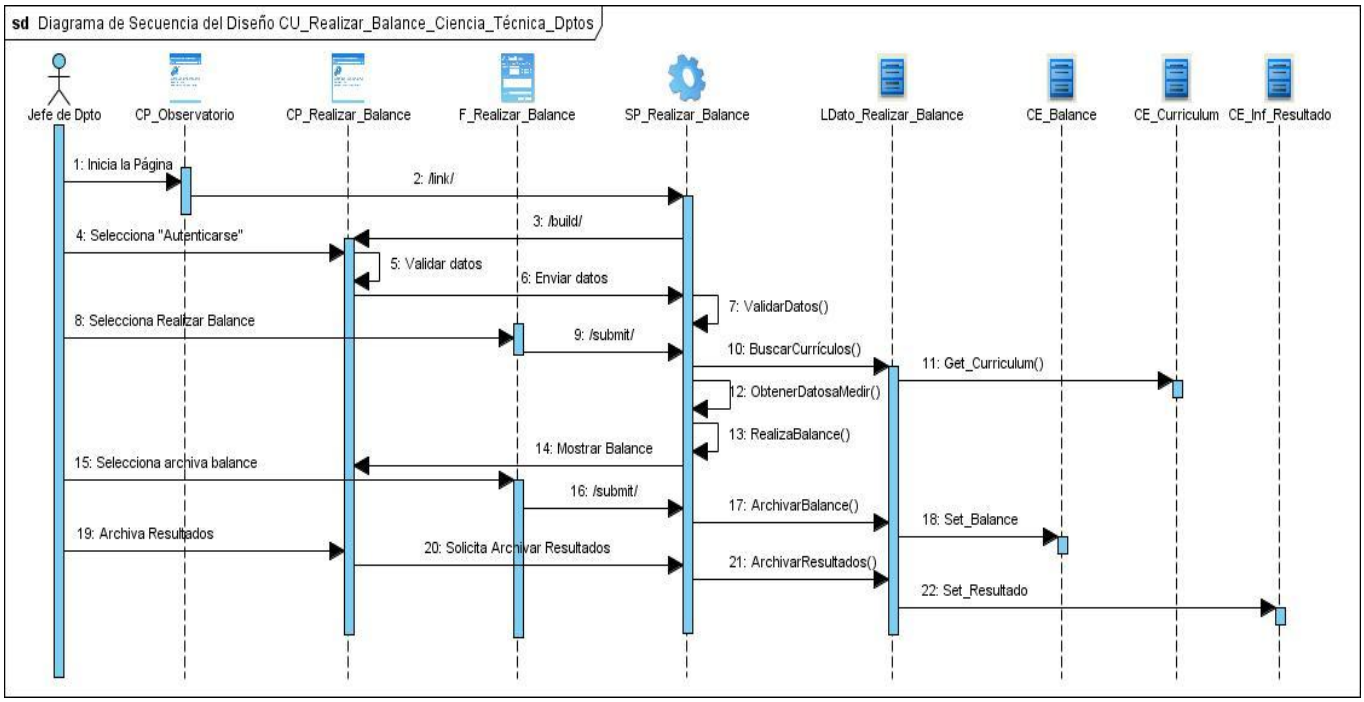


Figura 11. Diagrama de Secuencia del Diseño del CUS: “Realizar Balance de Ciencia-Técnica para Departamentos.”

2.7.3 Patrones de Diseño Utilizados

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles". (Booch, Grady)

Los patrones de diseño proponen soluciones simples y exitosas a problemas habituales que se pueden presentar durante el diseño, basadas en la experiencia. Un patrón de diseño soluciona problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas

etapas del desarrollo desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

El catálogo de patrones más famoso es el contenido en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”, el de la banda de los 4, también conocido como el LIBRO GOF (Gang-Of-Four Book). El grupo GOF agrupó los patrones en tres grandes categorías de acuerdo con su propósito. Este criterio describe la función que el patrón cumple. Los patrones de diseño pueden tener propósito creacional que abstraen el proceso de instancias, estructural que se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño y los de comportamiento que corresponden a los algoritmos y a la asignación de responsabilidades entre objetos. Seguidamente se detallan los patrones del grupo mencionado anteriormente que fueron utilizados en el diseño de la solución.

Patrones Creacionales

- **Fábrica abstracta** (Abstract Factory): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- **Instancia única** (Singleton): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones de Comportamiento

- **Iterador** (Iterator): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.
- **Mediador** (Mediator): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.

Patrón Estructural

- **Fachada** (Facade): El patrón fachada trata de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace de pantalla o fachada.

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas. Los patrones 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 nombre de este grupo se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. A continuación se describen los patrones básicos de asignación de responsabilidades que se manifestaron en el diseño del sistema.

- **Patrón Experto:** Consiste en asignar una responsabilidad al experto en información; a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender.
- **Patrón Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que para conectar con el objeto producido en cualquier evento dando soporte al bajo acoplamiento.
- **Patrón Controlador:** Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente, un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad.

2.8 Conclusiones Parciales

- El análisis y diseño de un sistema ofrece la primera visión de lo que pudiera ser la solución en el desarrollo del mismo.
- En este capítulo se han mostrado las principales funcionalidades que podría tener el sistema a través de una serie de modelos, que garantizan un mayor entendimiento del mismo.
- Se definieron las principales clases del análisis y el diseño mostrando sus relaciones así como la colaboración entre ellas mediante diferentes diagramas.
- Para lograr un diseño más legible se utilizaron algunos patrones que garantizan la claridad de los diagramas.

CAPÍTULO 3: IMPLEMENTACIÓN DEL SISTEMA

3.1 Introducción

El propósito esencial de este capítulo es definir cómo desarrollar la arquitectura comenzando con el resultado de la etapa de diseño e implementando las clases definidas en el capítulo anterior en términos de componentes. Se modelan los diagramas de despliegue y de componentes conformando el modelo de implementación del sistema, dando una visión de cómo quedará construida y distribuida la aplicación.

3.2 Modelo de Implementación

Los diagramas de implementación muestran los aspectos físicos del sistema. Incluyen la estructura del código fuente y la implementación, en tiempo de implementación. Estos diagramas muestran dependencias entre la estructura del sistema en ejecución (diagrama de despliegue) o las partes de código del sistema (diagrama de componentes).

