# DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DEL MÓDULO DE PLANIFICACIÓN: GESTIÓN DE TIEMPOS, RIESGOS Y PRODUCTOS ENTREGABLES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS.

LAURA INÉS ROJAS MUÑOZ
MARIO ANDRÉS JEREZ QUINTERO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

# DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DEL MÓDULO DE PLANIFICACIÓN: GESTIÓN DE TIEMPOS, RIESGOS Y PRODUCTOS ENTREGABLES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS.

LAURA INÉS ROJAS MUÑOZ
MARIO ANDRÉS JEREZ QUINTERO

Trabajo de grado para optar el título de Ingeniero de Sistemas

Director
ING. ENRIQUE TORRES LOPEZ
PROFESIONAL DIVISION DE SERVICIOS DE INFORMACION
UNIVERSIDAD INDUSTRIAL DE SANTANDER

Codirector
ING. GIOBANI SERRANO DURAN
PROFESIONAL OFICINA DE PLANEACION
UNIVERSIDAD INDUSTRIAL DE SANTANDER

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

## AGRADECIMIENTOS

A Dios, por permitirme cumplir esta meta tan importante en mi vida.

A mi familia, especialmente a mi mamá quién siempre me ha brindado su apoyo incondicional.

A mi compañero Mario Andrés Jerez Quintero por su constante apoyo y por compartir conmigo esta y muchas experiencias de vida. Te amo.

A nuestro director, el ingeniero Enrique Torres López por su ayuda y constante apoyo durante la elaboración del proyecto.

A nuestro codirector, ingeniero Giobani Serrano Duran por su colaboración y respaldo para la ejecución del proyecto.

A los ingenieros de la División de Servicios de información: Emilio Cárcamo, Humberto Ruiz y Elkin Suarez por su invaluable colaboración que hizo posible el desarrollo exitoso del presente proyecto.

A mis compañeros Yenny Tobo, Edwing García, Marco Ruiz y Pedro Otero, con quienes compartimos los momentos felices y los difíciles durante la realización del proyecto.

A la Universidad Industrial de Santander, lugar donde he crecido como persona y profesional.

Laura Inés Rojas Muñoz

## AGRADECIMIENTOS

A mis padres por su apoyo económico y moral incondicionales, que me han permitido llegar hasta esta etapa académica.

A mi compañera de proyecto Laura Inés Rojas Muñoz con quien formé un buen equipo de trabajo y también la persona que aguantó mi fuerte carácter. La estimo y aprecio mucho.

A nuestros directores de proyecto: el Ingeniero Enrique Torres López (profesional de la División de Servicios de Información de la UIS) y el Ingeniero Giobani Serrano Duran (profesional de la Oficina de Planeación) por la confianza y la colaboración que me brindaron a lo largo del proyecto.

A los ingenieros Emilio Cárcamo, Humberto Ruiz y Elkin Suarez por su importante y constante compañía y colaboración durante el desarrollo de este proyecto.

A los compañeros de los proyectos complementarios: Marco Ruiz y Pedro Otero, Edwin García y Yenny Tobo; por su apoyo ofrecido en todo este proceso.

A la Universidad Industrial de Santander en la que me formé como profesional, en la que he madurado y en la que he percibido un poco la realidad de la vida.

*"El madurar nunca será doloroso si nos percatamos de su menester."*

Mario Andrés Jerez Quintero

# CONTENIDO

# LISTADO DE TABLAS

# LISTADO DE FIGURAS

**RESUMEN**

**TITULO:** DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DEL MÓDULO DE PLANIFICACIÓN: GESTIÓN DE TIEMPOS, RIESGOS Y PRODUCTOS ENTREGABLES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS[1].

**AUTORES:**
ROJAS MUÑOZ, Laura Inés[2]
JEREZ QUINTERO, Mario Andrés[3]

**PALABRAS CLAVE:** Gestión de Proyectos, Planificación, JavaEE5, JSF, RichFaces, Hibernate, Seam.

**CONTENIDO**:

El presente trabajo de grado fue desarrollado en conjunto con la División de Servicios de Información, unidad encargada de administrar y desarrollar el software que soporta el funcionamiento de la UIS, y la Oficina de Planeación de la Universidad Industrial de Santander, encargada de la planificación institucional y de brindar asesoría y capacitación a todas las dependencias de la universidad que se encuentren en el proceso de planeación de sus diversos proyectos.

Actualmente la UIS cuenta con un sistema de información limitado, basado en tecnología JSP, producto de un proyecto de grado, que no brinda un soporte completo en la gestión de proyectos de inversión de la institución educativa; haciendo esta herramienta poco funcional. Adicionalmente la Oficina de Planeación plantea el uso de la metodología del PMI, la cual se basa en un conjunto de buenas prácticas, que sirven como guía en la Gestión de Programas y Proyectos de la UIS. Esto repercute directamente sobre los procesos que se ejecutan a lo largo del ciclo de vida de los proyectos, generando ciertos cambios; lo cual justifica aún más la actualización del sistema de información que apoya la administración.

En este proyecto se plantea el diseño y desarrollo de una herramienta software orientada a Web, con calidad de nivel empresarial, siguiendo los lineamientos de la metodología del PMI, que servirá de apoyo a la Gestión de Programas y Proyectos de la UIS, haciendo énfasis en la fase de Planificación de los Proyectos, en lo que concierne a la gestión de tiempos, riesgos y productos entregables. Para el desarrollo de este trabajo de grado se tomaron como base los estándares definidos por la División de Servicios de Información UIS, los cuales contemplan: herramientas de desarrollo, organización de directorios, diagramación UML y su documentación, organización y documentación del código fuente.

---

[1] Proyecto de grado en la modalidad de Investigación.
[23] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Ing. Enrique Torres López. Codirector: Ing. Giobanni Serrano Durán.

**ABSTRACT**

**TITLE:** DESIGN, IMPLEMENTATION AND COMMISSIONING OF PLANNING MODULE: TIME MANAGEMENT, RISKS AND DELIVERY PRODUCT SYSTEM PROJECT MANAGEMENT UIS[4].

**AUTHORS:**
ROJAS MUÑOZ, Laura Inés [5]
JEREZ QUINTERO, Mario Andrés [6]

**KEY WORDS:** Project Management, Planning, JavaEE5, JSF, RichFaces, Hibernate, Seam.

**CONTENT:**

The present graduation Project was developed with the Division of Information Services, a branch in charge of managing and developing the software that supports the functioning of UIS, and the Planning Office of Universidad Industrial de Santander, in charge of the institutional planning and providing advice and training to all the offices at the university that are in the planning process of the diverse projects.

Nowadays UIS has a very limited information system, based on JSP technology, product of a graduation project that doesn't provide a complete support of the management of investment projects at the educational institution, becoming this tool obsolete and little functional. In addition the Planning Office proposes the use of PMI (Project Management Institute) methodology, which bases on a set of lineaments and good practice that help as a guide in management of programs and projects at UIS. This has direct repercussions on processes that are executed along the life cycle of projects, generating certain changes what justifies even more the necessity of modernizing the information system that supports the management.

This project proposes the design and development of a software tool focused on Web, with quality of a managerial level according to the lineaments of PMI methodology that will support the management of programs and projects at UIS, by emphasizing on the phase of planning projects with respect to the management of time, risks and delivery products. To carry out this graduation project the defined standards by the Division of Information Services of UIS were taking as a base. They take into account the development tools, the organization of directories, the UML diagram and its documentation, organization and documentation of the source code.

---

[4]Degree Project in the modality Research
[5,6]Faculty of Physical-Mechanical Engineering.Systems and Computer Engineering.Director: Ing. Enrique Torres López. Codirector: Ing. Giobanni Serrano Durán.

# INTRODUCCIÓN

En la actualidad el entorno empresarial no sólo es uno de los medios más influyentes en el impulso y desarrollo de los sistemas de información, sino que también ha forjado un papel fundamental en la continua evolución y mejoramiento de esta herramienta, esencial para la toma de decisiones en cualquier organización, como también prestando atención a las necesidades del momento y procurando que las tecnologías disponibles abarquen con más eficacia todas las demandas de los negocios.

Tanto organizaciones privadas como públicas, están cada vez más interesadas y enfocadas en contar con herramientas software que generen confianza en la gestión de la información donde su disponibilidad, oportunidad e integridad; entre otras, no se pierdan, y en contraprestación, colabore en los diferentes niveles de decisión pertinentes.

La Universidad Industrial de Santander (UIS) ha venido consolidando políticas para el manejo de sus recursos de inversión y proporcionando elementos, como lo es el Banco de Programas y Proyectos de Inversión, desde hace más de una década; teniendo entre sus principales metas el uso óptimo de los recursos a disposición, escogiendo la forma más adecuada de asignarlos, y el fomento de una cultura de trabajo más estructurada, es decir, una cultura de proyectos. Esto supone que existen decisiones de vital importancia que tienen lugar en las diferentes unidades académicas y administrativas de la Institución, y como consecuencia, también es de suponer la presencia de los sistemas de información.

Las experiencias obtenidas a través del tiempo han indicado que para que se logren los objetivos concebidos por el Banco de Proyectos, se debe apoyar en el eficiente y eficaz uso de las tecnologías informáticas que proporcionan sistemas de información robustos, con la facultad de adaptarse al dinamismo de las organizaciones y que tomen como referencia las últimas tendencias de la Industria del Software. Además, particularmente para la Universidad es imprescindible tener la capacidad de proyectar, ejecutar y controlar sus recursos de inversión; tarea que sería mucho más compleja sin un Software que le ofrezca un adecuado soporte.

Por lo expuesto, desde la Oficina de Planeación con la colaboración de la División de Servicios de Información, ambas dependencias de la UIS, se han liderado proyectos concernientes al desarrollo de Sistemas de Información para poder realizar un seguimiento a los proyectos derivados de la labor institucional; proyectos que a su vez han aportado más conocimientos acerca de los requerimientos de la Entidad Educativa.

Lo que se plantea en este trabajo de grado es diseñar y desarrollar una herramienta software, orientada a Web, con calidad de nivel empresarial, para el Sistema de Gestión de Programas y Proyectos de la UIS (SGPUIS), enfocada en la fase de Planificación de los Proyectos formulados en la Institución, en lo relativo a la gestión de tiempos, de riesgos y de productos entregables.

# CAPÍTULO 1

# 1 CONTEXTO GENERAL

## 1.1 ESPECIFICACIONES DEL PROYECTO

### 1.1.1 Título

DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DEL MÓDULO DE PLANIFICACIÓN: GESTIÓN DE TIEMPOS, RIESGOS Y PRODUCTOS ENTREGABLES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS.

Director y Codirector:

Nombre : Ing. Enrique Torres López
Institución : Universidad Industrial de Santander
Cargo : Profesional adscrito a la División de Servicios de Información.

Nombre : Ing. Giobani Serrano Durán
Institución : Universidad Industrial de Santander
Cargo : Profesional adscrito a la Oficina de Planeación.

Autores:

Nombre : Laura Inés Rojas Muñoz
Código : 2042534
Carrera : Ingeniería de Sistemas

Nombre : Mario Andrés Jerez Quintero
Código : 2040049
Carrera : Ingeniería de Sistemas

Entidades Interesadas En El Proyecto:

Oficina de Planeación
División de Servicios de Información
Universidad Industrial de Santander

### 1.1.2 Objetivos

#### *1.1.2.1 Objetivo General*

Realizar el diseño, implementación y puesta en funcionamiento del módulo de Planificación, específicamente en la programación y seguimiento de las fases y actividades del proyecto, manejo de riesgos y manejo de productos entregables; mediante el uso de últimas tecnologías de programación web.

#### *1.1.2.2 Objetivos Específicos*

- Permitir el manejo de los tiempos del proyecto, lo que implica:
    - Creación de las fases del proyecto y sus actividades asociadas.
    - Creación del cronograma: definiendo la secuencia y duración de las actividades del proyecto.

- Registrar y controlar los cambios asociados al cronograma que surjan a lo largo del ciclo de vida del proyecto.
- Diseñar un sistema de alertas que contribuyan al cumplimiento de las actividades del cronograma.

- Apoyar la gestión de productos entregables, que abarca:
  - Definición de los productos entregables asociados a una fase del proyecto.
  - Supervisión y control de avances y entregas, mediante el registro de las actas de entrega y la generación de alertas de cumplimiento

- Facilitar los procesos de administración de riesgos, permitiendo:
  - Registrar los riesgos que pueden ocurrir durante el ciclo de vida del proyecto.
  - Planificar la respuesta y los planes de contingencia, en caso de que se presente alguno de los riesgos identificados.
  - Realizar un seguimiento a los riesgos presentados, y llevar un registro de los entregables afectados.

- Seguir las políticas de seguridad de la universidad Industrial de Santander teniendo en cuenta:

  - Establecer roles para los usuarios con el fin de definir los permisos de acceso al sistema.
  - Definir los alcances y funciones que puede desempeñar cada usuario en el sistema, buscando así mantener la integridad y la seguridad del mismo.

- Servir de Herramienta de apoyo para la realización de un proceso confiable de auditorías, fijando las pistas de auditoría necesarias como parte de la estructura de datos establecida.

### 1.1.3 JUSTIFICACIÓN

La Universidad Industrial de Santander, es una entidad de educación superior en continuo cambio: el aumento del número de estudiantes, la creación permanente de nuevos programas académicos, avances en proyectos de investigación, entre otros aspectos, hacen que el trabajo de las Unidades académico administrativas (UAA) de apoyo a la gestión institucional sea más complejo y difícil de realizar. Su estructura tradicional, aunque funciona en pro de las necesidades del entorno académico – administrativo, se ve limitada en eficiencia y velocidad en los procesos de soporte académico y de investigación, dado que los tiempos de respuesta a las solicitudes y a las necesidades de implementación de las políticas no es el mejor. En ese sentido, la gestión de proyectos se convierte en una herramienta, que hace más dinámica la aplicabilidad de las políticas de desarrollo institucional.

La aplicación para la formulación y seguimiento de proyectos, que utiliza actualmente la UIS, no satisface las expectativas de los usuarios, dado que no se adapta a la naturaleza dinámica de la institución y no permite un óptimo control de los proyectos en cada una de las fases de su ciclo de vida; de ahí, que, el nuevo sistema de Gestión de Proyectos de la UIS (SGPUIS) surge como alternativa para gestionar de manera adecuada los proyectos identificados en el marco del Plan de Desarrollo Institucional, el cual tiene una cobertura hasta el 2018 y cuenta con el respaldo económico de Colciencias, Recursos de Estampilla ProUIS, y recursos propios orientados a la generación de proyectos.

Este proyecto está siendo coordinado por la Oficina de Planeación con el apoyo de la División de Servicios de Información y contará con la participación de algunos actores potenciales que harán uso de esta herramienta, entre ellos, la

Dirección de Contratación, División Financiera, División de Planta Física, División de Mantenimiento Tecnológico, Vicerrectoría Académica, Vicerrectoría Administrativa, Vicerrectoría de Investigación y Extensión, las cuales tienen bajo su responsabilidad la formulación y ejecución de la mayor parte de proyectos que adelanta la institución.

Los beneficios que aporta esta investigación son los siguientes:

- Brindar a la Dirección de la Universidad, una herramienta gerencial que le permita orientar los recursos de inversión hacia las estrategias de desarrollo plasmadas en el Plan de Desarrollo Institucional PDI.

- Facilitar el seguimiento y control del desarrollo de un proyecto durante todas sus fases.

- Aportar información para valorar los resultados de las inversiones realizadas versus los avances académicos e institucionales proyectados en los planes de desarrollo de la Universidad.

- Dotar a las UAA de la Universidad de los elementos procedimentales para el desarrollo de los proyectos, con el fin de garantizar la coherencia en la maduración y desarrollo de las propuestas.

- Simplificar la identificación, control y seguimiento de los riesgos que puedan presentarse en los proyectos.

- Realizar un óptimo seguimiento del cronograma de actividades, recursos y entregables de los proyectos.

# CAPÍTULO 2

# 2 FUNDAMENTO TEÓRICO

## 2.1 Estado del arte

Con la instauración del Acuerdo 103 de 1997, el Consejo Académico confirió un avance importante enfocado en la formalización y organización de la inversión en la Universidad Industrial de Santander (UIS), dictando la creación del Banco de Proyectos de Inversión.

Este Banco de proyectos, junto con la asistencia de herramientas tecnológicas disponibles ha tenido la misión de incentivar una cultura de proyectos y sobre todo, otorgar a la dirección de la Universidad una visión clara y oportuna de todas las ideas propuestas por las diferentes unidades académicas y administrativas, para que las decisiones tomadas por las distintas autoridades de la institución tengan un argumento basado en la información y faculte el uso eficiente y eficaz de sus recursos. Adicionalmente, el Banco de Proyectos debe satisfacer todos los requerimientos de la Universidad para cumplir las metas de desarrollo institucional, contempladas en el Plan de Desarrollo.

Como toda entidad de gran envergadura y proyección, la gestión de la UIS debía encontrar una solución para lograr manejar el considerable volumen de información que va incrementándose con el transcurrir del tiempo, siendo una alternativa el uso de las tecnologías de información disponibles en el mercado que permitan satisfacer a largo plazo las necesidades tecnológicas de la universidad.

Un primer intento por satisfacer las demandas tecnológicas del Banco de Proyectos fue el sistema de información desarrollado en 1997 por la estudiante Melva Janeth Hernández Ariza, lo que constituyó su trabajo de grado para obtener su título de Ingeniera de Sistemas.

El software fue realizado con el manejador de bases de datos Access versión II y estuvo disponible para toda la comunidad universitaria. No obstante, esta primera herramienta de apoyo, además de las limitaciones tecnológicas, red interna de la institución, tuvo problemas importantes que culminaron convirtiéndola en una herramienta obsoleta e ineficaz.

Las directivas de la UIS requerían un sistema de información más ligado al proceso de inversión, que contara con funciones, procedimientos, objetivos y reglamentación adecuados; donde la asistencia técnica relacionara mejor a los usuarios con la herramienta y los cambios de actualización de software y/o hardware no afectaran de manera ostensible al sistema, objetivos que no cumplía la herramienta desarrollada.

Esta primera experiencia tuvo un impacto enriquecedor para posteriores desarrollos de aplicaciones destinadas a servir al Banco de Proyectos, ya que se pudo concebir los proyectos de una manera más profunda, con respecto a todos los procesos y funciones involucrados en la gestión de los mismos; dando origen a un modelo mejorado.

El segundo Sistema de Información surgió en el año 2001, como trabajo de grado de los estudiantes Edwin Hernández Ramírez y Silvia Suarez Barón con la supervisión tanto de la Oficina de Planeación como de la División de Servicios de Información. Este sistema ya contaba con una reglamentación más madura de las

políticas de la gestión de proyectos en la Universidad, ventaja que junto con las tecnologías disponibles de la época terminó en el diseño de un Sistema de Información más robusto, que intentaría abarcar todo el ciclo de vida de un proyecto y que estaría disponible en la intranet de la Institución.

Para llevar a cabo la nueva propuesta se resolvió orientar un modelo de desarrollo Lineal Secuencial o Ciclo de Vida Clásico porque se ajustaba a los requerimientos del Banco de Proyectos; el Sistema de gestión de proyectos se basó en tecnologías como la Arquitectura DNA de Microsoft (estándar para el desarrollo de aplicaciones de internet por capas) y ASP (tecnología para crear páginas Web), entre otras; con el fin de disponer de las tecnologías de punta y así poder resarcir las necesidades del Banco de Proyectos.

Otros aspectos a destacar del nuevo sistema son la realización del mantenimiento al software (delegado a la División de Servicios de Información), la seguridad implementada tanto a nivel de bases de datos como también de aplicación (autenticación de usuarios), la contemplación de las distintas metodologías para la formulación proyectos y las respectivas asesorías de la Oficina de Planeación en el uso de la herramienta; con estas políticas el sistema no quedaría desamparado y por el contrario se abrió el camino para su continuo desarrollo.

