Universidad de las Ciencias Informáticas

Facultad 8

Sistema para la Gestión de Información de los Proyectos Productivos en la Facultad 8 (SGIFP).

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

Autores: Helen Caridad García González.

Mario Enrique Olivares Ferreira.

Tutor: Ing. Arcel Labrada Batista.

Ciudad de La Habana

Junio de 2008

"Año 50 de la Revolución"

"La virtud, como el arte, se consagra constantemente a lo que es difícil de hacer y cuanto más dura es la tarea, más brillante es el éxito."

Aristóteles

Declaración de Autoría.

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con el mismo.

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

_____
Helen Caridad García González.
Firma del Autor.

_____
Mario Enrique Olivares Ferreira.
Firma del Autor.

_____
Ing. Arcel Labrada Batista.
Firma del Tutor.

Agradecimientos.

De Nosotros:

A la Revolución, por poner en nuestras manos la Universidad de las Ciencias Informáticas (UCI).

A nuestras familias, por guiarnos siempre por el mejor de los caminos.

Al profe Renier Portelles Cobas, nuestro guía doblemente, en el grupo y en los momentos mas difíciles de la tesis. Por su ayuda, sus consejos y su empeño en que saliéramos adelante.

A Arcel Labrada Batista nuestro tutor, por los momentos que nos dedicó.

A Albert y Jose, compañeros de batalla de Symfony que nunca nos abandonaron ante las dudas y dificultades que nos asaltaron a lo largo del trabajo.

A Arieskien, el mejor de todos los compañeros, siempre dispuesto a ayudar de verdad.

A todos nuestros compañeros de siempre: Dani, Yanirys, Danay y Yoandry (los piojos), Lisesita y Pedrito (los morrocoyos).

A todo el grupo en general por responder a nuestras preguntas y por escucharnos siempre en los momentos buenos y malos.

A todo el que de alguna manera ha contribuido a hacer nuestro gran sueño realidad.

Y el más especial de todos los agradecimientos para el tío Vladimir.

Dedicatoria.

De Helen:

A mi mamita, por ser lo más grande que tengo en este mundo, por siempre estar ahí por mí y para mí.

A mimi, mi segunda mamá, lo otro grande de mi vida.

A tita, papi, rá, papito, el brother, la amiga, los nenes, por ser mi familia, por su confianza y apoyo.

A Ane, por sus consejos y amistad.

A Mirio y Martica.

Al chuchito, por TODOOOO y TANTOOOO.

De Mario:

Dedico este trabajo y sobre todo mi esfuerzo que es mucho más valioso que el resultado final:

A mi tío Vladimir, primeramente por ser faro y guía para mí a lo largo de toda mi vida.

A mi mamá, mi papá, mi abuela, mi abuelo, Remigio, mis tíos y tías, que de una forma u otra siempre se han sacrificado para ayudarme como un hijo más y siempre los tengo presente a todos.

A mis hermanos, a mis primos, a mis abuelos que ya no están físicamente conmigo.

A mis amigos, que han sido parte importante siempre para mí.

Y a mi novia Helen, que compartió todos los momentos malos y buenos que pasamos en esta travesía.

Resumen.

En la Facultad 8 de la Universidad de las Ciencias Informáticas (UCI) la información relacionada con los proyectos productivos que en ella se desarrollan es controlada manualmente, ya que no se cuenta con un sistema encargado de su gestión, por lo que pueden sucederse retrasos en las búsquedas y obtención de reportes especializados, así como pérdidas de datos. En el presente Trabajo de Diploma titulado: "Sistema para la Gestión de Información de los Proyectos Productivos en la Facultad 8" (SGIFP), bajo la disciplina de desarrollo propuesta por la metodología RUP, se llevan a cabo el modelamiento del negocio, el levantamiento de requisitos, el análisis y diseño, así como la implementación y prueba de un sistema que permite gestionar con mayor eficiencia la información referente a los proyectos productivos de la facultad en cuestión, basada en la rapidez a partir de la automatización de varios de los procesos que se desarrollan en dicha área.

Palabras claves: sistema de gestión de información y proyectos productivos.

Índice.

Índice de Contenidos.

Índice de Tablas.

Índice de Figuras.

Introducción.

La información es considerada un grupo organizado de datos, que al relacionarse adquieren un sentido, constituyendo un mensaje sobre un determinado fenómeno. Históricamente se ha sistematizado en archivos y en sistemas bibliotecarios. Su manejo por el hombre, implica que este exprese voluntad de poder, manifestado en el saber que ella contiene y que su consciente actuación motiva transformaciones culturales.

Con el devenir de los años, la información se ha ido manifestando cambiantemente. En un comienzo, su almacenamiento, acceso y uso limitado, se realizaba en las bibliotecas de los monasterios. Más tarde con el nacimiento de la imprenta, se comenzaron a fabricar los libros en serie y surgen los primeros periódicos. Luego irrumpen la radio, la televisión e Internet como nuevas alternativas.

Ya en el siglo XXI, con el creciente desarrollo tecnológico que se ha sucedido, se cuenta con enormes volúmenes de información existentes en medios muy complejos, con capacidades ascendentes de almacenamiento y en soportes cada vez más reducidos. La proliferación de redes de transmisión de datos, de bases de datos con acceso en línea, ubicadas en cualquier lugar, localizables mediante Internet, permiten el hallazgo de otras redes y centros de información de diferentes tipos, en cualquier momento, desde disímiles lugares.

Con el propósito fundamental de aprovechar al máximo todas las ventajas brindadas por las tecnologías, los centros docentes de estos tiempos deben contar con herramientas que permitan almacenar, centralizar, organizar y controlar la información creada durante el proceso docente-educativo que realiza, con el principal objetivo de garantizar la disponibilidad y facilitar el manejo de los datos ya sea de sus estudiantes, profesores, trabajadores o actividades que estos efectúen.

Actualmente en la Facultad 8 de la Universidad de las Ciencias Informáticas (UCI) se desarrollan varios proyectos productivos, los cuales se controlan de forma manual y en algunos casos con el apoyo de las herramientas ofimáticas, ocasionando con el paso del tiempo pérdidas por deterioro, tras la constante manipulación de la información archivada en papel y en el caso de los documentos digitalizados, por la aparición de virus o fallas en el Sistema Operativo; además de demoras al realizar búsquedas, provocado por el almacenamiento disipado de los datos y la falta de automatización de los procesos de obtener y brindar información.

Por tal situación se plantea como **problema** a resolver que: no existe un sistema que gestione la información de los proyectos productivos que se desarrollan en la Facultad 8 de la UCI, ocasionando

retrasos en las búsquedas y obtención de reportes especializados, así como pérdidas de datos. Se define como **objeto de estudio**: los procesos de gestión de información que tienen lugar en el área de producción de la Facultad 8 de la UCI. Donde el **campo de acción** se encuentra delimitado por: los procesos de gestión de información de los proyectos productivos que pertenecen a dicha área.

Como **objetivo general** se plantea: realizar el análisis, diseño e implementación de un sistema que gestione la información de los proyectos productivos que se desarrollan en la Facultad 8 de la UCI. Del cual se derivan los **objetivos específicos** que se listan a continuación:

1. Investigar los procesos de gestión de información de los proyectos productivos que se llevan a cabo en la Facultad 8.
2. Realizar el análisis, diseño, implementación y prueba del sistema.
3. Elaborar la documentación del sistema.

El trabajo defiende la siguiente **idea**: Si se realiza el análisis, diseño e implementación de un sistema que gestione la información de los proyectos productivos que se desarrollan en la Facultad 8 de la UCI, entonces se contribuirá a realizar con mayor eficiencia este proceso.

Para dar cumplimiento a los objetivos trazados se requiere desarrollar **tareas** tales como:

1. Realizar entrevistas al Vicedecano de Producción de la Facultad 8 para conocer todos los procesos que tienen lugar en el área de producción y cómo se realizan.
2. Investigar sobre la existencia de otros sistemas similares.
3. Conocer las tendencias y tecnologías actuales.
4. Realizar la selección de las herramientas más apropiadas para el desarrollo del sistema, así como del lenguaje y metodología a utilizar.
5. Realizar el análisis, diseño e implementación del sistema.
6. Realizarle pruebas al sistema.
7. Elaborar la documentación en el formato requerido con las características establecidas.

Los **aportes prácticos** que se esperan obtener son:

1. Facilitar un servicio que permitan gestionar con mayor eficiencia la información referente a los proyectos productivos en la Facultad 8.

2. Brindar mayor rapidez en la búsqueda y obtención de información, a partir de la automatización de varios de los procesos que se llevan a cabo en el área productiva de la facultad en cuestión.
3. Lograr mayor aprovechamiento de las tecnologías que posee la Universidad de las Ciencias Informáticas (UCI).
4. Ahorrar tiempo y facilitar el trabajo del Vicedecano de Producción.
5. Posibilitar a los diferentes directivos obtener importantes reportes del estado en que se encuentran los proyectos productivos en la facultad.
6. Hacer más eficiente el control de la información en el proceso productivo de la facultad.
7. Proporcionar la información relacionada al tema de la producción en la Facultad 8 de manera centralizada para un fácil acceso.
8. Contar con el código fuente, permitiendo así actualizaciones de acuerdo a nuevas necesidades que puedan surgir.
9. Contar con la documentación del sistema desarrollado en el formato requerido y con las características establecidas.

Fueron utilizados como **métodos científicos de investigación**, los métodos teóricos y empíricos. Dentro de los teóricos se empleó el análisis histórico lógico durante la primera parte del presente Trabajo de Diploma, para realizar un estudio del estado del arte, con el objetivo de conocer cómo han madurado los conceptos relacionados a los sistemas de gestión de información; herramientas, metodologías y lenguajes mas utilizados para su desarrollo.

Dentro de los métodos empíricos, la entrevista individual fue la seleccionada, con el objetivo de recoger los datos necesarios para la comprensión de los procesos que se llevan a cabo en el área de producción de la Facultad 8.

El trabajo se encuentra estructurado para su mejor comprensión en cinco capítulos:

Capítulo 1. Fundamentación Teórica: En este capítulo se exponen los elementos teóricos que soportan el trabajo realizado. Estos son: el estudio de sistemas existentes destinados a la gestión de información, así como algunas de las características más relevantes de herramientas, lenguajes y metodologías, dentro de las que se encuentran las empleadas durante el desarrollo de la propuesta de solución, acompañadas de una breve justificación de su selección.

Capítulo 2. Características del Sistema: Se brinda una breve descripción de los procesos del negocio en cuestión, así como sus actores, trabajadores y casos de usos. Se presentan además el diagrama de casos de usos del negocio y el modelo de objetos y se definen tanto los requerimientos funcionales como los no funcionales, a la vez que se presentan los actores del sistema y el diagrama de casos de uso del mismo.

Capítulo 3. Análisis y Diseño del Sistema: Durante este capítulo se exponen los diagramas de clases del análisis y del diseño. Se muestran también el diseño de la base de datos y la descripción de cada una sus tablas.

Capítulo 4. Implementación y Pruebas: Aquí se muestran como está implementado el sistema, a través de los diagramas de componentes y de despliegue, además se exponen y detallan las diferentes pruebas que se le realizan al mismo.

Capítulo 5. Estudio de Factibilidad. Durante este capítulo se realiza el estudio de factibilidad del proyecto a desarrollar.

Capítulo I: Fundamentación Teórica.

1.1 Introducción.

En el presente capítulo, se presentan los principales conceptos y sistemas informatizados vinculados al campo de acción. A la vez que se brinda una breve panorámica de las tendencias y tecnológicas actuales que permitieron la selección de las herramientas, lenguajes de programación y metodologías, que fueron utilizados durante el desarrollo de la propuesta de solución para el problema planteado.

1.2 Gestión de información.

La información es un elemento fundamental para el desarrollo. Con el suceder de los años, la gestión de la información ocupa, cada vez más, un espacio mayor en la economía de los países a escala mundial y no es más que el proceso de analizar y utilizar la información que se ha recabado y registrado para permitir a los administradores (de todos los niveles) tomar decisiones documentadas. Se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de una organización.

La gestión de la información implica:

- ✓ Determinar la información que se precisa.
- ✓ Recoger y analizar la información.
- ✓ Registrarla y recuperarla cuando sea necesario.
- ✓ Utilizarla.
- ✓ Divulgarla.

Para que la información tenga un uso adecuado tiene que compartirse con los demás interesados o usuarios. En este contexto, debe entenderse que las tecnologías de información y las telecomunicaciones no son más que un medio para facilitar esta gestión.

1.3 Existencia de sistemas automatizados referentes al campo de acción.

1.3.1 Sistema de Gestión de Información de la Facultad 8 (SGIF).

Durante el curso 2006-2007 surgió una propuesta titulada: “Sistema de Gestión de Información de la Facultad 8” (SGIF) para dar solución a los problemas existentes en la Facultad 8 de la UCI, resumidos al comienzo de este Trabajo de Diploma.

Como su nombre lo indica, SGIF es un sistema para la gestión de información de la Facultad 8 de la Universidad de las Ciencias Informáticas (UCI), compuesto por varios módulos, cada uno con responsabilidades específicas. Un módulo para la gestión de la Residencia Estudiantil de dicha facultad, otro para el control de las Investigaciones Científicas que se desarrollen a cualquier nivel, con sus estudiantes y profesores participantes. Así como el encargado del control de la Producción. Conjuntamente a estos, se encuentran los módulos referentes a los términos Docencia, Cursos Optativos, Planificación Docente, Sindicato y Unión de Jóvenes Comunistas (UJC).

El módulo Producción, destinado a gestionar la información arrojada por los proyectos que se desarrollen en dicha facultad, no pudo ponerse en funcionamiento, debido a que el proceso productivo presenta cambios relevantes, que no se recogen en la aplicación existente o que cambiaron su enfoque completamente. Este es el caso de la introducción de nuevos conceptos como “polos productivos”, dentro de los que comienzan a agruparse los proyectos de acuerdo a la temática que aborden y que vienen a reemplazar la anterior estructura existente en el área de producción de la facultad en cuestión. Además, en la aplicación anterior no se realiza de forma eficiente el proceso de obtención de los datos relacionados a los estudiantes y profesores, necesario para conformar un nuevo proyecto. También faltan funcionalidades imprescindibles como terminar proyecto dando la posibilidad de adicionar un nuevo producto. Así como brindar una mayor cantidad reportes que aporten información más útil y detallada a los diferentes usuarios.

1.4 Tendencias y tecnologías actuales.

1.4.1 Los Servicios Web.

Los Servicios Web (en inglés Web service) son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí mediante una colección de protocolos y estándares, con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar. [1]

En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP. Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc. SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto por un envelope (sobre), cuya estructura está formada por los siguientes elementos: header (cabecera) y body (cuerpo). [1]

Por otro lado, WSDL (Lenguaje de Descripción de Servicios Web), permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes. [1]

1.4.2 Servidores Web.

Un Servidor Web, es un tipo de software que funciona en un ordenador y maneja la entrega de los componentes de las páginas como respuesta a peticiones de los navegadores de los clientes.

Básicamente, sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante el protocolo HTTP. [2] Los archivos para cada sitio de Internet se almacenan y se ejecutan en el servidor.

1.4.2.1 Internet Information Server (IIS).

Internet Information Server, desarrollado por Microsoft, constituye una serie de servicios para los ordenadores que funcionan con sistema operativo Windows que convierte a un ordenador en un servidor de Internet o Intranet, esto no es mas, que en las computadoras que tienen este servicio instalado se pueden publicar páginas Web tanto local como remotamente (Servidor Web). Es rápido y recomendable para la plataforma Windows 2000, dado por la integración que presenta con su Servicio de Directorios que permite el desarrollo de aplicaciones basadas en la Web fiables y escalables.

IIS 5.0 ofrece una administración muy sencilla que se realizará mediante el Administrador de servicios de Internet. La versión 5.0 de IIS permite que el desarrollo de aplicaciones Web sea mucho más robusto y la creación de sitios Web sea más configurable y completa. Ofrece un entorno escalable basado en los componentes cliente/servidor que se pueden integrar dentro de las aplicaciones Web. [3] IIS 5.0 también es capaz de impedir que aquellos usuarios con direcciones Internet Protocol (IP)

conocidas obtengan acceso no autorizado al servidor, permitiendo especificar la información apropiada en una lista de restricciones.

Una de las formas que tiene IIS de asegurar los datos es mediante Secure Sockets Layer (SSL). Lo que proporciona un método para transferir datos entre el cliente y el servidor de forma segura, permitiendo también que el servidor pueda comprobar al cliente antes de que inicie una sesión de usuario. Otra característica nueva es la autenticación implícita que permite a los administradores autenticar a los usuarios de forma segura a través de servidores de seguridad y proxy.

1.4.2.2 Servidor Web Apache.

Apache, es el servidor Web hecho por excelencia. Presenta características que hacen que cada vez millones de servidores reiteren su confianza en este programa, destacándose: su configurabilidad, robustez y estabilidad. Provee un alto grado de calidad y fortaleza para las implementaciones que utilizan el protocolo HTTP (de páginas Web).

Comenzó a desarrollarse en febrero de 1995, por Rob McCool. Es una tecnología gratuita, de código fuente abierto, que posee una licencia descendiente de la licencias BSD (no es GPL), la cual permite hacer lo que se quiera con el código fuente, siempre que se les reconozca su trabajo. Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal (Ejemplo: Linux, Novell, Unix y Windows).

El servidor Web Apache:

- ✓ Tiene interfaz de autenticación con todos los sistemas.
- ✓ Facilita la integración como "plugins" de lenguajes de programación de páginas Web dinámicas.
- ✓ Tiene integración en estándar del protocolo de seguridad SSL.
- ✓ Provee interfaz a todas las bases de datos.
- ✓ Tiene una alta configurabilidad en la creación y gestión de logs, ya que permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.
- ✓ Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto.

- ✓ Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades de este servidor, ya que se pueden escribir módulos para realizar determinadas funciones, lo que implica que haya gran cantidad de ellos disponibles para su utilización.

Los módulos de Apache pueden clasificarse en tres categorías:

- ✓ **Módulos Base:** Módulo con las funciones básicas del Apache.
- ✓ **Módulos Multiproceso** (para manejar las peticiones)**:** Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones. Se han diseñado varios módulos multiprocesos para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.
- ✓ **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor. Simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.

1.4.3 Lenguajes de programación para el desarrollo de Aplicaciones Web.

Un lenguaje de programación es un medio necesario para que las máquinas, en particular las computadoras, puedan interpretar las instrucciones que se les dan y para que se pueda además controlar su comportamiento. Consiste en un grupo de reglas sintácticas que definen su estructura y en un grupo de reglas semánticas que definen el significado de sus elementos. Estas reglas son utilizadas por el programador a través de las cuales crea un programa o subprograma. Por otra parte, las instrucciones que forman dicho programa son conocidas como código fuente.

1.4.3.1 Lenguajes del lado servidor.

Los lenguajes del lado del servidor, son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Son independientes del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo. Por otra parte, los scripts son almacenados en el servidor, quien los ejecuta y traduce a HTML, por lo que permanecen ocultos para el cliente. Este hecho puede resultar a todas luces una forma legítima de proteger el trabajo intelectual realizado. [4]

1.4.3.1.1 Python.

Python es un lenguaje de scripting. Es interpretado, por lo que no es necesario realizar la compilación del código fuente para ejecutarlo. Al mismo tiempo es gratuito (incluso para propósitos empresariales),

independiente de plataforma y orientado a objetos. Se encuentra preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso páginas Web.

En los últimos años este lenguaje se ha hecho muy popular debido a la cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero. Otras de las características que han influido en el auge de Python son su sencillez y velocidad para crear los programas.

1.4.3.1.2 Hypertext Preprocessor (PHP).

PHP es un lenguaje de programación interpretado de alto nivel, embebido en páginas HTML y ejecutado en el servidor, gratuito, independiente de plataforma, rápido ya que generalmente es utilizado como modulo de Apache, con una gran librería de funciones y mucha documentación.
Inició como una modificación a Perl, escrita por Rasmus Lerdorf a fines de 1994. En los siguientes tres años, se fue convirtiendo en lo que se conoce como PHP/FI 2.0. Esta forma de programar llegó a muchos usuarios, pero el lenguaje no tomó el peso actual hasta que Zeev Surasky y Andi Gutmans le incluyeron nuevas características en 1997, que dio por resultado el PHP 3.0. [5]

La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. Este lenguaje posee como una de sus características más importantes el soporte para: (a) gran cantidad de gestores de bases de datos, entre los que pueden mencionarse InterBase, mSQL, MySQL, Oracle, Informix, PostgreSQL, etc. (b) la mayoría de los servidores Web de hoy día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros.

PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI. Ofrece además la integración con las varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en PDF hasta analizar código XML y brinda una solución simple y universal para las paginaciones dinámicas del Web de fácil programación. Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día que el código comparables en otros lenguajes. Debido a su amplia distribución está perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto, goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

1.4.3.2 Lenguajes del lado cliente.

Los lenguajes de lado cliente son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pretratamiento. Son totalmente independientes del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts de lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas. [4]

1.4.3.2.1 Hypertext Markup Language (HTML).

HTML es un conjunto de símbolos o palabras que definen varios componentes de un documento Web; estos se pueden ver siempre dentro de las etiquetas "<", ">". Es un lenguaje simple de marcado utilizado para crear documentos de hipertexto para WWW. [6]

Fue creado por Tim Berners-Lee en 1991; en un principio con objetivos divulgativos, sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, si que se han ido incorporando modificaciones con el tiempo, estos son los estándares del HTML. En la actualidad, es el lenguaje que utilizan todos los navegadores para mostrar la información final.

Algunas ventajas muy importantes que posee el HTML es la de ser muy fácil de aprender, lo que permite que cualquier persona, pueda enfrentarse a la tarea de crear una Web; la facilidad con que se pueden actualizar los contenidos y que permite utilizar estilos en formato CSS en las páginas para una mayor facilidad en su modificación. Aunque no es un lenguaje de descripción de estructura de uso general, su amplia difusión y el número de documentos estructurados según sus normas es tan grande que su consideración como lenguaje de definición de estructura se hace obligatoria.

HTML, no solo permite establecer hiperenlaces entre diferentes documentos, sino que es un lenguaje de descripción de independiente de la plataforma en que se utilice. Es decir, un documento HTML contiene toda la información necesaria sobre su estructura y con el usuario, es luego el browser que se utilice el responsable de asegurar que el documento tenga un aspecto coherente, independiente del tipo de máquina desde donde se acceda al documento. [6]

1.4.3.2.2 JavaScript.

JavaScript es un lenguaje interpretado que permite incluir macros en páginas Web. Estas macros se ejecutan en el ordenador del visitante de nuestras páginas, y no en el servidor. JavaScript es también un lenguaje de scripts desarrollado por Netscape para incrementar las funcionalidades del lenguaje

HTML que comenzó a ofrecerlo como parte de su Navegador v.2.0. JavaScript sólo sirve para incluirse en documentos HTML y fuera de ellos no tiene ninguna vigencia, no crea aplicaciones autónomas. Se utiliza embebido en el código HTML, entre las tags <script> y </script>. Proporciona los medios para:

- ✓ Controlar las ventanas del navegador y el contenido que muestran.
- ✓ Evitar depender del servidor Web para cálculos sencillos.
- ✓ Capturar los eventos generados por el usuario y responder a ellos sin salir a Internet.
- ✓ Simular el comportamiento de las macros CGI cuando no es posible usarlas.
- ✓ Comprobar los datos que el usuario introduce en un formulario antes de enviarlos.
- ✓ Comunicarse con el usuario mediante diversos métodos.

La característica de JavaScript que más simplifica la programación es que, aunque el lenguaje soporta cuatro tipos de datos, no es necesario declarar el tipo de las variables, argumentos de funciones ni valores de retorno de las funciones. El tipo de las variables cambia implícitamente cuando es necesario, lo que dificulta el desarrollo de programas complejos, pero ayuda a programar con rapidez macros sencillas. En esto, JavaScript se separa totalmente de lenguajes como C, C++ o Java.

1.4.4 Integrated Development Environment ('IDE').

Un entorno de desarrollo integrado(IDE), es un entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

1.4.4.1 Zend Studio.

Zend Studio es un programa de la casa Zend (uno de los mayores impulsores de de la tecnología de servidor PHP), orientada a desarrollar aplicaciones Web con PHP. Proporciona un buen número de ayudas (creación y gestión de proyectos, depuración del código), además de servir de editor de texto para páginas PHP. Son muchos los desarrolladores que trabajan con él, por lo que se considera uno de los mejores IDE. El estar escrito en Java le ha permitido a Zend lanzar versiones del producto para Windows, Linux y MacOS, con relativa facilidad.

Zend Studio contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben recibir. Dispone de herramientas para gestionar los proyectos, muy útiles para mejorar la productividad en la programación. Consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Permite además hacer depuraciones simples de scripts.

1.4.4.2 EasyEclipse.

El proyecto EasyEclipse empaqueta el entorno de desarrollo Eclipse junto con una cuidada selección de "plugins" Open Source para obtener un IDE final excepcionalmente bueno para el desarrollo de aplicaciones en PHP, Python, Ruby y por supuesto Java, con todos los plugins ya instalados y configurados para que el desarrollador final sólo tenga que preocuparse del código de su aplicación y no de afinar su IDE.