- **Diagramas de despliegue:** Se utilizan para modelar la vista de despliegue estática, muestran la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes.
- **Diagramas de componentes:** Se utilizan para modelar la vista de implementación estática de un sistema, muestran las dependencias entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable.

3.2.1 Diagrama de Despliegue

El Diagrama de Despliegue se utiliza para modelar la arquitectura en tiempo de ejecución de un sistema. Muestra la configuración de los elementos de hardware (nodos) utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Se representa mediante un grafo de nodos unidos por conexiones de comunicación, los nodos son elementos físicos de ejecución que representa un recurso computacional, que generalmente tiene por lo menos memoria y a veces también capacidad de proceso. Los nodos sirven para modelar la topología del hardware sobre el que se ejecuta el sistema. Un nodo representa, generalmente, un procesador o un dispositivo sobre el que se pueden desplegar los componentes y debe tener un nombre asignado que lo distinga del resto de nodos. En el siguiente diagrama se muestra un nodo que representa a la

computadora(PC) desde donde se pueden conectar el usuario (PC Cliente) para acceder a la aplicación, una PC Servidor Web que es donde estará situada la aplicación, una PC Servidor de Base de Datos en donde se encontrará la base de datos del sistema.

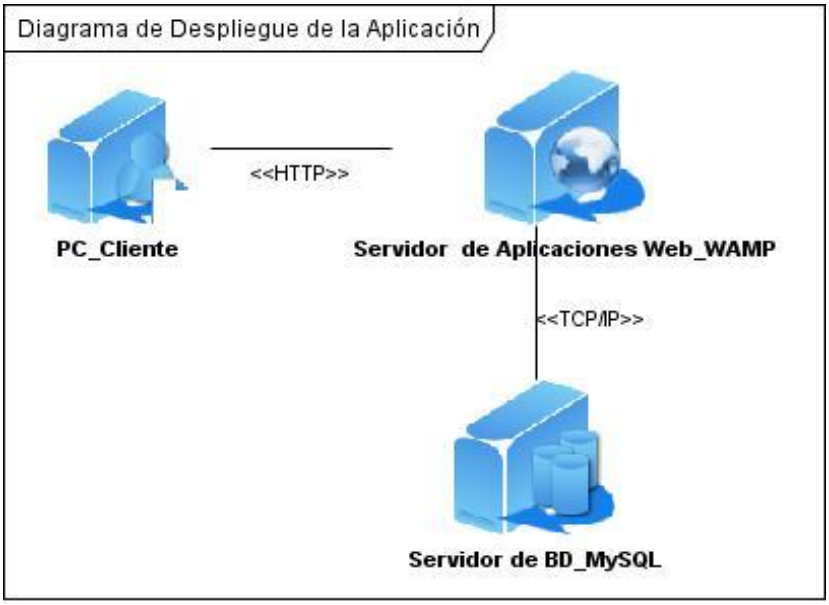


Figura 12. Diagrama de Despliegue de la Aplicación.

3.3.2 Diagrama de Componentes del Sistema

Los diagramas de componentes representan cómo un sistema de software es dividido en componentes, mostrando las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables o paquetes. Se utilizan para modelar la vista estática y dinámica de un sistema, muestran la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, habitualmente se realizan por partes, donde cada diagrama describe un apartado del sistema.

Los diagramas pueden ser usados para modelar y documentar cualquier arquitectura de sistema, los elementos de modelado dentro de un diagrama de componentes son componentes y paquetes. Los diagramas de componentes muestran los componentes software que constituyen una parte reusable, sus interfaces, y sus interrelaciones. Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida.

Seguidamente se muestra la figura del diagrama de componentes de uno de los casos de uso críticos del sistema. Para chequear los diagramas correspondientes al resto de los casos de uso consultar el documento: “Modelo de Implementación”.

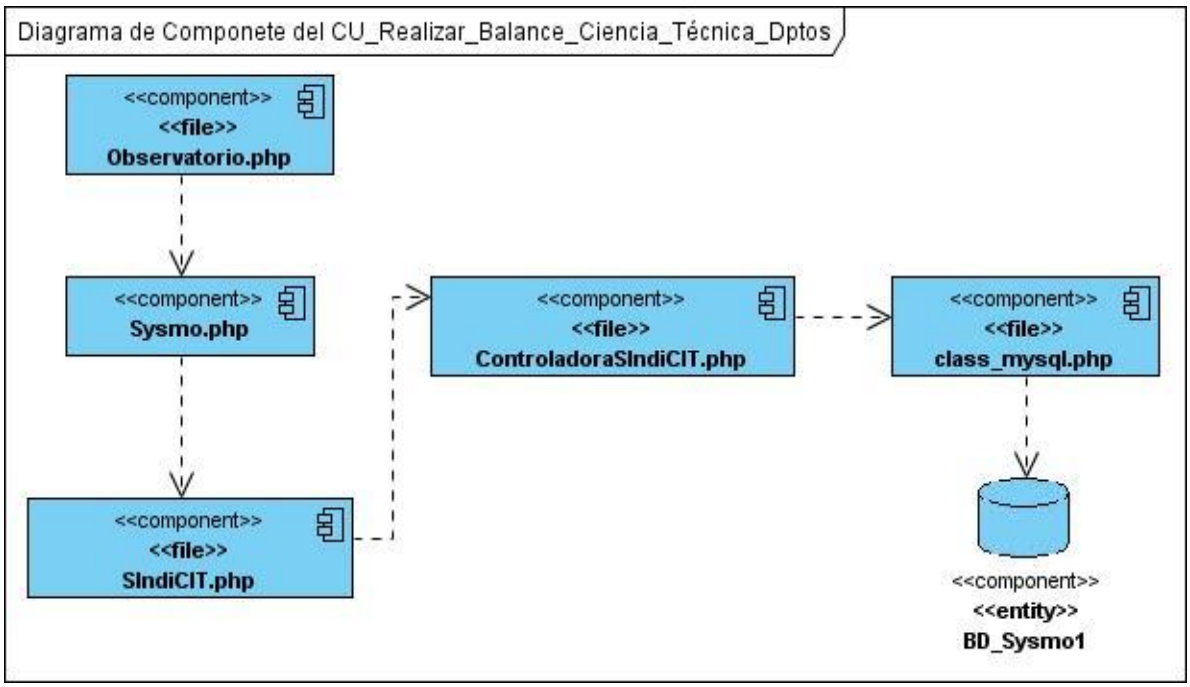


Figura 13. Diagrama de Componentes del CUS: “Realizar Balance de Ciencia-Técnica para Departamentos.”