A pesar del planteamiento anterior y dado que las tecnologías informáticas continuaron evolucionando, los modelos de gestión de proyectos fueron modificados tornándose más complejos para el Sistema y la seguridad, que cada vez tomó un papel vital para resguardar la información.

Adicionalmente, con la consolidación de los estándares para el desarrollo de aplicaciones web y las tendencias de los sistemas de información en el ámbito empresarial, el Banco de Proyectos empezó a carecer de herramientas robustas que soportaran todos los procesos de gestión y tuviera en cuenta la naturaleza

dinámica de la Universidad; razones que impedían un completo seguimiento a los proyectos a lo largo de su ciclo de vida.

Tal circunstancia motivó a la Oficina de Planeación y a la División de Servicios de Información a plantear un nuevo sistema de información, utilizando las tecnologías de información de punta y reestructurando las políticas de seguridad; proponiendo no sólo desarrollar una herramienta de vanguardia sino también orientando esfuerzos hacia la calidad exigida en el nivel empresarial.

## 2.2 Gestión de Proyectos

### 2.2.1 ¿Qué es un Proyecto?

Un proyecto es el conjunto de actividades y herramientas acopladas coherentemente, con el fin de lograr los objetivos propuestos para dar una respuesta adecuada a un problema o a una oportunidad, teniendo como base una metodología adaptada a las necesidades particulares de cada entidad.

Se caracteriza por ser temporal, es decir, todo proyecto tiene un inicio y un fin definido, se espera que este final se dé cuando se hayan cumplido completa y satisfactoriamente los objetivos, en caso contrario se llega al final cuando se cancela el proyecto. Otra característica importante es su desarrollo gradual, lo que implica que el proyecto se elabora por pasos y los objetivos se van alcanzando gradualmente.

Los posibles resultados de un proyecto pueden ser: un producto, la capacidad de realizar un servicio y/o un documento.

### 2.2.2 Dirección de Proyectos

La dirección de proyectos se enfoca en aplicar y articular consecuentemente las etapas que conforman la gestión de proyectos: inicio, planificación, ejecución, seguimiento, control y cierre; esta división en etapas se da con el objetivo de facilitar la gestión de los proyectos, de manera que guíen el proyecto de principio a fin, dentro del tiempo estipulado y con el presupuesto asignado para él, constituyendo así el Ciclo de Vida del Proyecto.

En la actualidad la dirección de proyectos abarca tanto prácticas tradicionales muy utilizadas como ideas innovadoras que han nacido en el día a día, debido al avance y la incursión de las herramientas tecnológicas que han permitido la constante evolución en la Dirección de Proyectos; cabe destacar la Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK), desarrollada por el Project Management Institute, que comprende las bases de la Gestión de Proyectos.

### 2.2.3 Ciclo de vida de un proyecto


**Figura 1. Etapas de un Proyecto**

***Inicio:*** Esta etapa constituye el proceso de identificación del proyecto a realizar, la definición preliminar del alcance y la constitución del proyecto.

***Planeación:*** Una vez se ha constituido el proyecto, se define cómo se ejecuta, supervisa, controla y aprueba un proyecto. Esta etapa incluye:

- Plan de gestión
- Preparación
- Evaluación
- Viabilidad
- Elegibilidad
- Aprobación
- Programación

Especialmente el presente proyecto contempla las siguientes actividades de esta etapa:

**Definir Actividades:** este proceso consiste en determinar cada una de las actividades necesarias para realizar los entregables del proyecto.

**Secuenciar las actividades:** Una vez definidas las actividades se procede a identificar las relaciones entre cada una de las actividades del proyecto.

**Establecer la duración de las actividades**: Definir aproximadamente el tiempo o periodos de trabajo necesarios para culminar con éxito cada una de las actividades definidas.

**Desarrollar el cronograma**: en este proceso se lleva a cabo el análisis del orden de las actividades que deben llevarse a cabo para cumplir con los objetivos del proyecto, además de su duración y los recursos que son necesarios para llevarlas a cabo.

**Gestión de entregables:** Consiste en:

- Identificar los entregables correspondientes a cada fase del proyecto.
- Establecer la relación de cada entregable con los objetivos específicos del proyecto.
- Llevar a cabo el seguimiento y control de cada entregable.

**Gestión de Riesgos:** Este proceso involucra:

- *Identificar Riesgos:* Se definen los riesgos que puedan presentarse en cualquier etapa del ciclo de vida del proyecto, especificando su descripción.
- *Análisis Cualitativo de Riesgos*: Consiste en determinar la probabilidad de ocurrencia del riesgo, sus posibles causas y el evento probable que desencadenaría ese riesgo.
- *Respuesta a Riesgos*: En este proceso se describen las acciones que se deben llevar a cabo en caso tal de que se presente un riesgo.
- *Realizar el respectivo seguimiento y control a cada riesgo*.

***Ejecución:*** En esta etapa se realizan las actividades necesarias y programadas para llevar a cabo el plan de gestión, con el propósito de cumplir satisfactoriamente con los requisitos planteados en el proyecto. Los procesos realizados son:

- Dirigir y gestionar la ejecución del proyecto
- Realizar aseguramiento de la calidad
- Gestionar las adquisiciones
- Adquirir y desarrollar el Equipo del Proyecto
- Distribuir la Información

***Seguimiento y control:*** Esta etapa abarca todo el ciclo de vida del proyecto, en ella se llevan a cabo todas las tareas de verificación y control, como:

- Control de costos
- Control y verificación del alcance
- Control del cronograma
- Control de calidad
- Seguimiento y control de riesgos
- Control integrado de cambios

***Cierre:*** Contempla la finalización de obras de infraestructura y contratos, y el cierre definitivo del proyecto. En esta etapa se deben realizar las siguientes tareas:

- Archivar documentación del proyecto.
- Medir resultados del proyecto
- Registrar lecciones aprendidas y errores cometidos.

### 2.2.4 Áreas de conocimiento de la dirección de proyectos

#### *2.2.4.1 Gestión de la integración del proyecto*

Constituye la identificación e integración de los procesos indispensables para gestionar el proyecto y culminarlo exitosamente, los procesos son:

- Mediante un acta realizar la constitución del proyecto.
- Determinar claramente los alcances del proyecto.
- Elaborar el plan de Gestión del proyecto.
- Liderar y administrar la elaboración del proyecto.

- Auditar el trabajo del proyecto.
- Gestionar los cambios.
- Cierre del proyecto.

#### *2.2.4.2 Gestión del Alcance del proyecto*

Abarca los recursos requeridos para garantizar que el proyecto lleve a cabo exclusivamente el trabajo que sea necesario para finalizarlo satisfactoriamente. En esta etapa se define clara y específicamente que contiene y que no contiene el proyecto, los procesos que facilitan esta gestión son:

- Elaborar un plan para administrar el alcance del proyecto.
- Enunciar detalladamente el alcance del proyecto.
- Definir la Estructura de Desglose del trabajo (EDT).
- Verificar el alcance.
- Controlar los cambios a realizar, para que estos no afecten el alcance del proyecto.

#### *2.2.4.3 Gestión del Tiempo del Proyecto*

Incluye la ejecución de los procesos que permitirán finalizar el proyecto en la fecha estipulada, estos procesos se enuncian a continuación.

- Establecer específicamente las actividades del cronograma.
- Definir el secuenciamiento de las actividades.
- Estimar los recursos necesarios para llevar a cabo cada una de las actividades.
- Calcular en periodos laborales el tiempo necesario para cumplir con cada actividad.

- Elaborar el cronograma de actividades.
- Supervisar los cambios que se presenten en el cronograma del proyecto.

#### *2.2.4.4 Gestión de los Costes del Proyecto*

Esta etapa involucra los siguientes procesos, imprescindibles para llevar a cabo el proyecto con el presupuesto asignado.

- Calcular los costos en que se incurre por cada actividad.
- Presupuestar los costos generales, sumando los costos individuales por actividad.
- Supervisar los cambios que puedan afectar el costo estimado para el proyecto.

#### *2.2.4.5 Gestión de Calidad del Proyecto*

Aplica un sistema de gestión de calidad que involucra todos los procesos y actividades, y una continua mejora de estos últimos a lo largo del proyecto, para garantizar que las políticas, los objetivos y las responsabilidades cumplan con las expectativas de la organización.

Sus procesos comprenden:

- Programación de calidad.
- Ejecutar la verificación de la calidad.
- Llevar a cabo la auditoria de Calidad.

### *2.2.4.6 Gestión de los Recursos Humanos del proyecto*

Lo constituyen todos los procesos enfocados a la organización y dirección del equipo del proyecto. La gestión de Recursos Humanos contiene los siguientes procesos:

- Identificación y Documentación de Roles del Proyecto, y creación del Plan de Gestión.
- Agenciar el Grupo del Proyecto.
- Fortalecimiento de Competencias y Relaciones entre los miembros del equipo del proyecto.
- Dirigir el Grupo del Proyecto.

### *2.2.4.7 Gestión de las Comunicaciones del Proyecto*

Muestra todos los procesos que permiten una administración adecuada de la información, con el fin de hacer hincapié en la necesidad de manejar buena comunicación entre los miembros del proyecto.

Los procesos involucrados son:

- Establecimiento de requerimientos de información y comunicación para los que intervienen en el proyecto.
- Difusión de la Información.
- Informar la Productividad del Proyecto.
- Dirigir las comunicaciones para mantener buenas relaciones entre los integrantes del proyecto.

### *2.2.4.8 Gestión de los Riesgos del Proyecto*

Contiene todos los procesos que gestionan aspectos relativos de los riesgos, para aprovechar los eventos positivos y minimizar la aparición y el impacto de los sucesos negativos.

Estas pautas incluyen:

- Planificación de la gestión de Riesgos.
- Conocer todos los posibles eventos negativos que puedan afectar la ejecución del proyecto.
- Distinción Cualitativa de los Riesgos.
- Distinción Cuantitativa de los Riesgos.
- Desarrollo de planes de contingencia.
- Supervisión de los riesgos identificados.

### *2.2.4.9 Gestión de las Adquisiciones del Proyecto*

Abarca las actividades que deben realizarse para comprar los artículos y servicios pertinentes para la realización del proyecto; los procesos involucrados en la gestión de adquisiciones son:

- Definir los productos a comprar, cuándo y cómo se debe realizar dicha compra.
- Determinar la contratación, especificándolas características de los productos y servicios.
- Realizar cotizaciones, licitaciones o escuchar propuestas, según corresponda.
- Seleccionar el proveedor y elaborar con él un contrato escrito.
- Realizar la gestión del contrato.
- Finalización del contrato.

### 2.2.5 Gestión de la Inversión en la Universidad Industrial de Santander

Mediante el acuerdo Nº 032 del año 2002 expedido por el Consejo Superior de la Universidad Industrial de Santander se aprobó la reglamentación de la inversión de esta institución.

El motivo por el cual se implementó esta regulación fue "brindar las herramientas normativas, técnicas y administrativas para la asignación de recursos a las diferentes Unidades Académico Administrativas de la Universidad, teniendo como base criterios de claridad, transparencia, conforme a los lineamientos de desarrollo de la institución y mediante la aplicación del Sistema de Planificación Institucional de la Universidad"[7].

Los nuevos lineamientos del Banco de Proyectos están constituidos por una variedad de componentes que hacen de la gestión de proyectos un eslabón indispensable para cumplir con "las políticas de desarrollo institucional, en la aplicación del proyecto educativo institucional (PEI) y en todas aquellas iniciativas que apuntarán al crecimiento de la Universidad"[8]. Por tal razón, con el tiempo fue siendo pieza fundamental para el Sistema de Planificación Institucional, junto con el Sistema de Inversión Institucional y el Programa de Gestión Anual, para darle eficiencia y eficacia a la labor de la Dirección de la entidad educativa.

[7] SERRANO, Giobani. Plan de proyecto: propuesta de modelo para la Gestión de Proyectos de Inversión en una Institución Pública de Educación Superior en Colombia: una perspectiva desde el Pensamiento Sistémico. Universidad Industrial de Santander. Bucaramanga. 2009. P. 23.
[8] Ibíd. P 23.

### 2.2.6 Banco de Programas y Proyectos de Inversión Nacional– BPIN

En el epílogo de la década de los 80, se marca un hecho histórico en Colombia que cambió el paradigma de la inversión pública en todo el orden territorial con la creación del Banco de Proyectos de Inversión Nacional (BPIN); mediante la Ley 38 de 1989. Esta herramienta fue creada para "la racionalización del gasto público y para el fortalecimiento de las actividades de reinversión";[9] aspectos claves para la toma de decisiones y para la optimización del uso del Presupuesto General de la Nación.

El BPIN además de contar con un soporte legal e institucional, incluía tres componentes básicos desde su diseño:

#### *2.2.6.1 Componente de Metodología:*

Con este elemento se busca dar una nueva presentación y proposición para la ejecución de los recursos de inversión del Presupuesto General de la Nación; formulando ordenadamente las propuestas a través de proyectos, los cuales deben cumplir unas condiciones mínimas y procedimientos impuestos por el Departamento Nacional de Planeación (DNP), para todas las entidades del ámbito público.

[9]CADENA RUIZ, Ana Maria. Antecedentes BPIN. {En línea}. {08 Febrero de 2010}. Disponible en: (http://www.dnp.gov.co/PortalWeb/Portals/0/archivos/documentos/DIFP/Presupuesto/Antecedentes_Bpin.pdf)

#### *2.2.6.2 Componente de Capacitación*

La capacitación tiene el objetivo de fomentar el uso de las Metodologías sancionadas por el DNP y también, la divulgación del uso de técnicas concernientes a la evaluación de proyectos para finalmente lograr una cultura de proyectos en todo el territorio nacional.

#### *2.2.6.3 Componente de Sistemas de Información*

Con la creciente adopción de los Sistemas de Información como pieza clave para la toma de decisiones y principal responsable del almacenamiento de gran cantidad de información, el DNP no descarta el desarrollo de una herramienta computacional con el fin de darle funcionalidad al BPIN en la identificación de proyectos viables para su posterior aprobación o rechazo.

### 2.2.7 Banco de Programas y Proyectos de Inversión de la UIS (BPPIUIS)

Es el mecanismo del Sistema de Planificación Institucional que se emplea para la definición de las inversiones, asignando los recursos correspondientes a cada uno de los proyectos o programas admisibles; para operar en el marco del Plan de Desarrollo, el Programa de Gestión Anual y el Presupuesto de Inversiones. Igualmente admite implantar procesos de análisis, programación y ejecución de la Inversión, y la respectiva supervisión de la gestión y los resultados.

Los cometidos del BPPIUIS deben abarcar los siguientes aspectos:

a) Asegurar que la información presente en el Banco sea tan competente y pertinente como la programación de los proyectos de inversión de la Universidad.

b) Ampliar los alcances de la dirección de la Institución en la gestión de las inversiones, para que los recursos adscritos sean justos y transparentes.

c) Vincular los procesos de planeación con la estructuración de los proyectos y la destinación de los recursos de inversión de la Universidad.
d) Afianzar una sólida cultura de trabajo por proyectos en la UIS.

e) Mejorar el proceso de toma de decisiones en la adscripción de recursos mediante antecedentes de programas y proyectos inscritos en el Banco de Proyectos.

f) Enriquecer con eficiencia y eficacia todas las etapas de la gestión de la inversión, que involucran: planeación, presupuestación, programación, ejecución, evaluación y gestión de la inversión.

g) Desarrollar un sistema de información que relacione el presupuesto de inversión y el programa de gestión.

h) Permitir obtener información de problemas y sus posibles orígenes en la fase de ejecución de las propuestas de inversión de la Institución.

## 2.3 Diagramación UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es una herramienta que permite al diseñador de aplicaciones o sistemas, abstraer ideas en un modelo visible, que puede abarcar toda la complejidad del sistema, y que a su vez sirve como interface de comunicación con el desarrollador.

UML se basa en un estándar para describir el prototipo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.
Además este lenguaje de modelado no sólo fue desarrollado para entender mejor los requerimientos del Sistema (o del Software) sino también para contar con un prototipo orientado a objetos; vital en el desarrollo de Software dado que se identifica con el paradigma de programación orientada a objetos.

Con el propósito de contextualizar las ventajas que ofrece la utilización del UML como herramienta de diseño, específicamente en el desarrollo de aplicaciones Web; es correcto afirmar que el éxito de un proyecto de esta índole depende en gran medida de la comunicación entre el Analista (persona que se encarga de documentar los problemas o necesidades del cliente) y los desarrolladores (Diseñadores y Programadores). En este punto es evidente la importancia del uso del UML ya que permite dar forma de una manera convencional a los requerimientos del cliente a través de diagramas que representan el lenguaje de comunicación entre diseñador(es) y desarrollador(es).

Los Diagramas UML se dividen en tres grandes grupos, según el énfasis que presenta cada diagrama sobre el sistema:

- Los **Diagramas de Estructura** enfatizan en los elementos que deben existir en el sistema modelado. Ej. diagramas de clases, de objetos y de componentes.

- Los **Diagramas de Comportamiento** enfatizan en lo que debe suceder en el sistema modelado. Ej. diagramas de casos de uso y diagramas de estados.

- Los **Diagramas de Interacción** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado. Ej. diagramas de secuencia y colaboración.[10]

A continuación se describen los diagramas UML que de acuerdo con los estándares establecidos por la División de Servicios de Información, deben ser contemplados en los proyectos de desarrollo de software que se realicen para la universidad:

### 2.3.1 Diagrama de Casos de Uso

El diagrama de casos de uso tiene el objetivo de evidenciar todas las funcionalidades del sistema y además debe llevarse a cabo desde la perspectiva del usuario, ya que modela la interacción directa de este con el sistema; de su análisis se puede concluir si el sistema cumple satisfactoriamente con los requisitos de los usuarios.

Para el prototipo de casos de uso se identifican tres elementos básicos:

---

[10] Enciclopedia Virtual Wikipedia. Lenguaje Unificado de Modelado. {En línea}. {15 diciembre de 2010}. Disponible en: (http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado#Diagramas).

- ***Actores:*** Corresponden a los diferentes roles que los usuarios del sistema pueden representar. Cada rol debe ser único, es decir, distinguible de los demás, contando con un nombre específico y responsabilidades particulares al momento de interactuar con el sistema. No obstante, es necesario aclarar que un rol representa a un tipo o categoría de usuarios del sistema e incluso un usuario puede tener asignado más de un rol en el sistema. Un actor es usualmente identificado como se muestra la figura 2:

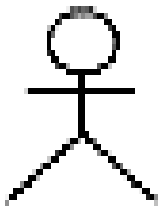

**Figura 2. Actor**

- ***Caso de Uso:*** Es una secuencia de transacciones relacionadas, ejecutadas por uno o más actores y el sistema en un diálogo determinado. En su conjunto, los casos de uso son los que describen todas las funcionalidades del sistema y cada uso de ellos está constituido por una secuencia de mensajes. En la figura 3 indica cómo se grafican los casos de uso en el diagrama.

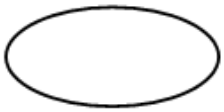

**Figura 3. Caso de Uso**

Un caso de uso especifica dos tipos de secuencias o comportamientos:

a) *Secuencia Básica o Comportamiento Normal:* Son las acciones que ejecuta el actor sobre el sistema en el orden establecido en cada caso de uso.

b) *Secuencia o Comportamiento Alternativo:* Es el camino que el Actor invoca cuando este no lleva a cabo la secuencia básica en forma satisfactoria.

- ***Relaciones:*** Son los elementos que conectan los anteriores elementos en el diagrama (Actores y Casos de Uso), los cuales confieren un significado a cada vínculo que establecen.

  En un diagrama de casos de uso pueden existir los siguientes enlaces:

  - *Relación de Generalización entre Actores:* Se utiliza cuando un actor hereda ciertas características de otro más general.

  - *Relación de Generalización entre casos de Uso:* Indica cuando un caso de uso hereda y adiciona características de otro más general.