EasyEclipse dispone de varias "distribuciones": para desarrollo Java de servidor, de aplicaciones Java de escritorio, para dispositivos móviles, para LAMP, para PHP, para Python y para Ruby, y por supuesto, EasyEclipse es open source y multiplataforma. Es fácil descargarlo e instalarlo y simple de mantener sin cuestiones de versiones ni dependencias.

Todo desarrollador, necesita un grupo de herramientas específicas y robustas tan pequeñas y simples como sea posible, esto es lo que EasyEclipse provee. Cada distribución es empaquetada para un entorno específico con las funcionalidades justas para ese ambiente. No más complejo que lo necesario y una forma fácil de descargar e instalar.

1.4.5 Sistemas Gestores de Base de Datos (SGBD).

Un Sistema Gestor de Base de Datos es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad; [7] además de coordina la organización, almacenamiento y extracción de los datos, e interpreta las consultas a la base de datos.

1.4.5.1 MySQL.

MySQL un sistema de gestión de base de datos desarrollado por MySQL AB como software libre en un esquema de licenciamiento dual. MySQL AB fundada por David Axmark, Allan Larsson y Michael Widenius en 19995 en Suecia pertenece a Sun Microsystems desde enero de 2008, quien posee el copyright de la mayor parte del código. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero las empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia específica que les permita este uso.

MySQL es muy utilizado en aplicaciones Web como MediaWiki o Drupal, en múltiples plataformas, incluyendo: AIX/GNU-Linux/Windows 95/Windows 98/Windows NT/Windows 2000/Windows XP/ Windows Vista y otras versiones de Windows/Mac OS X/Solaris/SunOS-Apache-MySQL-PHP/Perl/Python) y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación Web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

1.4.5.2 Oracle.

Oracle es básicamente una herramienta que se basa en la tecnología cliente/servidor para la gestión de bases de datos. Surgió entre finales de los años 70 y principio de los años 80, desarrollado por la compañía “Oracle Corporación” y conocida entonces como “Relational Software”.

En la actualidad es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas Web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos.

1.4.5.3 PostgreSQL.

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) de código abierto más avanzada del mundo. Comenzó como un proyecto denominado Ingres en la Universidad Berkeley de California. Ingres fue más tarde desarrollado comercialmente por la Relational Technologies/Ingres Corporation. En 1986 otro equipo dirigido por Michael Stonebraker de Berkeley continuó el desarrollo del código de Ingres para crear un sistema de bases de datos objeto-relacionales llamado Postgres. En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, Postgres fue renombrado a PostgreSQL, tras un breve periplo como Postgres95. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. [8]

PostgreSQL posee muchas características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x. [9]

1. **DBMS Objeto-Relacional**: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multa-versión, soporte multa-usuario, optimización de consultas, herencia, y arrays. [9]
2. **Altamente Extensible**: PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario. [9]
3. **Soporte_SQL_Comprensivo:** PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92. [9]

4. **Integridad Referencial:** PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. [9]
5. **API Flexible:** La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike. [9]
6. **Lenguajes Procedurales:** PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido. [9]
7. **MVCC:** MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles. [9]
8. **Cliente/Servidor:** PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL. [9]
9. **Write Ahead Logging (WAL):** La característica de PostgreSQL conocida como "Write Ahead Logging" incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. [9]

1.4.6 Lenguajes de Modelado Visual.

Un lenguaje de modelado visual es usado para especificar, visualizar, construir y documentar artefactos de un sistema de software. No es más que un medio que permite la construcción de un modelo, donde este último constituye una simplificación de la realidad. El objetivo del modelado de un sistema es capturar las partes esenciales del sistema. Para facilitar este modelado, se realiza una abstracción y se plasma en una notación gráfica. Esto se conoce como modelado visual.

El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar. Cuando se intenta construir un sistema complejo, es necesario abstraer la complejidad en modelos que el ser humano pueda entender. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando alguno de los lenguajes existentes para este modelado visual se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

1.4.6.1 Unified Modeling Language (UML).

UML, son las siglas de Lenguaje Unificado de Construcción de Modelos, es la notación con que se construyen sistemas por medio de conceptos orientado a objetos. Fue creado por los miembros de Rational: Grady Booch, James Rumbaugh e Ivar Jacobson. Es el lenguaje de mayor difusión que la OMG ha convertido desde 1997 en el estándar para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación. Es un lenguaje para describir principalmente sistemas orientados a objetos independientes de cualquier lenguaje de programación específico, es simple de aprender, bastante flexible y consistente desde el planeamiento hasta el despliegue. Los beneficios de usar UML incluyen la trazabilidad mejorada, inteligibilidad entre los usuarios y un mantenimiento realmente simplificado.

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Proporciona valor conceptual y visual, facilidad para extender el lenguaje y para representar elementos particulares a determinados tipos de aplicaciones. Mediante este lenguaje se hace posible además de la visualización, la especificación, construcción y documentación de sistemas que involucran gran cantidad de software con tecnología orientada a objetos.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. La finalidad de los diagramas UML es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

A través de un conjunto de símbolos y diagramas del UML los creadores de sistemas pueden generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas que estén involucradas en el proceso de desarrollo de los sistemas.

El modelo gráfico de UML tiene un vocabulario que se presenta a continuación:

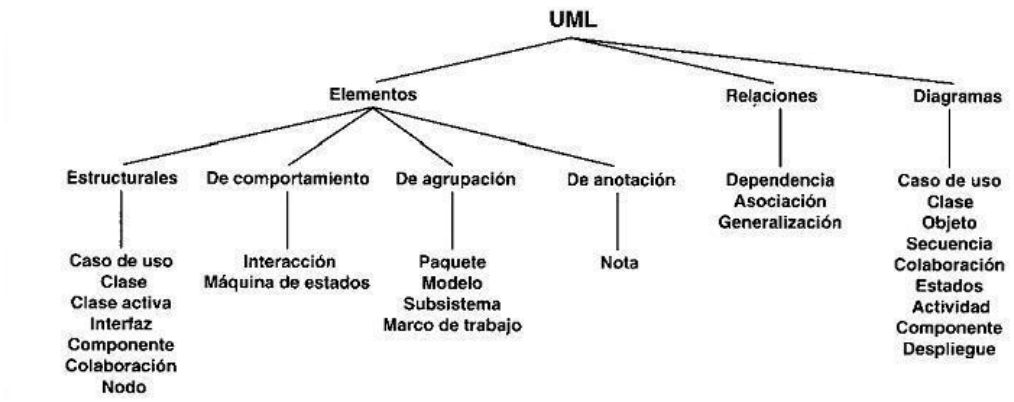

Figura 1: El vocabulario de UML.

1.4.7 Metodologías de Desarrollo de Software.

En la ingeniería de software de un proyecto, la metodología de desarrollo de software define quién debe hacer qué, cuándo y cómo debe hacerlo. Una metodología es un proceso que se encargada de elaborar estrategias de desarrollo de software. Abarca todo el ciclo de vida del producto, y es definido por Jacobson, Booch y Rumbaugh en su libro “El Proceso Unificado de Desarrollo de Software” como: “el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software”, adoptando la misma se obtiene un producto software más predecible.

Dentro de las metodologías de desarrollo de software podemos encontrar las llamadas metodologías tradicionales y las metodologías ágiles.

1.4.7.1 Metodologías Ágiles.

Durante los últimos años han surgido las llamadas metodologías ágiles. Estas aportan nuevas técnicas y métodos de trabajo para el desarrollo de cada etapa de un software. En general esta metodología hace un balance entre los procesos y el esfuerzo, ya que tratan de centrarse en las cuestiones necesarias sin perderse en las burocráticas. Estas metodologías tiene como prioridad satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor, capturan los cambios para que el cliente tenga una ventaja competitiva, construir el proyecto en torno a individuos

motivados y lograr que el personal del negocio y los desarrolladores trabajen juntos a lo largo del proyecto, entre otras.

Dentro de las más mencionadas de estas metodologías, se encuentran Extreme Programing (XP) y Microsoft Solution Framework (MSF).

1.4.7.1.1 Extreme Programing (XP).

XP es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo y corto equipo, donde cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. [10]

En la figura 2 se puede obserbar la metodología XP.

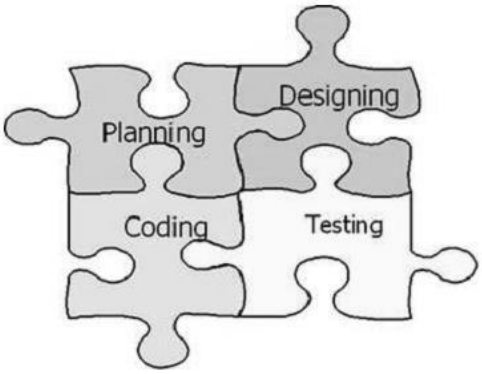

Figura 2: Metodología Extreme Programing.

La metodología Extreme Programing (XP) se basa en las Pruebas Unitarias, la Refabricación y la Programación en pares como características fundamentales. Además XP convierte a los clientes o usuarios en miembros del equipo. Propone que los clientes tienen el derecho de decidir que se implementa, de saber el estado real del proyecto, su progreso, de añadir, cambiar o quitar requerimientos en cualquier momento, como también obtener lo máximo de cada semana de trabajo y de un sistema funcionando cada 3 o 4 meses. XP plantea derechos que posee el desarrollador como son: el decidir como se implementan los procesos, crear el sistema con la mejor calidad posible, pedir al cliente en cualquier momento aclaraciones de los requerimiento, estimar el esfuerzo para implementar el sistema y cambiar los requerimientos en base a nuevos descubrimientos.

Lo fundamental de este tipo de metodología es la comunicación, entre los usuarios y los desarrolladores, así como la simplicidad, al desarrollar y codificar los módulos del sistema, además de la retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

1.4.7.1.2 Microsoft Solution Framework (MSF).

MSF es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. [10]
En la figura 3 se puede obserbar la metodología MSF.

Figura 3: Metodología MSF.

Esta metodología presenta características como: el ser una tecnología agnóstica, adaptable, escalable y flexible. A continuación se explican brevemente.

1. **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar. [10]
2. **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más. [10]
3. **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente. [10]
4. **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. [10]

MSF está compuesto de varios modelos los cuales son los encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto, ellos son: el Modelo de Arquitectura del Proyecto, el Modelo de Equipo, el Modelo de Proceso, el Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el Modelo de Aplicación. [10]

1.4.7.2 Metodologías Tradicionales.

Las Metodologías Tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, haciendo énfasis en la previa y total planificación del proyecto a desarrollar. Una vez detallado todo el trabajo a realizar es que comienza el ciclo de desarrollo del software. Todo lo anteriormente planteado se realiza con el objetivo de conseguir un software más eficiente y predecible.

RUP es la metodología más conocida dentro de esta clasificación. Mundialmente es muy utilizada, al igual que en la Universidad (UCI).

1.4.7.2.1 Rational Unified Process (RUP).

La metodología RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Desarrollo 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

Como proceso en sí, define como sus elementos más importantes: (a) los trabajadores, que indican el "quién". Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. (b) Actividades, que definen el "cómo", una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos. (c) Los artefactos, que no son más que el "qué", productos tangibles del proyecto que son producidos, modificados y usados por las actividades (modelos, elementos dentro del modelo, código fuente y ejecutables). Por último el Flujo de actividades, que identifica el "Cuándo", secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP, tiene como característica fundamental en su ciclo de vida ser:

1. Dirigido por casos de uso.
2. Centrado en la arquitectura.
3. Iterativo e Incremental.

Divide en 4 fases el desarrollo del software:

- ✓ **Inicio**: El Objetivo en esta etapa es determinar la visión del proyecto.
- ✓ **Elaboración**: En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ **Construcción**: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ **Transición**: El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

El ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

Disciplina de Desarrollo:

- ✓ **Ingeniería de Negocios:** Entendiendo las necesidades del negocio.
- ✓ **Requerimientos:** Trasladando las necesidades del negocio a un sistema automatizado.
- ✓ **Análisis y Diseño:** Trasladando los requerimientos dentro de la arquitectura de software.
- ✓ **Implementación:** Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- ✓ **Pruebas:** Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte:

- ✓ **Configuración y administración del cambio:** Guardando todas las versiones del proyecto.
- ✓ **Administrando el proyecto:** Administrando horarios y recursos.
- ✓ **Ambiente:** Administrando el ambiente de desarrollo.
- ✓ **Distribución:** Hacer todo lo necesario para la salida del proyecto.

En la figura 4 se pueden obserbar las distintas fases e iteraciones de la metodología RUP.

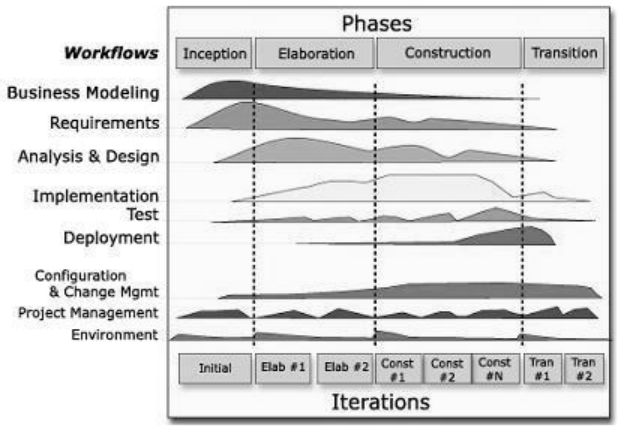

Figura 4: Fases e iteraciones de la metodología RUP.

Los elementos del RUP son:

- ✓ **Actividades:** Son los procesos que se llegan a determinar en cada iteración.
- ✓ **Trabajadores:** Vienen hacer las personas o entes involucrados en cada proceso.
- ✓ **Artefactos:** Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

RUP es una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software dado por la particularidad de que en cada ciclo de iteración, se hace exigente el uso de artefactos.

1.4.8 Herramientas CASE de modelado con UML.

Las herramientas CASE como su nombre lo indica "Computer Aided Software", en español: "Ingeniería de Software Asistida por Computadora", constituyen 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 nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Abarcan todos los pasos del proceso de software, y también aquellas actividades generales que se aplican a lo largo de todo el proceso. Son un complemento de la caja de herramientas del ingeniero del software, además ayudan a asegurar que la calidad sea algo diseñado antes de llegar a construir el producto.

Todas las herramientas CASE prestan soporte a un lenguaje de modelado para acompañar la metodología y un alto porcentaje de ellas soportan UML, debido al alto grado de aceptación que ha tenido este lenguaje desde su lanzamiento.

1.4.8.1 Enterprise Architect (EA).

Enterprise Architect es una herramienta comprensible de diseño y análisis UML, multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Cubre el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Ofrece salida de documentación flexible y de alta calidad. Soporta generación e ingeniería inversa de código fuente para varios lenguajes, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP.

- ✓ **Velocidad, estabilidad y rendimiento:** El Lenguaje Unificado de Modelado provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es

posible mantener la trazabilidad de manera consistente. Enterprise Architect soporta este proceso en un ambiente fácil de usar, rápido y flexible. [11]

- ✓ **Trazabilidad de extremo a extremo:** Enterprise Architect provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue. Combinados con la ubicación de recursos y tareas incorporados. [11]
- ✓ **Construido sobre las bases de UML 2.1**: Las bases de Enterprise Architect están construidas sobre la especificación de UML 2.0, Usa Perfiles UML para extender el dominio de modelado, mientras que la Validación del Modelo asegura integridad. [11]

1.4.8.2 Visual Paradigm.

Visual Paradigm es una herramienta CASE que utiliza "UML": como lenguaje de modelado. Es una Suite de herramientas CASE, dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software, lo cual garantiza la calidad del producto final. Es fácil de usar, es amigable, soporta ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI, generador de informes, editor de figuras, etc. Posibilita crear un gran conjunto de artefactos de las distintas fases del desarrollo del software. Presenta sincronización entre diagramas de entidad-relación y diagramas de clases, interoperabilidad con otras aplicaciones, facilidad para redactar especificaciones de casos de uso del sistema, generación de documentos y de código ORM; disponibilidad de múltiples versiones, para cada necesidad, disponibilidad de integrarse en los principales IDEs y en múltiples plataformas. Se integra con varias herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic.

1.4.8.3 Rational Rose Enterprise Edition.

Rational Rose Enterprise Edition en una herramienta CASE que permite el diseño detallado del software y la generación de código fuente de programas y bases de datos e ingeniería inversa (obtención del diseño a partir del código fuente), basado en modelos con soporte UML. Fue desarrollada por los creadores de UML Booch, Rumbaugh y Jacobson. Cubre todo el ciclo de vida de un proyecto desde la concepción y formalización del modelo, pasando por la construcción de los componentes, transición a los usuarios, hasta la certificación de las distintas fases y entregables.

El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de

Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción. [12]
Esta herramienta propone la utilización de cuatro tipos de modelos para realizar el diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas, creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. [12]

Rational Rose utiliza un proceso de desarrollo iterativo controlado (Controlled Iterative Process Development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que estos se hagan mínimos. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, esta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración. [12]

Rational Rose permite que hayan varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. [12]

1.4.9 La Arquitectura de Software (AS).

La Arquitectura es el esqueleto o base de una aplicación done es analizada desde varios puntos de vista. Representa la organización fundamental de un sistema constituida en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Constituye la nueva visión que los desarrolladores tienen de los sistemas de software en la última década. Desde los pequeños programas hasta los sistemas más grandes poseen una estructura y un comportamiento que los hace clasificables según su "arquitectura".

Este nuevo aspecto hace posible el estudio de sistemas ya implementados así como el desarrollo de nuevos, según la categoría del problema que resuelven y el tipo de "arquitectura". Los distintos niveles de abstracción de la funcionalidad de los sistemas, están asociados con diferentes aspectos y componentes de su "arquitectura". Esta asociación se manifiesta en forma clara en las distintas etapas

del proceso de desarrollo de software. Existen distintos tipos de arquitecturas de sistemas que se clasificación en patrones de diseño y que tienen aplicaciones a lo largo del proceso de desarrollo.

1.4.9.1 Arquitectura n-Capas.

Con el objetivo de solventar los problemas de escalabilidad, disponibilidad, seguridad e integración que pueden presentan las aplicaciones compactas, se ha generalizado la división de las aplicaciones en capas.

La capa que se agrega es la que surge de separar definitivamente las reglas de negocio de la de Acceso a Datos. Esta arquitectura trae consigo la ventaja de aislar definitivamente la lógica de negocios de todo lo que tenga que ver con el origen de datos, de modo que ante cualquier eventual cambio, solo se deberá tocar un módulo específico, así como al momento de plantear la escalabilidad del sistema no se afrontarán grandes modificaciones.

Dentro de este tipo de arquitectura una de las divisiones muy utilizada en el desarrollo de sistemas es la arquitectura en tres capas.

1.4.9.1.1 Arquitectura en tres capas.

La arquitectura en tres capas cuenta con una interfaz gráfica que facilita al usuario el uso del sistema (**Capa 1:** Capa de Presentación.), con una capa para centralizar la lógica de negocio (**Capa 2:** Lógica de Negocio.) y por último una capa que servirá para guardar los datos (**Capa 3:** Base de Datos.).
A continuación se muestra la figura 5 que presenta la arquitectura en 3 capas.

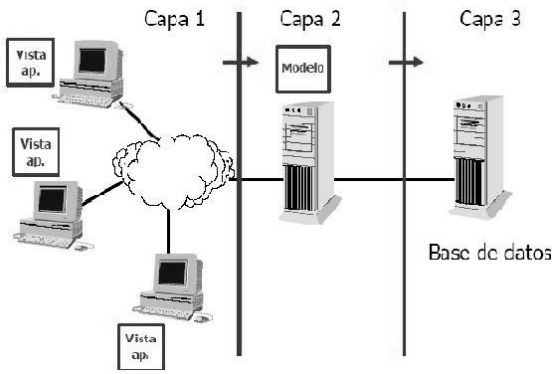

Figura 5: Arquitectura en 3 capas.

Esta arquitectura nos otorga algunas ventajas:

- ✓ Centralización de los aspectos de seguridad y transaccionalidad, que serían responsabilidad del modelo. [13]
- ✓ No replicación de lógica de negocio en los clientes: esto permite que las modificaciones y mejoras sean automáticamente aprovechadas por el conjunto de los usuarios, reduciendo los costes de mantenimiento. [13]
- ✓ Mayor sencillez de los clientes. [13]

1.4.9.2 Patrón de arquitectura Modelo Vista Controlador (MVC).

El MVC (Model/View/Controller) fue introducido como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. MVC es un patrón de arquitectura de software clásico que está formado por tres niveles:

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

En la figura 6 se puede observar el patrón MVC.

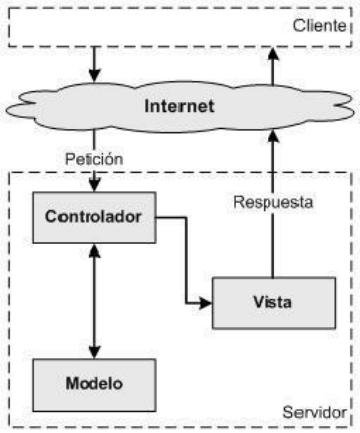

Figura 6: El Patrón MVC.

El patrón MVC separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre vista y controlador es menos clara.

Este modelo de arquitectura presenta varias ventajas:

- ✓ Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado.
- ✓ Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- ✓ La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

1.4.10 Framework.

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver tareas comunes. Además proporciona estructura al código fuente,

forzando al desarrollador a crear código más legible, más fácil de mantener y facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. [14]

1.4.10.1 Symfony.

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. [14] Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Agrupa a sus creadores dentro de la empresa francesa Sensio Labs, cuyo co-fundador es Fabien Potencier y líder del proyecto Symfony.

Este framework proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una aplicación. [14]

Symfony está desarrollado completamente en PHP5. Ha sido probado en numerosos proyectos reales y en sitios Web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de base de datos como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. [14] Como software libre, cuenta con una gran comunidad en varios idiomas, incluyendo español, y esta seriamente respaldado por sus creadores (Sensio Labs), quienes además de usarlo en sus aplicaciones, dan conferencias a lo largo de todo el mundo. Actualmente está en la versión estable 1.011 y en beta 1.1.

Symfony está basado en un patrón clásico del diseño Web, conocido como arquitectura MVC, toma lo mejor de esta arquitectura y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. En primer lugar, el controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno de cada uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática.

Las clases de la capa del modelo, también se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera el código necesario. Cuando Propel encuentra restricciones de claves foráneas (o externas) o cuando encuentra datos de tipo fecha, crea métodos especiales para acceder y modificar esos datos, por lo que la manipulación de datos se convierte en un juego de niños. La abstracción de la base de datos es completamente invisible al programador, ya que la realiza otro componente específico llamado Creole. Así, si se cambia el sistema gestor de bases de

datos en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración. Por último, la lógica de la vista se puede transformar en un archivo de configuración sencillo, sin necesidad de programarla.

En la figura 7 se puede observar el flujo de trabajo de Symfony.

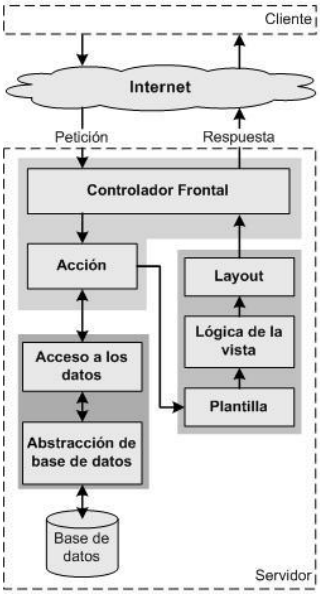

Figura 7: El flujo de trabajo de Symfony.

1.5 Conclusiones.

En este capítulo se realizó un estudio de las tendencias y tecnologías actuales para el desarrollo de aplicaciones Web. Dentro del gran número de herramientas, metodologías y lenguajes existentes de hoy día, se analizaron aquellas que gozan de mayor popularidad y se partió del particular concepto de "independencia tecnológica" que persigue Cuba en el desarrollo de aplicaciones e informática en general. Por supuesto que la UCI no se queda detrás en este tema, por eso, la selección realizada se caracteriza por ser software libre y multiplataforma en su mayoría. Se tuvo en cuenta además, el conocimiento previo sobre estas tecnologías, con el objetivo de minimizar el tiempo de desarrollo del Trabajo de Diploma a realizar.

Se decidió seleccionar, para el desarrollo y construcción del sistema, como lenguajes de programación del lado cliente, HTML, lenguaje de definición de estructura más ampliamente difundido y para incrementar sus funcionalidades, JavaScript, ambos muy conocidos y utilizados en la Universidad. Del lado servidor, PHP fue el escogido, partiendo de que se tiene un poco más de experiencia en su uso comparado con el resto de los mencionados en el capítulo; es "open source" y puede ser utilizado en

cualquiera de los principales sistemas operativos. Posee además una de las comunidades más grandes de desarrolladores en el mundo, incluyendo la UCI; un código simple que facilita su trabajo, una gran librería de funciones y mucha documentación. Una de las características que se tuvieron en cuenta es la facilidad de PHP para funcionar con el servidor Web Apache, ya que se integra como un módulo de este y justifica la selección del mismo, ambos poseen gran compatibilidad y por su parte Apache es también gratuito, multiplataforma y presenta gran modularidad que lo hace personalizable a la vez que configurable y flexible. Como gestor de base de datos PostgreSQL se adaptó perfectamente a las necesidades planteadas.