En la imagen anterior se puede observar que para realizar el balance de ciencia y técnica para un departamento el usuario debe acceder a la página principal del observatorio (**Observatorio.php**) y seleccionar la opción Sistema de Indicadores entonces el sistema le muestra la página **Sysmo.php** que es donde se autentica la persona. Si el usuario tiene los permisos requeridos entonces el sistema le muestra la interfaz donde podrá realizar el balance para su departamento (**SindiCIT.php**). Las páginas mencionadas anteriormente son las que conforman la capa de presentación para funcionalidad seleccionada. El resto de las clases que intervienen en dicha funcionalidad son **ControladoraSindiCIT.php** y **class_mysql.php** las cuales son transparentes al usuario pues se encuentran en las capas de negocio y accesos a datos respectivamente realizando las operaciones seleccionadas por el usuario en la interfaz.

A continuación se brinda una breve descripción de las principales secciones que tiene el portal del observatorio implementado. La aplicación cuenta con una interfaz principal donde se brindan todos los servicios agrupados en varias secciones:

- La sección "***Formación***" brinda dos servicios: "***Currículum***" donde los profesores pueden crear su currículum en línea y los jefes de Departamento revisarlos. El otro servicio de este apartado es "***Sistema de Indicadores***" a través del cual los jefes de Departamento y asesores de investigación pueden realizar el balance de ciencia y técnica tanto para cada Departamento como para la Facultad.
- La sección "***Investigación y Postgrado***" ofrece servicios como: "***Próximos Eventos***" donde se publica la información referente a los eventos científicos y técnicos de interés para la Facultad. "***Revistas para Publicaciones***" contiene acceso a las revistas donde los docentes puedan publicar trabajos investigativos. También presenta vínculos a otros portales web de universidades donde se oferten cursos de postgrado y maestrías.
- La sección "***Nichos Investigativos***" facilita información referente a las líneas investigativas de la facultad, así como enlaces a portales web de otras entidades que también manipulen términos asociados con estos perfiles de investigación.

3.4 Conclusiones Parciales

- En este capítulo se ha mostrado los principales aspectos físicos del sistema mediante una serie de modelos, mostrando una visión de cómo quedaría construida y distribuida la aplicación.
- Se definieron los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución para modelar la vista de despliegue estática del sistema.
- Se identificaron componentes de software, incluyendo componentes de código fuente, binario y ejecutable para modelar la vista de implementación estática del sistema.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

4.1 Introducción

Durante todo el proceso de implementación se fue realizando la gestión de la calidad del producto mediante el uso de métricas y técnicas de pruebas de software. En este capítulo se mostrarán los resultados obtenidos luego de aplicar métricas a nivel de componentes y orientadas a clases en el modelo del diseño. Como parte de las técnicas se abordarán las pruebas de caja blanca que fueron realizadas a la capa de presentación, a la capa de lógica del negocio y a la integración de ambas capas, y las pruebas de caja negra que se llevaron a cabo sobre la interfaz del software. Se realizará un análisis de los resultados obtenidos durante el proceso de prueba, para conocer si se cumplió con el objetivo propuesto.

4.2 Métricas del Modelo de Diseño

Las medidas y las métricas son componentes claves para comprender mejor la calidad del producto, estimar la efectividad de los procesos y mejorar la calidad del trabajo realizado a nivel de proyecto. Las métricas de diseño proporcionan al diseñador una mejor visión interna que le permitirá guiar el diseño hacia un nivel superior de calidad.

Métricas de diseño a nivel de componentes: se concentran en las características internas de los componentes del software. (Pressman, 2005)

- **Métricas de cohesión**: proporcionan una indicación de la cohesión de las clases del diseño. A continuación se muestra la ecuación para calcular la Cohesión Funcional Débil:

$$CFD = \frac{\text{número de adhesivos (i)}}{\text{número de elementos (i)}}$$

(i): se define como la muestra.

Adhesivo: se le llamará adhesivo a un elemento que aparece en dos o más clases.

Súper adhesivo: se le llamará súper adhesivo a un elemento que aparece en todas las clases.

Mientras más cercano esté el valor de CFD a 1 mayor será la cohesión del módulo.

Tabla 9. Usabilidad de las clases.

No	Nombre de la clase	Usabilidad
1	SP_Visualizar_Eventos	3
2	E_Evento	3

3	LDatos_Buscar_Eventos	1
4	SP_Visualizar_Curso	2
5	LDatos_Visualizar_Curso	1
6	E_Curso	1
7	SP_Conf_Curriculum	2
8	E_Curriculum	4
9	SP_Realizar_Balance	2
10	LDatos_Realizar_Balance	1
11	E_Balance	2
12	E_Inf_Resultados	2
13	SP_Gestionar_Eventos	2
14	SP_Revisar_Currículum	2

$$CFD = \frac{\text{número de adhesivos (i)}}{\text{número de elementos (i)}}$$

$$CFD = \frac{10}{14} = 0.71$$

La relación del número de clases adhesivas con el número total de elementos de la muestra determina que el diseño de clases posee una cohesión funcional alta para un 71% de fortaleza.

Métricas orientadas a clases: Hacen hincapié en el encapsulamiento, la herencia, complejidad de clases y polimorfismo. A continuación se describen las que se le aplicaron a los diagramas de clases del diseño. (Pressman, 2005)

- **Árbol de profundidad de herencia (APH):** La métrica fue propuesta por Chidamber y Kemerer (CK). Mide la longitud máxima desde el nodo hasta la raíz del árbol. Si es demasiado profundo las clases inferiores heredan muchos métodos y hay mucha complejidad en el diseño. Esto da lugar a posibles dificultades cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda (con un valor grande de APH) lleva también a una mayor complejidad de diseño. Por el lado positivo, los valores grandes de APH implican que se pueden reutilizar muchos métodos.

  Al aplicar la métrica se demostró que el sistema no presenta un alto nivel de jerarquía de herencia pues el **APH** toma valor 1, ya que solo hay presencia de herencia entre las clases y

las interfaces de las cuales heredan sus funciones y las redefinen. Por tanto, se concluye que existe bajo acoplamiento entre las clases garantizando que los cambios sean fáciles de controlar.