  - *Relación de Extensión:* Es un enlace entre casos de uso que define cuando un caso de uso es extendido por otro, es decir, cuando un caso de uso tiene variantes en su secuencia básica, la cual indica una extensión hacia la secuencia básica de otro.
  - *Relación de Inclusión:* Señala cuando el comportamiento normal de un caso de uso por su naturaleza incorpora el comportamiento normal de otro.

  - *Relación de Asociación:* Se utiliza para conectar un actor con un caso de uso e implica la existencia de una comunicación entre ellos.

  La siguiente figura 4 ilustra los diferentes tipos de relaciones que pueden presentarse en un diagrama de casos de uso:

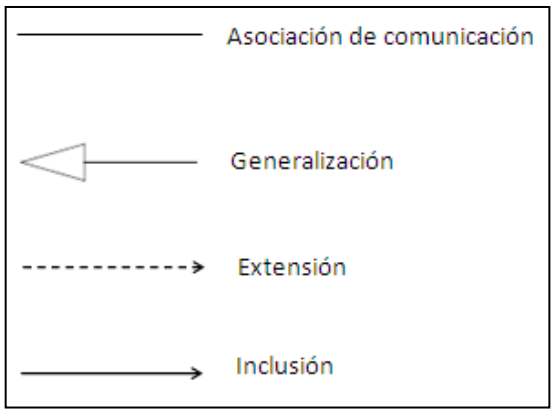


**Figura 4. Tipos de Relaciones Casos de uso**

### 2.3.2 Diagrama de Clases

Sirve para modelar las clases que involucra el sistema, pero es preciso aclarar que el diagrama de clase representa el prototipo estático del sistema, ya que no explica el comportamiento del sistema en el tiempo.

Un diagrama de clases se compone de dos elementos: Clases y Relaciones.

#### *2.3.2.1 Clases*

Para entender el concepto de clase es imprescindible tener claridad acerca del concepto de objeto.

"Un objeto puede ser una abstracción, concepto o cosa con límites bien definidos y con significado en el sistema"[11]

[11] DE AMESCUA SECO, Antonio; CUADRADO GALLEGO, Juan José; ERNICA LAFUENTE, Emilio; GACÍA GUZMÁN, Javier; GARCÍA SÁNCHEZ, Luis; MARTÍNEZ FERNÁNDEZ, Paloma; SÁNCHEZ SEGURA M[a] Isabel. Análisis y Diseño Estructurado y Orientado a Objetos de Sistemas Informáticos. Quinta Edición. Universidad Carlos III de Madrid, España. McGraw-Hill, 2003. P 25.

Adicionalmente un objeto presenta ciertas características:

- *Estado:* Es definido por los atributos que contiene el objeto y por sus posibles relaciones con otros objetos.
- *Comportamiento:* Explica la funcionalidad de un objeto, señalando todas las operaciones que este puede realizar.
- *Identidad:* Implica la unicidad de cada objeto así comparta el mismo estado con otro(s).

"Una clase es una descripción de un conjunto de objetos con las mismas propiedades (atributos), el mismo comportamiento (operaciones), las mismas relaciones entre objetos y la misma semántica"[12]. Los atributos o propiedades hacen referencia a valores concretos que pueden ser numéricos, alfabéticos o alfa-numéricos.

Una clase es representada en la figura 5:

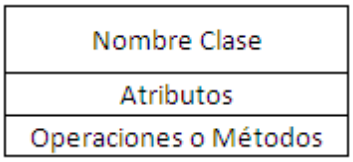


**Figura 5. Representación de clase**

### *2.3.2.2 Relaciones*

Es la connotación entre los objetos de dos o más clases. Las clases pueden ser interrelacionadas por medio de las siguientes relaciones:

[12] Ibíd. P 25.

- ***Relación de Asociación:*** Representa un conjunto de enlaces entre dos clases distintas, los cuales pueden ser unidireccionales o bidireccionales. Además contempla la Multiplicidad de Asociaciones que es la cantidad de objetos de cada clase en una relación.

- ***Relación de Agregación:*** Es la relación que existe entre una clase que envuelve o incluye a otras clases, cuyos tiempos de vida no dependen del tiempo de vida de la clase que las incluye. Cuando el tiempo de vida de un objeto de una clase depende del tiempo de vida de otro objeto que lo incluye, se dice que es una ***Relación de Composición.***

- ***Relación de Herencia:*** Es el vínculo entre una Súper clase y una o más subclases hijas, quienes heredan atributos y comportamientos de su clase padre y pueden extender las propiedades de dicha Súper clase adicionando atributos que proporcionan un mayor detalle de lo que la clase padre representa.

  La herencia puede darse de dos formas:

  - *Generalizada:* Ocurre cuando la Súper clase encapsula las propiedades y comportamientos de una o más subclases. En este tipo de relación las subclases no pueden heredar.

  - *Especializada:* Se da cuando las subclases heredan las propiedades y comportamientos de una clase mayor dándole a esta última un mayor nivel de detalle.

La figura 6 exhibe las relaciones que pueden existir en el diagrama de clases, anteriormente descritas:

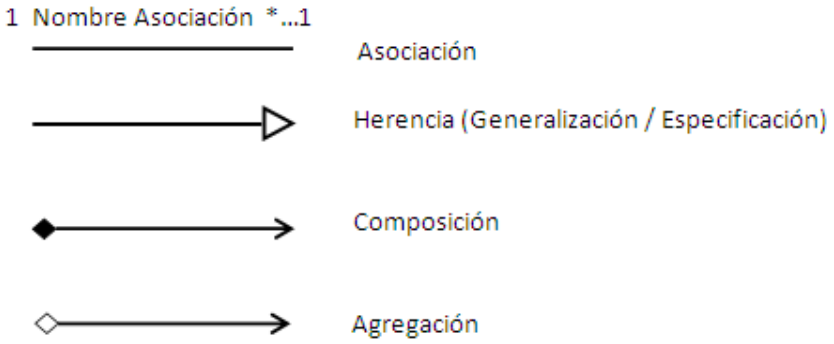


**Figura 6. Tipos de Relaciones entre Clases**

### 2.3.3 Diagrama de Secuencia

"Representa la interacción entre clases del modelo de estructuras estáticas, ordenadas temporalmente"[13], donde el Actor es el responsable de iniciarla en el momento que efectúa un mensaje u operación sobre el sistema. El diagrama de secuencia se lee de izquierda a derecha y de arriba abajo; cada caso de uso tiene asociado un diagrama de secuencia, sin embargo, puede tener más de uno dependiendo de los posibles comportamientos alternos por los que el caso de uso deba optar.

Los principales elementos que involucra un diagrama de secuencia se muestran en la figura 7:

[13] DE AMESCUA SECO, Antonio; CUADRADO GALLEGO, Juan José; ERNICA LAFUENTE, Emilio; GACÍA GUZMÁN, Javier; GARCÍA SÁNCHEZ, Luis; MARTÍNEZ FERNÁNDEZ, Paloma; SÁNCHEZ SEGURA M[a] Isabel. Análisis y Diseño Estructurado y Orientado a Objetos de Sistemas Informáticos. Quinta Edición. Universidad Carlos III de Madrid, España. McGraw-Hill, 2003. P 64.

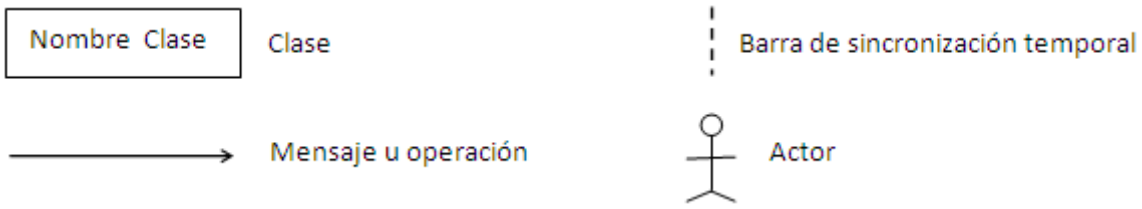


**Figura 7. Elementos del Diagrama de Secuencia**

## 2.4 Tecnologías de Desarrollo de Aplicaciones Web

### *2.4.1 JAVA EE5*

La edición empresarial de Java ha hecho más eficiente el desarrollo de aplicaciones Java empresariales, teniendo como principal objetivo brindar a los desarrolladores un grupo de API que permita reducir el tiempo de desarrollo, la complejidad y el rendimiento de las aplicaciones.

La plataforma Java EE introduce un modelo de programación simplificado, convirtiendo los descriptores de despliegue XML en opcionales. En su lugar, un desarrollador puede inyectar la información como una anotación dentro del fichero de código fuente Java, y el servidor se encarga de configurar el componente en el momento del despliegue y en tiempo de ejecución.

Otra de las ventajas que ofrece esta plataforma es que permite a las inyecciones de dependencias ser aplicadas a todos los recursos necesarios para un componente. Estas inyecciones se pueden utilizar en contenedores EJB, contenedores Web y clientes de aplicación.

La API de persistencia permite un mapeo de objetos a relaciones para manejo de datos relacionales en beans empresariales, componentes web y clientes de aplicación. También puede ser utilizado en aplicaciones Java SE, fuera del ambiente Java EE.

#### *2.4.1.1.1 Seguridad*

La plataforma Java EE dispone de reglas estándar para controlar el acceso. Estas son definidas por el desarrollador y se implementan al desplegar la aplicación en el servidor. Java EE pone a disposición de los desarrolladores mecanismos de acceso predefinidos para que ellos no tengan que implementarlos en sus aplicaciones. Una sola aplicación trabaja en diversos ambientes de desarrollo sin necesidad de modificar el código fuente.

#### *2.4.1.1.2 Componentes Java EE*

Un componente es una unidad de software que está contenida y ensamblada en una aplicación Java EE, posee sus clases relacionadas e interactúa con los demás componentes que así lo especifiquen.
La plataforma Java EE define los siguientes componentes:
Aplicaciones cliente y Applets, estos componentes se ejecutan en el cliente.

Los componentes Web:Servlets, JavaServer Faces, y tecnología JavaServerPagesTM (JSPTM);se ejecutan en el servidor.

Los componentes JavaBeansTM (EJBTM) empresariales (beans empresariales), son llamados componentes de negocios y se ejecutan en el servidor.

Los componentes Java EE se desarrollan con el lenguaje de programación Java y su compilación es igual que cualquier programa en este lenguaje. A diferencia de las clases Java estándar, los componentes Java se ensamblan en una aplicación y se verifica que estén bien formados, de acuerdo a las especificaciones de Java EE, se despliegan en ejecución donde son manejados por el servidor.

Comunicaciones del servidor Java EE

La siguiente imagen ilustra los componentes que pueden formar la capa cliente, en los casos en que la interacción del cliente con la capa del negocio se lleva a cabo en el servidor Java EE de forma directa, y en el caso de un cliente que se ejecuta en un navegador mediante una página JSP o un Servlet que se ejecuta en la capa web

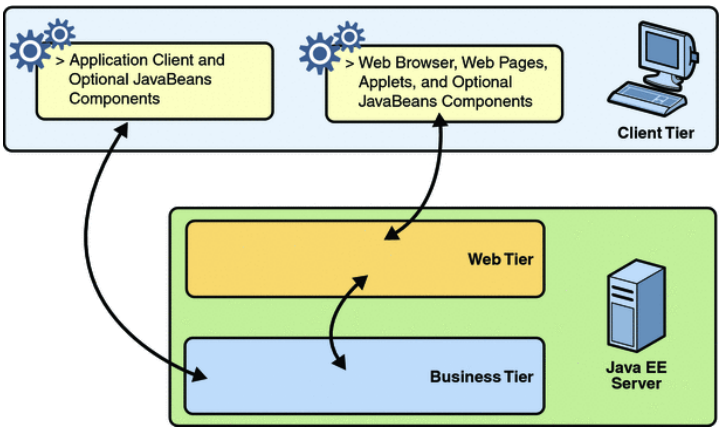


**Figura 8. Comunicación del servidor**[14]

14 Oracle. *Documentación Java EE 5.* {En línea}. {2 de Octubre de 2010}. Disponible en: http://java.cabezudo.net/trabajos/JEE5/manual/jee5.v0.01.00/ababac.html

En caso de que la aplicación Java EE haga uso de un navegador o una aplicación cliente pesada se debe tener cuidado en la decisión de cuál de los dos utilizar para equilibrar la funcionalidad en el cliente y acercarse al usuario para cargarle al servidor toda la funcionalidad que sea posible, esto facilita la distribución, despliegue y manejo de la aplicación.

#### *2.4.1.2 Contenedores Java EE*

La arquitectura basada en componentes e independiente de la plataforma de la arquitectura Java EE hace que las aplicaciones Java EE sean fáciles de escribir ya que la lógica de negocio está organizada en componentes reutilizables. Adicionalmente el servidor Java EE brinda servicios de capas bajas en forma de contenedores para cada tipo de componente.

#### *2.4.1.3 Soporte para servicios web*

Los servicios web son aplicaciones empresariales que utilizan un estándar abierto basado en XML y protocolos de transporte para intercambiar datos entre clientes. La plataforma proporciona una API para XML y las herramientas que se necesitan para diseñar, desarrollar, probar y desplegar rápidamente servicios web y clientes que operan totalmente con otros servicios web y clientes,los cuales se ejecutanen plataformas no propias de Java.

Para escribir servicios web y clientes con las APIs de XML de Java EE se deben pasar los datos en los parámetros de las llamadas a los métodos y procesar los datos retornados; en el caso de servicios web orientados a documentos, es necesario enviar documentos que contengan la comunicación del servicio en ambos sentidos. No se necesita programación a bajo nivel dado que la API de XML hace el trabajo de traducir los datos de la aplicación desde y hacia el flujo de

datos basado en SML que es enviado a través de los protocolos estandarizados de transporte basados en XML.

La traducción de los datos a un flujo de datos estandarizado basado en XML es lo que hace que los servicios web y clientes escritos con las APIs de Java EE puedan operar entre ellos totalmente. Esto no necesariamente significa que los datos transportados incluyan etiquetas XML porque el transporte de los datos puede ser texto plano, datos XML o cualquier tipo de dato binario como audio, vídeo, mapas, ficheros de programa, documentos CAD o lo que sea requerido.

#### *2.4.1.4 Ensamblaje y despliegue de una aplicación Java EE*

Una aplicación Java EE es empaquetada en una o más unidades estándar para ser desplegada en cualquier sistema compatible con la plataforma Java EE. Cada unidad contiene:

- Un componente o componentes funcionales (como un bean empresarial, página JSP, servlet o Applet).
- Un descriptor de despliegue que describe su contenido.

Una vez que una unidad Java EE ha sido producida, está lista para ser desplegada. Sudespliegue típicamente involucra el uso de una herramienta para especificar la información de ubicación específica, como una lista de usuarios locales que pueden acceder a esta y el nombre de la base de datos local. Una vez desplegado en una plataforma local, la aplicación esta lista para ejecutarse.

#### *2.4.1.5 Empaquetado de aplicaciones*

Una aplicación Java EE es distribuida en un Archivo Empresarial (EAR) que es un Archivo Java estándar (JAR) con una extensión .ear. El uso de archivos EAR y módulos hace posible ensamblar una gran cantidad de aplicaciones Java EE utilizando alguno de los mismos componentes. No se necesita codificación extra; es solo un tema de ensamble (o empaquetado) de varios módulos Java EE en u fichero EAR de Java EE.
Un fichero EAR contiene, como muestra la figura, módulos Java EE y descriptores de despliegue.

Un **descriptor de despliegue** es un documento XML con una extensión .xml que describe la configuración de despliegue de una aplicación, un módulo o un componente. Dado que la información en el descriptor de despliegue es declarativa, esta puede ser cambiada sin la necesidad de modificar el código fuente. En tiempo de ejecución, el servidor Java EE lee el descriptor de despliegue y actúa sobre la aplicación, módulo o componente como corresponde.

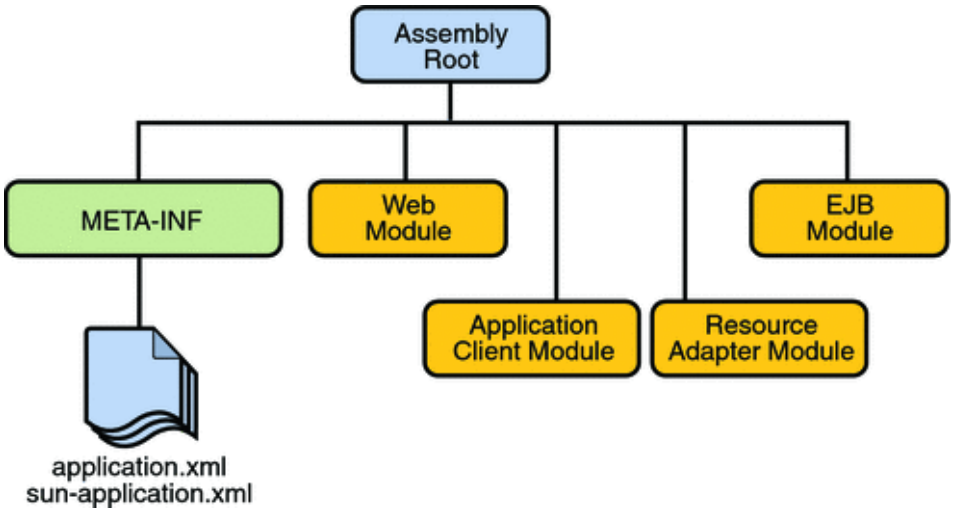


**Figura 9. Estructura de un fichero EAR**[15]

### 2.4.2 Aplicaciones web

Una aplicación web es una extensión dinámica de una web o servidor de aplicación. Hay dos tipos de aplicaciones web:

- **Orientada a la presentación**: Una aplicación web orientada a la presentación genera páginas web interactivas que contienen varios tipos de lenguajes de marcas (HTML, XML y demás) y contenido dinámico en respuesta a las peticiones.

- **Orientadas a los servicios**: Una aplicación web orientada a los servicios implementa el punto final de un servicio web. Las aplicaciones orientadas a la presentación son a menudo clientes de aplicaciones orientadas a servicios.

---

15 Oracle. *Documentación Java EE 5.* {En línea}. {2 de Octubre de 2010}. Disponible en: http://java.cabezudo.net/trabajos/JEE5/manual/jee5.v0.01.00/acadad.html

Desde la introducción de la tecnología de Servlets y JSP, han sido desarrollados marcos de trabajo para construir aplicaciones web interactivas. La figura 10 muestra estas tecnologías y sus relaciones.

16

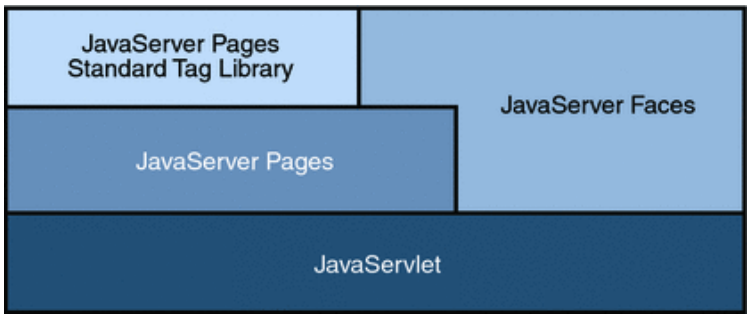


**Figura 10. Tecnologías Java para aplicaciones web**

Debe notarse que la tecnología Java Servlet es la base para todas las tecnologías de aplicaciones web. Cada tecnología agrega un nivel de abstracción que hace la creación de prototipos y el desarrollo más rápido y la aplicación web por si misma más fácil de mantener, de escalar y más robusta.

### *2.4.2.1 Ciclo de vida de una aplicación web*

Una aplicación web consiste en componentes web, ficheros de recursos estáticos como imágenes, clases de ayuda y librerías. El contenedor web proporciona muchos de los servicios de soporte para mejorar las habilidades de los componentes web y los hace fáciles de desarrollar. Sin embargo, dado que una

16 Oracle. *Documentación Java EE 5.* {En línea}. {2 de Octubre de 2010}. Disponible en: http://java.cabezudo.net/trabajos/JEE5/manual/jee5.v0.01.00/acadab.html

aplicación web debe tomar estos servicios en una cuenta, el proceso de crear y ejecutar una aplicación web es diferente que el utilizado para las clases Java autónomas.