Para efectuar la implementación de la aplicación, nos apoyamos en el IDE "open source" EasyEclipse, que presenta entre sus distribuciones para diferentes lenguajes una para PHP. La metodología de desarrollo de software escogida fue RUP, debido a que es la más usada en la Universidad y sobre la que más dominio se posee, abarca de manera organizada todo el ciclo de vida del software y permite obtener un producto más eficiente y predecible, basado en la previa y total planificación del proyecto, lo que posibilita además contar con una detallada y amplia documentació del proyecto. La metodología RUP utiliza el UML para preparar todos los esquemas de un sistema software. Se utiliza este lenguaje para el modelado del sistema, ya que constituye un estándar de facto a nivel internacional para especificar, visualizar, construir y documentar artefactos de un software, lo que brinda un entendimiento común por parte de los diferentes desarrolladores, clientes y todos los involucrados en el proceso de desarrollo. Además es el lenguaje para describir principalmente sistemas OO. Rational Rose es una herramienta CASE para el modelado con UML y que integra todos los elementos que propone la metodología RUP para cubrir todo el ciclo de vida de un proyecto. Es la herramienta que se emplea en la docencia de la UCI y con la que se aprende a trabajar en la asignatura de Ingeniería de Software (ISW), fue la elegida.

Se aprovecharon los beneficios brindados por Symfony, framework que se adaptó perfectamente a las necesidades reales presentadas para la propuesta de solución, software libre, que cuenta con una gran comunidad en varios idiomas, incluyendo español y que permitió agilizar el trabajo. Basado en el patrón de arquitectura MVC, de aquí que este haya sido el implementado en el sistema desarrollado, conjuntamente con las distintas ventajas que nos brinda su uso: el poder separar los componentes de un sistema permitiendo implementarlos por separado y la conexión dinámica entre el Modelo y sus Vistas.

Capítulo 2: Características del Sistema.

2.1 Introducción.

Para poder comenzar a desarrollar el sistema, es de suma importancia detenernos a analizar las características y organización de los procesos que se desarrollan en el área que se pretende automatizar, con el objetivo de lograr una mayor comprensión del problema que se desea resolver. Con este propósito se realiza la modelación del negocio y también con la intención de establecer el común entendimiento entre clientes y desarrolladores.

En el presente capítulo se describe el objeto de estudio, se realiza la modelación del negocio, se analizan tanto los requerimientos funcionales como no funcionales y se muestra entre otros l diagramas el de casos de uso del sistema.

2.2 Descripción general de los procesos involucrados en el negocio.

Los proyectos productivos son introducidos en la Facultad 8 de la UCI, a raíz de propuestas de clientes que se dirigen al Vicedecano de Producción para plantearle la necesidad de comenzar un proyecto. En algunos casos, estos son asignados directamente por la Dirección de Producción UCI, quien tiene también la responsabilidad de darle a la facultad, temáticas para la creación de sus polos productivos.

El Vicedecano de Producción, es el encargado de conformar los proyectos aprobados tras un estudio de sus características. Agrupándolos según el tema que aborden en polos productivos y les asigna estudiantes, profesores, un líder encargado de guiar y responder por el proyecto, uno o varios laboratorios y las PCs correspondiente atendiendo a la magnitud del mismo. Además de registra toda esta información para la elaboración de posteriores reportes, atendiendo a solicitudes de directivos dentro y fuera de la facultad, ya sean decanos, vicedecanos, jefes de departamentos y de proyectos etc.

2.3 Información que se maneja en el negocio.

- Listado de estudiantes de la facultad: Registra todos los estudiantes de la facultad y es un documento de suma importancia a la hora de conformar un proyecto y de modificarlo.
- Listado de profesores de la facultad: Registra todos los profesores de la facultad y es también de suma importancia para conformar y modificar un proyecto.

- Listado de laboratorios de la facultad: Registra todos los laboratorios de la facultad y al igual que los anteriores se necesita en el momento de conformar y modificar un proyecto.
- Registro de polos productivos: Guarda todos los datos relacionado a cada uno de los polos productivos existentes en la facultad.
- Registro de proyectos productivos de la facultad: Guarda todos los datos relacionado a cada uno de los proyectos productivos que se desarrollan en la facultad.

2.4 Objeto de automatización.

Se desean automatizar los procesos de registro, modificación y obtención de toda la información relacionada con los proyectos y polos productivos de la Facultad 8 de la UCI.

2.5 Propuesta del sistema.

Se propone implementar un sistema con el que se pretende facilitar el manejo y control de la información que se manipula en la Facultad 8 relacionada con sus proyectos productivos y que brinde además un gran número de reportes en el menor tiempo posible, proporcionando así grandes beneficios para aquellas personas que deseen información de manera rápida y precisa.

El sistema brinda las opciones de asignarle a un proyecto: estudiantes, laboratorios (con un número de PC), profesores, así como un líder y de ubicarlo en un determinado polo productivo dentro de los existentes en la facultad, atendiendo a la temática que aborde.

Por otra parte, se procura también que el sistema brinde la opción de modificar cualquiera de los datos anteriores, atendiendo a solicitudes de cambios que se puedan efectuar durante el ciclo de vida de un proyecto productivo.

2.6 Modelamiento del negocio.

Durante el flujo de trabajo modelamiento del negocio, se logra una mayor comprensión del problema a resolver, mediante el estudio de la estructura y dinámica de la organización; donde se identifican sus procesos, roles y responsabilidades mediante los modelos de casos de uso del negocio y de objetos y se garantiza el entendimiento común ente desarrolladores y usuarios finales del software que se pretende desarrollar.

2.6.1 Actores y trabajadores del negocio.

Lo que se modela como un actor del negocio es al rol que se juego al interactuar con el negocio para beneficiarse de sus resultados. Es un individuo o grupo de individuos, organización, máquina o sistema de información externo al negocio y que interactúa con él.

Por otra parte, un trabajador del negocio define el rol de un individuo o grupo de individuos, máquina o sistema automatizado que participan directamente en los procesos que se llevan a cabo en el negocio, ya que son los que realizan las actividades y son propietarios de elementos pero no obtienen ningún beneficio con los resultados del proceso.

A continuación se muestran las tablas 1 y 2, con los actores y trabajadores del negocio respectivamente y sus justificaciones correspondientes:

Actores del Negocio.	Justificación.
Dirección de Producción UCI.	Asigna los polos productivos a la facultad e informa de los cambios que se necesitan hacer sobre ellos. Además también puede plantear la necesidad de comenzar un proyecto productivo.
Cliente.	Plantea la necesidad de comenzar un proyecto productivo.
Líder de Proyecto.	Informa los cambios que son necesarios realizar en los proyectos productivos. Así como un proyecto ha llegado a su fin.
Directivo.	Solicita información referente a los polos y proyectos productivos de la facultad

Tabla 1: Descripción de los actores del negocio.

Trabajadores del Negocio.	Justificación.
Vicedecano de Producción.	Registra los proyectos y polos productivos. Conforma los reportes que son solicitados por parte de los directivos. Además es el encargado de controlar los cambios que se realizan tanto en proyectos como en polos de la facultad.
Secretaria de Producción.	Es la persona que asiste al Vicedecano de Producción. Es la encargada de elaborar los listados de los estudiantes, los profesores y laboratorios de la facultad que utiliza el Vicedecano de Producción para conformar los proyectos productivos.

Tabla 2: Descripción de los trabajadores del negocio.

2.6.2 Modelo de casos de uso del negocio.

El modelo de casos de uso del negocio, se describe mediante diagramas de casos de uso. Es una técnica que permite la descripción de los procesos de negocio de una organización en términos de casos de uso y actores del negocio, que se corresponden con los procesos del negocio y los clientes, respectivamente.

A continuación se muestra la figura 8 correspondiente al diagrama de casos de uso del negocio:

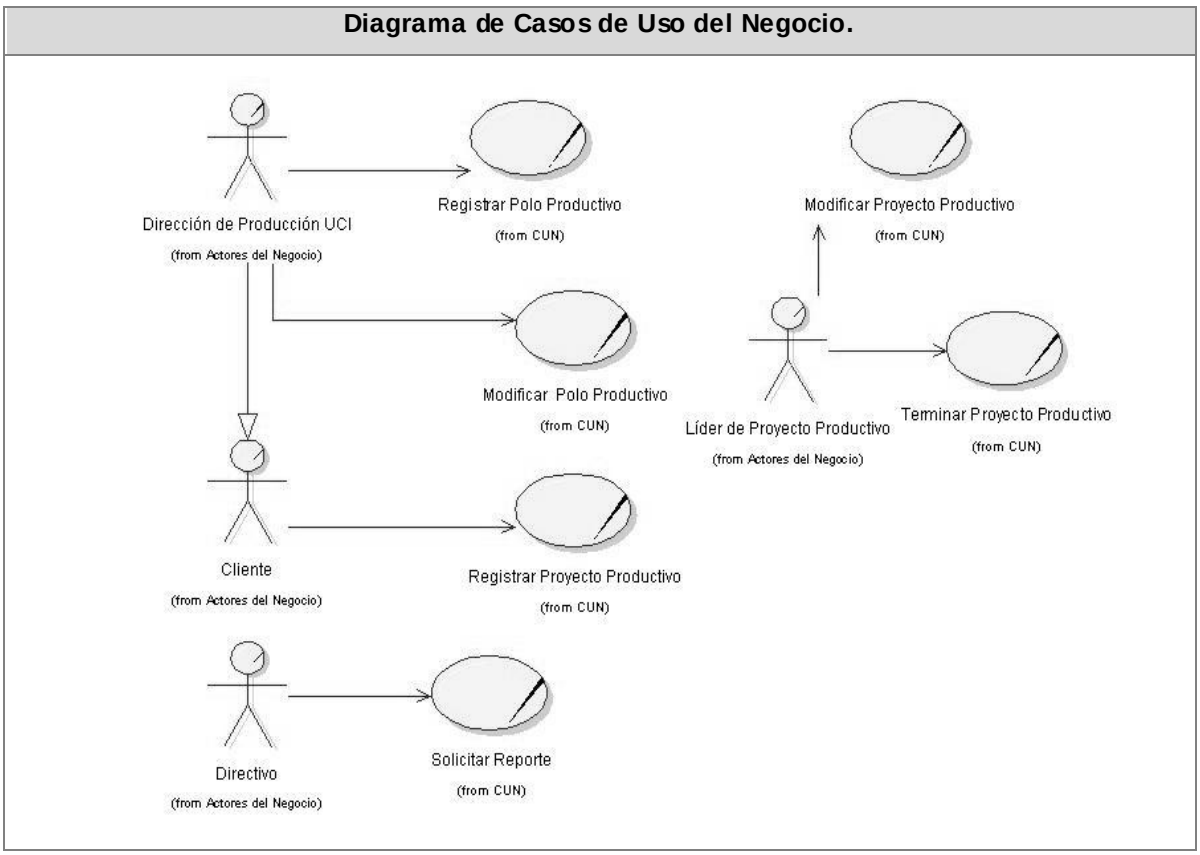

Figura 8: Diagrama de casos de uso del negocio.

2.6.3 Realización de los casos de uso de del negocio.

El detallar los casos de uso del negocio, tiene como objetivo fundamental describir su flujo de sucesos, desde cómo comienza hasta cómo termina, pasando por la forma de interactuar con los actores, brindando así una mayor comprensión de los procesos del negocio a clientes y desarrolladores.

Se describe de forma textual como se llevan a cabo las actividades dentro del negocio y quienes las realizan. Además, por medio de los diagramas de actividades se muestra el flujo de los procesos de manera gráfica. A través de las calles, se especifican las responsabilidades de los actores y trabajadores del negocio y a través del flujo de objetos cómo se utilizan las entidades del negocio.

A continuación se muestran las descripciones textuales de los casos de uso del negocio y sus diagramas de actividades correspondientes:

Caso de Uso:	**Registrar Proyecto Productivo.**
Actores:	Cliente.
Trabajadores:	Vicedecano de Producción y Secretaria de Producción.
Resumen:	El caso de uso inicia cuando un cliente se acerca a la facultad y plantea la necesidad de comenzar un proyecto productivo. El Vicedecano de Producción solicita los datos del proyecto y le asigna el laboratorio, los estudiantes, etc. que formarán parte del proyecto. Registra el proyecto con todos sus datos. Finaliza el caso de uso.
Flujo normal de los eventos.	
Acción del Actor.	**Respuesta del Negocio.**
1. El cliente se dirige a la facultad para proponer un proyecto productivo.	1.1 El Vicedecano de Producción solicita los datos del proyecto.
2. El cliente entrega los datos del proyecto que desea comenzar.	2.1 El Vicedecano de Producción analiza los datos del proyecto.
	2.2 Si aprueba los datos del nuevo proyecto y decide que puede emprenderlo, solicita a la secretaria los listados de los estudiantes, profesores y laboratorios de la facultad.
	2.3 La secretaria le entrega los listados al Vicedecano de Producción.
	2.4 El Vicedecano de Producción conforma el proyecto con los estudiantes, profesores y le asigna uno o varios laboratorios en dependencia de la magnitud del proyecto y registra el nuevo proyecto que ha quedado conformado.
Flujos Alternos.	
Acción del Actor.	**Respuesta del Negocio.**

2. a	2.2 Si el Vicedecano de Producción no aprueba los datos del nuevo proyecto y decide que no puede emprenderlo, le comunica al cliente que su proyecto no ha sido aprobado.
Mejoras:	Se mantendrá un registro automatizado y actualizado de toda la información de los proyectos productivos, permitiendo tener un mayor control sobre los mismos y aportando a la vez mayor facilidad en el momento de buscar alguna información requerida por algún directivo.

Tabla 3: Descripción textual del caso de uso del negocio "Registrar Proyecto Productivo".

Diagrama de actividades del caso de uso del negocio "Registrar Proyecto Productivo".

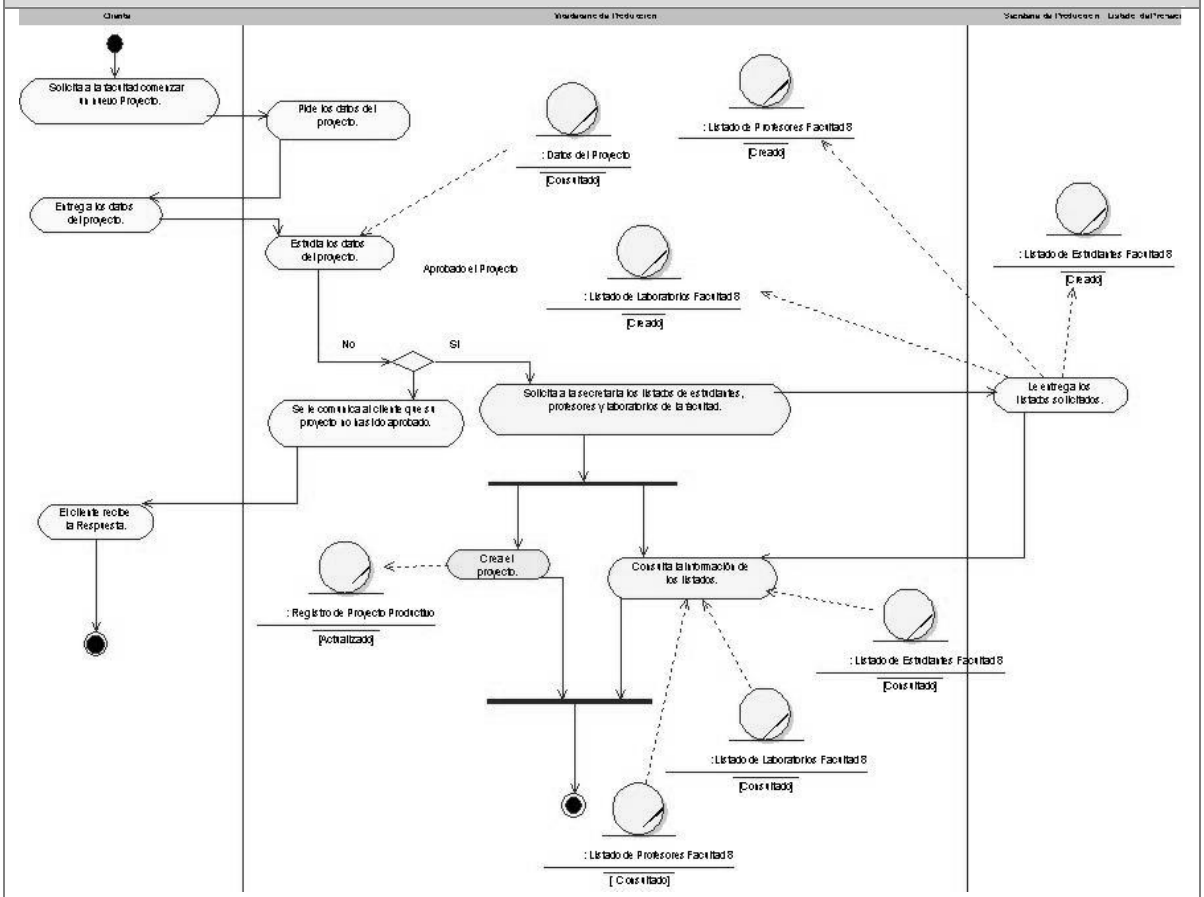

Figura 9: Diagrama de actividades del CUN: "Registrar Proyecto Productivo".

Caso de Uso:	**Modificar Proyecto Productivo.**
Actores:	Líder de Proyecto.
Trabajadores:	Vicedecano de Producción.
Resumen:	El caso de uso inicia cuando el líder de proyecto solicita realizar algún cambio en el proyecto, ya sea sustituir estudiantes, incrementar número de PC, etc. El Vicedecano de Producción busca los estudiantes disponibles, las PC, etc. y se las asigna al proyecto.
Flujo normal de los eventos.	
Acción del Actor.	**Respuesta del Negocio.**
1. El líder de proyecto solicita realizar cambios en el proyecto productivo.	1.1 El Vicedecano de Producción pregunta que cambios necesita realizar en el proyecto productivo.
2. Le muestra los detalles del cambio que necesita realizar.	2.1 El Vicedecano de Producción analiza la magnitud del cambio.
	2.2 Si el Vicedecano de Producción aprueba el cambio asigna los nuevos recursos requeridos por el líder de proyecto.
	2.3 Actualiza el registro de proyecto con los nuevos datos.
Flujos Alternos.	
Acción del Actor.	**Respuesta del Negocio.**
2. a	2.2 Si el Vicedecano de Producción no aprueba el cambio le comunica al líder de proyecto que no se aprueban las modificaciones requeridas por el para el proyecto productivo.
Mejoras:	Se contará con un proceso de modificar cualquier recurso de un proyecto de forma automática, permitiendo mantener en todo momento un registro actualizado de la información de los proyectos de la facultad.

Tabla 4: Descripción textual del CUN: "Modificar Proyecto Productivo".

Diagrama de actividades del caso de uso del negocio "Modificar Proyecto Productivo".

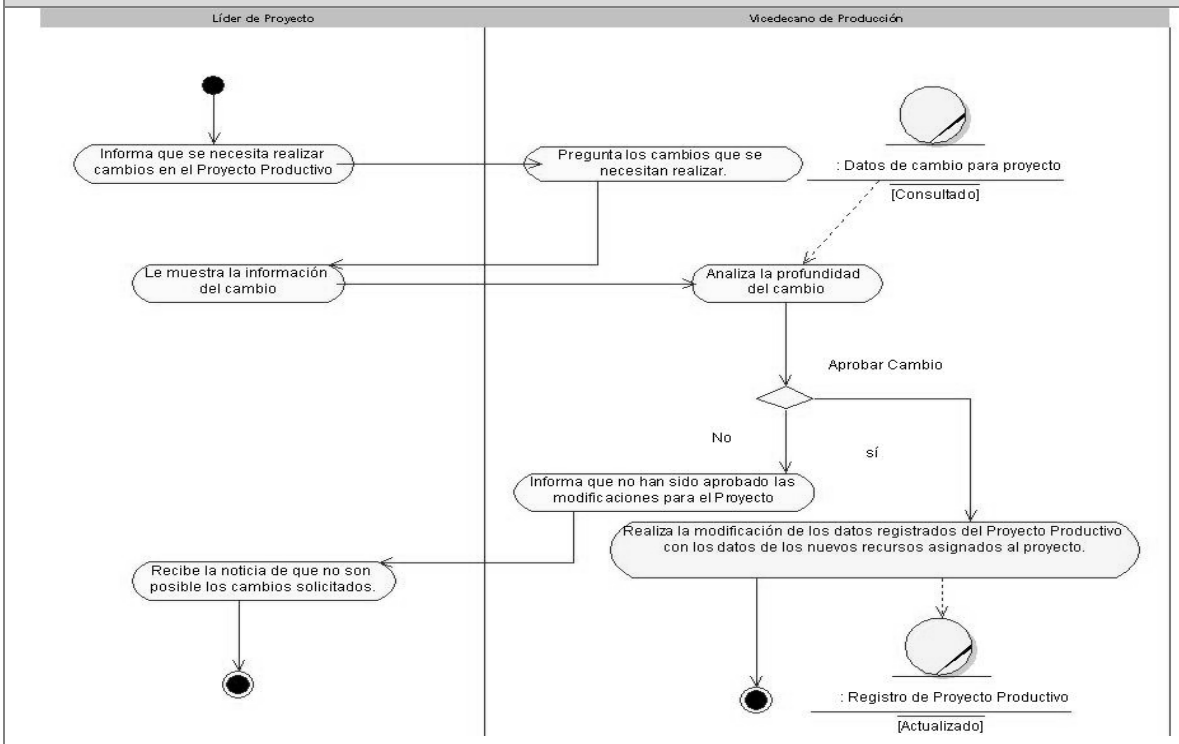

Figura 10: Diagrama de actividades del CUN: "Modificar Proyecto Productivo".

Caso de Uso:	**Terminar Proyecto Productivo.**
Actores:	Líder de Proyecto.
Trabajadores:	Vicedecano de Producción.
Resumen:	El caso de uso inicia cuando el líder de proyecto le informa al Vicedecano de Producción que el proyecto ha concluido. El Vicedecano de Producción registra el nuevo producto obtenido en el proyecto que concluye.
Flujo normal de los eventos.	
Acción del Actor.	**Respuesta del Negocio.**
1. El líder de proyecto comunica que ha concluido el proyecto productivo.	1.1 El Vicedecano de Producción pide los datos del nuevo producto terminado.
2. El líder de proyecto entrega los datos del producto terminado.	2.1 El Vicedecano de Producción registra el nuevo producto.
	2.2 El Vicedecano de Producción actualiza el

	registro de proyectos de la facultad.
Mejoras:	Se mantendrá un registro automatizado y actualizado, que permitirá un mejor control sobre los productos desarrollados por la facultad.

Tabla 5: Descripción textual del caso de uso del negocio "Terminar Proyecto Productivo".

Diagrama de actividades del caso de uso del negocio "Terminar Proyecto Productivo".

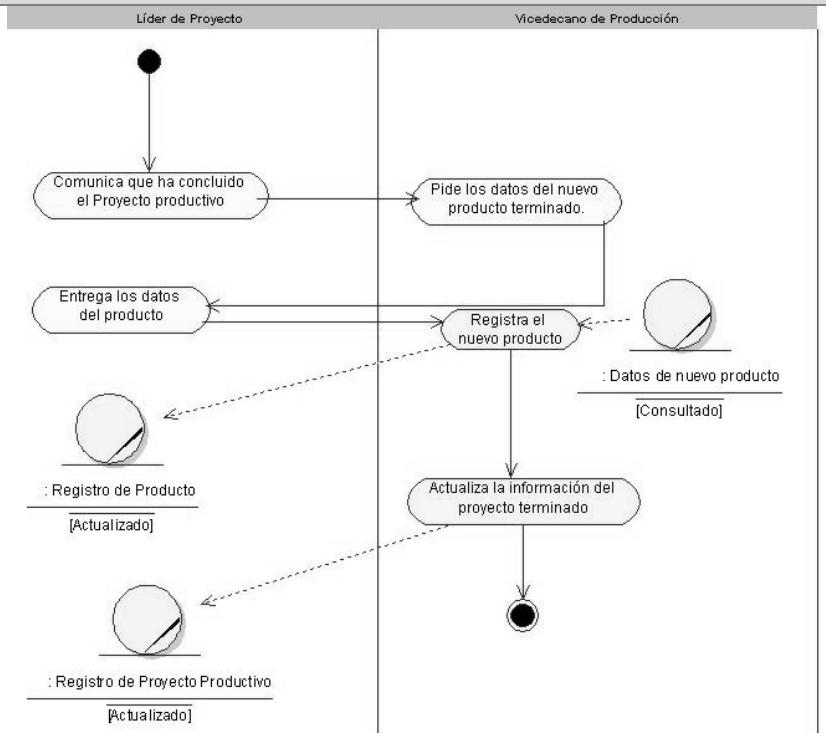

Figura 11: Diagrama de actividades del CUN: "Terminar Proyecto Productivo".