- **Tamaño de la clase (TC):** Esta métrica pertenece a la familia de métricas propuesta por Lorens y Kidd. Ellos proponen que para determinar el tamaño de una clase se debe tener en cuenta las medidas siguientes:

  El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.

  El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

  El TC es el resultado del promedio general de las dos medidas anteriores para el sistema completo. Si existen valores grandes de TC éstos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilización de la clase y complicará la implementación y la comprobación.

  Se aplicó esta métrica para las principales clases de las tres capas definidas:

Tabla 10. Tamaños de clases.

No	Nombre de la clase	Número de atributos	Número de operaciones
1	SP_Visualizar_Eventos	0	3
2	E_Evento	13	0
3	LDatos_Buscar_Eventos	2	3
4	SP_Visualizar_Curso	0	4
5	LDatos_Visualizar_Curso	4	3
6	E_Curso	7	0
7	SP_Conf_Curriculum	0	6
8	E_Curriculum	15	0
9	SP_Realizar_Balance	0	11
10	LDatos_Realizar_Balance	5	24
11	E_Balance	5	0
12	E_Inf_Resultados	9	0
13	SP_Gestionar_Eventos	0	6

14	LDatos_ Gestionar_Eventos	2	6

De esta forma, el umbral queda con los datos mostrados a continuación:

Tabla 11. Resultados de la métrica: "Tamaño de Clase".

Umbral	Tamaño	Cantidad de Clases
<= 20	Pequeño	13
>20 y <=30	Medio	1
>30	Grande	0

Esta métrica se le aplica a conjunto de clases pertenecientes a las tres capas definidas para la arquitectura del sistema. De acuerdo con los umbrales que se presentan en la tabla anterior, el 93% de las clases se consideran pequeñas, el 7% son medianas y no existen clases grandes, por tanto, el diseño de clases no es complejo.

4.3 Pruebas de Software

Desarrollar un software implica una serie de actividades de producción la cuales son realizadas por personas y éstas cometen errores y cambian de ideas. Los errores pueden empezar a darse desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea e imperfecta; así en los posteriores pasos del diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad.

Las pruebas constituyen la actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es realizada de algún aspecto del sistema o componente. 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. (Jacobson, y otros, 2000)

4.4 Tipos de Pruebas

En la comúnmente llamada fase de pruebas se pueden encontrar diferentes niveles y tipos de pruebas, definiéndose como: (Jacobson, y otros, 2000)

Pruebas de Integración: Se comprueba la compatibilidad y funcionalidad de los interfaces entre las distintas 'partes' que componen un sistema, estas 'partes' pueden ser módulos, aplicaciones

individuales, aplicaciones cliente/servidor, etc. Este tipo de pruebas es especialmente relevante en aplicaciones distribuidas.

Pruebas de Validación: Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento con los requisitos especificados.

Pruebas de Sistema: el software ya validado se integra con el resto del sistema donde algunos tipos de pruebas a considerar son:

- **Rendimiento:** determinan los tiempos de respuesta, el espacio que ocupa el módulo en disco o en memoria, el flujo de datos que genera a través de un canal de comunicaciones, etc.
- **Resistencia:** determinan hasta donde puede soportar el programa determinadas condiciones extremas.
- **Robustez:** determinan la capacidad del programa para soportar entradas incorrectas.
- **Seguridad:** se determinan los niveles de permiso de usuarios, las operaciones de acceso al sistema y acceso a datos.
- **Usabilidad:** se determina la calidad de la experiencia de un usuario en la forma en la que éste interactúa con el sistema, se considera la facilidad de uso y el grado de satisfacción del usuario.
- **Instalación:** se determinan las operaciones de arranque y actualización del software.

Pruebas de Aceptación*:* Son las pruebas que hará el cliente, se determina que el sistema cumple con lo deseado y se obtiene la conformidad del cliente.

Durante todo el proceso de implementación se fue realizando paralelamente la gestión de la calidad del producto mediante el uso de técnicas de pruebas de software. Las técnicas de prueba ayudan a definir conjuntos de casos de prueba aplicando un cierto criterio. Los casos de prueba quedarán determinados por los valores a asignar a las entradas en su ejecución. Algunas de las técnicas que se han utilizado son las pruebas de caja blanca que fueron realizadas a la capa de presentación, a la capa de lógica del negocio y a la integración de ambas capas; y las pruebas de caja negra que se llevaron a cabo sobre la interfaz del software. Se realizará un análisis de los resultados obtenidos durante el proceso de prueba, para conocer si se cumplió con el objetivo propuesto inicialmente.

4.5 Pruebas de Caja Blanca

Las pruebas de caja blanca se basan en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Para dichas pruebas se utilizó el framework

PHPUnit, el cual brinda un conjunto de librerías que se integran fácilmente al IDE de desarrollo seleccionado: NetBeans, permite realizarle pruebas a las clases implementadas en ambas capas. Mediante estas pruebas el ingeniero del software puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello, que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican al software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. (Pressman, 2005)

Algunos de los métodos empleados en las pruebas de caja blanca son los siguientes:

Prueba del camino básico: Es una técnica que le permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Algunos elementos y conceptos utilizados alrededor de este método son:

- *Grafo de flujo o grafo del programa:* representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. (Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control)
- *Complejidad ciclomática:* es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. Esta medida ofrece al probador de software un límite superior para el número de pruebas que debe realizar para garantizar que se ejecutan por lo menos una vez cada sentencia.
- *Camino independiente:* cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición.

Prueba de la estructura de control: dentro de este tipo de pruebas se contempla el método del camino básico mencionado anteriormente pero además existen otras pruebas asociadas que permiten ampliar la cobertura de la prueba y mejorar su calidad. Estas son: (Pressman, 2005)

Prueba de condición: es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. Algunos conceptos empleados alrededor de esta prueba son los siguientes:

- *Condición simple:* es una variable lógica o una expresión relacional.
- *Condición compuesta:* está formada por dos o más condiciones simples, operadores lógicos y paréntesis.

Prueba del flujo de datos: Se eligen caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de bucle: es una técnica que se centra exclusivamente en la validez de las construcciones de bucles (bucles simples, anidados, concatenados y no estructurados).

Pruebas unitarias: En este tipo de pruebas se analiza una porción de código que pueda ser analizada de manera aislada, como por ejemplo funciones y métodos. A los que después de pasarle unos parámetros de entrada se deben obtener otros parámetros de salida claramente definidos.