El proceso para crear, desplegar y ejecutar una aplicación web puede resumirse como sigue:

- Desarrollar el código del componente web.
- Desarrollar el descriptor de despliegue de la aplicación web.
- Compilar los componentes de la aplicación web y clases de ayuda referenciada por los componentes.
- Opcionalmente empaquetar la aplicación en una unidad que se pueda desplegar.
- Desplegar la aplicación en un contenedor web.
- Acceder a una URL que referencia la aplicación web.

### 2.4.3 Tecnología Servlet Java

#### *2.4.3.1 ¿Qué es un Servlet?*

Un servlet es una clase del lenguaje de programación Java que es utilizada para extender las habilidades de los servidores que guardan aplicaciones a las cuales se accede mediante el modelo petición-respuesta.

A pesar de que los servlets pueden devolver a cualquier tipo de respuesta, estos son comúnmente utilizados para extender las aplicaciones almacenadas en servidores web. Para estas aplicaciones, la tecnología Servlet Java define las clases servlets específicas para HTTP.

### ❖ *Ciclo de vida de un servlet*

El ciclo de vida de un servlet está controlado por el contenedor en donde el servlet ha sido desplegado. Cuando una petición es mapeada a un servlet, el contenedor realiza los siguientes pasos:

1. Si una instancia del servlet no existe, el contenedor web:
   - Carga la clase servlet.
   - Crea una instancia de la clase servlet.
2. Inicializa la instancia del servlet llamando al método init.
3. Invoca el método service, pasando los objetos request y response.
4. Si el contenedor necesita quitar el servlet, este finaliza el servlet llamando el método destroy.

### ❖ *Inicializando un Servlet*

Luego de que el contenedor carga e instancia la clase servlet y antes que distribuya solicitudes de los clientes, el contenedor Web inicializa el servlet. Para construir a medida este proceso y que el servlet cargue datos de configuración persistente, se inicializan recursos y se realiza cualquier otra actividad a tiempo, usted debe sobrecargar el método init de la interface Servlet. Un servlet que no puede completar su proceso de inicialización debe lanzar una excepción UnavailableException.

❖ ***Manteniendo el estado del cliente***

Muchas aplicaciones necesitan que una serie de solicitudes de un cliente sean asociadas con otro. Las aplicaciones basadas en Web son responsables por mantener este estado, llamado sesión, ya que HTTP no tiene estado. Para soportar las aplicaciones que necesitan mantener el estado, la tecnología Servlet de Java proporciona una API para manejo de sesiones y permite varios mecanismos para implementar sesiones.

*Accediendo a una sesión*

Las sesiones son representadas por un objeto HttpSession. Se accede a la sesión llamando el método getSession de un objeto request. Este método retorna la sesión actual asociada con esa solicitud, aunque si dicha solicitud no tiene una sesión entonces esta última es creada.

❖ ***Finalizando un servlet***

Cuando un contenedor de servlet determina que un servlet debe ser eliminado del servicio, el contenedor llama al método destroy de la interface Servlet. En este método, usted libera todos los recursos que el servlet está utilizando y guarda cualquier estado persistente.

Todos los métodos de un servicio dado por un servlet deben ser completados cuando un servlet es eliminado. El servidor trata de asegurarse de esto llamando el método destroy solo luego de que todas las solicitudes de servicio han retornado o después de un período de gracia específico del servidor, lo que suceda primero. Si su servlet tiene operaciones que toman un tiempo largo en ejecutar, las operaciones pueden seguir ejecutándose luego de que el método destroy es llamado. Es preciso asegurar que todos los hilos manejan las solicitudes de clientes de forma completa realizando los siguientes procesos:

- Mantener la cuenta de cuantos hilos siguen ejecutando el método service.
- Proveer un apagado limpio con el método destroy notificando a los hilos que tardan mucho tiempo en ejecutase de la baja y esperar a que estos completen.
- Hacer un sondeo periódico de los métodos que tardan mucho en ejecutar y si es necesario, detener el trabajo, hacer limpieza y retornar.

### 2.4.4 JAVA SERVER FACES

#### *2.4.4.1 Características principales*

La tecnología JavaServer Faces constituye un marco de trabajo (*framework*) de interfaces de usuario del lado de servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).
Los principales componentes de la tecnología JavaServer Faces son:
• Una API y una implementación de referencia para:
- Representar componentes de interfaz de usuario (UI-*User Interface*) y manejar su estado.
- Manejar eventos, validar en el lado del servidor y convertir datos.

- Definir la navegación entre páginas.
- Soportar internacionalización y accesibilidad.
- Proporcionar extensibilidad para todas estas características.

• Una librería de etiquetas JavaServerPages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP. Este modelo de programación bien definido junto con la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado servidor. Con un mínimo esfuerzo, es posible:

• Conectar eventos generados en el cliente a código de la aplicación en el lado del servidor.
• Mapear componentes UI a una página de datos en el lado servidor.
• Construir una interfaz de usuario con componentes reutilizables y extensibles.
Como se puede apreciar en la figura, la interfaz de usuario que se crea con la tecnología JavaServer Faces se ejecuta en el servidor y se renderiza en el cliente.
[17]

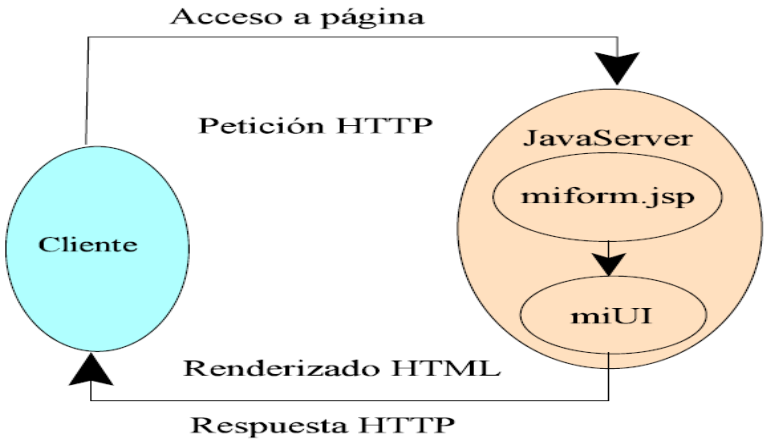


**Figura 11. Diagrama de una aplicación jsf**

[17] SICUMA: Sistemas de Información Cooperativos Universidad de Málaga. *Tutorial de JavaServer Faces.*P. 5 {En línea}. {10 de Diciembre de 2010}. Disponible en: http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf

En la figura 11, la página JSP (**miform.jsp**), especificalos componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por latecnología JavaServer Faces. La UI de la aplicación web (representada por *miUI* en la figura 12)maneja los objetos referenciados por la JSP, que pueden ser de los siguientes tipos:

• Objetos componentes que mapean las etiquetas sobre la página JSP.
• Oyentes de eventos, validadores y conversores registrados y asociados a loscomponentes.
• Objetos del modelo que encapsulan los datos y las funcionalidades de loscomponentes específicos de la aplicación (lógica de negocio).

#### *2.4.4.2 Beneficios de la Tecnología JavaServer Faces*

Una de las ventajas de que JSF sea una especificación estándar es que pueden encontrarse implementaciones de distintos fabricantes. Esto permite no vincularse exclusivamente con un proveedor concreto, y poder seleccionar el más adecuado según los requerimientos de la aplicación; según el número de componentes que suministra, el rendimiento de éstos, soporte proporcionado, precio, política de evolución, etc.

JSF trata la vista (la interfaz de usuario) de una forma algo diferente a lo que se está acostumbrado en las aplicaciones web, donde la programación de la interfaz se desarrolla a través de componentes y está basada en eventos (pulsación de un botón, cambio en el valor de un campo, etc.).

JSF es muy flexible ya que permite personalizar tanto los componentes como la recarga de la vista de las páginas, con el fin elaborar interfaces de usuario en la forma que más nos convenga.

La tecnología JavaServer Faces permite construir aplicaciones web que introducen realmente una separación entre el comportamiento y la presentación, separación sólo ofrecida tradicionalmente por arquitecturas UI del lado del cliente y parcialmente por la tecnología JSP.

Separar la lógica de negocio de la presentación también permite que cada miembro del equipo de desarrollo de la aplicación web se centre en su parte asignada del proceso diseño, y proporciona un modelo sencillo de programación para enlazar todas las piezas.

Otro objetivo importante de la tecnología JavaServer Faces es mejorar los conceptos asociados con componente-UI y capa-web sin limitarse a una tecnología de *script* particular o un lenguaje de marcas. Aunque la tecnología JavaServer Faces incluye una librería de etiquetas JSP personalizadas para representar componentes en una página JSP, las APIs de JavaServerFaces se han creado directamente sobre el API *JavaServlet*. Esto permite, teóricamente, hacer algunas cosas avanzadas: usar otra tecnología de presentación junto a JSP, crear componentes propios directamente desde las clases de componentes, y generar salida para diferentes dispositivos cliente; entre otras.

#### *2.4.4.3 ¿Qué es una aplicación JavaServer Faces?*

En su mayoría, las aplicaciones JavaServer Faces son como cualquier otra aplicación web Java. Se ejecutan en un contenedor de servlets de Java y, típicamente, contienen:

• Componentes JavaBeans conteniendo datos y funcionalidades específicas de la aplicación.
• Oyentes de Eventos.
• Páginas, (principalmente páginas JSP).
• Clases de utilidad del lado del servidor, como beanspara acceder a las bases de datos.

Además de estos ítems, una aplicación JavaServer Faces también tiene:
• Una librería de etiquetas personalizadas para implementar componentes UI en una página.
• Una librería de etiquetas personalizadas para representar manejadores de eventos, validadores y otras acciones.
• Componentes UI representados como objetos con estado en el servidor.

Toda aplicación JavaServer Faces debe incluir una librería de etiquetas personalizadas que define las etiquetas que representan componentes UI, así como una librería de etiquetas para controlar otras acciones importantes, como validadores y manejadores de eventos. La implementación de JavaServer Faces, de SunMicrosystems, proporciona estas dos librerías. La librería de etiquetas de componentes elimina la necesidad de codificar componentes UI en HTML u otro lenguaje de marcas, lo que se traduce en el empleo de componentes completamente reutilizables. Y la librería principal (core) hace fácil registrar eventos, validadores y otras acciones de los componentes.

Finalmente, la tecnología JavaServer Faces permite convertir y validar datos sobre componentes individuales e informar de cualquier error antes de que se actualicen los datos en el lado del servidor.

A continuación listamos los roles principales de un equipo de desarrollo típico

• **Autores de páginas**, que utilizan un lenguaje de marcas, como HTML, para construir páginas para aplicaciones web. Cuando se utiliza la tecnología JavaServer Faces, los autores de páginas casi siempre usarán exclusivamente la librería de etiquetas.

• **Desarrolladores de aplicaciones**, que programan los objetos del modelo, los manejadores de eventos, los validadores, y la navegación de páginas. Los desarrolladores de aplicaciones también pueden proporcionar las clases de utilidad necesarias.

• **Escritores de componentes**, que tienen experiencia en programar interfaces de usuario y prefieren crear componentes personalizados utilizando un lenguaje de programación. Los programadores expertos pueden crear sus propios componentes desde cero, o pueden extender los componentes estándares proporcionados por JavaServer Faces.

• **Vendedores de herramientas**, que proporcionan herramientas que mejoran la tecnología JavaServer Faces para hacer incluso más sencilla la construcción de interfaces de usuario en el lado servidor.

### 2.4.5 MODELO VISTA CONTROLADOR EN JSF

El patrón MVC (Modelo Vista Controlador), permite separar la lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) y la lógica de presentación (cómo interaccionar con el usuario).

Utilizando este tipo de patrón es posible conseguir más calidad, un mantenimiento más fácil, perder el miedo al folio en blanco (existe un patrón de partida por el que empezar un proyecto), etc. al margen de todo esto, una de las cosas más importantes que permite el uso de este patrón consiste en normalizar y estandarizar el desarrollo de Software.

En la figura 12 se muestra un esquema del patrón de diseño MVC:

[18]

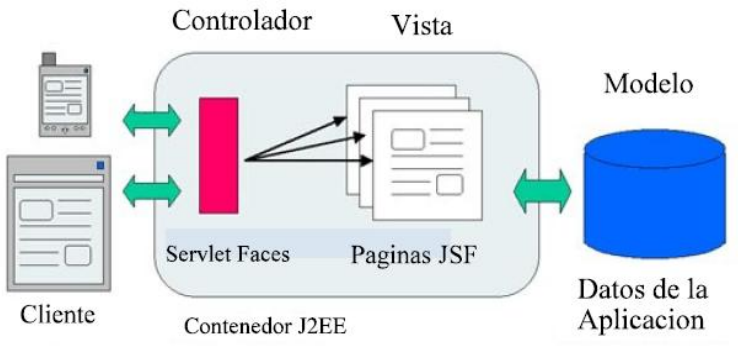


**Figura 12. Modelo vista controlador**

Este modelo de arquitectura presenta otras importantes ventajas:

• Hay una clara separación entre los componentes de un programa; lo cual admite implementarlos por separado.

---

[18] SICUMA: Sistemas de Información Cooperativos Universidad de Málaga. *Tutorial de JavaServer Faces*.P. 18 {En línea}. {10 de Diciembre de 2010}. Disponible en: http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf

- Se cuenta con una API muy bien definida; cualquiera que use la API, podrá reemplazar el modelo, la vista o el controlador, sin dificultad.

- La conexión entre el modelo y sus vistas es dinámica: se produce en tiempo de ejecución, no en tiempo de compilación.

#### *2.4.5.1 Modelo*

El modelo es el objeto que representa y trabaja directamente con los datos del programa: gestiona los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los diferentes controladores y/o vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a las vistas cuándo deben reflejar un cambio en el modelo.

#### *2.4.5.2 Vista*

La vista es el objeto que maneja la presentación visual de los datos gestionados por el Modelo. Genera una representación visual del modelo y muestra los datos al usuario e interacciona con el modelo a través de una referencia al propio modelo.

#### *2.4.5.3 Controlador*

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Entra en acción cuando se realiza alguna operación, ya sea un cambio en la información del modelo o una

interacción sobre la Vista. Se comunica con el modelo y la vista a través de una referencia al propio modelo.

Además, JSF opera como un gestor que reacciona ante los eventos provocados por el usuario, procesa sus acciones y los valores de estos eventos, y ejecuta código para actualizar el modelo o la vista.

19

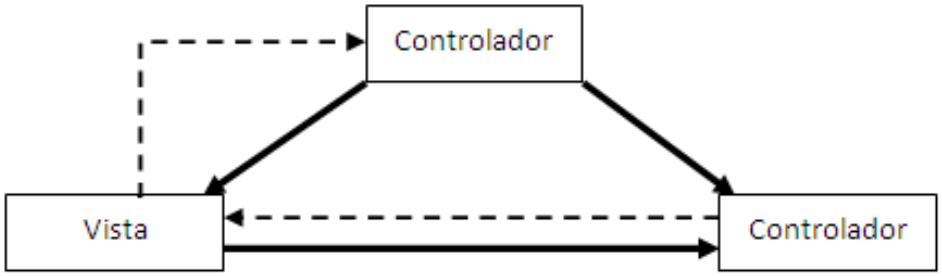


**Figura 13. Modelo Vista-Controlador**

En la figura 13 se puede observar las relaciones entre el Modelo, la Vista y el Controlador. Cabe destacar en la figura las líneas continuas que significan una relación directa como también las líneas discontinuas que implican una relación indirecta. También es importante tener en cuenta que aunque puede haber cierta "referencia indirecta" entre el Modelo y la Vista, el primero sigue sin saber nada del segundo.

El modo en que interactúan los tres elementos del modelo MVC se describe a continuación:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

[19] Imagen tomada de : http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (del modelo) a la vista aunque puede dar la orden a la vista para que se actualice.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

#### *2.4.5.4 Ciclo de vida de una página JavaServer Faces*

El ciclo de vida de una página JavaServer Faces page es similar al de una página JSP:

El cliente hace una petición HTTP de la página y el servidor responde con la página traducida a HTML. Sin embargo, debido a las características extras que ofrece la tecnología JavaServer Faces, el ciclo de vida proporciona algunos servicios adicionales mediante la ejecución de algunos pasos extras.

Los pasos del ciclo de vida se ejecutan dependiendo de si la petición se originó o no desde una aplicación JavaServer Faces y si la respuesta es o no generada con la fase de renderizado del ciclo de vida de JavaServer Faces.

En la figura 14 se visualiza todo el ciclo de vida de una petición-respuesta de una página JSF:

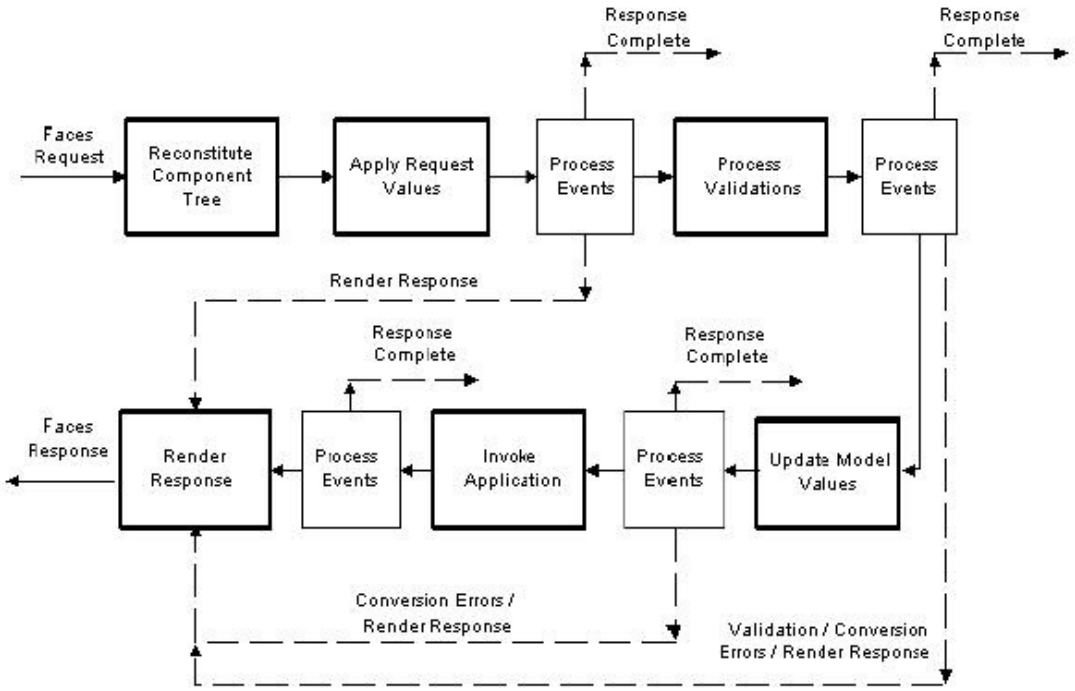


**Figura 14. Ciclo de vida petición-respuesta de una página JSF**

20

### ❖ *Ciclo de Vida Estándar de Procesamiento de Peticiones*

Conociendo lo que la tecnología JavaServer Faces realiza para procesar una página, un desarrollador de aplicaciones JavaServer Faces no necesitará preocuparse de los problemas de renderizado asociados con otras tecnologías UI. El ciclo de vida JavaServer Faces consta de las siguientes fases:

*Reconstituir el árbol de componentes*

---

20 Tutorial Java ServerFaces. {En línea}. {27 de Noviembre de 2010}. Disponible en: http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf

Cuando se hace una petición de una página JavaServer Faces, o cuando se produce un evento como pulsar sobre un enlace o un botón, el sistema JavaServer Faces entra en el estado reconstituir el árbol de componentes.
Durante esta fase, la implementación JavaServer Faces construye el árbol de componentes de la página JavaServer Faces, conecta los manejadores de eventos y los validadores y graba el estado en el FacesContext.