Caso de Uso:	**Solicitar Reporte.**
Actores:	Directivo.
Trabajadores:	Vicedecano de Producción.
Resumen:	El caso de uso inicia cuando algún directivo, ya sea dentro de la facultad o fuera de ella, solicita información detallada acerca de los proyectos o polos existentes en la facultad. El Vicedecano de Producción pregunta el tipo de información exacta que el directivo desea

	obtener y elabora el reporte.
Flujo normal de los eventos.	
Acción del Actor.	**Respuesta del Negocio.**
1. El directivo solicita información.	1.1 El Vicedecano de Producción pregunta el tipo de información exacta que se desea obtener.
2. El directivo le detalla la información que desea obtener.	2.1 El Vicedecano de Producción elabora el reporte con la información señalada y se lo entrega.
3. El directivo recibe el reporte elaborado por el Vicedecano de Producción.	
Mejoras:	Se obtendrá una automatización de los reportes requeridos por los directivos, permitiendo una mayor rapidez a la hora de buscar información y permitiendo un mejor control sobre la misma.

Tabla 6: Descripción textual del caso de uso del negocio "Solicitar Reporte".

Diagrama de actividades del caso de uso del negocio "Solicitar Reporte".

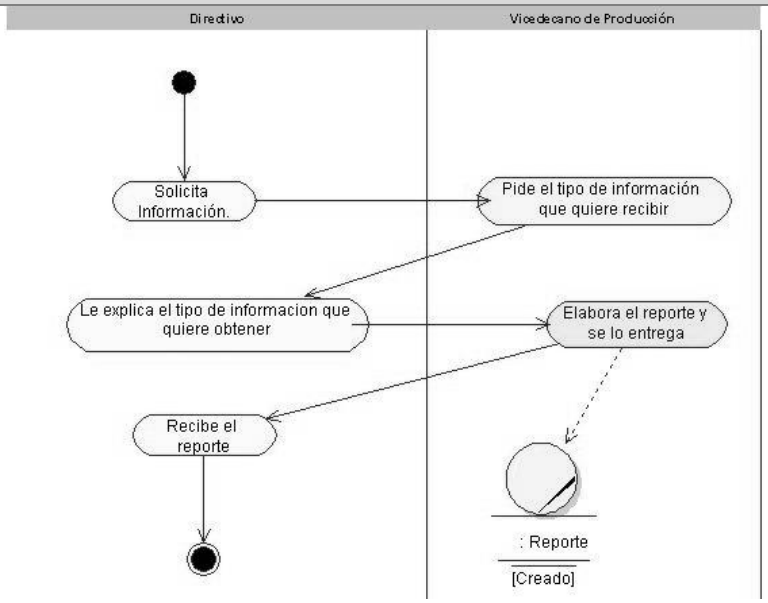

Figura 12: Diagrama de actividades del CUN: "Solicitar Reporte".

Nota 1: El resto de los diagramas de actividades, correspondientes a los casos de uso del negocio "Registrar Polo Productivo" y "Modificar Polo Productivo", como cada una de sus descripciones textuales, se encuentran en el **Anexo 1**.

2.6.4 Modelo de objetos del negocio.

El modelo de objetos del negocio, es un modelo interno a un negocio, muestra la relación entre los trabajadores y entidades del negocio dentro del flujo de trabajo de modelamiento del negocio. Describe cómo cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo.

A continuación se muestra la figura 13 correspondiente al modelo de objetos del negocio:

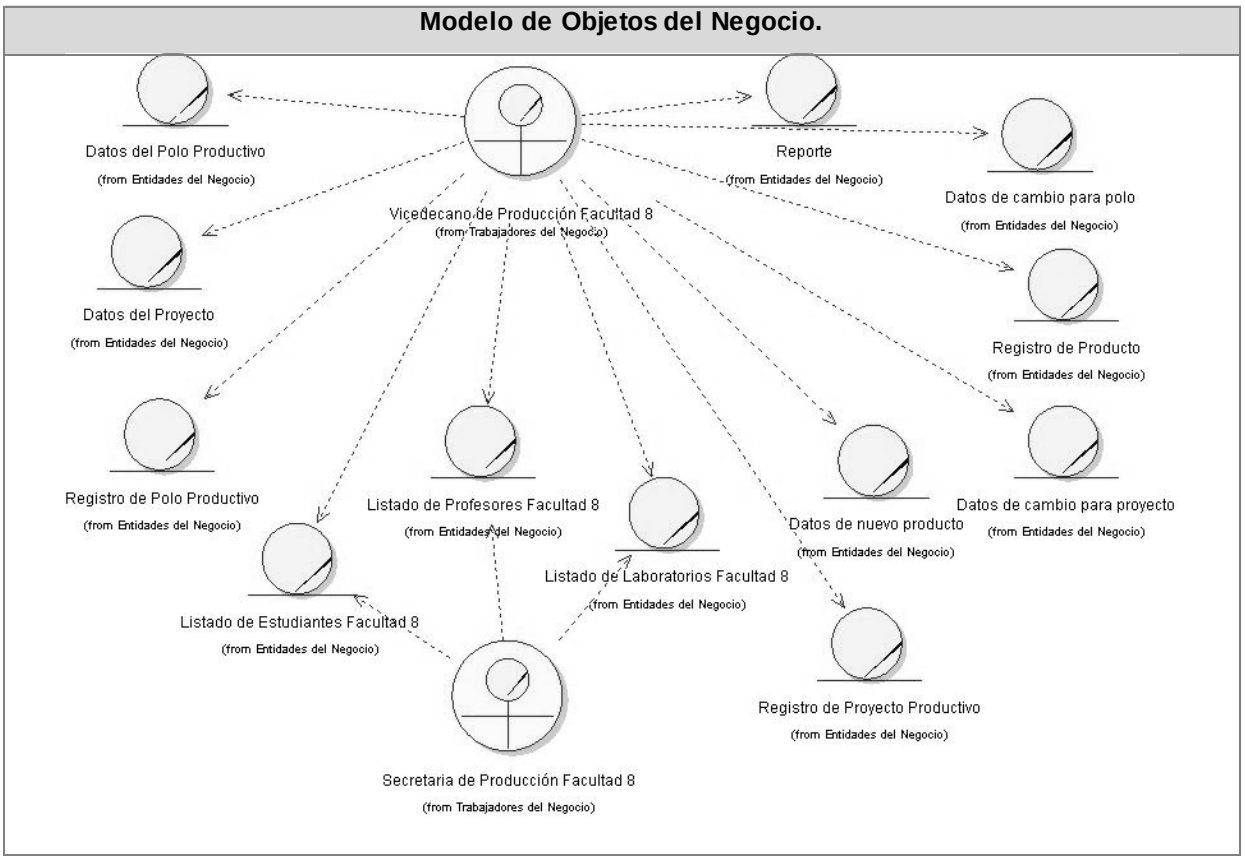

Figura 13: Modelo de objetos del negocio.

2.6.5 Reglas del negocio.

Las reglas del negocio regulan aspectos del negocio, ya que describen políticas que deben cumplirse o condiciones que deben ser satisfechas. Son las restricciones que existen en un negocio dado;

entiéndase por esto las acciones no válidas que la aplicación debe controlar para que el negocio no colapse.

A continuación se listan las reglas identificadas en el presente negocio:

- ✓ Un proyecto puede pertenecer solamente a un polo productivo.
- ✓ Un polo productivo no puede abarcar más de una línea temática.
- ✓ Un estudiante puede pertenecer a solo un proyecto productivo.
- ✓ Un proyecto productivo debe tener al menos un líder.
- ✓ Un polo productivo debe tener un responsable.

2.7 Captura de requisitos.

El flujo de trabajo de requerimientos es uno de los más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que se construya. Pudiera verse a los requisitos como al contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen.

Los requisitos se dividen en dos grupos. Los requisitos funcionales y los no funcionales:

2.7.1 Requisitos funcionales.

Los requerimientos funcionales son aquellas capacidades o condiciones que el sistema debe cumplir. Ellos definen qué es lo que el sistema debe hacer y permiten identificar las funcionalidades requeridas. No alteran la funcionalidad del producto, es decir, los requerimientos funcionales se mantienen invariables sin importarle con que propiedades o cualidades se relacionen.

A continuación se enumeran las funcionalidades que el sistema debe cumplir:

RF 1 Permitir autenticarse.

1.1 Autenticar usuario.
1.2 Cerrar sesión.

RF 2 Gestionar proyectos productivos.

2.1 Adicionar proyectos.
2.2 Ubicar los proyectos en los polos de acuerdo a la temática que aborden.
2.3 Asignar un cliente a cada proyecto.
2.4 Asignarles estudiantes a los proyectos.
2.5 Asignarles profesores a los proyectos.

2.6 Asignarles roles a los estudiantes y profesores.

2.7 Ubicar los proyectos en laboratorios.

2.7.1 Asignarles PCs a los proyectos.

2.8 Buscar un determinado proyecto.

2.9 Actualizar los datos de un proyecto seleccionado.

2.10 Terminar un proyecto.

2.11 Registrar un nuevo producto.

RF 3 Gestionar polos productivos.

3.1 Adicionar polos.

3.2 Buscar un determinado polo.

3.3 Actualizar los datos de un polo seleccionado.

3.4 Eliminar un polo.

RF 4 Generar reportes.

4.1 Listar estudiantes de la facultad.

4.2 Listar profesores de la facultad.

4.3 Listar proyectos dado un polo.

4.4 Total de proyectos de la facultad.

4.5 Total de estudiantes que pertenecen a proyectos en la facultad.

4.6 Total de profesores vinculados a proyectos en la facultad.

4.7 Listar estudiantes que no pertenecen a proyecto.

4.8 Listar profesores que no pertenecen a proyecto.

4.9 Total de polos existentes en la facultad.

4.10 Total de productos por polos.

4.11 Total de productos de la facultad.

RF 5 Gestionar usuarios del sistema.

5.1 Adicionar nuevos usuarios.

5.2 Modificar los datos de un usuario.

5.3 Eliminar usuarios.

5.4 Asignarle permisos a los usuarios del sistema.

RF 6 Permitir cancelar cualquiera de las operaciones anteriores si no se desea continuar con su ejecución.

2.7.2 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. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente 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.

A continuación se presentan los requisitos no funcionales del sistema a desarrollar:

Requerimientos de Software:

- ✓ **RNF 1:** Navegador Internet Explorer v6.0 o superior, Mozilla Firefox v2.x.
- ✓ **RNF 2:** Servidor Web Apache v2.x o superior con módulo PHP 5.2.3 disponible, este debe estar configurado con las extensiones “pgsql”, “soap”, “ldap”.
- ✓ **RNF 3:** Como Sistema Gestor de Base de Datos: PostgreSQL preferiblemente v8.x en adelante.

Requerimientos de Hardware:

- ✓ **RNF 4:** Tarjeta de red.
- ✓ **RNF 5:** Para los servidores tanto Web como SGBD: PENTIUM IV o superior con 256 MB de RAM o más.
- ✓ **RNF 6:** Capacidad de disco duro, preferiblemente mayor a los 10 GB.

Restricciones para el diseño e implementación:

- ✓ **RNF 7:** Utilizar como lenguaje del lado del servidor al PHP v5.2.3 o superior y del lado del cliente: HTML y JavaScript.
- ✓ **RNF 8:** Para la programación en PHP se recomienda el IDE: EasyEclipse.
- ✓ **RNF 9:** Se recomienda el uso de la arquitectura MVC.
- ✓ **RNF 10:** El código deberá ser reutilizable.

Requerimientos de Apariencia o interfaz externa:

- ✓ **RNF 11:** Estará diseñado para resolución de 1024x768, aunque deberá verse en 800x600 o cualquier resolución superior a esta.

- ✓ **RNF 12:** La interfaz debe ser agradable para el usuario, que combine correctamente los colores, tipo de letra y tamaño, que los iconos estén en correspondencia con lo que representan no debe contener muchas imágenes que demoren las respuestas al usuario.
- ✓ **RNF 13:** La navegabilidad debe ser sencilla.

Requerimientos de Seguridad:

- ✓ **RNF 14:** Se podrá acceder al sistema solamente después de autenticarse.
- ✓ **RNF 15:** Autorización (Atribución a los usuarios respecto a sus funciones de trabajo.)
- ✓ **RNF 16:** Contar con un sistema de permisos y usuarios para el acceso a al información.
- ✓ **RNF 17:** Chequeo de seguridad sobre las operaciones no reversibles (adicionar, modificar, eliminar).
- ✓ **RNF 18:** Tiene que ser capaz de anular cualquier acción incorrecta que atente contra la integridad de los datos.
- ✓ **RNF 19:** El sistema debe estar disponible las 24 horas del día (disponibilidad).

Requerimientos de Usabilidad:

- ✓ **RNF 20:** El sistema debe permitir el acceso concurrente de diferentes usuarios, en dependencia del nivel de usabilidad que este definida para cada uno.
- ✓ **RNF 21:** RNF 19: El sistema podrá ser usado por cualquier persona que posea conocimientos básicos de computación y trabajo en la Web.

Requerimientos de Rendimiento:

- ✓ **RNF 22:** El sistema deberá ser capaz de gestionar toda la información y dar respuesta a las solicitudes lo más rápido posible.

Requerimientos de Soporte:

- ✓ **RNF 23:** El sistema debe ser de fácil instalación, mantenimiento y configuración.
- ✓ **RNF 24:** Contar con una etapa de prueba para eliminar la mayor cantidad de errores.

Requerimientos de Portabilidad:

- ✓ **RNF 25:** El sistema deberá ser multiplataforma, haciendo énfasis en Linux y Windows.

Requerimientos Políticos Culturales:

- ✓ **RNF 26:** El sistema solo podrá ser utilizado en territorio cubano.

- ✓ **RNF 27:** El sistema debe tener una interfaz que esté acorde con el lugar donde se implantará, es decir, que refleje los ideales de la organización.
- ✓ **RNF 28:** El producto no debe contener palabras en otros idiomas.
- ✓ **RNF 29:** El producto debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.

Requerimientos Legales:

- ✓ **RNF 30:** La mayoría de las herramientas de desarrollo son libres y del resto, las licencias están avaladas.
- ✓ **RNF 31:** Reconocido y autorizado por instancias superiores tales como la directiva de la UCI.
- ✓ **RNF 32:** Documentación legal de uso como Declaración de Autoría.
- ✓ **RNF 33:** La plataforma sobre la que se va a ejecutar el sistema esté bajo la licencia de software libre.

Requerimientos de Confiabilidad:

- ✓ **RNF 34:** Los reportes que se obtendrán deben ser 100% precisos y reales.
- ✓ **RNF 35:** Deben establecerse los mecanismos necesarios para el restablecimiento del sistema ante fallos de comunicación u otros.
- ✓ **RNF 36:** Garantiza un control estricto sobre el tráfico de información.
- ✓ **RNF 37:** Chequeo constante de la integridad y consistencia en los datos.

2.8 Modelado del sistema.

2.8.1 Definición de los actores del sistema.

Los actores del sistema representan terceros fuera del sistema que colaboran con él. Estos pueden ser tanto los usuarios como los sistemas externos de un sistema que interactúan con él y que pueden ser expresados mediante uno o más actores del sistema.

A continuación se muestra la tabla7, con los actores del sistema y su correspondiente justificación:

Actores del Sistema.	Justificación.
Vicedecano de Producción.	Es el encargado de gestionar toda la información relacionada con los proyectos y polos productivos (adicionar, actualizar, eliminar etc.). Así como de administrar el sistema y obtener información del mismo con previa autenticación.
Usuario.	Es todo aquel que hace uso del sistema para obtener información con previa autenticación.

Tabla 7: Actores del sistema.

2.8.2 Modelo de casos de uso del sistema.

Los casos de uso del sistema son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento desde el punto de vista del usuario. Son un conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor. Dicho en otras palabras, no son más que fragmentos de funcionalidad que el sistema ofrece a los actores que interactúan con el mismo, reportándoles beneficios.

El modelo de casos de uso del sistema describe lo que hace el sistema para cada tipo de usuario. Permite que se establezca un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema y proporciona la entrada fundamental para el análisis y diseño.

A continuación se muestra la figura 14 perteneciente al diagrama de casos de uso del sistema:

Diagrama de Casos de Uso del Sistema.

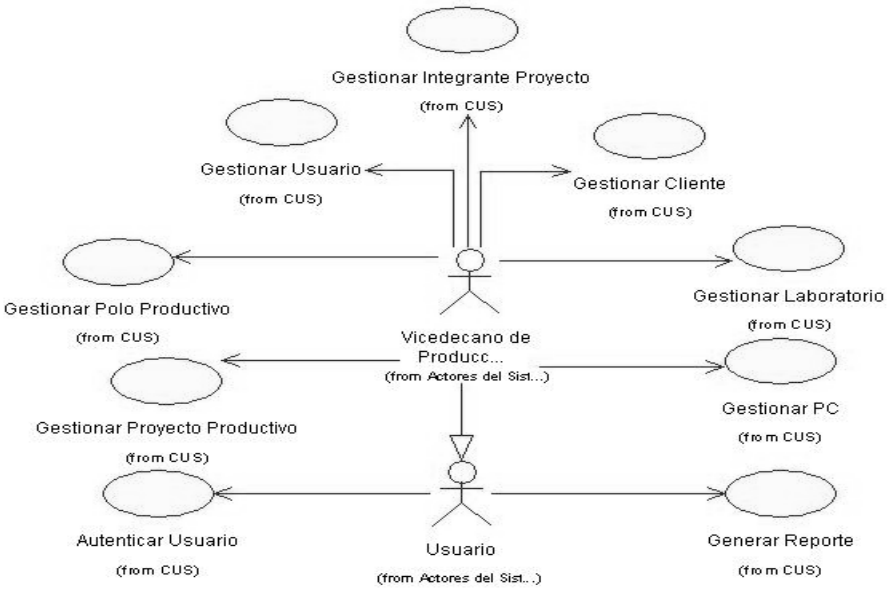

Figura 14: Diagrama de casos de uso del sistema.

2.8.3 Descripción de los casos de uso del sistema.

La descripción de los casos de uso del sistema, detallan las acciones que tienen lugar durante la interacción actor-sistema, es decir, describe el flujo de actividades que realiza el actor al hacer uso del sistema y las correspondientes respuestas del mismo.

A continuación se muestran las descripciones de los casos de uso del sistema:

Nombre del Caso de Uso:	**Autenticar Usuario.**
Actores:	Usuario (Inicia).
Propósito:	Permite a los usuarios del sistema identificarse previamente para hacer uso de él, según los permisos que posee y por último, el cierre de sesión al terminar su trabajo.
Resumen:	El caso de uso se inicia cuando un usuario del sistema desea hacer uso del mismo. Para esto es necesario realizar una autenticación previa y el sistema da el acceso a los recursos de acuerdo al usuario identificado.

	Termina el caso de uso cuando el usuario entra al sistema o se le deniega el acceso.
Referencias:	RF 1
Precondiciones:	----------
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
1. El usuario se encuentra frente a la página principal de la aplicación Web.	1.1 El sistema muestra el formulario en el cual el usuario debe introducir los datos necesarios para ser identificado (usuario y contraseña) y así otorgarle los permisos.
2. El usuario introduce los datos que ha solicitado el sistema (usuario y contraseña para ser identificado como usuario del sistema) y presiona el botón "Aceptar".	2.1 El sistema verifica usuario y contraseña introducidos.
	2.2 Si los datos introducidos por el usuario se corresponden con los de un usuario registrado en la base de datos el sistema le permite acceder al sistema de acuerdo a los permisos que posea.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
2. a	2.2 Si los datos introducidos por el usuario no se corresponden con los de un usuario registrado en la base de datos el sistema muestra un mensaje de error y le indica al usuario que no es válido (le deniega el acceso).
Postcondiciones:	El usuario queda autorizado para acceder a los recursos del sistema de acuerdo con los permisos que posea.
Prioridad:	Secundario.

Tabla 8: Descripción del caso de uso del sistema "Autenticar Usuario".

Nombre del Caso de Uso:	**Gestionar Proyecto Productivo.**
Actores:	Vicedecano de Producción (Inicia).
Propósito:	Permitir, registrar, actualizar, buscar y terminar un proyecto.
Resumen:	El caso de uso se inicia cuando el Vicedecano de Producción selecciona la opción "Gestionar Proyecto" el sistema muestra una interfaz con todos los tipos de gestiones que se pueden efectuar: adicionar, actualizar, buscar y terminar proyecto. El Vicedecano de Producción escoge la opción que desea realizar. El caso de uso culmina cuando el Vicedecano de Producción termina con la opción seleccionada.
Referencias:	RF 2
Precondiciones:	1. Tiene que haberse autenticado correctamente con los permisos necesarios.
	2. Si la opción seleccionada es la de actualizar un proyecto, buscar los datos referentes a él, o terminar un proyecto, dicho proyecto debe haber sido adicionado ya a la base de datos.
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
1. El Vicedecano de Producción selecciona la opción de "Gestionar Proyecto".	1.1 El sistema muestra las opciones de gestionar un proyecto: ✓ Adicionar Proyecto. ✓ Actualizar proyecto. ✓ Buscar Proyecto. ✓ Terminar Proyecto.
Sección: Adicionar proyecto productivo.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción "Adicionar Proyecto".	2.1 El sistema muestra el formulario para introducir los datos necesarios para adicionar

	un nuevo proyecto.
3. El Vicedecano de Producción introduce los datos requeridos por el sistema para la adición de un nuevo proyecto y luego presiona el botón “Adicionar”.	3.1 El sistema verifica la validez de los datos introducidos.
	3.2 Si los datos son válidos, el sistema los adiciona en la base de datos y crea al nuevo proyecto.
	3.3 El sistema muestra un mensaje de que el proyecto se ha adicionado correctamente en la base de datos. Finaliza el caso de uso.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
3. a	3.2 Si los datos no son válidos el sistema muestra un mensaje de error indicando que los datos no son válidos.
3. b El Vicedecano de Producción decide que no desea adicionar un nuevo proyecto y presiona el botón “Cancelar”.	3.1 El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	El proyecto queda adicionado en la base de datos.
Sección: Actualizar proyecto productivo.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Actualizar Proyecto”.	2.1 El sistema muestra los proyectos existentes en la base de datos.
3. El Vicedecano de Producción selecciona el proyecto al cual desea modificar sus datos.	3.1 El sistema obtiene todos los datos referentes al proyecto seleccionado y los muestra.
4. El Vicedecano de Producción modifica los datos del proyecto seleccionado. Presiona el botón “Aceptar”.	4.1 El sistema verifica la validez de los datos introducidos.
	4.2 Si los datos son válidos el sistema actualiza en la base de datos los datos que se modificaron del proyecto seleccionado.

	4.3 El sistema muestra un mensaje de que los datos del proyecto se han modificado satisfactoriamente.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
4. a	4.2 Si los datos no son válidos el sistema muestra un mensaje de error indicando que los datos no son válidos.
4. b El Vicedecano de Producción decide que no desea actualizar los datos de un proyecto y presiona el botón “Cancelar”.	4.1 El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	El proyecto queda actualizado en la base de datos.
Sección: Buscar proyecto productivo.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Buscar Proyecto”.	2.1 El sistema muestra los proyectos existentes en la base de datos.
3. El Vicedecano de Producción selecciona el proyecto del cual desea ver sus datos.	3.1 El sistema obtiene todos los datos referentes al proyecto seleccionado y los muestra.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
3. a El Vicedecano de Producción decide que no desea buscar los datos de un proyecto y presiona el botón “Cancelar”.	3. El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	Los datos de un proyecto quedan mostrados en la pantalla para que puedan ser consultarlos.
Sección: Terminar proyecto productivo.	
Curso normal de los eventos.	
Acción del actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Terminar Proyecto”.	2.1 El sistema muestra los proyectos existentes en la base de datos.

3. El Vicedecano de Producción selecciona el proyecto que termina.	3.1 El sistema actualiza el estado del proyecto en la base de datos con terminado.
	3.2 El sistema adiciona un nuevo producto a la tabla productos de la base de datos.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
3. a El Vicedecano de Producción decide que no desea terminar un proyecto y presiona el botón "Cancelar".	3.1 El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	1. El proyecto queda en la base de datos con estado terminado.
	2. Queda adicionado un nuevo producto en la base de datos.
Prioridad:	Crítico.

Tabla 9: Descripción del caso de uso del sistema "Gestionar Proyecto Productivo".

Nombre del Caso de Uso:	**Gestionar Integrante Proyecto.**
Actores:	Vicedecano de Producción (Inicia).
Propósito:	Permitir, registrar, actualizar y buscar la información de un integrante de proyecto, así como eliminarla.
Resumen:	El caso de uso inicia cuando el Vicedecano de Producción elige la opción de "Gestionar Integrante Proyecto", se muestran las opciones de gestionar un integrante proyecto. El Vicedecano de Producción selecciona una de las opciones. El caso de uso finaliza cuando el Vicedecano de Producción termina la acción seleccionada.
Referencia:	RF 2
Precondiciones:	1. Tiene que haberse autenticado correctamente con los permisos necesarios.