Seguidamente se muestran una imagen que representan la realización de pruebas unitarias que se les aplicaron a las funcionalidades implementadas en las clases de las capas de presentación y lógica de negocio.

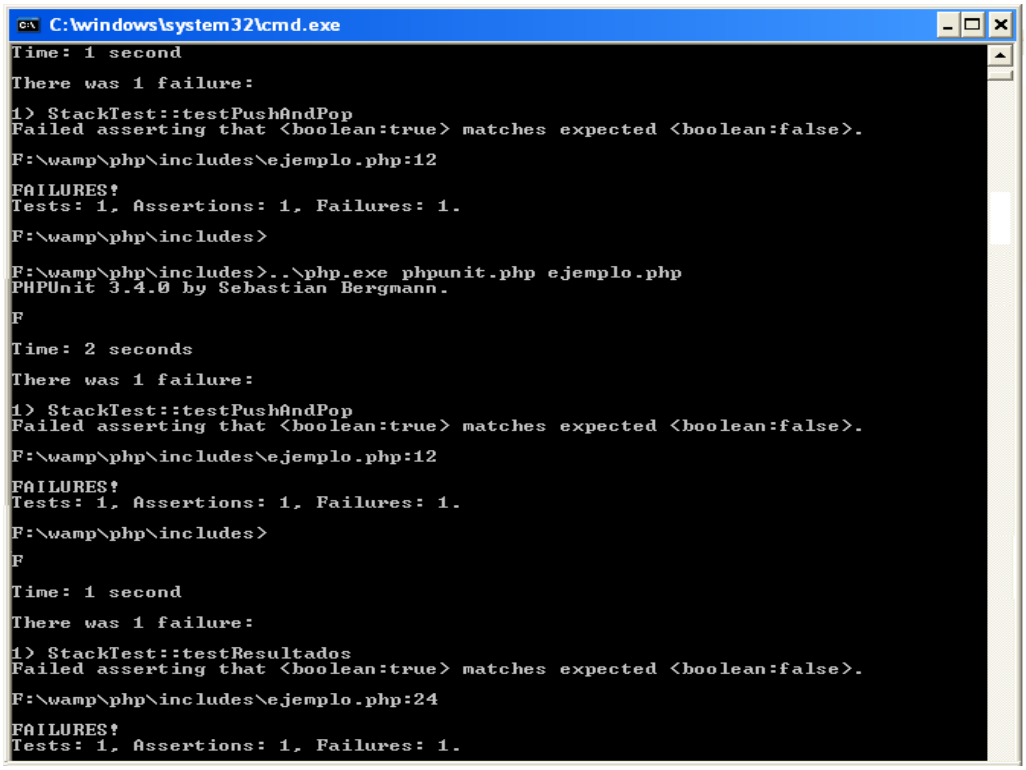


Figura 14. Primera iteración de pruebas unitarias.

La imagen anterior muestra una sección de los resultados arrojados durante la realización de la primera iteración de pruebas unitarias. Releva tres métodos, que luego de ser probados, retornaron fallos. Los errores fueron tratados y en una segunda iteración de pruebas fueron reevaluados logrando los resultados que aparecen en la siguiente imagen.

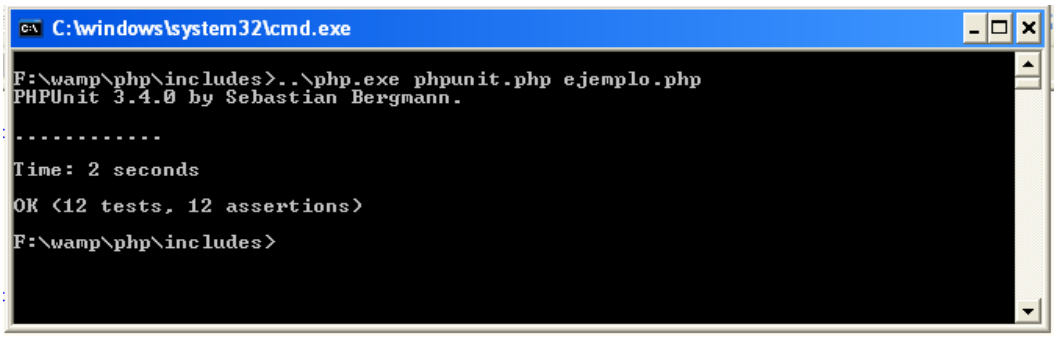


Figura 15. Segunda iteración de pruebas unitarias.

La figura anterior muestra los resultados de 12 métodos donde se encuentran los tres métodos que en la primera iteración resultaron fallidos y 9 métodos que corresponden a las funcionalidades más

importantes del sistema. Como se puede apreciar en dicha imagen, los resultados arrojados esta vez fueron satisfactorios.

4.6 Pruebas de Caja Negra

Las pruebas de Caja Negra se centran principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Las pruebas de Caja Negra no son una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Estas pruebas permiten encontrar: (Pressman, 2005)

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes. Pueden ser datos válidos o inválidos para el programa según lo que se desea, si es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: (Pressman, 2005)

- *Técnica de la Partición de Equivalencia:* esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- *Técnica del Análisis de Valores Límites:* esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- *Técnica de Grafos de Causa-Efecto:* es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Una de las pruebas a desarrollar, utilizando la técnica de caja negra, es la de **“Validación”**, que es realizada sobre la interfaz del sistema. Esta prueba se realiza luego de integrar todos los módulos

que conforman al sistema para comparar las especificaciones obtenidas durante el análisis y diseño del sistema contra el funcionamiento del sistema final. La correcta realización de estas pruebas es de gran importancia, pues con ellas se pueden detectar varios errores. Para realizar dichas pruebas se diseñaron casos de prueba para cada caso de uso del sistema con el objetivo de encontrar errores en las funcionalidades implementadas.

4.7 Diseño de Casos de Prueba

Con el diseño de casos de pruebas se obtienen un conjunto de pruebas que tienen como misión encontrar defectos y errores al sistema. Cada caso de prueba brinda un número de valores de entradas en las pruebas, condiciones de ejecución y resultado esperados en de las mismas para verificar una determinada funcionalidad del sistema.

El objetivo principal de las pruebas es comprobar que el producto cumpla con las necesidades y exigencias de los usuarios finales del producto. Se diseñaron casos de pruebas para cada una de las funcionalidades del sistema los cuales pueden ser consultados en el documento "***Casos de Pruebas***".