*Aplicar valores de la petición*

Una vez construido el árbol de componentes, cada componente del árbol extrae su nuevo valor desde los parámetros de la petición con su método decode. Enseguida, el valor se almacena localmente en el componente. Si la conversión del valor falla, se genera un mensaje de error asociado con el componente y se pone en la cola de FacesContext. Este mensaje se mostrará durante la fase *Renderizar la respuesta*, junto con cualquier error de validación resultante de la fase *Procesar validaciones*.

Si durante esta fase se produce algún evento, la implementación JavaServer Faces comunica estos eventos a los oyentes interesados.

En este punto, si la aplicación necesita redirigirse a un recurso de aplicación Web diferente o generar una respuesta que no contenga componentes JavaServer Faces, puede llamar a FacesContext.responseComplete.

Hasta el momento, se han puesto los nuevos valores en los componentes y los mensajes y eventos se han puesto en sus colas.

*Procesar validaciones*

Durante esta fase, el sistema JavaServer Faces procesa todas las validaciones registradas con los componentes del árbol. Examina los atributos del componente especificados por las reglas de validación y evalúa las reglas con los valores de dichos atributos. Si un valor incumple una regla, la implementación JavaServer Faces añade un mensaje de error al FacesContexty el ciclo de vida avanza directamente hasta la fase *Renderizar las respuestas* para que la página sea dibujada de nuevo incluyendo los mensajes de error. Si había errores de conversión de la fase aplicar los valores a la petición, también se mostrarán.

*Actualizar los valores del modelo*

Una vez que la implementación JavaServer Faces determina que el dato es válido, puede pasar por el árbol de componentes y actualizar los valores del modelo con los nuevos valores pasados en la petición. Sólo se actualizarán los componentes que tengan expresiones *valueRef*. Si el dato local no se puede convertir a los tipos especificados por los atributos del objeto del modelo, el ciclo de vida avanza directamente a la fase *Renderizar la respuesta*, durante la que se dibujará de nuevo la página mostrando los errores, similar a lo que sucede con los errores de validación.

*Invocar Aplicación*

Durante esta fase, la implementación JavaServer Faces maneja cualquier evento a nivel de aplicación, como enviar un formulario o enlazar a otra página.

En este momento, si la aplicación necesita redirigirse a un recurso de aplicación web diferente o generar una respuesta que no contenga componentes JavaServer Faces, puede llamar a FacesContext.responseComplete.

Posteriormente, la implementación JavaServer Faces configura el árbol de componentes de la respuesta a esa nueva página y, por último, transfiere el control a la fase *Renderizar la Respuesta*.

*Renderizar la Respuesta*

Durante esta fase, la implementación JavaServer Faces invoca los atributos de codificación de los componentes y dibuja los componentes del árbol de componentes grabado en el FacesContext.

Si se encontraron errores durante las fases *Aplicar los valores a la petición*, *Procesar validaciones* o *Actualizar los valores del modelo*, se dibujará la página original. Si las páginas contienen etiquetas output_errors, cualquier mensaje de error que haya en la cola se mostrará en la página.

### 2.4.6 SEAM

Seam es un framework de aplicaciones de Java Enterprise. Inspirada en los siguientes principios:

- Un tipo de cosas

Seam define un modelo de componentes uniformes para la lógica del negocio en su aplicación. Un componente seam puede ser con estado, que se encuentra asociado a cualquiera de los diversos contextos bien definidos, incluyendo el de larga duración, el contexto de persistencia, de procesos de negocio y el contexto de la conversación, que se conserva en todas las solicitudes múltiples en una interacción con el usuario.

En seam no hay distinción entre los componentes del nivel de presentación y componentes de la lógica del negocio. El desarrollador puede estratificar la aplicación de acuerdo con la arquitectura que se haya diseñado.
Los componentes seam pueden simultáneamente tener acceso al estado asociado a la solicitud web y al estado de recursos transaccionales.

- Integrar JSF con EJB 3.0

EJB 3 es un modelo de componentes a nivel de lógica de negocio y de persistencia, del lado del servidor, mientras que JSF es un modelo de componentes de la capa de presentación.

JSF y EJB3 funcionan mejor juntos, a pesar de que Java EE5 no proporciona un estándar para integrar estos modelos de componentes.No obstante los creadores de ambos modelos (JSF y EJB3) proporcionan puntos de extensión estándar para permitir la integración con otros marcos.

Seam unifica los modelos de componentes de JSF y EJB3, dejando al desarrollador el único problema de diseñar la lógica del negocio.

- Integrar AJAX

Seam soporta mejor las soluciones de código abierto basado en AJAX JSF: JBossRichFaces e ICEfaces. Estas soluciones le permiten agregar la capacidad de AJAX para la interfaz de usuario sin necesidad de escribir código JavaScript.

Por otra parte, seam proporciona una capa incorporada del Javascript que le permite llamar a los componentes JavaScript de forma asincrónica del lado cliente sin la necesidad de una capa de acción intermedia.

Estos enfoques funcionan bien porque seam incorporo la gestión de concurrencia y el estado, que aseguran que las peticiones asincrónicas Ajax se manejan de forma segura y eficiente en el servidor.

- Procesos de negocio como el primer constructor de clase

Seam ofrece transparencia en la gestión de procesos de negocio a través de JBPM, incluso permite definir la capa de presentación pageflow utilizando el mismo lenguaje que jBPM utiliza para la definición de procesos de negocio.

- Biyección

La biyección es dinámica y bidireccional, podríamos pensar en esto como un mecanismo de alias para variables contextuales (nombres en alguno de los contextos enlazados al hilo de ejecución actual) a atributos del componente.

La biyección permite auto ensamblaje de componentes con estado por el contenedor, esto incluso permite a un componente asegurar y fácilmente manipular el valor de una variable contextual, simplemente asignándola a un atributo del componente

- Preferir anotaciones a XML

La comunidad java ha estado confundida sobre el tipo de meta información con la que cuenta la configuración; las anotaciones de java han logrado cambiar esto.

EJB 3.0 abarca las anotaciones y la configuración de excepción como la forma más fácil de proporcionar información al contenedor en forma declarativa. SEAM extiende las anotaciones de EJb3 con una serie de anotaciones para la administración del estado declarativo y demarcación del contexto declarativo.

- La prueba de integración es fácil

Los componentes de seam, siendo simples clases Java, son por naturaleza comprobables. Sin embargo, para aplicaciones complejas, las pruebas unitarias por sí solas son insuficientes. SEAM proporciona la capacidad de prueba de aplicaciones seam como una característica central del framework. El usuario puede escribir las pruebas JUnit o TestNG que reproducen una interacción completa con un usuario. Estas pruebas pueden ser ejecutadas directamente en el IDE, donde seam automáticamente implementa los componentes EJB con JBoss Embebido.

- Las especificaciones técnicas no son perfectas

Existen limitaciones en el ciclo de vida de JSF para las solicitudes GET que fija seam. Los autores de seam actualmente trabajan con expertos JCP para asegurarse de que las correcciones pertinentes sean realizadas para la próxima versión.

- Hay más de una aplicación web que sirve páginas HTML

Un framework de aplicaciones web realmente completo debe abordar problemas como la persistencia, la concurrencia, a sincronía, administración del estado, seguridad, correo electrónico, mensajería, pdf, generación de gráficos, servicios web, caché, entre otros.

Seam integra JPA e Hibernate3 para persistencia, EJB TimerService y Quartz para ligeras a sincronías, jBPM para flujo de trabajo, JBoss rules para reglas del negocio, Meldware Mail para correo electrónico, HibernateSearch y Lucene para búsqueda de texto, JMS para mensajes y JBoss Caché para la página de almacenamiento en caché.

21

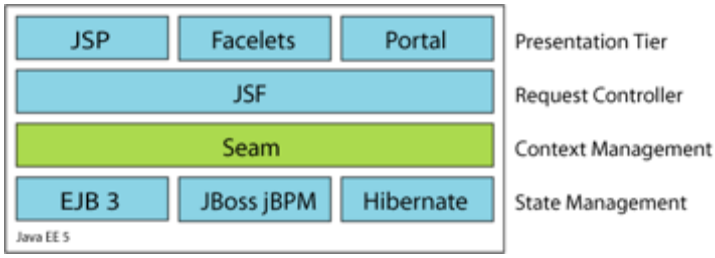


**Figura 15. Framework Seam**

21 JBoss. Página oficial Seam. *Community documentation.* {En línea}. {27 de Noviembre de 2010}. Disponible en: http://docs.jboss.org/seam/2.1.2/reference/en-US/html_single/#Book-Preface

### *2.4.6.1 EL MODELO DE COMPONENTES CONTEXTUALES*

Para conocer el núcleo del Framework de Seam es necesario entender dos conceptos esenciales: los contextos y los componentes.

- Contextos de Seam*:*

Los contextos son contenedores gestionados por el mismo framework de Seam en los cuales residen todos los componentes instanciados y que pueden ser demarcados por medio de anotaciones.

**Contexto Sin Estado o Stateless*:*** Es un contexto que contiene únicamente componentes sin estado (Stateless) lo que significa una ausencia de contexto ya que las resoluciones de una instancia de Seam no son almacenadas. No obstante se han desarrollado y utilizado ya que son una parte importante de cualquier aplicación de Seam.

**Contexto de Evento**: Considerado como el contexto de estado "más estrecho", proporciona una generalización de la noción de contexto de petición Web para cubrir otros tipos de eventos. Un claro ejemplo de contexto de evento tiene que ver con el ciclo de vida de una petición JSF en el cual los componentes invocados por una solicitud JSF en un contenedor de evento, son eliminados inmediatamente después responder con la petición.

**Contexto de Página**: Este contexto permite asociar el estado de un componente con la instancia del llamado de una página, es decir, Seam crea el contexto al momento de direccionar una página. Es muy útil al momento de requerir listas de

hacer clic (Combo Box) donde cada lista es devuelta por el cambio en los datos en el servidor. Los componentes perduran mientras se permanezca en la misma página.

**Contexto de Conversación**: Es posiblemente el lugar ideal para guardar el estado de una aplicación ya que permite al desarrollador implementar casos de uso relativamente extensos o realizar transacciones relativamente largas. Una "conversación" es una unidad de trabajo desde el punto de vista del usuario, lo que abarca varias peticiones e incluso varias transacciones con la base datos. La conversación mantiene su estado asociándolo a cada ventana en forma individual con el fin de evitar posibles colisiones entre conversaciones, ya que un usuario puede estar manejando múltiples conversaciones al mismo tiempo (por ejemplo en el caso de tener la misma página en más de una instancia del navegador). Las conversaciones también se conocen como "tareas", en consecuencia, una tarea es una conversación significativa en términos de un proceso de negocio extenso y tiene el potencial para disparar una transición del estado de un proceso de negocio cuando este es satisfactoriamente culminado. Seam controla el estado de las conversaciones mediante un tiempo de espera que se puede configurar con el propósito de evitar el incremento de conversaciones inactivas de un usuario en sesión; dado el caso que un usuario abandone la conversación. Otra característica particular del contexto de conversación es la posibilidad de tener conversaciones anidadas; en otras palabras una conversación contenida dentro de otra mayor.

**Contexto de Sesión**: Mantiene el estado de los componentes asociados al inicio de sesión de un usuario. Este contexto puede ser asociado con la interface HttpSession, sin embargo el contexto de sesión de Seam fue diseñado para manejar la Sincronización (Serialización de peticiones para evitar colisiones de solicitudes) y la Clusterización (facilidad en la distribución de componentes); ventajas que le confieren una verdadera robustez a cualquier aplicación web.

**Contexto de Proceso de Negocio**: Se encuentra asociado con una ejecución de proceso de negocio largo que abarca múltiples interacciones entre múltiples usuarios lo que implica que el estado sea compartido, manejado y persistido por el motor BPM (Business Process Management). Seam carece de anotaciones para realizar la demarcación de este contexto por lo que se debe manejar en forma externa utilizando un Lenguaje de Definición de Procesos.

**Contexto de Aplicación**: Es el contexto más general de la especificación de servlets. Este contexto se utiliza generalmente para guardar información estática como los datos de configuración, de referencia o meta-modelos. Seam hace uso de este contexto al establecer su propia configuración y meta-modelos.

*Prioridad de búsqueda de los Contextos Seam*

Algunas veces las instancias de componentes son obtenidas desde un ámbito (Scope) particular conocido, pero cuando no se conoce el ámbito del componente a instanciar Seam consulta en todos los Scopes con estado bajo un cierto orden de prioridad:

- Contexto de Evento.
- Contexto de Página.
- Contexto de Conversación.
- Contexto de Sesión.
- Contexto de Proceso de Negocio.
- Contexto de Aplicación.

❖ ***Componentes Seam:***

Los componentes son objetos con estado (Stateful), generalmente son EJBs (Enterprise JavaBeans), cuya instancia implica una asociación con un contexto en la cual a cada objeto se le asigna una única identidad.

Los componentes Seam son POJOs (acrónimo de Plain Old Java Object), es decir, son instancias de clases que no extienden o implementan nada adicional (como la implementación de interfaces en Hibernate). A pesar de que Seam es un framework desarrollo para integrarse profundamente con el estándar EJB 3.0, sus componentes pueden utilizarse por fuera del contenedor EJB 3.0.

**Bean de Sesión con Estado:** Estos componentes no solo son capaces de mantener el estado de la aplicación a través de múltiples invocaciones a un Bean sino también a lo largo de múltiples peticiones. Una de las características exclusivas de Seam es la manera como se mantiene la información de una conversación en curso, ya que esta es almacenada en variables de instancia de Sesión Stateful asociadas a este contexto, en lugar de adherirla directamente en el HttpSession.

A menudo los Bean de Sesión con Estado son utilizados como oyentes de acciones JSF aunque nunca deberían ser enlazados ni con el contexto de página ni con el contexto Stateless. Además en el ámbito de sesión, las peticiones concurrentes de Beans de Sesión Stateful estarán serializadas evitando posibles colisiones; siempre y cuando los interceptores de Seam no estén deshabilitados para el Bean.

**Beans de Entidad**: Los Beans de Entidad pueden ser ligados a una variable de contexto o a una función como componentes Seam. Como las entidades tienen una identidad de persistencia adicional a su identidad contextual, las instancias de

entidad están explícitamente ligadas al código Java, en lugar de ser creadas implícitamente por el framework.

Como los Beans de Entidad son componentes que no soporta la biyección no suelen usarse como oyentes de acciones JSF pero si pueden emplearse como Beans de soporte para proveer propiedades a los componentes JSF tanto en el envío como en el despliegue de formularios.

Los Beans de Entidad están destinados al contexto de conversación por defecto y nunca debe pertenecer a un contexto Stateless; también hay que tener en cuenta que es más eficiente tener una referencia al Bean de entidad en un Bean de Sesión Stateful en ambientes distribuidos.

**Java Beans**: Estos componentes pueden ser utilizados con los Beans de Sesión Stateful, sin embargo no cuenta con las mismas funcionalidades de este último, entre las que se encuentran:

- Demarcación de transacciones declarativa.
- Seguridad declarativa.
- Replicación del estado distribuido eficiente.
- Persistencia EJB 3.0.
- Métodos de Tiempo de Espera.

❖ ***Modelo de Concurrencia***

En la actualidad las aplicaciones web deben lidiar con la concurrencia de solicitudes ya que estas deben soportar procesos asíncronos como las peticiones AJAX y en consecuencia Seam debe gestionar algunos Contextos contemplando la posibilidad de una colisión entre solicitudes.

Los contextos de Sesión y Aplicación son multi-hilo en tanto que los contextos de evento y de página son naturalmente de un solo hilo. No obstante en el contexto de conversación se debe proteger de las solicitudes concurrentes y es por ello que Seam maneja un modelo de un sólo hilo por proceso para cada conversación, identificando cada solicitud con un serial para tener un control adecuado sobre las peticiones concurrentes.

### *2.4.6.2 Eventos, interceptores y el manejo de excepciones*

❖ ***Eventos seam***

El modelo de componentes Seam fue desarrollado para su uso con aplicaciones orientadas a eventos, específicamente para permitir el desarrollo de componentes de grano fino, débilmente acoplado en un modelo de grano fino. Los eventos en seam son de varios tipos:

- JSF eventos
- jBPM eventos de transición
- Seam acciones de página

- Seam impulsado en componentes eventos
- Seam contextuales eventos

❖ ***Página de acciones***

Una acción de página seam es un evento que ocurre antes de redirigir una página. El método de acción de página puede devolver un resultado JSF. Si el resultado no es nulo, seam utiliza las reglas de navegación que se hayan definido para navegar a una vista.

La identificación mencionada en el elemento <page> no tiene por qué corresponder a una página real o una página JSP Facelets, por lo tanto, se puede reproducir la funcionalidad de un marco tradicional orientado a la acción como Struts o WebWork con acciones de página. Esto es muy útil para hacer cosas complejas en respuesta a la non-faces (por ejemplo, las solicitudes HTTP GET).

❖ ***Página de parámetros***

Una petición a JSF Faces (envío de un formulario) encapsula una acción y los parámetros de entrada. Los parámetros de página pueden utilizarse con o sin acción.

Navegación:
Puede utilizar las reglas estándar de navegación JSF se define en el faces-config.xml en una aplicación de seam. Sin embargo, las reglas de navegación JSF tienen una serie de limitaciones:

- No es posible especificar parámetros de la petición usada para redirigir la página.

- No es posible iniciar o finalizar las conversaciones de una regla.
- En la evaluación del método de retorno no es posible evaluar una expresión EL arbitraria.

❖ ***Eventos impulsados por componentes***

Los componentes de seam pueden interactuar simplemente llamando a cada uno de los demás métodos. Los componentes con estado pueden incluso poner en práctica el modelo observador/observable. Pero permitir que los componentes que interactúen de una manera débilmente acoplado es posible cuando los componentes llaman a otros métodos directamente. Seam proporciona eventos impulsados por componentes.

❖ ***Eventos Contextuales***

Seam define una serie de funciones que pueden ser usados para tipos especiales de integraciones con el framework. Algunas de ellas son:

- org.jboss.seam.validationFailed: invocada cuando falla la validación JSF.
- org.jboss.seam.noConversation:invocada cuando no hay una conversación larga y es requerida.
- org.jboss.seam.postDestroyContext.<SCOPE> : invocada después de<SCOPE> el contexto es destruido.
- org.jboss.seam.beforePhase: llamada antes de iniciar una fase JSF.
- org.jboss.seam.security.notLoggedIn: llamada cuando el usuario no se ha autenticado y se requiere la autenticación.

❖ ***Interceptores Seam***

El ciclo de vida de los interceptores con estado es el mismo del componente interceptado, no es necesario mantener el estado de los interceptores y para esto seam permite optimizar el rendimiento permitiendo especificar el estado del interceptor.

Muchas de las aplicaciones implementadas por seam incluyen algunos interceptores, estos ya deben ser llamados en sus componentes.

❖ ***Administrar excepciones***

JSF es limitado en el manejo de excepciones, para esto seam permite definir una clase particular para anotar la clase de excepción o se declara en un archivo .xml. este servicio se combina con la anotación ApplicationException EJB 3.0 que especifica si la excepción debe provocar una reversión en la transacción.

### *2.4.6.3 Conversaciones y gestión de espacio de trabajo*

❖ ***Modelo de conversación seam***

Seam se propaga de forma transparente en el contexto de conversación mediante las devoluciones de datos JSF y las redirecciones. Si no se hace nada especial o no realiza una petición el contexto de la conversación no se propaga y se procesa en una nueva conversación temporal; este es el comportamiento deseado.

Si desea propagar la conversación seam a través de una solicitud non-faces se debe codificar explícitamente la conversación seam identificando el parámetro de la petición.

El modelo de conversación seam facilita crear aplicaciones con múltiples ventanas; algunas de estas aplicaciones requieren adicionalmente:

- La conversación se extiende por unidades más pequeñas, que se ejecutan en serie o a la vez.
- El usuario puede alternar en diferentes conversaciones dentro de una misma ventana, esto se llama gestión del área de trabajo.