	2. Si la opción seleccionada es la de actualizar un integrante de un proyecto, buscar los datos referentes a él, o eliminar un integrante de un proyecto determinado, dicho integrante debe haber sido adicionado a un proyecto ya en la base de datos.
Curso normal de los eventos.	
Acción del Actor:	**Respuesta del Sistema:**
1. El Vicedecano de Producción selecciona la opción de “Gestionar Integrante Proyecto”.	1.1 El sistema muestra las opciones de gestionar un laboratorio: ✓ Adicionar Integrante Proyecto. ✓ Actualizar Integrante Proyecto. ✓ Buscar Integrante Proyecto. ✓ Eliminar Integrante Proyecto.
Sección: Adicionar Integrante Proyecto.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Adicionar Integrante Proyecto”.	2.1 El sistema muestra el formulario para introducir los datos necesarios para adicionar un integrante a un proyecto.
3. El Vicedecano de Producción introduce los datos requeridos por el sistema para la adición de un nuevo integrante a un proyecto y luego presiona el botón “Adicionar”.	3.1 El sistema verifica la validez de los datos introducidos.
	3.2 Si los datos son válidos, el sistema los adiciona en la base de datos y crea al nuevo laboratorio.
	3.3 El sistema muestra un mensaje de que el integrante se ha adicionado correctamente a un proyecto en la base de datos. Finaliza el caso de uso.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
3. a	3.2 Si los datos no son válidos el sistema muestra un mensaje de error indicando que los datos no son válidos.
3. b El Vicedecano de Producción decide que no	3.1 El sistema redirecciona al Vicedecano de

desea adicionar un nuevo integrante a proyecto y presiona el botón “Cancelar”.	Producción para la página de Inicio.
Postcondiciones:	El proyecto queda adicionado en la base de datos.
Sección: Actualizar Integrante Proyecto.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Actualizar Integrante Proyecto”.	2.1 El sistema muestra los integrantes por cada proyecto existentes en la base de datos.
3. El Vicedecano de Producción selecciona al integrante de un proyecto al cual desea modificar sus datos.	3.1 El sistema obtiene todos los datos referentes al integrante seleccionado y los muestra.
4. El Vicedecano de Producción modifica los datos del integrante seleccionado. Presiona el botón “Aceptar”.	4.1 El sistema verifica la validez de los datos introducidos.
	4.2 Si los datos son válidos el sistema actualiza en la base de datos los datos que se modificaron del integrante seleccionado.
	4.3 El sistema muestra un mensaje de que los datos del integrante se han modificado satisfactoriamente.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
4. a	4.2 Si los datos no son válidos el sistema muestra un mensaje de error indicando que los datos no son válidos.
4. b El Vicedecano de Producción decide que no desea actualizar los datos de un integrante de proyecto y presiona el botón “Cancelar”.	4.1 El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	El integrante de proyecto queda actualizado en la base de datos.
Sección: Buscar Integrante Proyecto.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Buscar Integrante Proyecto”.	2.1 El sistema muestra los integrantes de proyecto existentes en la base de datos.

3. El Vicedecano de Producción selecciona el integrante de proyecto del cual desea ver sus datos.	3.1 El sistema obtiene todos los datos referentes al integrante de proyecto seleccionado y los muestra.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
3. a El Vicedecano de Producción decide que no desea Buscar los datos de un integrante de proyecto y presiona el botón “Cancelar”.	3.1 El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	Los datos del integrante de proyecto seleccionado quedan en la pantalla para que puedan ser consultarlos.
Sección: Eliminar Integrante Proyecto.	
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
2. El Vicedecano de Producción selecciona la opción “Eliminar Integrante Proyecto”.	2.1 El sistema muestra los integrantes de proyectos existentes en la base de datos.
3. El Vicedecano de Producción selecciona el integrante de proyecto que desea eliminar.	3.1 El sistema pregunta si está seguro de realizar la operación indicada (eliminar integrante de proyecto).
4. El Vicedecano de Producción presiona la opción aceptar.	4.1 El sistema elimina el integrante de proyecto de la base de datos.
	4.2 El sistema muestra un mensaje de que el integrante de proyecto ha sido eliminado.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
4. a El Vicedecano de Producción presiona la opción cancelar.	4.1 El sistema muestra un mensaje de que la acción ha sido cancelada y redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	Queda eliminado el integrante de proyecto seleccionado de la base de datos.
Prioridad:	Crítico.

Tabla 10: Descripción del caso de uso del sistema “Gestionar Integrante Proyecto”.

Nombre del Caso de Uso:	**Generar Reporte.**
Actores:	Usuario (Inicia).
Propósito:	Permitir que los usuarios obtengan la información que necesiten sobre un determinado tema.
Resumen:	El caso de uso se inicia cuando un usuario le indica al sistema que quiere obtener un reporte e indica el tipo de reporte que desea. El sistema le muestra todos los datos que tributan al reporte seleccionado. Termina el caso de uso.
Referencia:	RF 4
Precondiciones:	Requiere haberse autenticado previamente.
Curso normal de los eventos.	
Acción del Actor.	**Respuesta del Sistema.**
1. El usuario selecciona la opción de "Generar Reporte".	1.1 El sistema muestra las diferentes tipos de reporte que puede brindar: ✓ Listar estudiantes de la facultad. ✓ Listar profesores de la facultad. ✓ Listar proyectos dado un Polo. ✓ Total de proyectos de la facultad. ✓ Total de estudiantes que pertenecen a proyectos en la facultad. ✓ Total de profesores vinculados a proyectos en la facultad. ✓ Listar estudiantes que no pertenecen a proyecto. ✓ Listar profesores que no pertenecen a proyecto. ✓ Total de polos existentes en la facultad. ✓ Total de productos por polos. ✓ Total de productos de la facultad.
2. El usuario selecciona la opción del tipo de reporte	2.1 En caso de que el sistema necesite para

que desea obtener.	realizar el reporte algún parámetro, muestra el formulario en el cual introducir los datos requeridos.
3. El usuario introduce los datos que requiere el sistema. Presiona el botón "Enviar".	3.1 El sistema muestra los datos correspondientes con el reporte solicitado.
Cursos Alternos.	
Acción del Actor.	**Respuesta del Sistema.**
2.a	2.1 En caso de que el sistema no necesite para realizar el reporte ningún parámetro, muestra los datos correspondientes con el reporte solicitado.
3. a El usuario decide que no desea obtener ningún reporte y presiona el botón "Cancelar".	3.1 El sistema redirecciona al Vicedecano de Producción para la página de Inicio.
Postcondiciones:	El reporte queda mostrado en la pantalla para que pueda ser consultado.
Prioridad:	Secundario.

Tabla 11: Descripción del caso de uso del sistema "Generar Reporte".

Nota 2: El resto de las descripciones textuales, correspondientes a los casos de uso del sistema "Gestionar Usuario", "Gestionar Polo Productivo", "Gestionar Laboratorio", "Gestionar PC" y "Gestionar Cliente", se encuentran en el **Anexo 1**.

2.9 Conclusiones.

En este capítulo, a partir de la comprensión de los procesos de negocio, se definieron las principales funcionalidades que debe tener el sistema a desarrollar, estructurándose en casos de uso. Se elaboró el diagrama de casos de uso del sistema y se describieron textualmente cada uno de los casos de uso identificados.

Todo lo anterior, provee de una visión general de lo que el sistema debe hacer, por lo que se está en condiciones de pasar al siguiente capítulo, que expondrá cómo se realizarán las operaciones que aquí se describieron.

Capítulo 3: Análisis y Diseño del Sistema.

3.1 Introducción.

En este capítulo se muestran los resultados arrojados durante el flujo de trabajo de Análisis y Diseño del sistema a través de los diagramas de clases del análisis y del diseño, además de los diagramas de interacción y del diseño de la base de datos.

3.2 Análisis del sistema.

El objetivo fundamental del flujo de análisis consiste en obtener una visión del sistema orientada en ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Las actividades de análisis se desarrollan para facilitar la entrada al diseño, por lo que se convierten en un paso inicial y en una primera aproximación conceptual donde una vez comprendido los requisitos a este nivel, se aumenta la especificidad en aras de garantizar el cubrimiento de los requisitos funcionales y no funcionales.

3.2.1 Diagramas de clases del análisis.

Un diagrama de clases del análisis representa los conceptos en un dominio del problema, las cosas del mundo real, no de la implementación automatizada de estas cosas.

A continuación se muestran los diagramas de clases del análisis:

Diagrama de Clases del Análisis: CU Autenticar Usuario.

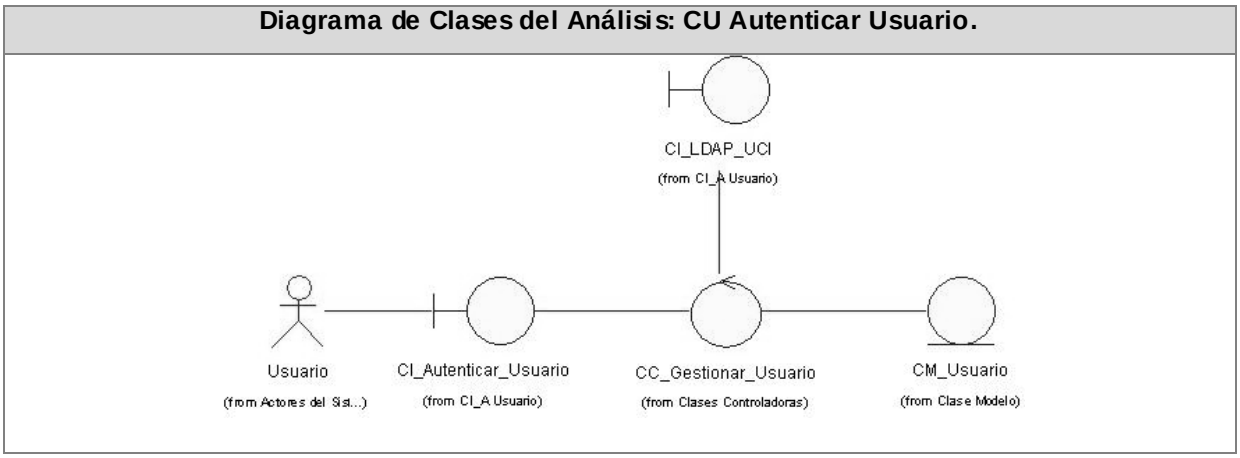

Figura 15: Diagrama de clases del análisis: CU Autenticar Usuario.

Diagrama de Clases del Análisis: CU Gestionar Proyecto Productivo.

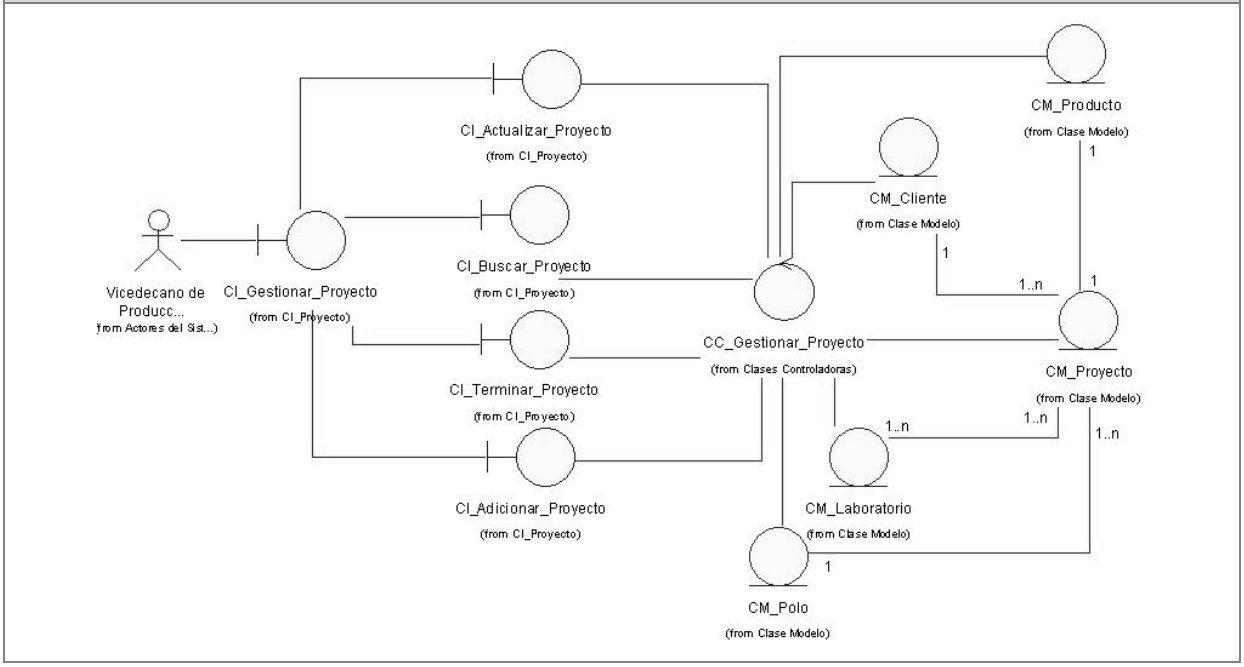

Figura 16: Diagrama de clases del análisis: CU Gestionar Proyecto Productivo.

Diagrama de Clases del Análisis: CU Gestionar Integrante Proyecto.

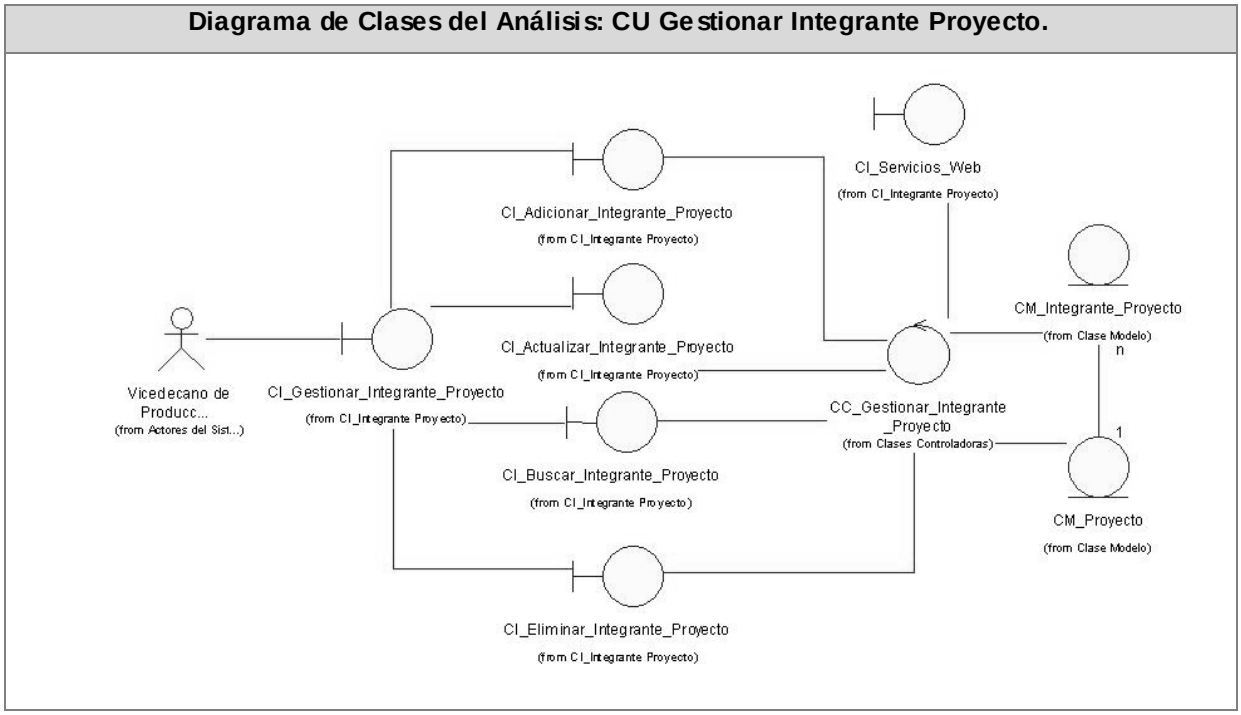

Figura 17: Diagrama de clases del análisis: CU Gestionar Integrante Proyecto.

Diagrama de Clases del Análisis: CU Generar Reporte.

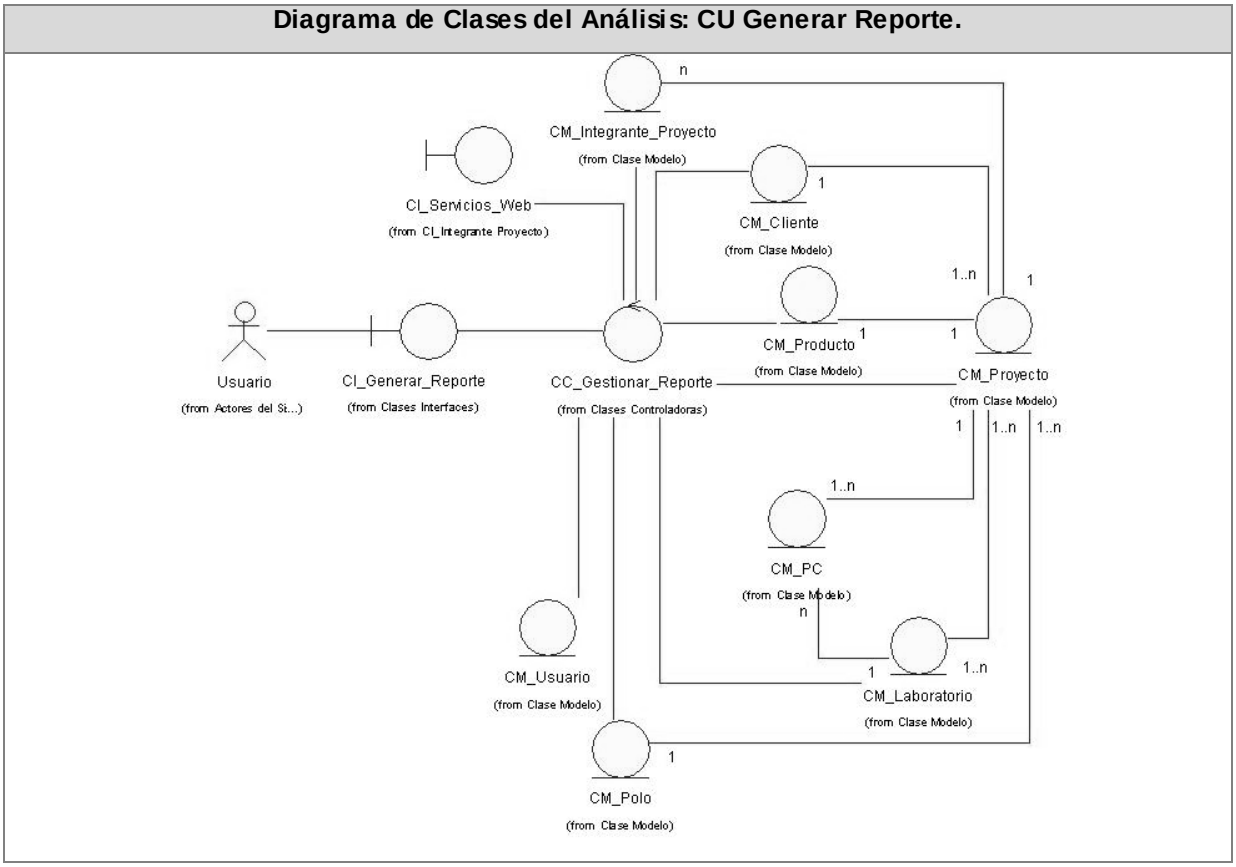

Figura 18: Diagrama de clases del análisis: CU Generar Reporte.

Nota 3: El resto de los diagramas de clases del análisis se encuentran en el **Anexo 2.**

3.3 Diseño del sistema.

Mediante el diseño se realiza un refinamiento del análisis, teniendo en cuenta los requisitos no funcionales, que no es más que ver cómo cumple el sistema sus objetivos y considerando además el entorno de implementación (Lenguaje de programación, plataforma, Sistemas heredados con los cuales interactuar, etc.). La ayuda del Rational define como propósitos del diseño:

- ✓ Transformar los requerimientos en un diseño de cómo debe ser el sistema.
- ✓ Desarrollar una robusta arquitectura del sistema.
- ✓ Adaptar el diseño para que se corresponda con el entorno de implementación, diseñando sus funcionalidades.

3.3.1 Diagramas de clases del diseño.

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones con sus relaciones estructurales y de herencia. Gráficamente es una colección de vértices y arcos. En el caso de las aplicaciones Web, representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase. En este tipo de aplicaciones son más importantes la modelación de la lógica y estado del negocio que los detalles de presentación.

A continuación se muestran los diagramas de clases del diseño Web:

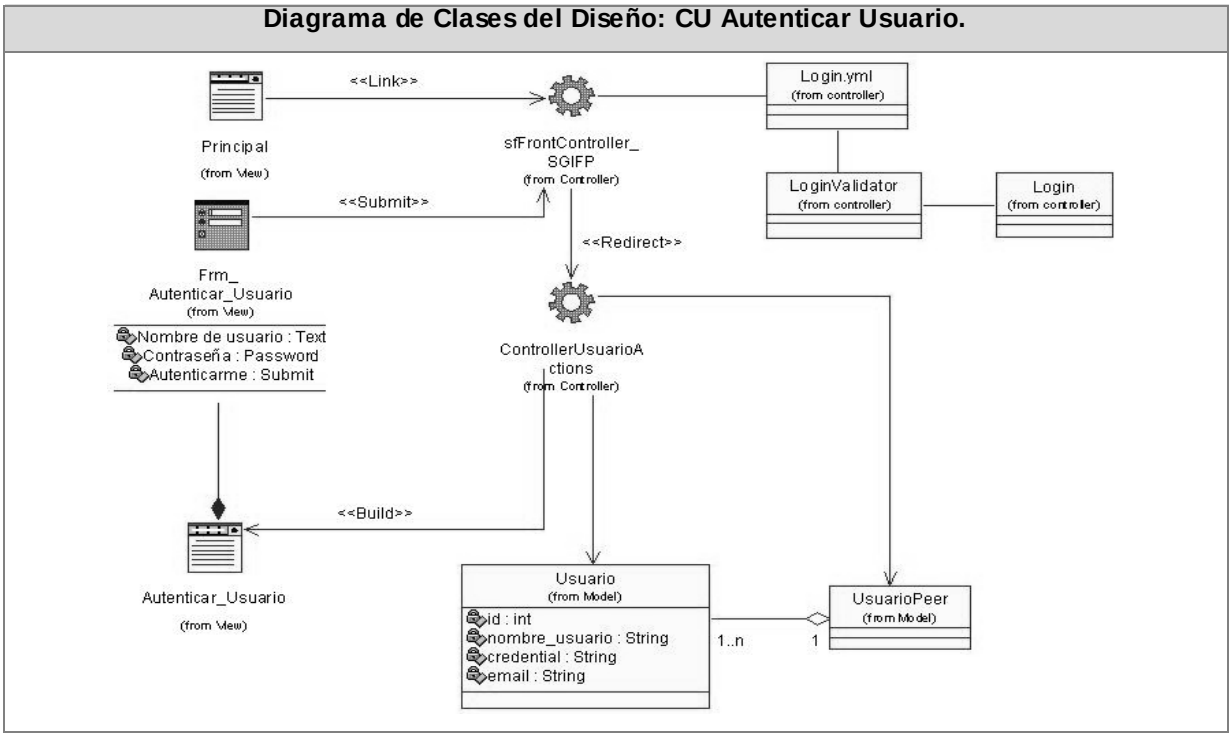

Figura 19: Diagrama de clases del diseño: CU Autenticar Usuario.

Diagrama de Clases del Diseño: CU Gestionar Proyecto Productivo: Sección Adicionar.

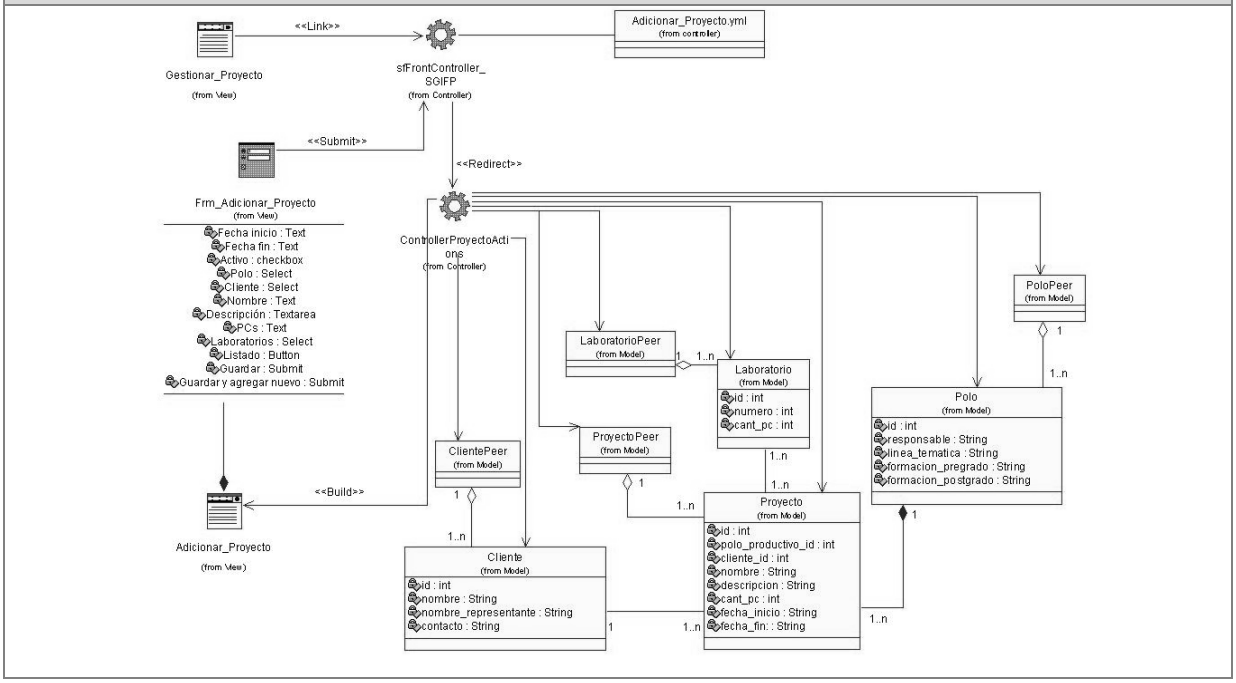

Figura 20: Diagrama de clases del diseño: CU Gestionar Proyecto Productivo. Sección Adicionar.