4.8 Resultados

Los resultados obtenidos de las pruebas realizadas al Observatorio Tecnológico para la Facultad 15 fueron satisfactorios. Se hicieron pruebas sobre el sistema enfocadas a las técnicas de caja blanca y caja negra. Además, las pruebas del sistema se dividieron en dos etapas. En la primera etapa los desarrolladores le hicieron dos iteraciones de pruebas. La segunda etapa comprende las revisiones realizadas por parte del equipo de calidad de la facultad. Mediante el desarrollo de los casos de pruebas se identificaron los errores que tenía el sistema los cuales fueron reevaluados a través de la realización de nuevas pruebas en una segunda iteración y donde no se encontraron nuevos errores.

La gráfica que a continuación que se presenta muestra los resultados obtenidos en las dos iteraciones de pruebas de validación realizadas al sistema. Incluye los resultados arrojados en las pruebas orientadas a la técnica de caja negra que se realizaron en las capas de presentación y negocio, y sobre la interfaz de la aplicación respectivamente.

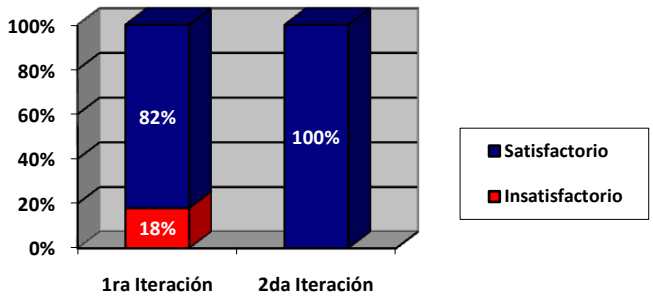


Figura 16. Resultados obtenidos en las pruebas de Sistema.

Después de la revisión del software por parte del personal de calidad de la facultad, y de los usuarios finales de la aplicación se realizó un análisis de los resultados obtenidos teniendo en cuenta los principales aspectos de medición, los cuales se describen a continuación:

Funcionalidad: La aplicación mostró ser completamente funcional, dando solución a todos los requisitos solicitados por los clientes, los cuales fueron identificados y validados en la etapa de Levantamiento de Requisitos.

Fiabilidad: Todos los problemas de tratamiento de errores que fueron descubiertos durante el desarrollo fueron solucionados, por tanto, la aplicación es capaz de detectar cualquier tipo de error, recuperarse y enviar a los interesados el tipo de error ocurrido.

Eficiencia: Debido a la estrategia de integración utilizada entre las capas de presentación y lógica de negocio, la aplicación es capaz de darle respuestas a todas las solicitudes del usuario en un tiempo relativamente corto, aunque en este tiempo de respuesta influyen de manera directa las condiciones de conexión que posean los clientes.

Usabilidad: Teniendo en cuenta que el software fue realizado para un grupo de usuario específico y el cual posee suficiente conocimiento de los procesos que se automatizaron, la aplicación mostró que es muy fácil la navegación por sus páginas, aún para personas que poseen pocos conocimientos de los procesos. Además de poseer una interfaz de usuario sencilla y agradable.

Mantenibilidad: Debido a que el sistema posee una arquitectura por capas darle mantenimiento es muy cómodo. Es fácil para cualquier persona que posea conocimientos del lenguaje de programación realizar los cambios solicitados, con vista a mejorar o adicionar nuevas funcionalidades.

Portabilidad: El software puede ser utilizado con varios navegadores web, como el Mozilla Firefox, Internet Explorer y desde cualquier sistema operativo.

Para certificar la veracidad de los parámetros anteriormente expuestos se confeccionaron actas de aceptación por parte del cliente y del equipo de calidad de la Facultad 15. Para consultar dicha información (Ver Anexo1).

4.9 Conclusiones Parciales

- Durante la etapa de pruebas se le aplicaron varias pruebas al software enfocadas a las técnicas de caja blanca y caja negra para comprobar el correcto funcionamiento del sistema y la integración de sus capas. Se utilizó la herramienta **PHPUnit** integrada al entorno de desarrollo integrado **NetBeans** para realizar las pruebas de unidad. También se utilizaron métricas para medir la calidad del producto final.
- Se mostraron los resultados arrojadas en cada una de las iteraciones de pruebas de la aplicación y se comprobó finalmente que el sistema cumplía con los requisitos determinados por los clientes y las normas de calidad establecidas por **CALISOFT**.

CONCLUSIONES

Al finalizar el presente trabajo se arribaron a las siguientes conclusiones:

- Con el estudio realizado, se lograron identificar los distintos observatorios tecnológicos existentes actualmente y se decidió no utilizar ninguno de estos. Cada observatorio es realizado para una determinada entidad, lo que implica establecer las características y los perfiles que interesan a sus usuarios. Los servicios que ofertan están orientados hacia las necesidades específicas de la propia entidad que lo sustenta.
- Se definieron las características del observatorio tecnológico que se necesitaba, resolviendo realizar una aplicación Web debido a las facilidades que brindan para aplicaciones "*En-Líneas*", una característica fundamental del observatorio. Además, soluciona problemas de compatibilidad y garantiza que el acceso a la aplicación sea sencillo, así como alta privacidad a los usuarios, estableciendo un sistema seguro de autenticación.
- En la etapa de análisis y diseño de la aplicación se obtuvo la primera visión de la solución mostrando las principales funcionalidades que se implementarían en el sistema.
- Se obtuvo una serie de modelos mediante los cuales se mostraron los principales aspectos físicos del sistema. Se mostró una visión de cómo quedaría construida y distribuida la aplicación. Se implementaron todas las funcionalidades del sistema según las necesidades y exigencias de los clientes.
- Para validar la solución se realizaron varias iteraciones de pruebas enfocadas a las técnicas de caja blanca y caja negra, auxiliadas por métricas para comprender mejor la calidad del producto y estimar la efectividad de los procesos. Por otra parte, los artefactos del proyecto, fueron liberados por el equipo de calidad de la facultad después de ser revisados.

Con el desarrollo de este trabajo se le dio cumplimiento al objetivo general de la investigación propuesta: desarrollar una solución informática que permita la gestión y control de la información de investigación y postgrado de la facultad 15.