❖ ***Conversaciones anidadas***

Se crean invocando el método @Begin(nested=”true”) dentro del ámbito de la aplicación de una conversación existente. Una conversación anidada posee su propio contexto de conversación, pero puede leer los valores de contexto de la conversación exterior que en este caso son de solo lectura.

- La anidación de una conversación se inicia en un contexto dentro de la conversación externa, esta es considerada como la conversación padre.
- Los objetos cargados directamente en el contexto de la conversación anidada no afectan los objetos accesibles en la conversación padre.
- La inyección en un contexto de búsqueda en la primera conversación, busca el valor en el contexto de la conversación en curso, si no encuentra ningún valor continúa por la conversación en pila si la conversación es anidada.

La conversación anidada se destruye y se reanuda la conversación externa cuando encuentra la etiqueta @End.

Las conversaciones pueden ser anidadas a cualquier profundidad arbitraria.

❖ ***Iniciando conversaciones con la solicitud GET***

JSF no define ningún tipo de detector de acción cuando una página se accede a través de una solicitud non-faces, esto puede ocurrir si el usuario marca la página o si accede a ella a través de un link.

Cuando se desea iniciar una conversación al acceder a una página, dado que no existe un método de acción JSF no se puede resolver el problema con la anotación @Begin.

Si la página necesita algún estado de la variable de contexto y este se lleva a cabo en un componente de seam, el estado puede alcanzarse en el método @Create. Si no, puede definirse el método @Create.

Si ninguna de las opciones funciona, seam permite definir una página de acciones en el archivo pages.xml.

❖ ***Una conversación de larga duración***

Algunas páginas requieren una conversación de larga duración, para esto seam ha incorporado un mecanismo para cumplir este requisito.

En el descriptor de seam, puede indicar que la conversación actual puede ser de larga duración (o anidados).

Cuando seam determina que una página es solicitada fuera de una conversación de larga duración toma las siguientes medidas.

- Un evento contextual llamado org.jboss.seam.noConversation es levantado.
- Se registra un mensaje de alerta con el conjunto de claves org.jboss.seam.NoConversation.
- Si se define el usuario es redirigido a una página alternativa.

#### *2.4.6.4 NAVEGACION EN SEAM*

❖ ***Modelo de navegación Stateless***

Existen dos formas para definir un modelo de navegación Stateless: (a) por medio de las reglas de Seam o (b) utilizando las reglas de JSF.

El modelo Stateless define un conjunto de salidas lógicas y salidas de nombre de un evento directamente a la página resultante de la vista. Las reglas de navegación son totalmente ajenas a cualquier estado en poder de la aplicación aparte de la página fuente del evento. Esto implica que los métodos de acción oyente (actionlistenermethods) deben en algunos escenarios tomar decisión con respecto al flujo de página, ya que sólo tienen acceso al estado actual de la aplicación.

Cuando una aplicación utiliza la paginación Stateless, Seam le permite al usuario navegar libremente por medio de los botones: *Regresar, Adelante* o *Refrescar*; siempre y cuando la aplicación se responsabilice de permanecer en el estado conversacional internamente consistente cuando se utilizan los botones anteriormente mencionados.

Una ventaja de implementar este tipo de navegación radica en utilizar reglas simples, de ser flexible, y además, de permitir el retorno de IDs de vista desde los

métodos de acción oyente. No obstante, la paginación Stateless no soporta procesos de negocios complejos.

❖ ***Modelo de Navegación Stateful***

Define un conjunto de transiciones entre un conjunto de estados lógicos y estados de nombres de la aplicación. Este modelo está orientado a los procesos de negocio en el que es posible establecer el flujo producido por cualquier interacción del usuario, en su totalidad, con la definición del flujo de página de JPDL e incluso escribir métodos de acción oyentes que desconocen por completo el flujo de la interacción. JPDL es un lenguaje xml simple y de fácil lectura, el cual es instaurado por el Motor de Gestión de procesos de Negocios de Jboss (JBPM). JPDL logra resolver los siguientes problemas:

- Definición del flujo de página involucrado en complejas interacciones de usuario (definir el flujo de página de una conversación en particular).
- Definición de los procesos de negocio globales (cuando los procesos de negocios involucran múltiples conversaciones con múltiples usuarios simultáneamente).

22

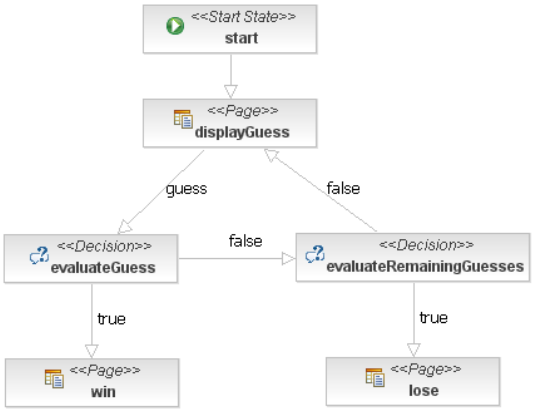


**Figura 16. Modelo de Navegación Stateful**

22 JBoss. Página oficial Seam. *Community documentation.* {En línea}. {27 de Noviembre de 2010}. Disponible en: http://docs.jboss.org/seam/2.1.2/reference/en-US/html_single/#d0e6695

A pesar de que la navegación Stateful tiene un diseño para soportar cualquier interacción del usuario, al mismo tiempo es un poco más restringida que la Stateless, ya que todos los posibles eventos deben estar definidos por JPDL y no pueden interactuar con métodos de acción oyente que personalizan las interacciones del usuario sobre la aplicación.

#### *2.4.6.5 SEAM Y EL MAPEO OBJETO-RELACIONAL*

Seam provee un extenso soporte para las Arquitecturas de Persistencia más importantes de Java: (a) Hibernate 3 y (b) Java Persistence API presentada por EJB 3.0. Entender cuál es la relación de Hibernate 3 y EJB 3.0 con Seam, o como se integran estos ORM con el Framework es de vital importancia para conocer las ventajas más importantes de Seam.

Una de las razones que impulsaron el desarrollo de Seam fue la dificultad que tenían otros Frameworks como Spring (que utilizaba Hibernate) para persistir los objetos, ya que cada transacción con la base de datos era atómica, es decir, que cada vez que una transacción finalizaba no sólo implicaba una interacción directa con la base de datos sino que también marcaba la pérdida del contexto de persistencia.

- ***Transacciones gestionadas por Seam***

El ORM EJB 3.0 fue el primero en introducir componentes Stateful (Bean de Sesión Stateful) con un contexto de persistencia extendido asociado al tiempo de vida del componente; pero esta era una solución parcial del problema de la persistencia ya que EJB 3.0 tenía los siguientes inconvenientes:

- El ciclo de vida de un Bean de Sesión Stateful debía ser manejado manualmente por medio de código en la capa Web (un problema sutil que era más difícil en la práctica de lo que parecía).
- La propagación del contexto de persistencia entre componentes Stateful en las transacciones era posible pero difícil.

No obstante Seam resuelve el primer inconveniente asociando los componentes de sesión Stateful a la conversación, pero la segunda falencia en los componentes de EJB, Seam pudo resolverlo trabajando mancomunadamente con Hibernate (otro ORM) cuyos resultados proporcionaron las siguientes soluciones:

- Usar el contexto de persistencia extendido en el ámbito de la conversación, en lugar de asociarlo a la transacción.
- Usar dos transacciones por solicitud: la primera abarca desde la restauración de la fase de vista hasta el final de la invocación de la fase de aplicación; y la segunda transacción abarca la fase de devolución de la respuesta.

*Sincronización de la Transacción*

La sincronización de transacciones tiene que ver con la devolución de llamadas (callbacks) que producen los eventos que inician y terminan dichas transacciones. Por defecto, Seam utiliza su propio componente de sincronización de transacciones el cual se usa explícitamente al momento de enviar una petición para que las devoluciones de llamada sean correctamente ejecutadas.

❖ ***Contextos de Persistencia gestionados por Seam***

Cuando se utiliza Seam en otro ambiente diferente a Java EE5 no se puede confiar el manejo del ciclo de vida del contexto de persistencia al contenedor, e

incluso, aun trabajando con la plataforma Java EE5 la propagación del contexto de persistencia entre los componentes es difícil y propensa a errores.
De cualquier manera se necesita un “Contexto de persistencia administrado” (especificado por JPA) o una “Sesión administrada” (de Hibernate) en los componentes.

El contexto de persistencia administrado por Seam es justo un componente Seam integrado que maneja una instancia del *EntityManager* (en JPA) o del *Session* (en Hibernate) el cual se puede inyectar en el momento deseado con la anotación *@In.*

Los contextos de persistencia que son gestionados por Seam son extremadamente eficientes en ambientes distribuidos, pues Seam es capaz de realizar una optimización de la especificación EJB3.0 y es que los contenedores se puedan usar para administrar contextos de persistencia extendidos. Seam también puede hacerse cargo de los errores en el contexto de persistencia extendido con la ventaja de llevar a cabo esta tarea en forma transparente en donde no existe la necesidad de replicar cualquier estado contexto de persistencia entre los nodos.

#### *2.4.6.6 EL MARCO DE APLICACIONES SEAM*

Con seam es más sencillo crear aplicaciones escribiendo clases java con anotaciones. El marco de aplicaciones de seam permite reducir la cantidad de código necesario para acceder a la base de datos en una aplicación web, para esto se usa Hibernate o JPA.

- ***Objetos Principales***

Proporcionan operaciones de persistencia para una entidad en particular, las operaciones pueden ser: persist (), remove (), update () y getInstance (). Antes de usar remove o update se debe establecer el identificador del objeto con el método SetId().

- ***Controlador de objetos***

Una parte opcional del frameworkseam es la clase del controlador y sus subclases EntityControllerHibernateEntityController y BusinessProcessController, que ofrecen algunos métodos de conveniencia para el acceso a los componentes integrados.

### *2.4.6.7 ALMACENAMIENTO EN CACHÉ DE SEAM*

En la mayoría de las aplicaciones empresariales, la base de datos es el principal cuello de botella como también es la capa menos escalable en el entorno de ejecución. En conclusión es casi imposible que una aplicación sea escalable mientras la interacción con la base de datos sea ostensible. Por lo tanto la escalabilidad de cualquier aplicación se puede favorecer si se disminuyen las transacciones directas con la base de datos, y esa es la principal misión de la caché.

***Almacenamiento en Caché multi-capa***

Con Seam se puede planificar para configurar un almacenamiento en una caché de manera individual para cada una de las capas de la aplicación.

- ***Caché de la Base de Datos***

La base de datos tiene su propia caché asignada pero a diferencia de la caché en la capa de Aplicación no es tan escalable.

- ***Caché de segundo nivel***

Independientemente del ORM que se emplee, en una aplicación Seam se dispone de una caché de segundo nivel de los datos de la base de datos, sin embargo es a menudo defectuosa. Su diseño la hace favorable en ambientes distribuidos en donde múltiples usuarios podrían utilizar los datos de esta caché siempre y cuando las modificaciones en dichos datos sean muy pocas.

- ***Caché a nivel de Contexto***

El contexto de conversación en Seam es una caché del estado conversacional. Los componentes que son inyectados en el contexto de conversación pueden mantener almacenado el estado relacionado con la interacción del usuario actual.

En particular, el contexto de persistencia actúa como un caché de los datos que han sido leídos en la conversación actual. Seam optimiza las respuestas de sus propios contextos de persistencia en un ambiente distribuido y no hay ningún requisito para mantener la consistencia transaccional con la base de datos.

Las aplicaciones pueden guardar estado transaccional el componente *cacheProvider* (Proveedor de caché) de Seam y este estado puede ser visible si la caché soporta el trabajo en un ambiente clusterizado. También pueden almacenar un estado no transaccional en el contexto de aplicación de Seam, el cual es invisible para los otros nodos del cluster.

- ***Caché a nivel de JSF***

Seam permite el almacenamiento en caché de los fragmentos recargados de una página JSF. A diferencia del segundo nivel de caché del ORM, este caché no es automáticamente inhabilitada cuando los datos cambian, así que su invalidación debe ser explícita en el código de la aplicación o establecer las políticas de expiración apropiadas.

### 2.4.7 Hibernate y ORM

A pesar de la innegable popularidad de los lenguajes orientados a objetos, las bases de datos relacionales han dominado el mercado por mucho tiempo. Esto ha dado lugar a un desfase entre las tecnologías y herramientas que procesan la información de un sistema y las que la almacenan. Se han intentado crear iniciativas de bases de datos orientadas a objetos pero no han tenido resultados afortunados. Entonces se ha optado por una alternativa diferente: El software de mapeo objeto-relacional (ORM por Object-relationalmapping).

Un software de ORM permite crear una correspondencia entre la información en una base de datos y objetos del lenguaje de programación en que se desee desarrollar una aplicación. Así, se crea la ilusión de una base de datos en memoria que el desarrollador puede manipular mediante técnicas y estructuras de datos típicas de la POO: Clases, métodos, casting, etc.

Los ORM también minimizan el impacto de diferencias radicales entre los objetos y las entidades de una base de datos relacional. Por ejemplo, un objeto es una estructura de datos que no sólo almacena valores, sino que se comporta diferente de acuerdo a los valores que tenga o reciba en un método. El ORM también se encarga de hacer las validaciones y conversiones necesarias entre tipos de datos de la base de datos y la máquina que ejecuta la aplicación, considerando que los tipos de datos primitivos de datos son de diferentes tamaños en diversos sistemas operativos y que las máquinas que albergan la aplicación y la base de datos pueden ser diferentes. El software ORM se vale de las ventajas de la encapsulación y los métodos en POO para ofrecer tal robustez.

Para Java, existe la librería Hibernate, software libre distribuido bajo la Licencia pública general reducida de GNU. Hibernate permite mapear las entidades de una base de datos a clases Java. También maneja todo tipo de cardinalidad de relaciones entre entidades, incluso relaciones reflexivas. Gracias a Hibernate, el código de lenguaje de consultas a introducir se simplifica en cantidad y complejidad de manera considerable, y facilita interactuar con la base de datos como si fuera esta tan sólo un objeto más en memoria. Además, como todo ORM, Hibernate se encarga del flujo de datos de la aplicación a la base de datos, efectuando todas las operaciones necesarias para que las claves, los datos, sus tipos y tamaños correspondan a las reglas de la base de datos establecidas en el mapeo, garantizando el éxito de las transacciones.

Hibernate permite el mapeo de entidades mediante archivos descriptores XML o mediante JPA (Java Persistence API). La segunda es la usada en el presente trabajo de grado.

**JPA**

La API (ApplicationProgrammers Interface) de persistencia de Java es el esfuerzo del grupo Java EE por brindar una solución integradora de todos los motores de ORM que existen para Java, puesto que Hibernate es sólo uno de una gran variedad. Su funcionalidad se basa en POJOs (Plain Old Java Objects) que no son más que objetos tradicionales de Java. El corazón de la funcionalidad del API se basa en anotaciones (palabras clave que inician con @) que le dan una característica especial a cada miembro de la clase, correspondiente con la base de datos relacional. Por ejemplo, la anotación @Id en la línea superior de la definición de un miembro, especifica la llave primaria de la entidad. De la misma manera, las anotaciones sirven para definir llaves foráneas, cardinalidad de las relaciones, restricciones del valor de los datos, entre otras funciones. Todo esto reduce signficativamente el código Java a escribir.

Una característica fascinante de JPA, es que permite que se hagan cambios al diseño de la base de datos sin tener que reescribir enteramente las aplicaciones. Para esto JPA introduce:

**JPQL**

Java PersistenceQueryLanguage el cual es el lenguaje de consultas que se usará dentro de la aplicación. Con él, el desarrollador jamás se refiere a las entidades directamente al momento de hacer las consultas, sino a los objetos mismos que fueron mapeados. Si el diseño de la BD cambiara, sólo habría que modificar las anotaciones de las entidades. El resto del código quedaría intacto y seguiría funcionando tal y como antes. Esta facilidad permite optimizar las bases de datos cuando se considere necesario, migrar a bases diferentes, o sencillamente corregir diseños defectuosos con un esfuerzo mínimo.

# CAPITULO 3

# 3 METODOLOGÍA DE DESARROLLO

## 3.1 Ciclo de vida del proyecto

### 3.1.1 Análisis de Requerimientos

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel de sistema y el diseño de programas.

En el análisis de requerimientos se especificará, junto con los funcionarios de la Oficina de Planeación, la función y comportamiento que deberán tener los programas a desarrollarse en el transcurso del proyecto, se indicará la interfaz con otros elementos del sistema y se establecerán los estándares de diseño que debe cumplir el sistema.

El análisis de requerimientos permite la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra los medios para valorar la calidad de los programas, una vez que se haya construido.

Es en esta etapa donde se harán reuniones periódicas con los funcionarios de la Oficina de Planeación, para que sean ellos los que definan las características del software que se desea desarrollar y que se adapte plenamente a las exigencias de la Gestión de proyectos en la Universidad Industrial de Santander.

### 3.1.2 Diseño

El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo).
En esta etapa de diseño se hace una traducción de los requisitos a una representación del software donde se pueda evaluar su calidad antes de que comience la codificación.

El diseño se efectuará, mediante modelos UML (Lenguaje de Modelado Unificado) que incluirá los diagramas que han sido seleccionados dentro de los estándares de desarrollo de software utilizados en la División de Servicios de Información, que son: de casos de uso, de clases y de secuencia, utilizando la herramienta de modelaje Enterprise Architect.

### 3.1.3 Implementación de la Aplicación

En esta etapa se procede a generar el software que se ha diseñado teniendo en cuenta los parámetros establecidos por la división de Servicios de Información, en cuanto a los estándares técnicos y de calidad que caracterizan las aplicaciones que son generadas para el servicio de la Universidad, teniendo como base la Arquitectura de Desarrollo de Aplicaciones de JAVA EE5 y la plataforma Informix como motor de base de datos.

### 3.1.4 Pruebas de Software

Son los procesos que permiten verificar la calidad de un producto software y el cumplimiento de los requerimientos establecidos en la fase de análisis de requerimientos.

Las pruebas de software se integran dentro de las diferentes fases del ciclo de desarrollo del software establecidas en la ingeniería de software.

Una vez generado el código, comienzan las pruebas, proceso utilizado para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Las pruebas se centran en los procesos lógicos internos del software, asegurando que todas las sentencias sean probadas y asegurar que la entrada definida produce los resultados esperados.

Las pruebas serán de carácter permanente a lo largo del desarrollo del proyecto por parte del equipo de trabajo, y habrá un periodo de tiempo para que los usuarios finales interactúen con la aplicación y detecten posibles ajustes.

### 3.1.5 Ajustes

Después de las Pruebas, cada uno de los errores detectados o las observaciones hechas por los usuarios debidamente analizadas, deben ser tenidas en cuenta para ajustar el sistema, de tal manera que se adapte plenamente a las necesidades de los mismos.

También estos se harán permanentemente a lo largo del desarrollo del proyecto a la par con las pruebas, en la medida en que se detecten errores o inconsistencias en el sistema que se ha desarrollado.

## 3.2 Metodología de Desarrollo

### 3.2.1 Modelo de construcción por prototipos.

Se eligió esta metodología debido a que es muy frecuente que los usuarios que están solicitando el sistema, en este caso la Oficina de Planeación, definan un conjunto de objetivos generales para el software, pero no identifican los requisitos detallados de entrada, proceso o salida. En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficacia de un algoritmo, o no haber comprendido plenamente el requerimiento del usuario. Para éstas y otras muchas situaciones, un *paradigma de construcción por prototipos* puede ofrecer el mejor enfoque, ya que la entrega de prototipos, que hacen parte integral del proyecto en su conjunto, permitirán la corrección temprana de errores o la redefinición del sistema en caso de ser necesario, y los prototipos funcionales permitirán la familiarización del usuario con el sistema que se está desarrollando.