Diagrama de Clases del Diseño: CU Gestionar Proyecto Productivo: Sección Actualizar.

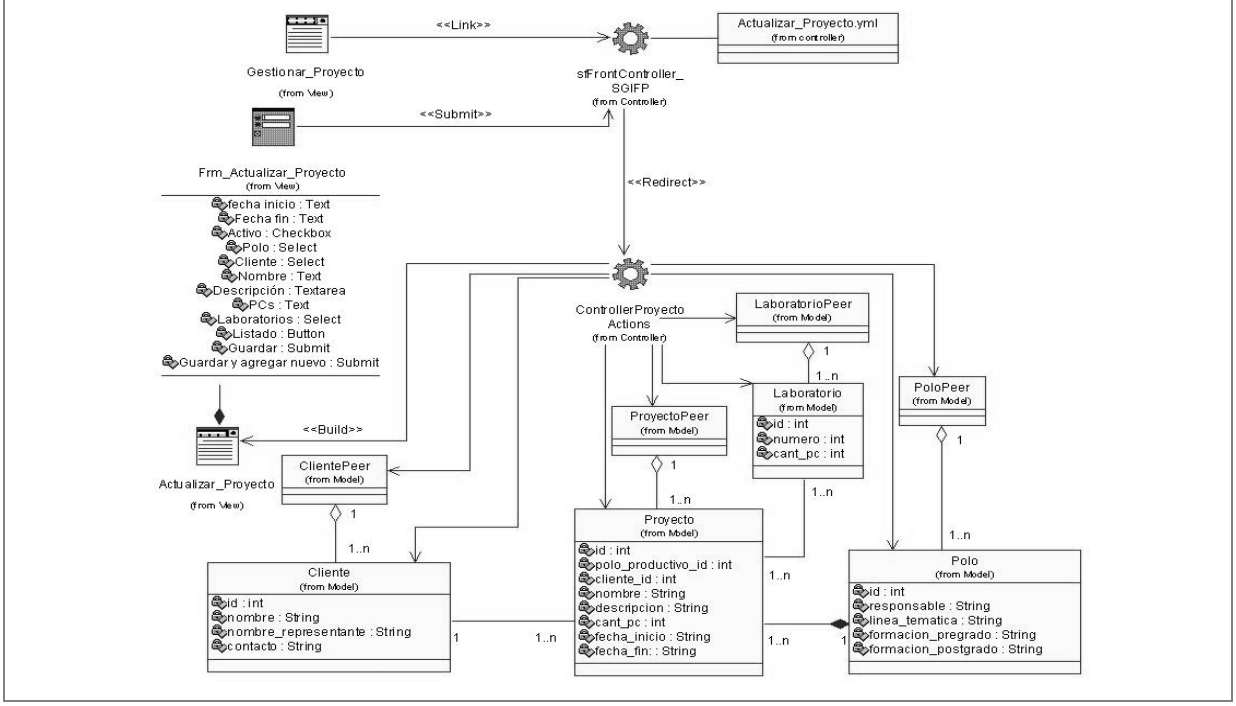

Figura 21: Diagrama de clases del diseño: CU Gestionar Proyecto Productivo. Sección Actualizar.

Diagrama de Clases del Diseño: CU Gestionar Proyecto Productivo: Sección Buscar.

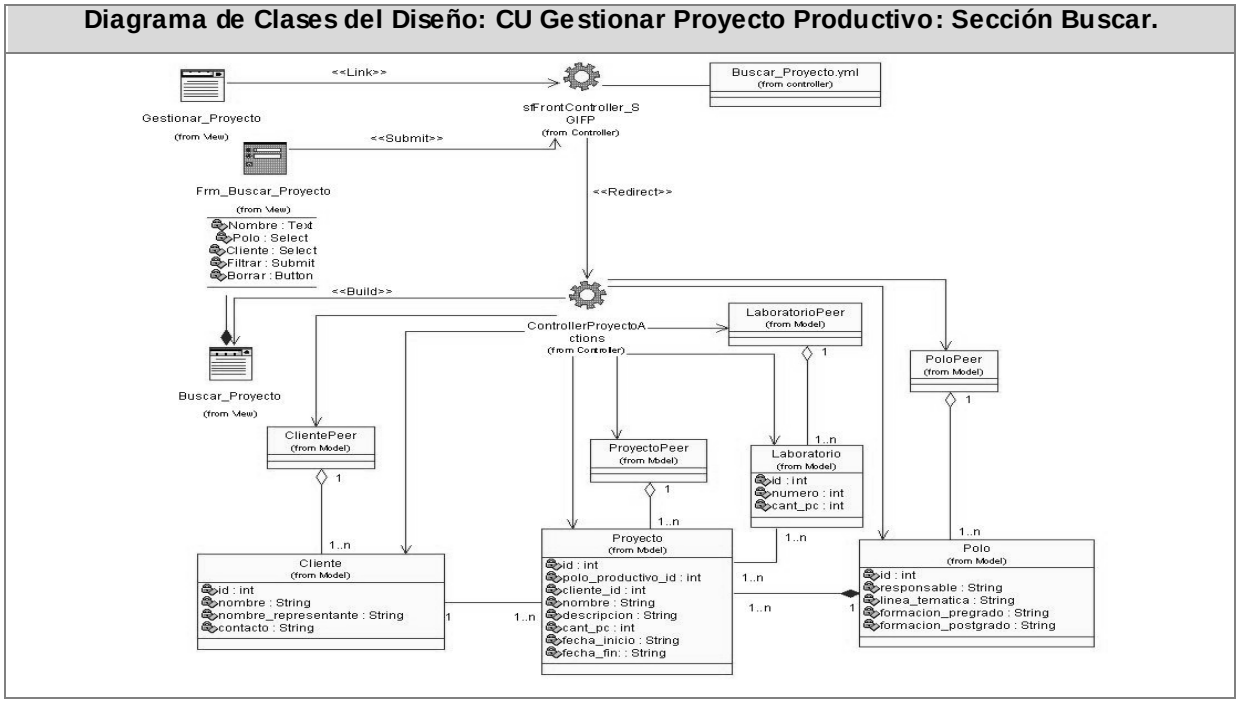

Figura 22: Diagrama de clases del diseño: CU Gestionar Proyecto Productivo. Sección Buscar.

Diagrama de Clases del Diseño: CU Gestionar Proyecto Productivo: Sección Terminar.

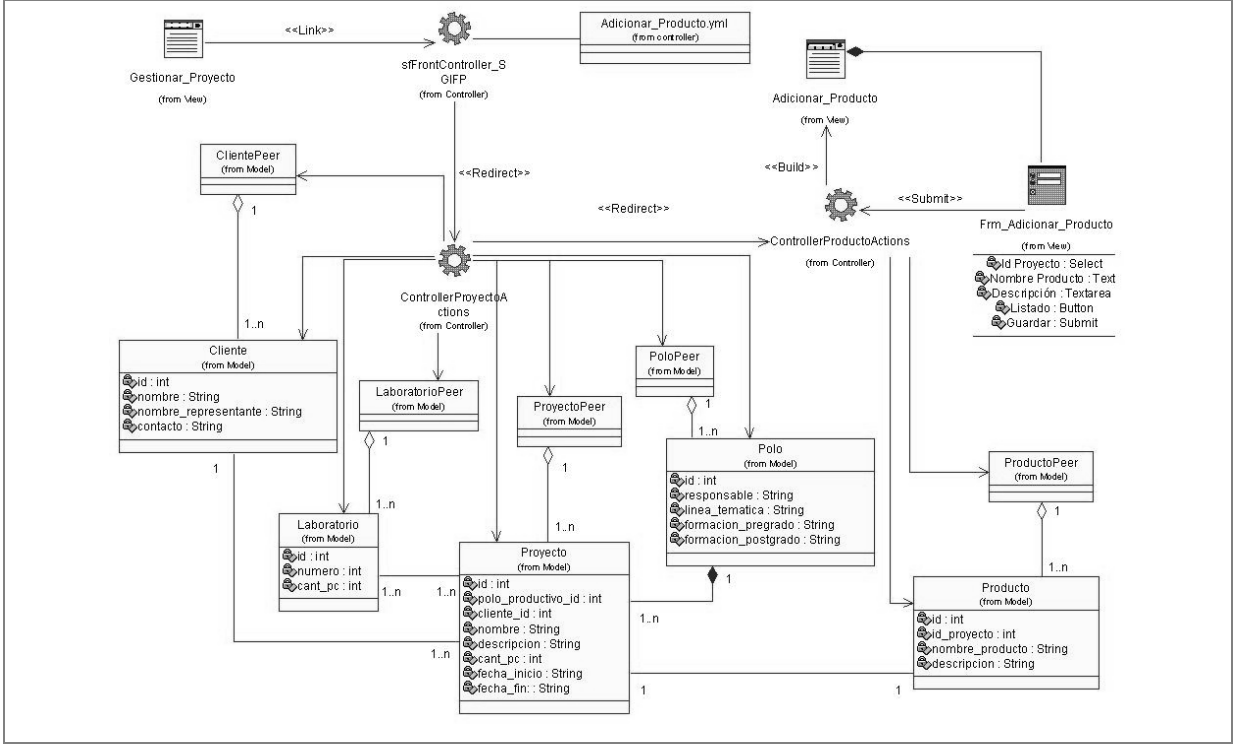

Figura 23: Diagrama de clases del diseño: CU Gestionar Proyecto Productivo. Sección Terminar.

Diagrama de Clases del Diseño: CU Gestionar Integrante Proyecto: Sección Adicionar.

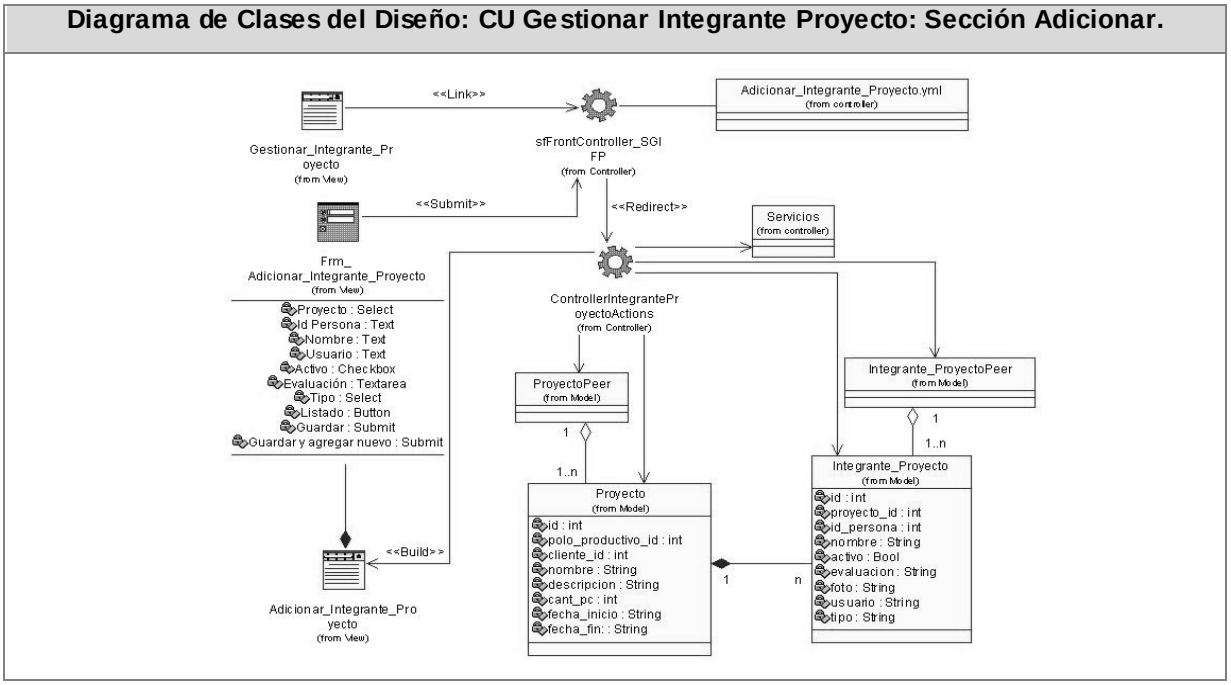

Figura 24: Diagrama de clases del diseño: CU Gestionar Integrante Proyecto. Sección Adicionar.

Diagrama de Clases del Diseño: CU Generar Reporte.

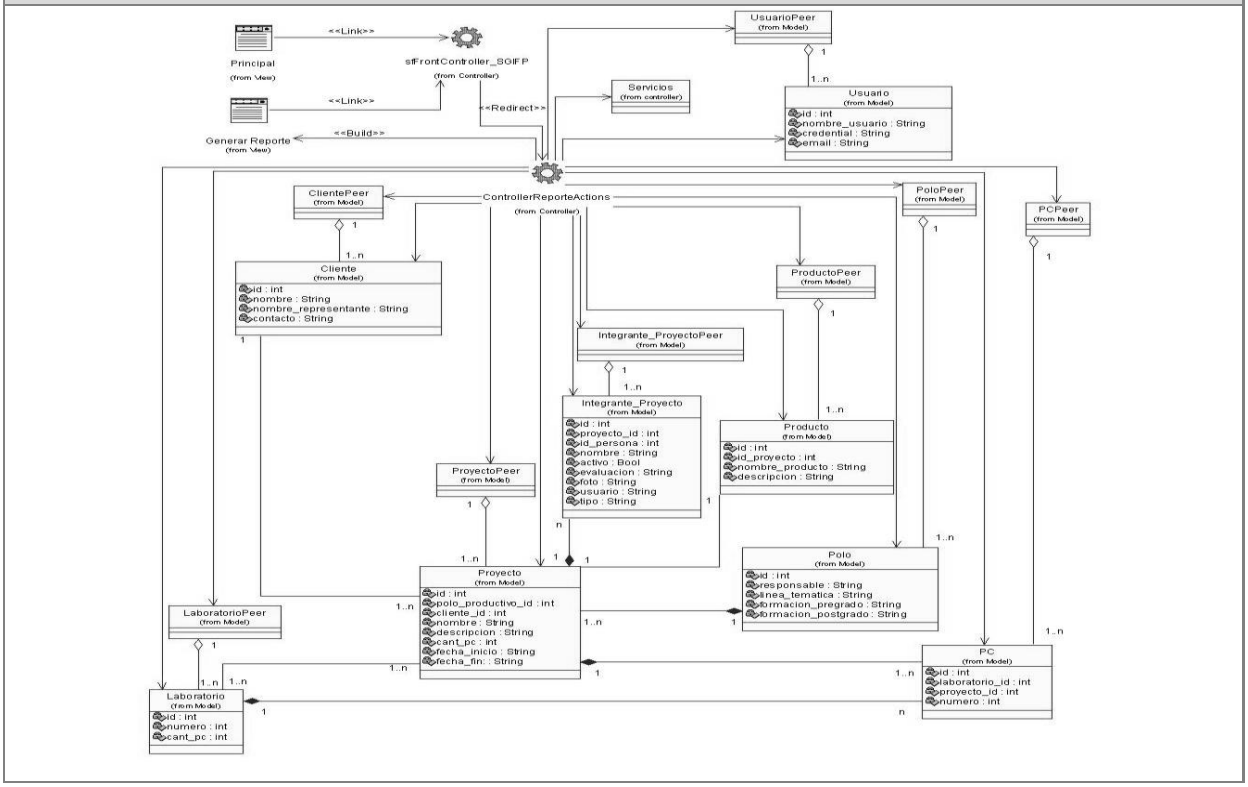

Figura 25: Diagrama de clases del diseño: CU Generar Reporte.

Nota 4: El resto de los diagramas de clases del diseño se encuentran en el **Anexo 2** al igual que las descripciones de las clases.

3.3.2 Diagramas de Interacción.

Los diagramas de secuencia y colaboración (diagramas de interacción) son diagramas de UML utilizados para modelar los aspectos dinámicos de un sistema y para construir sistemas ejecutables por medio de ingeniería directa e inversa.

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes y muestra interacciones basados en tiempo entre los objetos. Gráficamente, es una tabla que representa objetos, dispuestos a lo largo del eje X, y mensajes, ordenados según se suceden en el tiempo, a lo largo del eje Y.

Nota 5: Los diagramas de secuencia se muestran en el Anexo 2.

3.3.3 Diseño de la Base de Datos.

Existen distintos modos de organizar la información y representar las relaciones entre los datos en una base de datos. Los sistemas administradores de bases de datos convencionales usan uno de los tres modelos lógicos de bases de datos para hacer seguimiento de las entidades, atributos y relaciones. Los tres modelos lógicos principalmente de bases de datos son el jerárquico, de redes y el relacional. Cada modelo lógico tiene ciertas ventajas de procesamiento y también ciertas ventajas de negocios.

3.3.3.1 Diagrama de clases persistentes.

Un diagrama de clase persistentes es un tipo de diagrama que muestra un conjunto de objetos capaces de almacenar su estado en un medio permanente como una base de datos relacional.

A continuación se muestra el diagrama de clases persistentes del sistema:

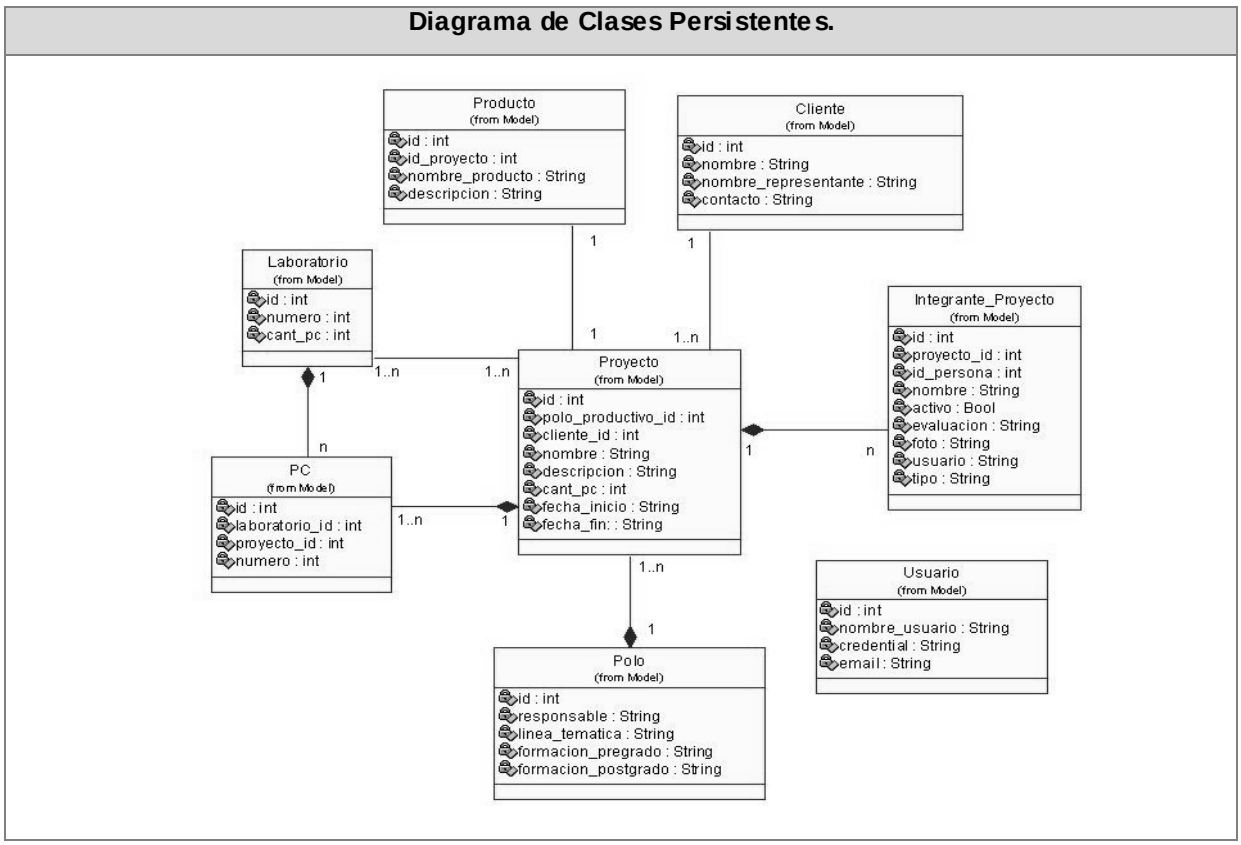

Figura 26: Diagrama de clases persistentes.

3.3.3.2 Modelo entidad relación.

El modelo entidad-relación (E-R) es uno de los varios modelos conceptuales existentes para el diseño de bases de datos. Se basa en una percepción de un mundo real. Los elementos esenciales del modelo son las entidades, los atributos y las relaciones entre las entidades. Su propósito es simplificar el diseño de bases de datos a partir de descripciones textuales de los requerimientos. Está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido.

A continuación se muestra el modelo entidad relación del sistema:

Modelo Entidad Relación.

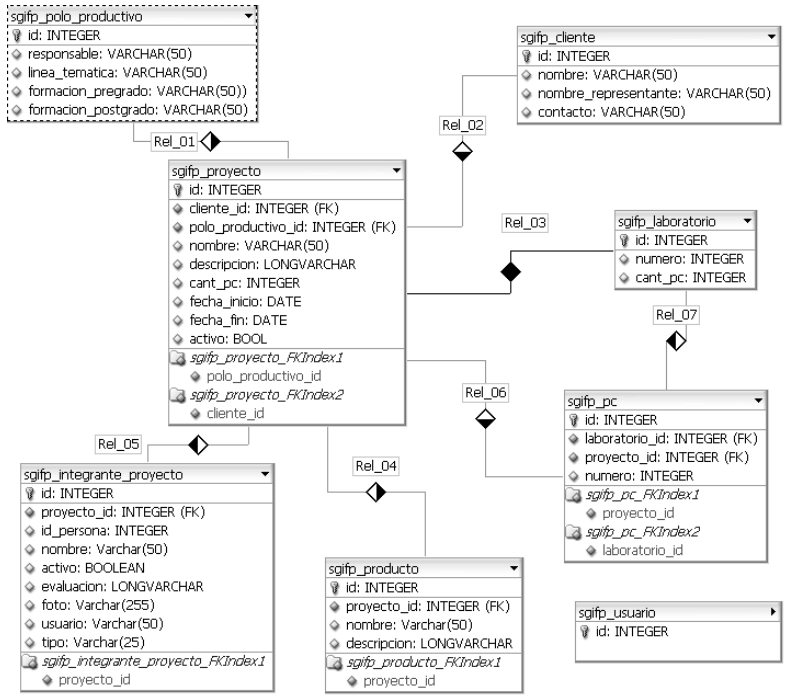

Figura 27: Modelo entidad relación.

3.3.3.3 Descripción de las tablas de la base de datos.

Nombre: sgif_usuario.		
Descripción: Almacena los datos de los usuarios del sistema.		
Atributo	**Tipo**	**Descripción**
id	Integer	El identificador del usuario.
nombre_usuario	Varchar	El usuario.
credential	Varchar	El tipo de permiso del usuario al sistema.
email	Varchar	Correo del usuario.

Tabla 12: sgif_usuario.

Nombre: sgifp_proyecto.		
Descripción: Almacena los datos de los proyectos de la facultad.		
Atributo	**Tipo**	**Descripción**
Id	Integer	El identificador del proyecto.
polo_productivo_id	Integer	El identificador del polo al que pertenece el proyecto.
cliente_id	Integer	El identificador del cliente del proyecto.
nombre	Varchar	Nombre del proyecto.
descripción	longvarchar	Descripción del proyecto.
cant_PC	Integer	Cantidad de PCs asignadas al proyecto.
fecha_inicio	Date	Fecha en que comienza el proyecto.
fecha_fin	Date	Fecha en que termina el proyecto.

Tabla 13: sgifp_proyecto.

Nombre: sgifp_polo_productivo.		
Descripción: Almacena los datos de los polos productivos existentes en la facultad.		
Atributo	**Tipo**	**Descripción**
id	Integer	El identificador del polo productivo.
responsable	Varchar	El responsable del polo productivo.
línea _temática	Varchar	El tema del polo productivo. Sobre que trata.
Formación_pregrado	Varchar	Si en el polo se imparten cursos de pregrado.
Formación_postgrado	Varchar	Si en el polo se imparten cursos de postgrado.

Tabla 14: sgifp_polo_productivo.