RECOMENDACIONES

- Teniendo en cuenta las ventajas que proporciona la solución propuesta, se recomienda hacerla extensiva al resto de las facultades e integrarla con las herramientas propuestas por la dirección de la Universidad.
- Mantener los temas abordados en el Observatorio actualizados para que los usuarios no dejen de acceder al sistema, y por consiguiente a las secciones de su interés.
- Agregar nuevos temas de interés para la Facultad en la aplicación, de manera que se logren incentivar las visitas de los usuarios a la misma.

BIBLIOGRAFÍA

AITEX. 2002. Observatorio Tecnológico. [En línea] 2002. [Citado el: 3 de 10 de 2009.] http://www.observatoriotextil.com/es/presentacion/presen_1024.htm.

Albaladejo, Xavier. 2009. SCRUM un proceso de trabajo. [En línea] 15 de 8 de 2009. http://www.proyectosagiles.org/scrum-proceso-trabajo-2-0.

Alvarez, Miguel Angel. 2003. Qué es Python. [En línea] 19 de 11 de 2003. [Citado el: 26 de 1 de 2010.] http://www.desarrolloweb.com/articulos/1325.php.

—. 2003. Qué es Python. [En línea] 19 de 11 de 2003. [Citado el: 26 de 1 de 2010.] http://www.desarrolloweb.com/articulos/1325.php.

—. 2003. Zend Studio. [En línea] 4 de 6 de 2003. [Citado el: 26 de 1 de 2010.] http://www.desarrolloweb.com/articulos/1178.php.

Barros, Oscar. 1994. *Reingeniería de Procesos de negocio.* Chile : Dolmen , 1994.

Carlos Galarza. 2009. Manual de Oracle y PI Sql. [En línea] 17 de 4 de 2009. [Citado el: 27 de 1 de 2010.] http://www.scribd.com/doc/3589792/Manual-De-Oracle-Y-Pl-Sql.

Dadiskod. 2010. Website de Dadiskod. [En línea] 2010. [Citado el: 26 de 1 de 2010.] http://dadisokd.netau.net/index.php/component/content/article/44-software-libre-dadisokd/94-netbeans-dadisokd.

Espinosa, Leidy Alfonso. 2008. *Análisis de Tendencias sobre Observatorios Tecnológicos a nivel nacional e internacional*. Departamento de Desarrollo de Productos y Servicios. Dirección de Información. 2008.

Expósito, Erly Delgado. 2008. Metodologías de desarrollo de software. ¿Cuál es el camino? [En línea] 2008. [Citado el: 21 de 11 de 2009.] http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml.

Franco, Miguel. 2008. Concepto de CMS- Aula 2.1. [En línea] 10 de 1 de 2008. [Citado el: 16 de 1 de 2010.] http://www.aula21.es/aula/spip.php?article6.

Galarza, Carlos. 2009. Manual de Oracle y PI Sql. [En línea] 17 de 4 de 2009. [Citado el: 27 de 1 de 2010.] http://www.scribd.com/doc/3589792/Manual-De-Oracle-Y-Pl-Sql.

Gamma, Erich, y otros. 1998. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley , 1998.

García, Ariadna y Bouza, Rita Raposo y Odalis. 2009. Observatorio Científico Tecnológico: Propuesta de un modelo para el sector empresarial cubano. [En línea] julio de 2009. http://www.sociedadelainformacion.com/16/observatoriocientifico.pdf.

Jacobson, Ivar y Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo Software.Capítulo 11 (Prueba).* s.l. : Pearson Addisson-Wesley, 2000. pág. 281 a la 299.

—. 2000. *Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley, 2000.

Joomlaos.net. 2009. Joomlaos.net. [En línea] 2009. [Citado el: 16 de 1 de 2010.] http://www.joomlaos.net/caracteristicas-de-joomla.php.

Kruchten, Philippe. 1995. *Architectural Blueprints The 4+1 View Model of Software Architecture.* 11 : IEEE Software, Institute of Electrical and Electronics Engineers, 1995. págs. 42-50.

—. 2000. *The Rational Unified Process: An Introduction.* s.l. : Addison Wesley, 2000.
La Coctelera. 2008. Blog La Coctelera. *Servicios de Vigilancia Tecnológica.* [En línea] 2008. [Citado el: 16 de 10 de 2009.] http://observatorio-tecnologico.espacioblog.com/post/2006/04/18/-servicios-vigilancia-tecnologica-existentes-.
—. 2008. Blog La Coctelera. *¿Por qué un Observatorio Tecnológico?* [En línea] 2008. [Citado el: 3 de 10 de 2009.] http://www.lacoctelera.com/observatorio-tecnologico/.
Martínez, Rolando Herrera. 2007. UXI – Revista de Software Libre de la UCI. [En línea] 9 de 2007. [Citado el: 8 de 2 de 2010.] http://revistauxi.files.wordpress.com/2007/10/uxi7.pdf.
Masip, David. 2002. Qué es Oracle. [En línea] 17 de 9 de 2002. [Citado el: 27 de 1 de 2010.] http://www.desarrolloweb.com/articulos/840.php.
Muras, Miguel Angel. 2009. Moisaic tecnología y comunicación multimedia. [En línea] 23 de 12 de 2009. [Citado el: 16 de 1 de 2010.] http://mosaic.uoc.edu/2009/12/23/una-aproximacion-a-los-gestores-de-contenidos-cms/.
Oracle ERP System. 2009. Actualidad y Tecnoligía. [En línea] 24 de 9 de 2009. http://estudiantessistemasucv.blogspot.com/.
Pressman, Roger S. 2005. *La ingeria del Software. Un enfoque práctico.* s.l. : McGraw-Hill, 5ta edición, 2005. págs. 281- 300 Capítulo 17.
Saha, Amit Kumar. 2008. MySQL: Introducción. Características y beneficios. [En línea] 2008. [Citado el: 11 de 2 de 2010.] http://www.gedlc.ulpgc.es/docencia/abd/Recursos/MySQL-Intro-features-benefits-SPANISH.pdf.
—. MySQL: Introducción. Características y beneficios. [En línea] [Citado el: 11 de 2 de 2010.] http://www.gedlc.ulpgc.es/docencia/abd/Recursos/MySQL-Intro-features-benefits-SPANISH.pdf.
SlideShare. 2009. Visual Paradigm For UML. [En línea] 2009. [Citado el: 23 de 1 de 2010.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.
Solórzano, L. F. 2007. *"El Observatorio Tecnológico como puente entre el mundo de la tecnología y el mercado".* 2007.
Sun Microsystems. 2009. Array Exception. Tecnologia y Desarrollo. [En línea] 20 de 12 de 2009. [Citado el: 26 de 1 de 2010.] http://www.arrayexception.com/desarrollo/java-desarrollo/sun-microsystems-lanza-netbeans-ide-68/.
—. 2009. Sun Microsystems lanza NetBeans IDE 6.8. [En línea] 14 de 12 de 2009. http://www.acceso.com/display_release.html?id=61473.
—. 2009. Sun Microsystems lanza MySQL 5.1. [En línea] 20 de 3 de 2009. http://www.infosertec.com.ar/blog/?p=4504.
Welicki, León E. Implementando Extreme Programming en la plataforma. [En línea] UPSAM, Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software, Madrid España. http://www26.brinkster.com/lwelicki/articles/Implementando%20Extreme%20Programming%20en%20la%20plataforma%20NET.pdf.