A continuación se observa la estructura del MODELO:

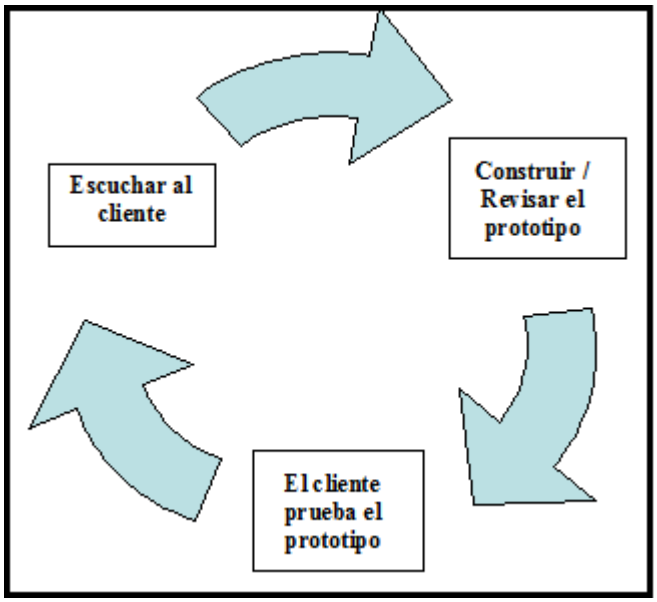


**Figura 17. Modelo Construcción Por Prototipos**

### *3.2.1.1 Estructura*

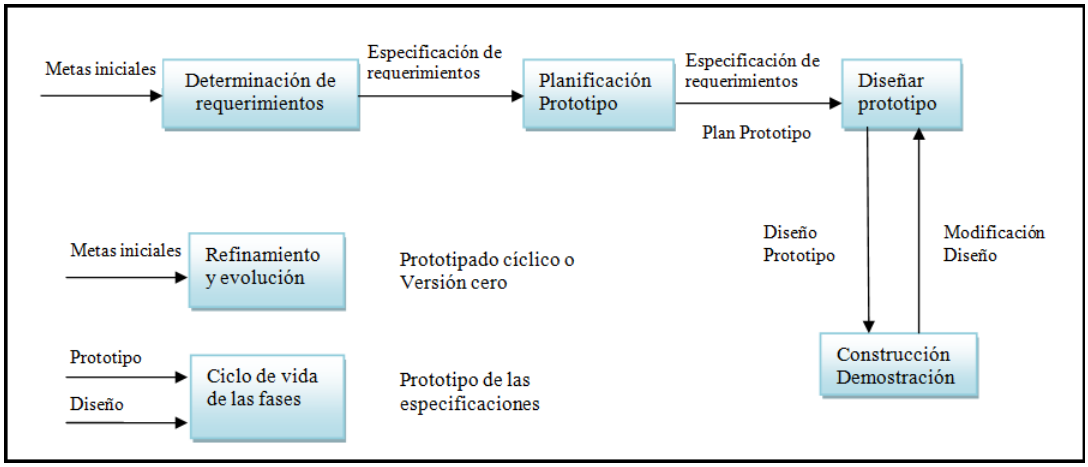


**Figura 18. Estructura**

Esta metodología es viable para el desarrollo del proyecto debido a que:

- En la creación de los prototipos iniciales se debe trabajar con unas ideas aproximadas de lo que desea la Oficina de Planeación, para presentar un producto o prototipo inicial, el cual puede evolucionar y refinarse dando como resultado un prototipo más maduro, que cumpla con todos los requerimientos de los usuarios.

- Con el uso del modelo de prototipos se da la facilidad de mejorar, de manera temprana los prototipos, teniendo en cuenta las sugerencias del usuario solicitante del proyecto, de tal forma que se cubran a cabalidad sus requerimientos.

- En este sistema de desarrollo se debe validar la versión actual del prototipo, para proceder a generar una nueva versión que contemple los nuevos requerimientos, con el fin de evitar retrocesos en el proceso de desarrollo.

## 3.3 Aplicación de la metodología

### 3.3.1 Diagramas UML

En la División de Servicios de Información (DSI) se han establecido los estándares concernientes al diseño de sistemas, el cual debe ser desarrollado por módulos y debe ceñirse al estándar definido por el Lenguaje Unificado de Modelado 2.1 (UML).

El diseño de un módulo, desarrollado en la DSI, debe contar con los siguientes diagramas:

- Diagrama de Casos de Uso
- Diagrama de Clases
- Diagrama de Secuencia

Teniendo en cuenta la dimensión de los diagramas UML elaborados para el proyecto, se ha tomado un ejemplo de cada uno de ellos para ser descritos de forma detallada.

#### *3.3.1.1 Diagrama de Casos de Uso*

Según los estándares definidos por la División de Servicios de Información UIS, por cada módulo del sistema que vaya a ser implementado debe realizarse un diagrama de casos de uso.

El caso de uso seleccionado como ejemplo, corresponde al módulo de entregables, en él se describe la forma en que los usuarios potenciales del sistema verán el módulo de entregables, en el cual se lleva a cabo la administración de los productos entregables de un proyecto.

En este módulo se llevan a cabo las siguientes acciones:

- Creación, edición y eliminación de productos entregables.
- Consulta de entregables.
- Asociación Entregables-Objetivos Específicos.
- Seguimiento y control en el cumplimiento de la planificación de los productos entregables.

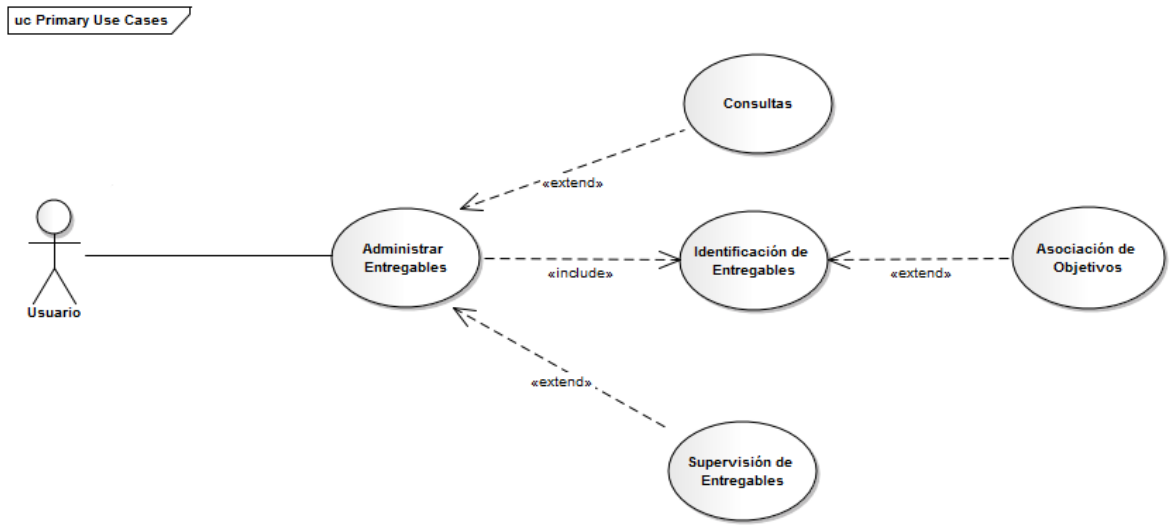


**Figura 19. Diagrama Casos de Uso Administrar Entregables**

Las siguientes tablas describen cada uno de los casos de uso ilustrados en la anterior figura.

**Tabla 1. Administrar Entregables**

| **Administrar Entregables** | |
|---|---|
| **Actor** | Usuario |
| **Propósito** | Los productos entregables son el resultado de los procesos realizados según se haya definido en el plan de gestión del proyecto. |
| **Resumen** | La administración de entregables contempla:<br>Identificación de Entregable.<br>Consultas.<br>Seguimiento y Control de Entregable. |
| **Flujo Principal** | **PLANIFICACION DE ENTREGABLES EXITOSA**<br>El usuario elige “Planificar Entregables” y decide por alguna de las siguientes opciones:<br>Consultas. |

| | |
|---|---|
| | Identificación de Entregables.<br>Supervisión de Entregables.<br>**SELECCIÓN**<br>El usuario:<br>Decide realizar una consulta el cual invoca el caso de uso Consultas.<br>Elige identificar un entregable el cual dispara el caso de uso Identificación de Entregables.<br>Desea realizar la auditoría de un entregable y esto conduce al siguiente caso de uso: Supervisión de Entregables. |
| **Precondición** | El usuario debe:<br>Estar autenticado.<br>Seleccionar un proyecto.<br>Identificar una fase. |

**Tabla 2. Consultas**

| **Consultas** | |
|---|---|
| **Actor** | Usuario |
| **Propósito** | Realizar una consulta a la base de datos de los productos entregables definidos para el proyecto. |
| **Resumen** | El usuario visualiza todos los entregables del proyecto, los cuales se encuentran organizados por fase. |
| **Flujo Principal** | **CONSULTA DE ENTREGABLES EXITOSA**<br>El usuario realiza la siguiente ruta básica al momento de consultar entregables:<br>**Ruta Básica:**<br>Elige “Consultar Entregables” lo que implica: |

| | |
|---|---|
| | Desplegar en pantalla el listado de todos los entregables identificados en el proyecto.<br>Seleccionar un entregable para ver el detalle de este, el cual visualiza en un panel modal todas las asociaciones con objetivos.<br>Escoge “Consultar Objetivos” lo que implica:<br>Un despliegue de información que corresponde a un listado de todos los objetivos identificados en una fase del proyecto.<br>Ver el detalle de un objetivo seleccionando una imagen, la cual muestra en pantalla un panel modal que contiene el listado de entregables asociados al objetivo indicado.<br>El sistema imprime en pantalla en caso de que la consulta este vacía.<br>**Ruta Alterna:**<br>El usuario entra en la ruta alterna si:<br>Ha elegido el paso 1 de la ruta básica y no se muestra la información de la consulta en donde se muestra en pantalla un aviso de error en la consulta.<br>Escoge el paso 2 de la ruta básica y no se muestra la información de la consulta en donde se muestra en pantalla un aviso de error en la consulta.<br>El error en la ruta alterna está asociado a problemas de conectividad con la Base de Datos o inconvenientes con el servidor. |
| **Precondición** | En el proyecto seleccionado se deben tener identificados:<br>Los objetivos.<br>Los entregables. |

**Tabla 3. Identificación de Entregables**

| **Identificación de Entregables** | |
|---|---|
| **Actor** | Usuario |
| **Propósito** | Identificar los productos entregables de cada fase del proyecto y almacenarlos en la base de datos. |
| **Resumen** | El usuario introduce en la base de datos los entregables identificados para cada fase del proyecto. |
| **Flujo Principal** | **IDENTIFICACIÓN DE ENTREGABLES EXITOSA**<br>**Ruta Básica:**<br>Elegir identificar un entregable.<br>El sistema despliega en pantalla el formulario para crear el nuevo registro de entregable.<br>Realizar el envío de la información por medio de un botón.<br>El sistema valida la información suministrada en el formulario.<br>El sistema imprime en pantalla un aviso de éxito en la transacción, de lo contrario se dirige a la Ruta Alterna 1.<br><br>**ERROR EN LA TRANSACCIÓN**<br>En este caso de uso pueden ocurrir las siguientes excepciones:<br>El usuario no introdujo información válida o no completó la información en el formulario; con lo cual el sistema le indica al usuario que campo(s) contiene información incorrecta o que campo(s) está vacío.<br>Hay un problema de conectividad con la Base de datos o algún inconveniente con el servidor, en donde el sistema visualiza el mensaje de excepción.<br><br>Adicionalmente el usuario puede optar por la siguiente alternativa: |

| | |
|---|---|
| | **Ruta Alterna:**<br>Elegir alguna de las siguientes acciones:<br>Ver detalle del Entregable: el usuario es redirigido a la interfaz que presenta la información detallada del entregable seleccionado.<br>Modificar Entregable: puede hacer modificación del entregable requerido, evento que implica desplegar la interfaz de modificación del entregable.<br>Eliminar Entregable: el usuario decide eliminar un entregable.<br>Si en usuario elige la acción *a* o *c* un entregable mediante un botón.<br><br>El sistema válida la información.<br>Se muestra en pantalla un mensaje de éxito en la transacción.<br><br>**ERROR EN LA RUTA ALTERNA**<br><br>En la ruta alterna es posible encontrar las siguientes excepciones:<br>Información incompleta o errónea en donde el sistema muestra el respectivo mensaje en pantalla.<br>El sistema restringe al usuario llevar a cabo la acción *b* o *c* para proteger la integridad de los datos en la base de datos (datos duplicados). |
| **Precondición** | Es necesario tener identificada la fase. |

**Tabla 4. Asociación de Objetivos**

| **Asociación de Objetivos** | |
|---|---|
| **Actor** | Usuario |
| **Propósito** | Realizar una consulta a la base de datos de los objetivos definidos para el proyecto. |
| **Resumen** | El usuario visualiza todos los objetivos del proyecto. |
| **Flujo Principal** | **ASOCIAR UN OBJETIVO EXITOSAMENTE**<br>El usuario lleva a cabo el siguiente camino:<br>**Ruta Básica:**<br>Elige la Ruta Alterna del caso de uso: *"Identificación de Entregable"*.<br>Se carga la interfaz de modificación de un entregable.<br>Despliega el Panel de asociación de entregables.<br>Diligencia el formulario de asociación de objetivos.<br>Confirma el envío de los datos por medio de un botón.<br>El sistema valida la información, marcando los campos vacíos sí este es su estado.<br>El sistema realiza la transacción y muestra un mensaje de éxito.<br>El usuario decide el siguiente camino:<br>**Ruta Alterna:**<br>El paso 1 y 2 de la ruta básica.<br>La modificación o eliminación de un objetivo asociado.<br>El sistema despliega un panel modal.<br>Se confirma el envío de la información por medio de un botón.<br>Si se desea modificar el sistema valida la información editada.<br>El sistema realiza la transacción mostrando en pantalla tanto un mensaje de éxito como también reflejando los cambios en la interfaz de modificación de entregable. |

| | |
|---|---|
| | **EXCEPCIONES EN LA ASOCIACIÓN**<br>La información del formulario es incompleta. El sistema dispara avisos indicando el problema.<br>Existen problemas de conexión ya sea a nivel de base de datos o con el servidor. El sistema recarga la pantalla para mostrar el mensaje de excepción en la transacción. |
| **Precondición** | Requiere que hayan sido creados los objetivos. |

**Tabla 5. Seguimiento de Entregables**

| **Supervisión de Entregables** | |
|---|---|
| **Actor** | Usuario |
| **Propósito** | El seguimiento y control de los entregables es el proceso de auditoría que permite verificar y analizar el estado de cualquier entregable con el objetivo de asegurar no sólo la calidad del producto terminado sino también evitar al máximo posibles retrasos en la entrega de productos (documentos, equipos, programas académicos, etc.), alertando al usuario de esta eventualidad en forma oportuna. |
| **Resumen** | Permite al usuario realizar seguimientos al entregable seleccionado. |
| **Flujo Principal** | **SEGUIMIENTO EXITOSO**<br>**Ruta Básica:**<br>El usuario elige la opción *"Supervisión de Entregables"* del caso de uso Administrar Entregables.<br>Ingresa a la interfaz de seguimiento de entregable.<br>Completa la información requerida en el formulario.<br>Confirma al sistema por medio de un botón. |

| | |
|---|---|
| | El sistema valida los datos introducidos.<br>La página es refrescada con la transacción exitosa indicándose con un aviso.<br>La transacción no fue exitosa y continua la ruta alterna.<br><br>**Ruta Alterna:**<br>El usuario no realiza de manera incorrecta el paso 3 de la ruta básica.<br>El sistema avisa al usuario al momento del envío del formulario.<br>El usuario desarrolló en forma correcta el paso 3 de la ruta básica pero existen problemas de conexión con la base de datos o con el servidor. En este caso el sistema imprime el mensaje de error una vez se refresca la interfaz. |
| **Precondición** | Debe existir un entregable. |

#### *3.3.1.2 Diagramas de Clases*

El diagrama de clases seleccionado como ejemplo, corresponde al módulo entregables del Sistema de Gestión de Proyectos UIS, dicho diagrama se aprecia en la figura 20.

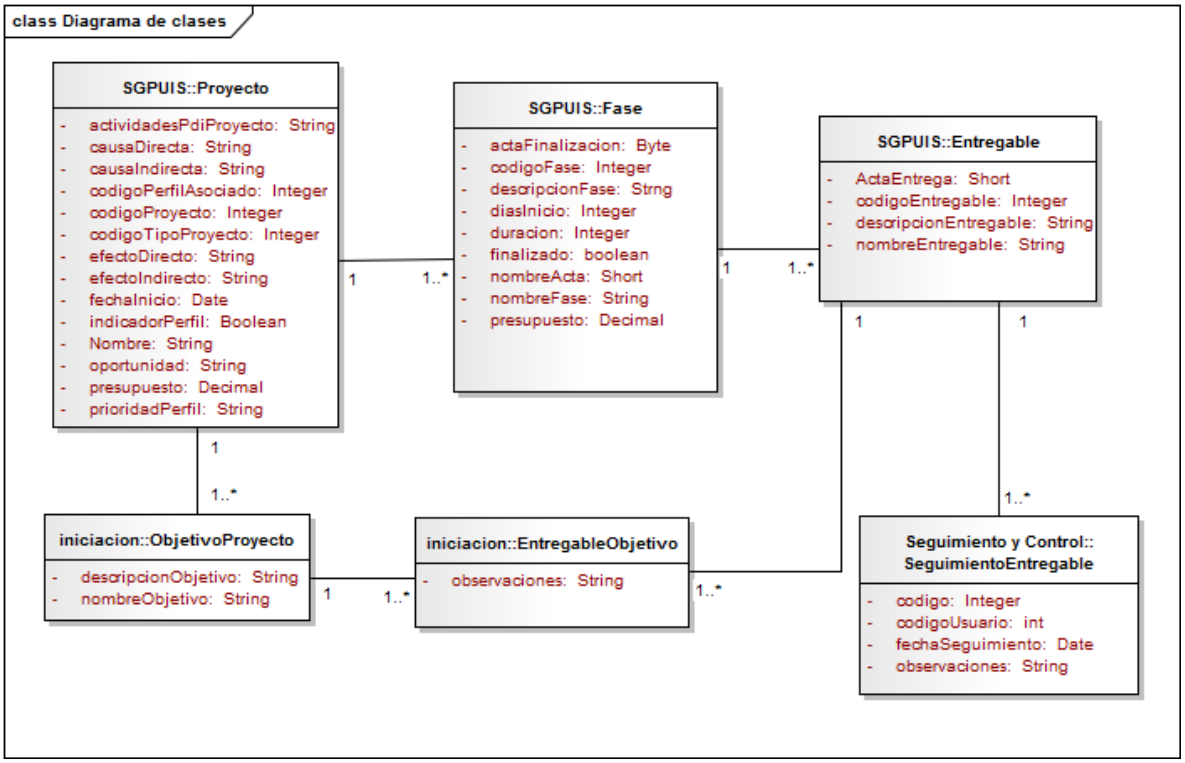


**Figura 20. Diagrama de Clases Entregables**

***Clase Entregable:*** Esta clase contiene atributos que proporcionan una descripción completa de los productos entregables de un proyecto.

**Tabla 6. Atributos de la Clase Entregable**

| Nombre | Tipo de datos | Descripción |
|---|---|---|
| codigoEntregable | Integer | Identificador a nivel de base de datos del entregable. |
| nombreEntregable | String | Identificador a nivel de usuario del entregable. |
| descripcionEntregable | String | Descripción clara del producto entregable. |
| actaEntrega | Short | Código del acta de entrega asociada. |

***Clase ObjetivoProyecto:*** Esta clase establece la relación entre un Proyecto y sus objetivos.

**Tabla 7. Atributos de la Clase ObjetivoProyecto**

| Nombre | Tipo de dato | Descripción |
|---|---|---|
| nombreObjetivo | String | Nombre asignado al Objetivo. |
| descripcionObjetivo | String | Descripción clara del Objetivo. |

***Clase SeguimientoEntregable:*** Esta entidad contiene información acerca del seguimiento que debe realizarse a los entregables a lo largo del proyecto.