Nombre: sgifp_cliente.		
Descripción: Almacena los datos de los clientes de los proyectos.		
Atributo	**Tipo**	**Descripción**
id	Integer	El identificador del cliente.
nombre	Varchar	Generalmente es el nombre de una entidad, etc.
nombre_representante	Varchar	Nombre de la persona que viene representando a la entidad.
contacto	Varchar	Correo, dirección, teléfono, etc.

Tabla 15: sgifp_cliente.

Nombre: sgifp_integrante_proyecto.		
Descripción: Almacena los datos de los integrantes de los proyectos.		
Atributo	**Tipo**	**Descripción**
id	Integer	El identificador de la persona que está integrada a un proyecto.
proyecto_id	Integer	El identificador del proyecto al cual pertenece la persona.
id_persona	Integer	El identificador de la persona que pertenece a un proyecto.
nombre	Varchar	El nombre de la persona.
activo	Boolean	Si la persona está activa o no en el proyecto.
evaluación	Longvarchar	Evaluación de la persona en el proyecto.
foto	Varchar	Identificador de la foto de la persona.
usuario	Varchar	El usuario.
tipo	Varchar	Estudiante o profesor.

Tabla 16: sgifp_integrante_proyecto.

Nombre: sgifp_PC.		
Descripción: Almacena los datos de las PCs que pertenecen a un proyecto.		
Atributo	**Tipo**	**Descripción**
id	Integer	El identificador de la PC.
laboratorio_id	Integer	El identificador del laboratorio en el que está ubicada la PC.
proyecto_id	Integer	El Identificador del proyecto al cual está asignada la PC.
numero	Integer	Número de la PC.

Tabla 17: sgifp_PC.

Nombre: sgifp_laboratorio.		
Descripción: Almacena los datos de los laboratorios de la facultad.		
Atributo	**Tipo**	**Descripción**
id	integer	El identificador del laboratorio.
número	integer	El número del laboratorio.
cant_PC	integer	La cantidad de PCs que tiene el laboratorio.

Tabla 18: sgifp_laboratorio.

Nombre: sgifp_laboratorio_proyecto.		
Descripción: Almacena los datos de los laboratorios que pertenecen a algún proyecto.		
Atributo	**Tipo**	**Descripción**
id	integer	El identificador del laboratorio de proyecto.
proyecto_id	integer	El identificador del proyecto al que es asignada la PC.
laboratorio_id	integer	El identificador del proyecto al que pertenece la PC.

Tabla 19: sgifp_laboratorio_proyecto.

Nombre: sgifp_producto.		
Descripción: Almacena los datos de los productos.		
Atributo	Tipo	Descripción
Id_proyecto	Integer	El identificador del producto.
nombre_producto	Varchar	El nombre del producto.
descripción	longvarchar	La descripción del producto.

Tabla 20: sgifp_producto.

3.3.4 Definiciones de diseño que se apliquen.

Symfony cumple con una gran gama de patrones de diseño que lo hacen muy robusto tanto en su código fuente como en las aplicaciones que se implementan con él. A continuación se presentan una serie de patrones utilizados en el diseño de la aplicación SGIFP.

Experto: Este patrón define que la responsabilidad de creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Esto se manifiesta ya que en la implementación del modelo-vista-controlador hecha por el framework la capa del controlador se divide en dos capas, la del controlador frontal que se encarga mediante la clase sfController de decodificar la petición y transferirla hacia el controlador de la de la acción correspondiente, el cual se encuentra en el módulo que maneja la información relacionada en la clase nombreModuloActions.

La capa del modelo, la cual se divide también en capa de acceso a datos manejada por Propel a través de las clases del modelo que encapsula toda la lógica de datos y la capa de abstracción de datos manejada por Creole encargada de manipular la conexión.
Y por ultimo la capa de vista, en la cual se manejan los datos referentes a la petición asi como la respuesta, de lo cual se encargan las clases sfRequest y sfResponse.

Creador: Este patrón nos ayuda a identificar quien es el responsable de la creación de una instancia. El mismo se observa en la clase del controlador del modulo nombreModuloActions, en el cual se encuentran definidas las acciones correspondientes a dicho módulo. En estas acciones se definen los objetos necesarios para realizar la lógica particular de la acción.

Alta cohesión: La información de una clase debe ser coherente y estar relacionada con la clase. La clase nombreModuloActions agrupa las acciones relacionadas con el fin particular del modulo. En ellas se definen variables para las plantillas y funcionalidades con un propósito a fin, lo que les da cierta independencia permitiendo asi mayor flexibilidad ante los cambios y una mayor usabilidad y reutilización.

Front-Controller: Todas las peticiones deben ser manejadas por un único controlador frontal, que es el punto de entrada a la aplicación. Este es el caso de front-controller definido por Symfony para cada entorno. Quien se encarga de recibir y delegar al controlador del modulo todas las peticiones asi como de mostrar las respuestas de dichas peticiones.

Singleton: En Symfony existe un Singleton del contexto mediante el cual se puede acceder a todos los objetos del núcleo del Framework a través del método getContext () en las acciones y de esta forma obtener cualquier tipo de información relacionada con la petición.

Decorator: Este patrón se manifiesta en la capa de la vista la cual se compone de un layout general que se puede definir para la aplicación entera o para un determinado modulo y que decora o que se compone en su interior de las plantillas.

Adapter: Symfony implementa el este patrón ya que es posible cambiar en cualquier de base de datos, simplemente configurando unos archivos y este cambio no seria problema para las clases que usan el modelo, ya que estas ultimas brindan una interfaz común independientemente del ambiente en el cual se ejecute.

3.3.5 Tratamiento de errores.

Para garantizar un correcto funcionamiento de cualquier sistema es muy importante identificar y controlar los posibles errores que se puedan producir al interactuar con él.

Symfony incluye un completo sistema de validación de errores, ofreciendo la posibilidad de realizar este proceso de múltiples formas. La vía seleccionada para el tratamiento de errores del sistema SGIFP fue la utilización de archivos YML, en vez de usar código PHP en la acción. Este método es mas factible, al ser reutilizable y mas sencillo de programar, ya que solo se realizan configuraciones. De esta manera sólo se efectúa la validación del lado del servidor, siendo esta la más importante. No se realiza la validación del lado del cliente.

3.3.6 Seguridad.

La seguridad es de suma importancia en todo sistema para proteger la información que en él se almacena de accesos no autorizados. Se requiere que cada usuario ingrese a la aplicación con los permisos que posee y que de esta misma forma haga su uso. Para garantizar que determinada información presente en el sitio sólo se muestre a los usuarios registrados y con autorización previa, se ha programado un módulo encargado de la seguridad. La implementación del mismo se ha hecho aprovechando las posibilidades que brinda el framework Symfony, el cual permite establecer la

seguridad por niveles, basados en las credenciales del usuario, las que establecen el acceso que tiene el mismo a las diferentes funcionalidades.

También se utiliza LDAP para la autenticación obteniéndose un sistema mas centralizado ya que se integra a los servicios que se brindan en la Universidad. Aprovechando una vez mas las posibilidades que brindan las tecnologías.

3.3.7 Interfaz.

El sistema SGIFP posee una interfaz sencilla, legible sin abuso de colores, basadose principalmente en el rendimiento de la aplicación y brindando una vista agradable al usuario.

Uso de los estandares CSS para manejar la presentacion asi como un codigo html semantico y organizado. Interfaz compatible con varios navegadores diferentes como Mozilla Firefox e Internet Explorer.

3.4 Conclusiones.

Durante este capítulo se llevó a cabo el flujo de trabajo Análisis y Diseño propuesto por la metodología RUP, mediante la elaboración de cada uno de los diagramas correspondientes. Contando como resultado final del diseño, con un plano del modelo de implementación, que permite pasar a la construcción de la propuesta de solución.

Capítulo 4: Implementación y Prueba del Sistema.

4.1 Introducción.

Con el resultado del diseño alcanzado durante el desarrollo del capítulo anterior se comienza el flujo de trabajo de implementación, donde se muestra las dependencias entre las partes de código y la estructura del sistema en ejecución mediante los diagramas de componentes y despliegue respectivamente. Además en este capítulo se le realizan pruebas al sistema SGIFP.

4.2 Implementación del sistema.

El flujo de trabajo de implementación se comienza con el resultado del diseño. Describe en términos de componentes cómo son implementados los elementos del modelo de diseño y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

Los diagramas de despliegue y componentes (artefactos generados en este flujo de trabajo), conforman el modelo de implementación, al describir los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará la aplicación. El propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo.

A continuación se presenta la estructura del modelo de implementación:

Estructura del Modelo de Implementación.

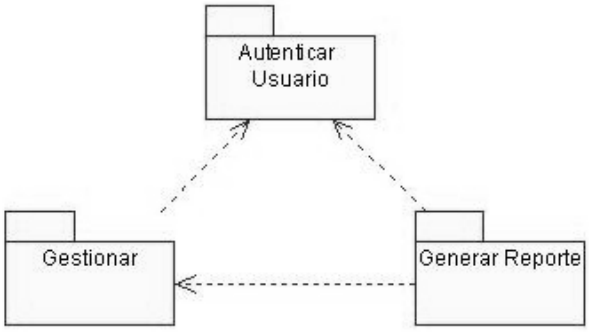

Figura 28: Estructura del modelo de implementación.

4.2.1 Diagrama de despliegue.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación que muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Se compone por nodos, dispositivos y conectores; donde los nodos son elementos de procesamiento con al menos un procesador, memoria, etc.; los dispositivos son nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela y los conectores expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

Mediante el diagrama de despliegue se captura la configuración de los elementos de procesamiento y sus conexiones y se visualiza la distribución de los componentes de software en los nodos físicos.

A continuación se muestra el diagrama de despliegue:

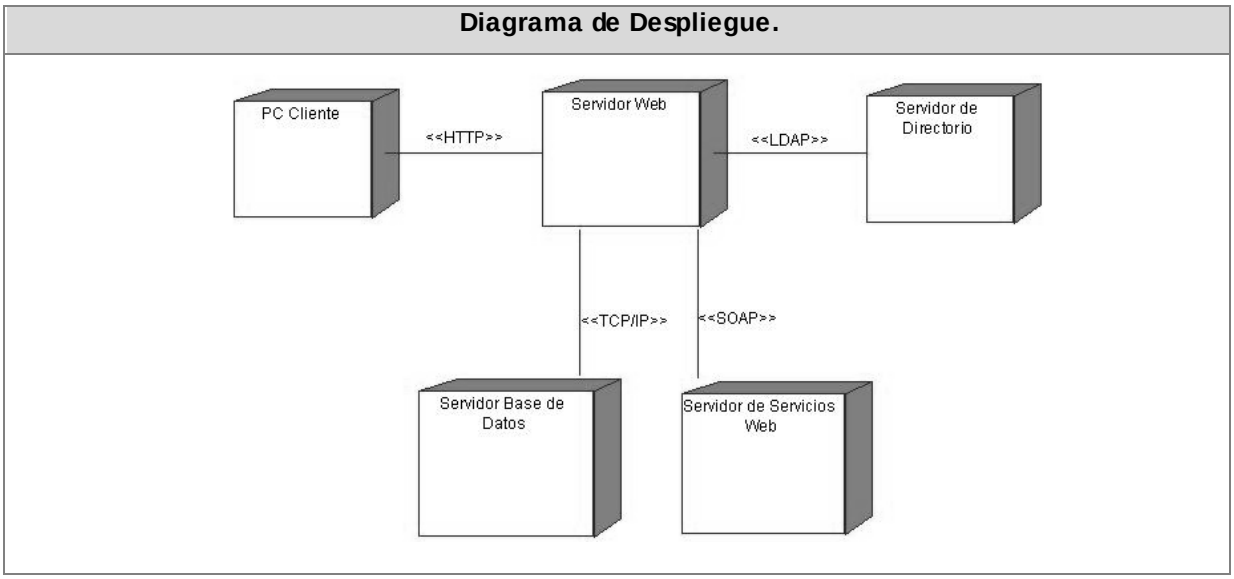

Figura 29: Diagrama de despliegue.

4.2.2 Diagramas de Componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. [15]

A continuación se muestran los diagramas de componentes:

Diagrama de Componentes: CU Autenticar Usuario.

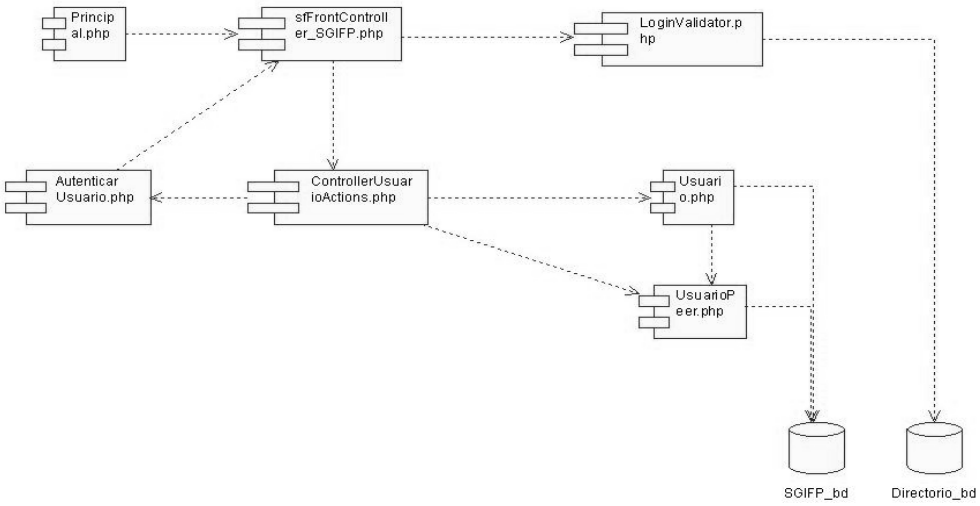

Figura 30: Diagrama de componentes: CU Autenticar Usuario.

Diagrama de Componentes: CU Gestionar Proyecto Productivo.

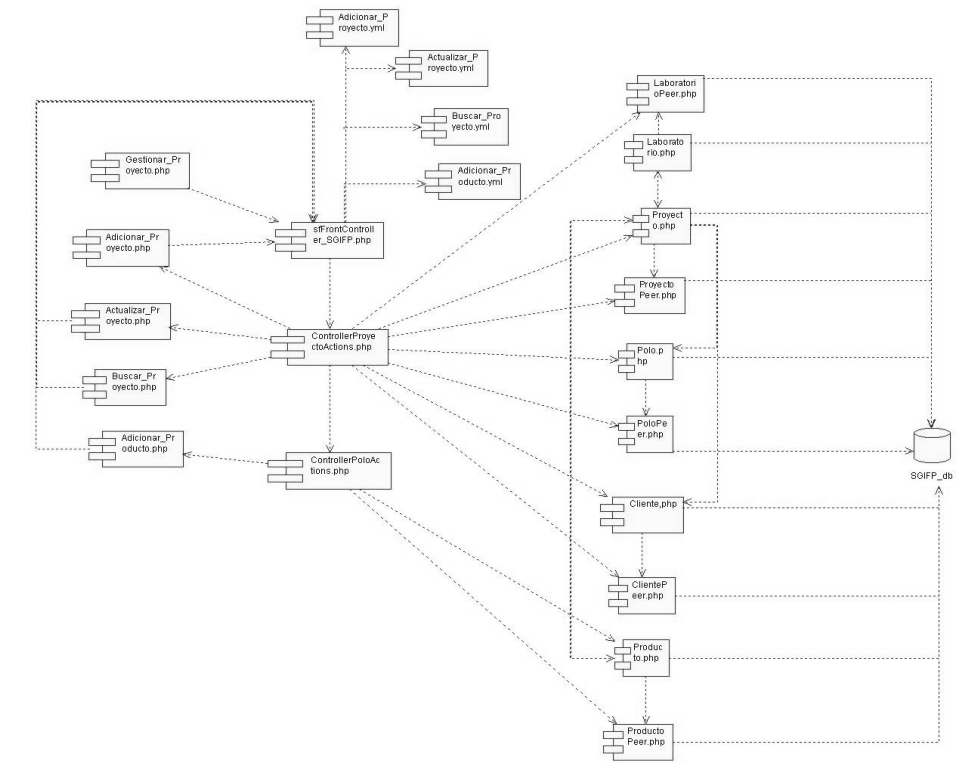

Figura 31: Diagrama de componentes: CU Gestionar Proyecto Productivo.

Diagrama de Componentes: CU Gestionar Integrante Proyecto.

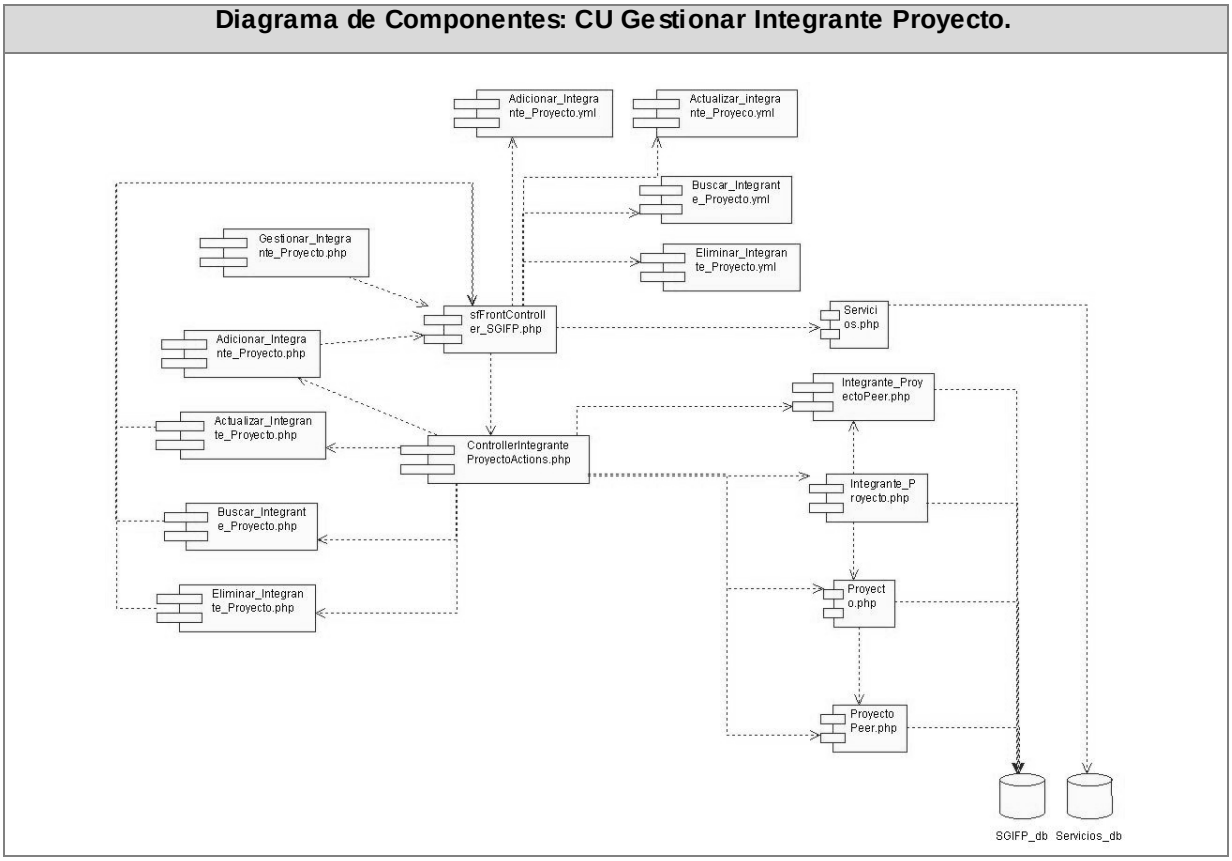

Figura 32: Diagrama de componentes: CU Gestionar Integrante Proyecto.

Nota 6: El resto de los diagramas de componentes se muestran en el **Anexo 3.**

4.3 Prueba.

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

Cualquier producto de ingeniería puede ser probado de una de estas formas:

- **Pruebas de Caja Negra:**

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

- **Pruebas de Caja Blanca:**

Conociendo el funcionamiento del producto se pueden desarrollar las pruebas de Caja Blanca que aseguren que "todas las piezas encajen", o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado deforma adecuada.

4.3.1 Pruebas de Caja Negra:

Las pruebas de Caja Negra son las que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

Estas pruebas examinan algunos aspectos del modelo, fundamentalmente del sistema, sin tener mucho en cuenta la estructura interna del software. Se centran principalmente en los requisitos funcionales del software. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Verifican las especificaciones funcionales y no consideran la estructura interna del programa. No validan funciones ocultas, como es el caso de las funciones que son implementadas pero no descritas durante la fase de diseño, por lo que los errores asociados a ellas, no serán encontrados. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Son realizadas sin el conocimiento interno del producto.

A continuación se presentan las pruebas de Caja Negra realizadas a cada caso de uso del sistema:

Tabla 21: Pruebas de Caja Negra para el caso de uso "Autenticar Usuario":

Condición de Entrada.	Casos Válidos.	Casos no Válidos.
Usuario	Cadena de caracteres.	Dejar vacío el campo usuario.
Contraseña	Cadena de caracteres.	Dejar vacío el campo contraseña.

Caso de Uso:	Autenticar Usuario.
Caso de Prueba:	Permitir autenticarse a un usuario que introduzca correctamente los datos.
Entrada:	El usuario introduce correctamente los datos necesarios para acceder al sistema. Ejemplo:

	Usuario: “vicedecano” Contraseña: “vicedecano_producción”
Resultado:	El sistema le da acceso al usuario autenticado.
Condiciones:	Usuario y contraseña introducidos correctamente.

Caso de Uso:	**Autenticar Usuario.**
Caso de Prueba:	Permitir autenticarse a un usuario que introduzca incorrectamente los datos.
Entrada:	El usuario introduce incorrectamente alguno de los datos necesarios para acceder al sistema o deja alguno de los campos vacío. Ejemplo: Usuario: “vicedecano” Usuario: ***“campo vacío”***
Resultado:	El sistema muestra un mensaje de error: (Llenar todos los campos correctamente.).
Condiciones:	Algún dato introducido incorrectamente. Algún campo vacío.

Tabla 22: Pruebas de Caja Negra para el caso de uso “Gestionar Proyecto Productivo”: Sección: Adicionar proyecto productivo:

Condición de Entrada.	Casos Válidos.	Casos no Válidos.
polo_productivo_id	Número	Ninguno (select).
cliente_id	Número	Ninguno (select).
nombre	Letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Números.
descripción	Letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Letras.
cant_pc	Números	Dejar el campo vacío. Caracteres extraños como: @ *

		; : / etc. Letras.
fecha_inicio	Formato: aaaa-mm-dd	Ninguno (calendario).
fecha_fin	Formato: aaaa-mm-dd	Ninguno (calendario).

Caso de Uso:	**Adicionar proyecto productivo.**
Caso de Prueba:	Adicionar un proyecto introduciendo correctamente todos los datos.
Entrada:	El Vicedecano de Producción introduce correctamente todos los datos requeridos para adicionar un proyecto: Ejemplo: polo_productivo_id: "1" cliente_id: "1" nombre: "Software Educativo" descripción: "Proyecto para la República Bolivariana de Venezuela." cant_pc: "20" fecha_inicio: "2008-05-21" fecha_fin: "2008-12-21"
Resultado:	El sistema adiciona el proyecto en la base de datos y muestra un mensaje indicando la acción realizada ("Proyecto adicionada correctamente").
Condiciones:	Los datos introducidos para adicionar correctamente un proyecto deben estar en el rango de los valores permitidos.

Caso de Uso:	**Adicionar proyecto productivo.**
Caso de Prueba:	Adicionar un proyecto introduciendo incorrectamente algún dato.
Entrada:	El Vicedecano de Producción introduce incorrectamente algún dato de los requeridos para adicionar un proyecto: Ejemplo: polo_productivo_id: "1" cliente_id: "1"

	nombre: "S**0**ftw@re**/**Educ@tivo" descripción: ***"campo vacío"*** cant_pc: ***"veinte"*** fecha_inicio: "2008-05-21" fecha_fin: "2008-12-21"
Resultado:	El sistema muestra un mensaje de error indicando la acción realizada incorrectamente ("Datos incorrectos").
Condiciones:	Algún dato introducido incorrectamente. Algún campo vacío.

Tabla 23: Pruebas de Caja Negra para el caso de uso "Gestionar Proyecto Productivo": Sección: Actualizar proyecto productivo:

Condición de Entrada.	Casos Válidos.	Casos no Válidos.
polo_productivo_id	Número	Ninguno (select).
cliente_id	Número	Ninguno (select).
nombre	Letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Números.
descripción	Letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Letras.
cant_pc	Números	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Letras.
fecha_inicio	Formato: aaaa-mm-dd	Ninguno (calendario).
fecha_fin	Formato: aaaa-mm-dd	Ninguno (calendario).