ANEXOS

Anexo 1: Actas de validación del sistema.

UCi

Informáticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Viernes, 11 de junio de 2010.

Luego de haber efectuado 2 iteraciones de revisiones a los artefactos: Especificación de Requisitos, Modelo de Negocio, Modelo de Sistema, Modelo de Diseño y Modelo de Despliegue del sistema Observatorio Tecnológico para la Facultad 15 y haberse detectado un promedio de 15 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que queda liberada la aplicación.

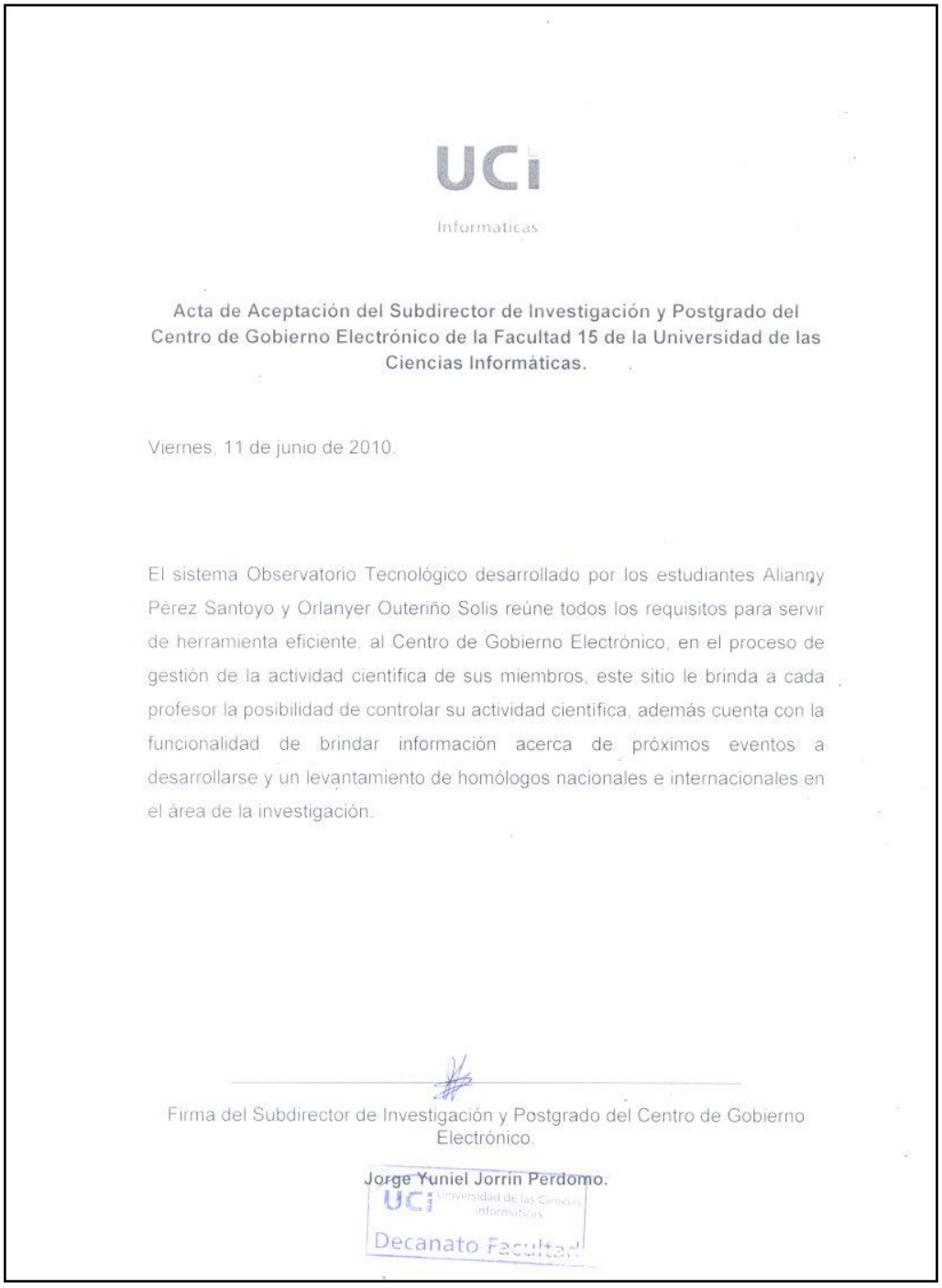
Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

UCi Universidad de las Ciencias Informáticas

Decanato Facultad

Figura 17. Acta de Liberación del sistema por parte del grupo de calidad.



UCi

Informaticas

Acta de Aceptación del Subdirector de Investigación y Postgrado del Centro de Gobierno Electrónico de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Viernes, 11 de junio de 2010.

El sistema Observatorio Tecnológico desarrollado por los estudiantes Alianny Pérez Santoyo y Orlanyer Outeriño Solis reúne todos los requisitos para servir de herramienta eficiente, al Centro de Gobierno Electrónico, en el proceso de gestión de la actividad científica de sus miembros, este sitio le brinda a cada profesor la posibilidad de controlar su actividad científica, además cuenta con la funcionalidad de brindar información acerca de próximos eventos a desarrollarse y un levantamiento de homólogos nacionales e internacionales en el área de la investigación.

Firma del Subdirector de Investigación y Postgrado del Centro de Gobierno Electrónico.

Jorge Yuniel Jorrín Perdomo.

UCi Universidad de las Ciencias Informáticas

Decanato Facultad

Figura 18. Acta de Aceptación del sistema por parte de los usuarios finales.