Caso de Uso:	**Actualizar proyecto productivo.**
Caso de Prueba:	Actualizar un proyecto introduciendo correctamente todos los datos.
Entrada:	El Vicedecano de Producción introduce correctamente todos los datos requeridos para actualizar un proyecto: Ejemplo: polo_productivo_id: "1" cliente_id: "1" nombre: "Software Educativo" descripción: "Proyecto para Cuba." cant_pc: "22" fecha_inicio: "2008-05-21" fecha_fin: "2008-12-21"
Resultado:	El sistema actualiza un proyecto en la base de datos y muestra un mensaje indicando la acción realizada ("Proyecto actualizado correctamente").
Condiciones:	Los datos introducidos para actualizar correctamente un proyecto deben estar en el rango de los valores permitidos.
Caso de Uso:	**Actualizar proyecto productivo.**
Caso de Prueba:	Actualizar un proyecto introduciendo incorrectamente algún dato.
Entrada:	El Vicedecano de Producción introduce incorrectamente algún dato de los requeridos para actualizar un proyecto: Ejemplo: polo_productivo_id: "1" cliente_id: "1" nombre: "Software Educativo" descripción: ***"campo vacío"*** cant_pc: ***"veintidós"*** fecha_inicio: "2008-05-21" fecha_fin: "2008-12-21"
Resultado:	El sistema muestra un mensaje de error indicando la acción realizada incorrectamente ("Datos incorrectos").

Condiciones:	Algún dato introducido incorrectamente. Algún campo vacío.

Tabla 24: Pruebas de Caja Negra para el caso de uso "Gestionar Proyecto Productivo": Sección: Terminar proyecto productivo:

Condición de Entrada.	Casos Válidos.	Casos no Válidos.
Id_proyecto	Número	Ninguno (select).
nombre_producto	Letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Números.
descripción	Letras	Dejar el campo vacío. Caracteres extraños como: @ * ; : / etc. Letras.

Caso de Uso:	**Terminar proyecto productivo.**
Caso de Prueba:	Adicionar un producto introduciendo correctamente todos los datos.
Entrada:	El Vicedecano de Producción introduce correctamente todos los datos requeridos para adicionar un producto: Ejemplo: Id_proyecto: "1" nombre_producto: "CNTI" descripción: "Proyecto de suma importancia para la educación en Cuba."
Resultado:	El sistema adiciona el producto en la base de datos y muestra un mensaje indicando la acción realizada ("Producto adicionada correctamente").
Condiciones:	Los datos introducidos para adicionar correctamente un producto deben estar en el rango de los valores permitidos.

Caso de Uso:	**Terminar proyecto productivo.**
Caso de Prueba:	Adicionar un producto introduciendo incorrectamente algún dato.
Entrada:	El Vicedecano de Producción introduce incorrectamente algún dato de los requeridos para adicionar un producto: Ejemplo: Id_proyecto: "1" nombre_producto: ***"campo vacío"*** descripción: "Proyecto de suma importancia para la educación en Cuba."
Resultado:	El sistema muestra un mensaje de error indicando la acción realizada incorrectamente ("Datos incorrectos").
Condiciones:	Algún dato introducido incorrectamente. Algún campo vacío.

Nota 7: El resto de las pruebas realizadas al sistema se pueden observar en el **Anexo 3.**

4.4 Conclusiones.

Al concluir el presente capítulo, se cuenta con los diagramas de despliegue y componentes, así como los resultados arrojados de las pruebas realizadas al sistema que se propone.

Capítulo 5: Estudio de Factibilidad.

5.1 Introducción.

El objetivo fundamental de la estimación, es determinar la posibilidad de llevar adelante el proyecto (el estudio de la factibilidad), de acuerdo a diferentes restricciones, dadas por características propias del mismo, como pueden ser: equipo, organizativas, económicas, técnicas, de tiempo, etc.
En este capítulo se estima el esfuerzo y se analizan los beneficios del sistema propuesto (SGIFP), utilizando el método de estimación por Puntos de Casos de Uso. Obteniendo importantes indicadores como son: esfuerzo y tiempo requerido de desarrollo.

5.2 Estimación de Esfuerzo:

La estimación mediante el análisis de Puntos de Casos de Uso es un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

A continuación, se realizará la estimación del Sistema para la Gestión de Información de los Proyectos Productivos de la Facultad 8 (SGIFP).

5.2.1 Paso 1. Identificar los Puntos de Casos de Uso Desajustados.

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar.
UAW: Factor de Peso de los Actores sin ajustar.
UUCW: Factor de Peso de los Casos de Uso sin ajustar.

Factor de Peso de los Actores sin Ajustar (UAW).

-Los actores del sistema se clasifican en:

Simple: Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API).

Medio: Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.

Complejo: Una persona que interactúa con el sistema mediante una interfaz gráfica.

-El factor de peso de cada actor está dado por su clasificación:

- ✓ Para un actor simple el factor de peso es 1.
- ✓ Para un actor medio el factor de peso es 2.
- ✓ Para un actor complejo el factor de peso es 3.

El sistema SGIFP cuenta con dos actores (Vicedecana de Producción y usuario). Estas son personas que interactúan con el sistema mediante una interfaz gráfica. Por tanto estos actores se clasifican en: Complejos y poseen factor de peso 3.

Para calcular UAW:

Tipo de Actor.	Descripción.	Peso.	Cant. Actores.	Cant. Actores * Peso.	Resultado.
Simple.	Sistema que interactúa con el SGIFP mediante un API.	1	0	1*0	0
Medio.	Sistema que interactúa con el SGIFP mediante un protocolo o una interfaz basada en texto.	2	0	2*0	0
Complejo.	Persona que interactúa con el sistema mediante una interfaz gráfica.	3	2	3*2	6
Total:					**6**

Tabla 25: Factor de peso de los actores sin ajustar.

Respuesta: UAW=6

Factor de Peso de los Casos de Uso sin ajustar (UUCW).

-Los casos de uso del sistema se clasifican en:

Simple: El Caso de Uso contiene de 1 a 3 transacciones.

Medio: El Caso de Uso contiene de 4 a 7 transacciones.

Complejo: El Caso de Uso contiene más de 8 transacciones.

-El factor de peso de cada caso de uso está dado por su clasificación:

- ✓ Para un caso de uso simple el factor de peso es 5.
- ✓ Para un caso de uso medio el factor de peso es 10.
- ✓ Para un caso de uso complejo el factor de peso es 15.

Nombre de los casos de uso del SGIFP----------Cantidad de transacciones.

1. Autenticar Usuario----------3
2. Generar Reporte----------3
3. Gestionar Proyecto Productivo----------6
4. Gestionar Polo Productivo----------6
5. Gestionar Usuario----------6
6. Gestionar Integrante Proyecto----------6
7. Gestionar Cliente----------6
8. Gestionar Laboratorio----------6
9. Gestionar PC----------6

Para calcular UUCW:

Tipo de Caso de Uso.	Descripción.	Peso.	Cant. de Casos de uso de ese tipo.	Cant. de Casos de uso de ese tipo * Peso.	Resultado.
Simple	De 1-3 transacciones.	5	2	5*2	10

Medio	De 4-7 transacciones.	10	8	10*7	70
Complejo	De 8 transacciones en adelante.	15	0	15*0	0
Total					**80**

Tabla 26: Factor de peso de los casos de uso sin ajustar.

Calculando UUCP:

UUCP = UAW + UUCW

UUCP= 6+80=86

Resultado del Paso 1: UUCP =86

5.2.2 Paso 2. Ajustar los Puntos de Casos de Uso.

UCP = UUCP * TCF * EF

Donde:

UCP: Puntos de Casos de Uso ajustados.
UUCP: Puntos de Casos de Uso sin ajustar.
TCF: Factor de complejidad técnica.
EF: Factor de ambiente.

Factor de complejidad técnica (TCF).

Para Calcular TCF:

TCF = 0.6 + 0.01 * Σ (Pesoi * Valori)

Donde:

Valor: es un número del 0 al 5.

-Significado de los valores:

0: No presente o sin influencia.
1: Influencia incidental o presencia incidental.

2: Influencia moderada o presencia moderada.

3: Influencia media o presencia media.

4: Influencia significativa o presencia significativa.

5: Fuerte influencia o fuerte presencia.

Factor	Descripción	Peso	Valor	Comentario	Σ (Pesoi * Valori)
T1	Sistema distribuido.	2	0	Sistema centralizado.	0
T2	Objetivos de performance o tiempo de respuesta.	1	4	Velocidad bastante rápida. Uso de caché.	4
T3	Eficiencia del usuario final.	1	4	Necesidad de eficiencia.	4
T4	Procesamiento interno complejo.	1	0	No hay cálculos complejos.	0
T5	El código debe ser reutilizable.	1	4	Código reutilizable.	4
T6	Facilidad de instalación.	0.5	4	Debe ser fácil de instalar.	2
T7	Facilidad de uso.	0.5	4	Debe ser fácil de usar.	2
T8	Portabilidad.	2	4	Sistema portable.	8
T9	Facilidad de cambio.	1	4	Se requiere un costo de mantenimiento bajo.	4
T10	Concurrencia.	1	4	Sí.	4
T11	Incluye objetivos especiales de seguridad.	1	3	Seguridad normal.	3
T12	Provee acceso	1	4	Los usuarios	4

	directo a terceras partes.			tienen acceso directo.	
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	1	Pocos usuarios internos, sistema fácil de usar.	1
Total:					**40**

Tabla 27: Σ (Pesoi * Valori) para el factor de complejidad técnica.

TCF = 0.6 + 0.01 * Σ (Pesoi * Valori)

Respuesta: TCF= 0.6+0.01*40=1

Factor de ambiente (EF).

Para Calcular EF:

EF = 1.4 - 0.03 * Σ (Pesoi * Valori)

Donde:

Valor: es un número del 0 al 5.

Factor	Descripción	Peso	Valor	Comentario	Σ (Pesoi * Valori)
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	4	El equipo está familiarizado con el modelo.	6
E2	Experiencia en la aplicación.	0.5	4	El equipo ha trabajado en este tipo de sistema.	2
E3	Experiencia en orientación a objetos.	1	4	El equipo tiene experiencia en la	4

				programación OO.	
E4	Capacidad del analista líder.	0.5	4	Tiene conocimientos de ISW.	2
E5	Motivación.	1	5	El equipo está altamente motivado.	5
E6	Estabilidad de los requerimientos.	2	4	No se esperan cambios.	8
E7	Personal part-time.	-1	0	El equipo trabaja a tiempo completo.	0
E8	Dificultad del lenguaje de programación.	-1	2	Se programará en PHP.	-2
Total:					**25**

Tabla 28: Σ (Pesoi * Valori) para el factor ambiente.

EF = 1.4 - 0.03 * Σ (Pesoi * Valori)

Respuesta: EF=1.4-0.03*25=0.65

UCP = UUCP * TCF * EF

UCP =86*1*0.65=55.9

Resultado del Paso 2: UCP=62.4

5.2.3 Paso 3. Calcular esfuerzo de FT Implementación.

E = UCP * CF

Donde:

E: esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: factor de conversión.

Factor de conversión (CF).

CF = 20 horas-hombre (si Total EF ≤ 2)

CF = 28 horas-hombre (si Total EF = 3 ó Total EF = 4)

CF = abandonar o cambiar proyecto (si Total EF ≥ 5)

Para calcular CF:

Total $_{EF}$ = Cant EF < 3 (entre E1 –E6) **+ Cant EF > 3** (entre E7, E8)

Total $_{EF}$ =2+0=2

Respuesta: CF=20 horas-hombre (porque Total $_{EF}$ ≤ 2)

E = UCP * **CF**

E=55.9*20 horas-hombre=1118 horas-hombre

Resultado del Paso 3: E=1118 horas-hombre

5.2.4 Paso 4. Calcular esfuerzo de todo el proyecto.

-El valor del esfuerzo calculado representa el esfuerzo del Flujo de Trabajo de Implementación, por comparación salen el resto de los esfuerzo y la suma de ellos es el esfuerzo total (ET).

Actividad	% esfuerzo	Valor esfuerzo
Análisis.	10%	280 horas-hombre
Diseño.	20%	559 horas-hombre
Implementación.	40%	1118 horas-hombre
Prueba.	15%	419 horas-hombre
Sobrecarga.	15%	419 horas-hombre
Total:	100%	2795 horas-hombre

Tabla 29: Esfuerzo de los flujos de trabajo.

Un mes tiene como promedio 30 días y se supone que una persona trabaja 8 horas por día; la cantidad de horas que puede trabajar una persona en 1 mes es 240 horas.

ET = 2795 horas-hombre y por cada 240 horas se tiene 1 mes, por tanto **ET = 12 mes-hombre.**

Esto quiere decir que 1 persona puede realizar el sistema SGIFP en aproximadamente 1 año.

Como este sistema estará desarrollado por dos personas, entonces SGIFP puede llevarse a cabo aproximadamente en 6 meses.

5.3 Beneficios tangibles e intangibles.

El Sistema para la Gestión de Información de los proyectos Productivos de la Facultad 8 (SGIFP) tiene como principal objetivo controlar y organizar toda la información relacionada a los proyectos productivos que se desarrollan en la facultad 8. Por lo que contará con el contante intercambio de usuarios, de ahí su interfaz gráfica sencilla y amigable.

Los beneficios más destacables brindados por este sistema son principalmente intangibles:

- ✓ Ahorrar tiempo y facilitar el trabajo del Vicedecano de Producción.
- ✓ Posibilitar a los diferentes directivos ver importantes reportes del estado en que se encuentran los proyectos de la facultad.
- ✓ Hacer más eficiente el control de la información en el proceso productivo de la facultad.
- ✓ Facilitar el acceso a la información que se necesite relacionada al tema de la producción en la Facultad 8.

Entre los beneficios tangibles podemos mencionar la obtención del sistema SGIFP, con la finalidad de que los usuarios interesados en el tema de la producción de la facultad, como es el caso de sus directivos, puedan estar actualizados del tema mediante la opción “Reportes” que brindará el sistema y no depender así del Vicedecano de Producción para ello.

5.4 Análisis de costos y beneficios.

El desarrollo de este sistema no requiere grandes gastos de recursos, ni de tiempo; los servidores que existen en la Universidad son capaces de soportar la base de datos que contiene la información, así como el software en su totalidad. Estará desarrollado casi en su totalidad por software libre. No reportará gastos por concepto de entrenamiento a los usuarios pues será sencillo y de fácil manejo. Pondrá en las manos de la Facultad 8 de la UCI una herramienta para la gestión de sus proyectos productivos mejorando un grupo de problemas (mencionados en la introducción a este Trabajo de Diploma) existentes por la ausencia de este tipo de software en la facultad.

Teniendo en cuenta todos los beneficios antes mencionados y que el esfuerzo necesario para el desarrollo de SGIFP es de 12 mes-hombre, se considera factible el desarrollo de esta aplicación.

5.6 Conclusiones.

En este capítulo se describió el estudio de la factibilidad realizado al sistema propuesto SGIFP. Se plantearon los principales beneficios que reportará al ser implantado y se arriba a la conclusión teniendo en cuenta el costo estimado, que es factible su desarrollo.

Conclusiones Generales.

Al término del presente Trabajo de Diploma, se cumplieron satisfactoriamente todos los objetivos trazados para su desarrollo. Se investigaron los procesos de gestión de información de los proyectos productivos que se desarrollan en la Facultad 8. Se llevó a cabo una correcta selección de las herramientas que permitieron realizar el análisis, diseño e implementación del sistema y se elaboró además la documentación del mismo en el formato requerido con las características establecidas.

Recomendaciones.

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

1. Usar el plugin sfGuard de Symfony para dar un manejo más completo de la seguridad.
2. Dar la posibilidad de registrar las actividades realizadas por los usuarios una vez autenticados para alcanzar una seguridad más robusta del sistema.
3. Mejorar la apariencia de las URLs mediante el sistema routing para hacerlas más entendibles a los usuarios.
4. Ampliar el número de reportes brindados por la aplicación para proporcionar mayor información a los usuarios.

Referencias Bibliográficas.

1. *W3C WORLD WIDE WEB*. (08 de 04 de 2008). Recuperado el 18 de Marzo de 2008, de W3C: http://www.w3c.es/divulgacion/guiasbreves/serviciosWeb/
2. *masadelante.com*. (2008). Recuperado el 18 de Marzo de 2008, de masadelante.com: http://www.masadelante.com/faq-servidor.htm
3. *formación.com*. (2004). Recuperado el 18 de Marzo de 2008, de formación.com: http://www.adrformacion.com/curso/intranet/leccion1/InternetInformationServer.htm
4. Álvarez, R. (s.f.). *desarrolloWeb.com*. Recuperado el 18 de Marzo de 2008, de desarrolloWeb.com: http://www.desarrolloWeb.com/articulos/239.php
5. S., C. V. (2008). *Maestros del Web.com*. Retrieved Marzo 18, 2008, from Maestros del Web.com: http://www.maestrosdelWeb.com/editorial/phpintro/
6. Vieyra, G. E. (23 de Mayo de 2000). *Conceptos de HTML*. Recuperado el 18 de Marzo de 2008, de Conceptos de HTML: http://www.fismat.umich.mx/~elizalde/tesis/node49.html
7. *Garbage Collector*. (01 de Noviembre de 2004). Recuperado el 21 de Marzo de 2008, de Garbage Collector: http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php
8. Worsley, J. (2001). *Proyecto S.O.B.L. Traducciones.* Recuperado el 21 de Marzo de 2008, de Proyecto S.O.B.L. Traducciones.: http://www.sobl.org/traducciones/practical-postgres/node12.html
9. Worsley, J. (2001). *Proyecto S.O.B.L. Traducciones.* Recuperado el 21 de Marzo de 2008, de Proyecto S.O.B.L.Traducciones.: http://www.sobl.org/traducciones/practical-postgres/node19.html
10. Sanchez, M. A. (7 de julio de 2004). *informatizate*. Recuperado el 21 de Marzo de 2008, de informatizate: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html
11. *SPARX SYSTEMS*. (2008). Recuperado el 21 de Marzo de 2008, de SPARX SYSTEMS: http://www.sparxsystems.com.ar/new/products/ea.php

12. Curbello, F. A. (Julio de 2007). Sistema de Gestión de Informació de la Faculatad 8. Módulo de Gestión de la Residencia estudiantil. *Sistema de Gestión de Informació de la Faculatad 8. Módulo de Gestión de la Residencia estudiantil.* Ciudad Habana, Cuba.
13. González, C. S. (s.f.). *ONess.* Recuperado el 21 de Marzo de 2008, de ONess: http://oness.sourceforge.net/proyecto/html/ch03s02.html
14. *librosWeb.es.* (s.f.). Recuperado el 21 de Marzo de 2008, de librosWeb.es: http://www.librosWeb.es/symfony/capitulo1/symfony_en_pocas_palabras.html
15. Vilas, A. F. (20 de 03 de 2001). Recuperado el 18 de Mayo de 2008, de http://www-gris.det.uvigo.es/~avilas/UML/node49.html

Bibliografía.

1. Álvarez, M. Á. (s.f.). *desarrolloWeb.com*. Recuperado el 21 de Marzo de 2008, de desarrolloWeb.com: http://www.desarrolloWeb.com/articulos/25.php
2. Álvarez, R. (s.f.). *desarrolloWeb.com*. Recuperado el 18 de Marzo de 2008, de desarrolloWeb.com: http://www.desarrolloWeb.com
3. *Ciberaula*. (2006). Recuperado el 18 de Marzo de 2008, de Ciberaula: http://linux.ciberaula.com/articulo/linux_apache_intro/
4. *desarrolloWeb.com*. (s.f.). Recuperado el 21 de Marzo de 2008, de desarrolloWeb.com: http://www.desarrolloWeb.com/articulos/1325.php
5. *EasyEclipse*. (2007). Recuperado el 21 de Marzo de 2008, de EasyEclipse: http://www.easyeclipse.org/site/about/index.html
6. González, C. S. (s.f.). *ONess.* Recuperado el 21 de Marzo de 2008, de ONess: http://oness.sourceforge.net/proyecto/html/ch03s02.html
7. Ivar Jacobson, G. B. (2004). *El Proceso unificado de Desarrollo de Software Volumen 1 y 2.* La Habana: Félix Varela.
8. *librosWeb.es.* (s.f.). Recuperado el 21 de Marzo de 2008, de librosWeb.es: http://www.librosWeb.es/symfony/capitulo2/el_patron_mvc.html
9. *MAZTERMAGAZINE*. (2004). Recuperado el 18 de Marzo de 2008, de MAZTERMAGAZINE: http://www.mastermagazine.info/termino/5560.php
10. *monografias.com*. (s.f.). Recuperado el 21 de Marzo de 2008, de monografias.com: http://www.monografias.com/trabajos4/basesdatos/basesdatos.shtml
11. Pozo, S. (Marzo de 2004). *MySQL con Clase*. Recuperado el 21 de Marzo de 2008, de http://mysql.conclase.net/curso/index.php
12. Quintero, J. B. (2005). Un estudio comparativo de herramientas para el modelado con UML. *Revista Universidad de EAFIT.* , 60-75.
13. Vilas, A. F. (20 de 03 de 2001). Recuperado el 18 de Mayo de 2008, de http://www-gris.det.uvigo.es/~avilas/UML/

Glosario de Términos.

API: Applications Programming Interface / Interfaz de Programación de Aplicaciones.

BSD: Licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

CCS: Cascading Style Sheets / Hojas de Estilo en Cascada. Tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación.

CASE: Computer Aided Software / Ingeniería de Software Asistida por Computadora.

CGI: Common Gateway Interface / Interface de Acceso Común o Interfaz Común de Puerta de Enlace. Es la interfaz entre un servidor con Protocolo de Transferencia de Hipertexto (HTTP),

CUN: Casos de Uso del Negocio.

CUS: Casos de uso del Sistema.

Dirección IP: Número que identifica a cada dispositivo dentro de una red con protocolo IP.

EF: Environment Factor / Factor de Ambiente.

GPL: General Public License / Licencia Pública General. Orientada principalmente a proteger la libre distribución, modificación y uso de software.

GUI: Graphical User Interface / Interfaz Gráfica de Usuario. Programa software que gestiona la interacción con el usuario de manera gráfica mediante el uso de íconos, menu, mouse, etc.

HTML: Hypertext Markup Language. Lenguaje de marcado de hipertexto, usado para escribir documentos para servidores World Wide Web.

HTTP: Hypertext Transfer Protocol / Protocolo de Transferencia de Hipertexto. Modo de comunicación para solicitar páginas Web.

IDE: Integrated Development Environment / Entorno de Desarrollo Integrado. Entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

IIS: Internet Information Server. Servidor Web recomendable para plataforma Windows 2000, dado por la integración que presenta con su Servicio de Directorios que permite el desarrollo de aplicaciones basadas en la Web fiables y escalables.

Internet: Red de computadoras alrededor de todo el mundo, que comparten información unas con otras por medio de páginas o sitios.

Intranet: Red de computadoras dentro de una red de área local (LAN) privada, empresarial o educativa que proporciona herramientas de Internet.

IP: Internet Protocol.

LDAP: Lightweight Directory Access Protocol. Es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. Habitualmente, almacena la información de login (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información.

Logs: Son archivos en los que se recogen las visitas que tienen las páginas de un sitio web.

Microsoft: Corporation Microcomputer Software. Empresa multinacional estadounidense dedicada al sector de la informática.

MVC: Model/View/Controller / Modelo Vista Controlador.

MVCC: Multi-Version Concurrency Control / Control de Concurrencia Multi-Versión.

MySQL: Sistema de gestión de bases de datos relacional.

MSF: Microsoft Solution Framework. Metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos.

OMG: Object Management Group. Asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas de la industria del software.

OO: Orientada a Objetos.

ORDBMS: Sistema de Gestión de Bases de Datos Objeto-Relacionales.

SO: Sistema Operativo.

PDF: Portable Document Format / Formato de Documento Portátil.

PHP: Hypertext Preprocessor. Lenguaje de programación del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas.

Plugins: Pequeños archivos que añaden mayores funcionalidades y características a un programa.

Protocolo IP: Un protocolo usado para la comunicación de datos a través de una red.

Proxy: Programa o dispositivo que realiza una acción en representación de otro. La finalidad más habitual es la del servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

RUP: Rational Unified Process / Proceso Unificado de Desarrollo. Metodología para el desarrollo de software.

Scripts: Conjunto de comandos secretos en un lenguaje interpretado para automatizar ciertas tareas de aplicación.

SMTP: Simple Mail Transfer Protocol / Protocolo Simple de Transferencia de Correo. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos (PDA's, teléfonos móviles, etc.).

SOAP: Protocolo Simple de Acceso a Objetos. Basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja.

SSL: Secure Sockets Layer. Protocolo de seguridad que proporciona un método para transferir datos entre el cliente y el servidor de forma segura.

Tags: Una etiqueta (o tag) es una marca con tipo que delimita una región en los lenguajes basados en XML.

TCF: Technical Complexity Factor / Factor de complejidad técnica.

UCI: Universidad de las Ciencias Informáticas.

UAW: Unadjusted Actor Weights / Factor de peso de los actores sin ajustar.

UML: Unified Modeling Language / Lenguaje Unificado de Modelado.

WAL: Write Ahead Logging. Registro que lleva todos los cambios que se van haciendo en la base de datos.

Windows: Familia de sistemas operativos desarrollados y comercializados por Microsoft. Existen versiones para hogares, empresas, servidores y dispositivos móviles, como computadores de bolsillo y teléfonos inteligentes.

WSDL: Lenguaje de Descripción de Servicios Web. Permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos.

WWW: World Wide Web.

XML: Extensible Markup Language / Lenguaje de Marcas Extensible. Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

XP: Extreme Programing. Metodología ágil para el desarrollo de software.