**Tabla 8. Atributos de la clase SeguimientoEntregable**

| Nombre | Tipo de datos | Descripción |
|---|---|---|
| codigo | Integer | Identificador a nivel de base de datos de la entidad SeguimientoEntregable. |
| codigoUsuario | Integer | Identificador a nivel de base de datos del usuario que realiza las observaciones. |
| fechaObservación | Date | Fecha en que se realizó la Observación. |
| observaciones | String | Anotaciones que el auditor considera importantes sobre la ejecución de un entregable. |

### *3.3.1.3 Diagrama de Secuencias*

De acuerdo a los estándares de la DSI se elaboran los diagramas de secuencia que sean considerados de mayor importancia.

El diagrama de secuencia que se presenta a continuación da una visión más amplia acerca del funcionamiento y la interacción del sistema con el usuario, en el caso de uso que corresponde a la creación de productos entregables.

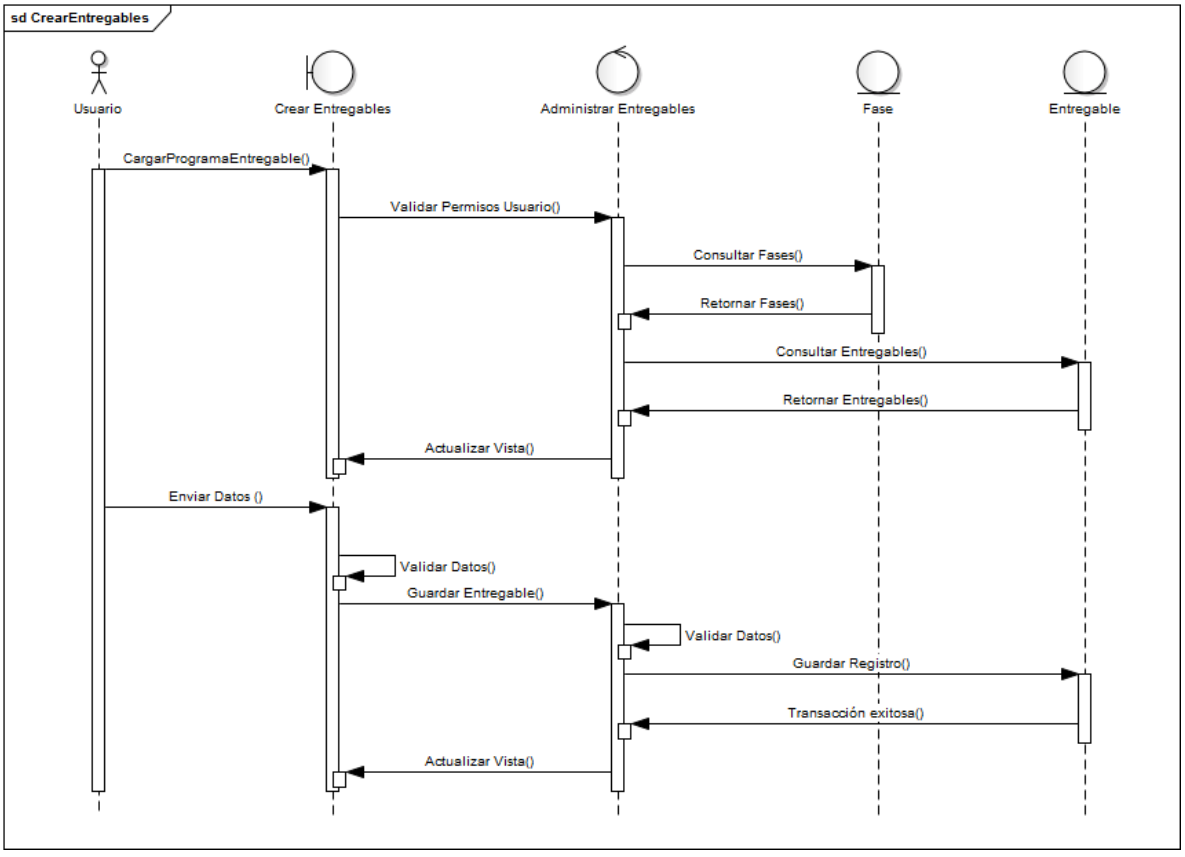


**Figura 21. Diagrama de Secuencias Entregables**

## 3.4 Prototipos

### 3.4.1 Primer prototipo

El primer prototipo elaborado consistió en el desarrollo de la interfaz de usuario de los módulos de fases, actividades, riesgos y entregables.

Dado que las actividades, riesgos y entregables están relacionadas directamente con las fases del proyecto, la primera interfaz en ser desarrollada fue la de planificación de las fases del proyecto.

Adicionalmente se implementaron las siguientes funcionalidades básicas:

- Crear registros de fases, actividades, entregables y riesgos.
- Ver información detallada de un registro seleccionado.
- Modificar los registros creados.
- Eliminar un registro seleccionado.

A continuación se muestra la interfaz que permite la creación de los entregables de un proyecto asociados a una fase:

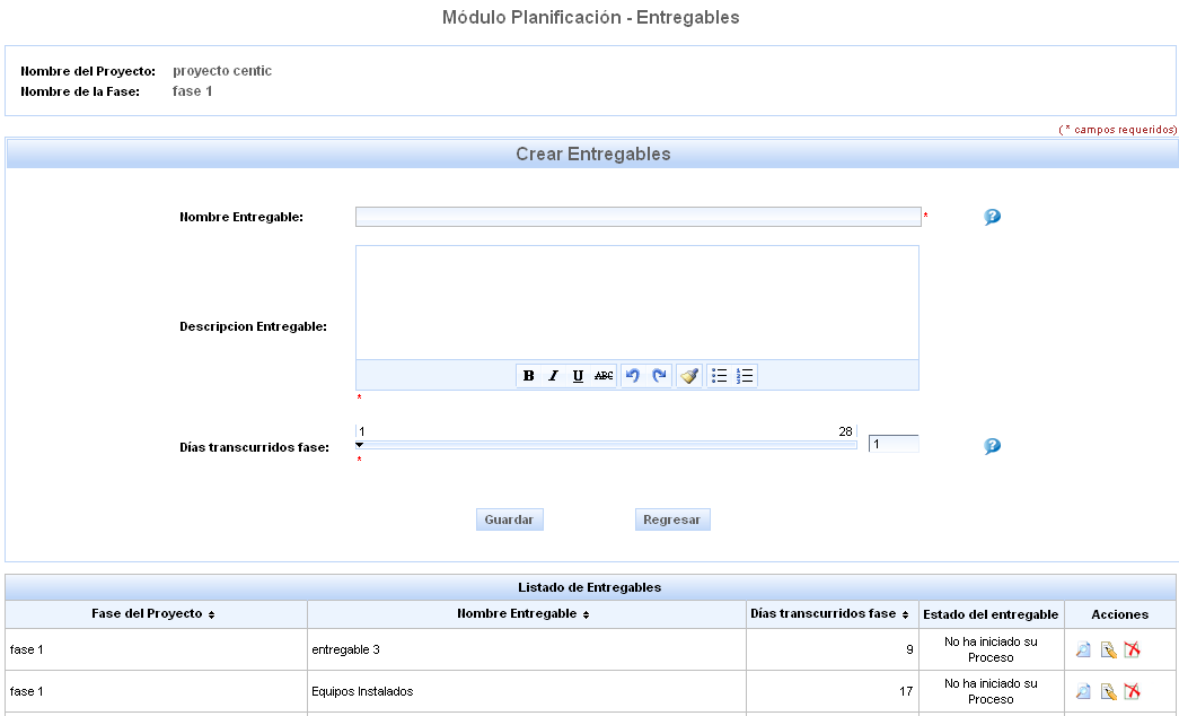


**Figura 22. Formato Crear Entregables**

La siguiente imagen pertenece al formato de las GUI (Graphical User Interface) de detalle, modificación o eliminación de los entregables:

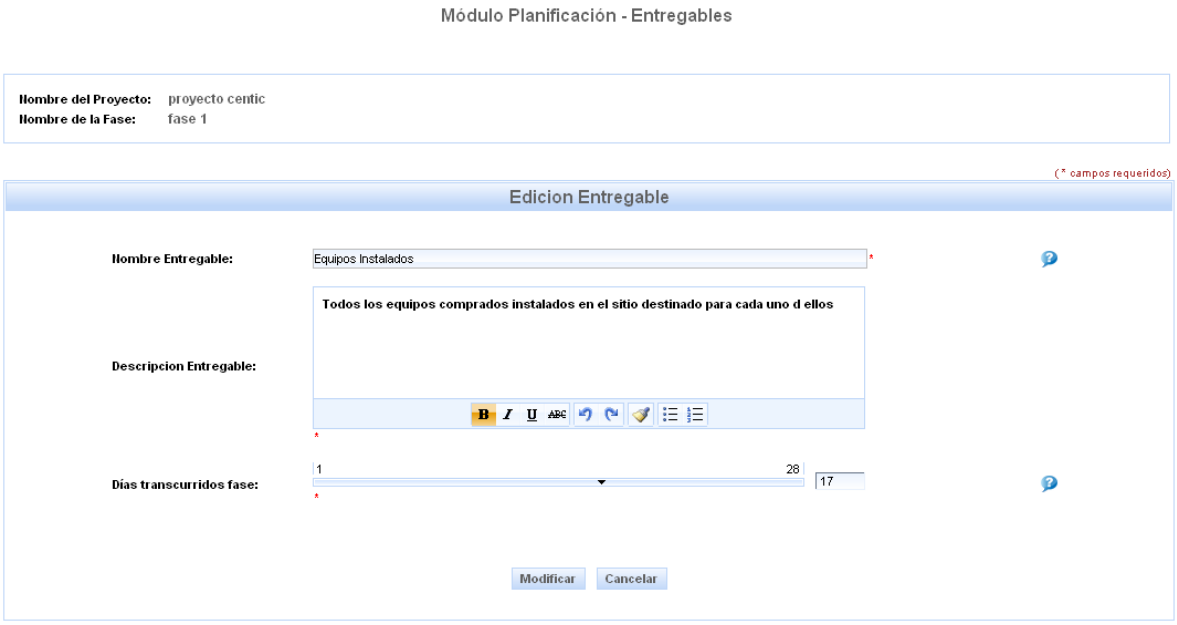


**Figura 23. Formato Detalle, Edición y Eliminación Entregable**

### 3.4.2 Segundo prototipo

❖ Módulo Entregables

Se elaboró la interfaz con sus respectivas acciones las cuales permiten la asociación de los entregables con los objetivos específicos. Las acciones realizadas fueron las siguientes:

- Crear la asociación de un entregable seleccionado con uno o varios objetivos específicos.
- Ver en detalle la relación entregable-objetivo seleccionada.
- Modificar los detalles de una asociación que se haya realizado.
- Eliminar las asociaciones realizadas.

La siguiente figura ilustra la interfaz para la asociación de un entregable a los objetivos específicos, en donde se distingue el formulario de asociación entregable-objetivo y además, el listado de las asociaciones realizadas y las diferentes acciones características del segundo prototipo (detalle, modificación o eliminación de las asociaciones realizadas).

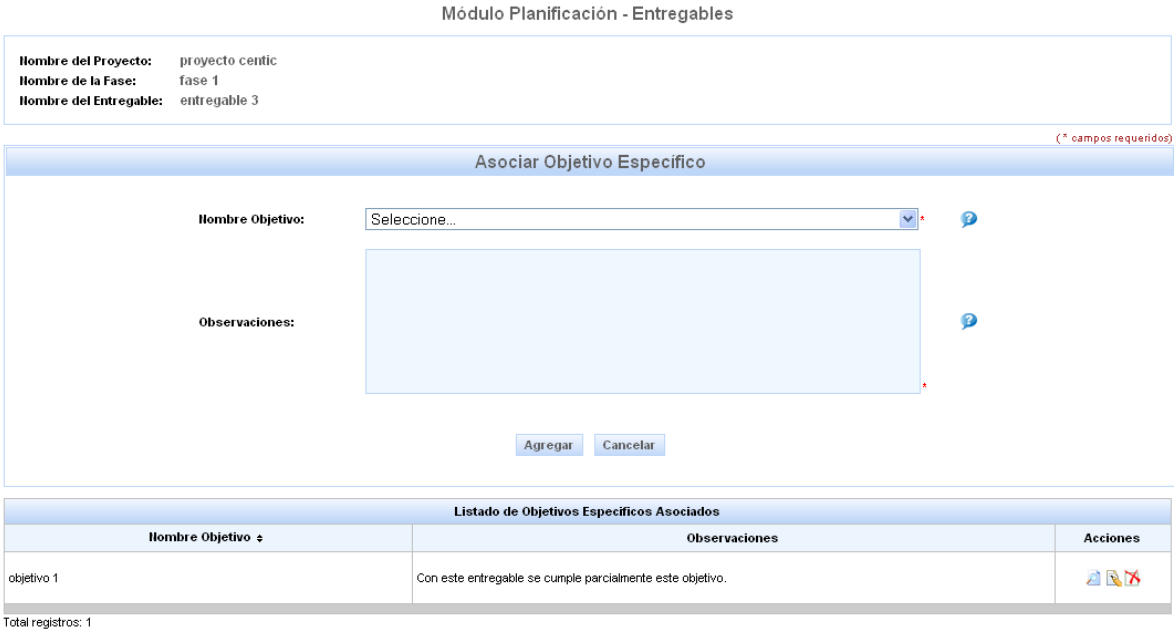


**Figura 24. Formato Asociación Objetivo Específico-Entregable**

❖ Módulo Riesgos

Para el administrador (Director del banco de proyectos) se desarrolló una interfaz con las acciones que permiten:

- Crear un nuevo riesgo predefinido o tomar riesgos creados en cualquier proyecto e incluirlos en el listado de riesgos predefinidos.
- Ver detalle riesgo predefinido.
- Modificar riesgo predefinido.
- Eliminar riesgo predefinido.

Se extiende la interfaz de riesgos del primer prototipo, desarrollando las acciones necesarias para visualizar la ayuda dirigida a los usuarios en el proceso de identificación de riesgos.

- Listar los riesgos predefinidos, en donde el usuario puede copiarlos y/o adaptarlos al proyecto en el cual esté trabajando.

En ambos módulos del prototipo actual se tienen en cuenta sus respectivas validaciones (datos incompletos o duplicados, restricciones en la modificación o eliminación de registros, etc.).

En la figura que se muestra a continuación se puede observar la validación en la cual se verifica que todos los datos deben ser ingresados en el formulario antes de guardar el registro.

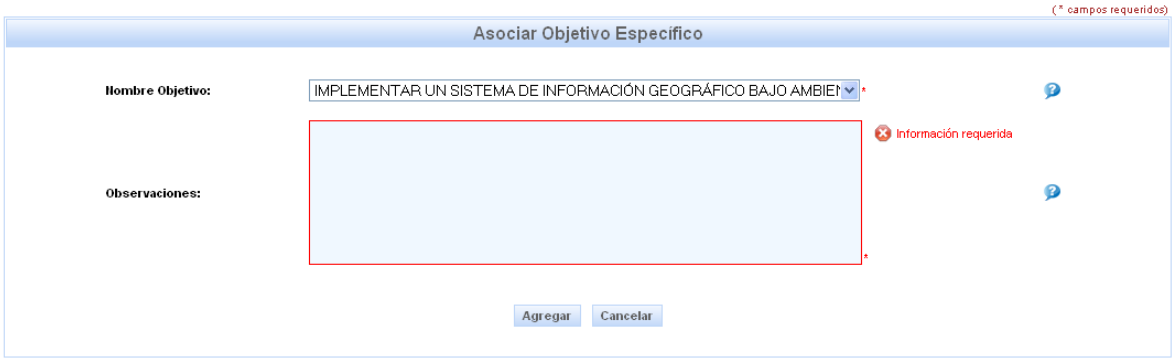


**Figura 25. Formato Entregable-Objetivo con validación**

La siguiente imagen exhibe la GUI de la Administración en cuanto al registro de riesgos estimados por ella misma o en otros proyectos registrados en el sistema.

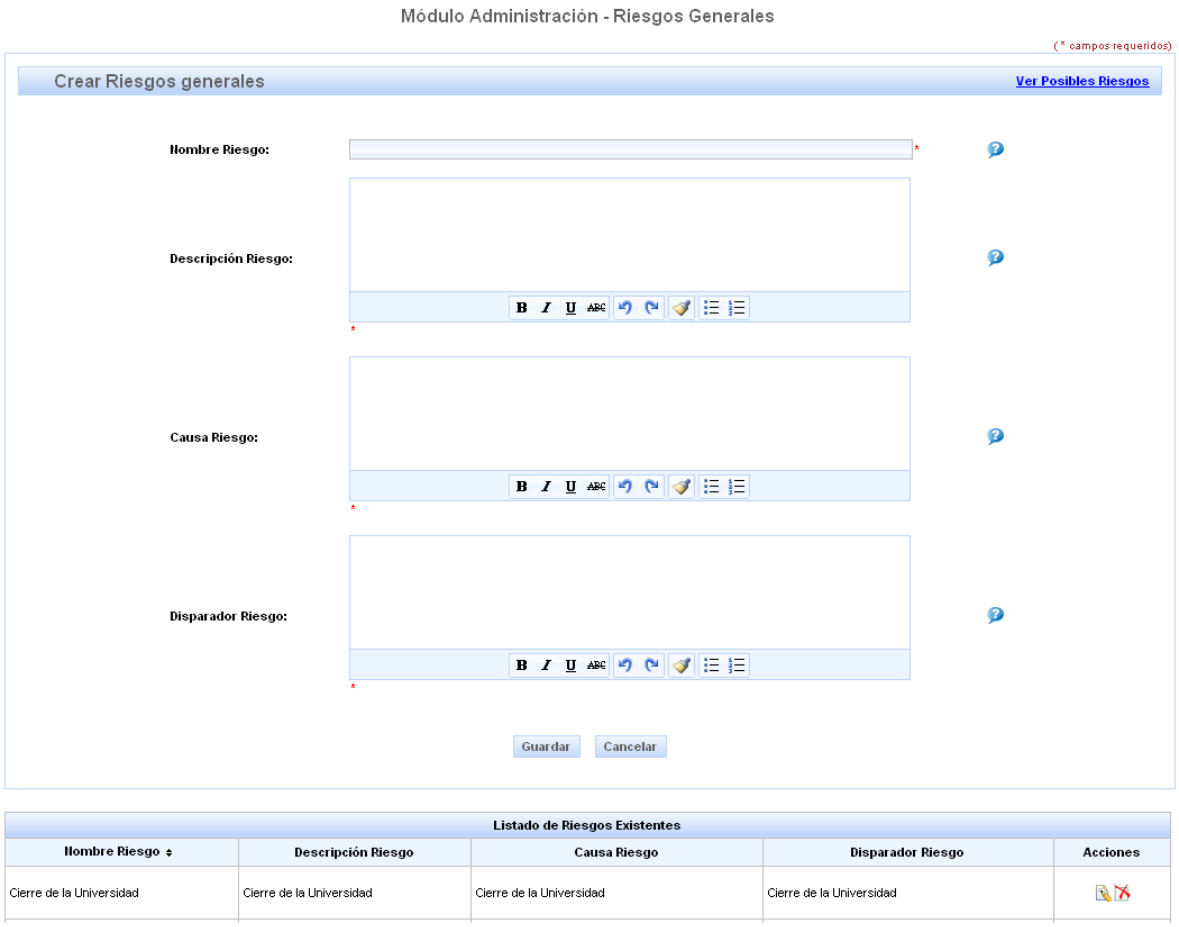


**Figura 26. Formato Creación de Riesgos Predefinidos por el Administrador**

### 3.4.3 Tercer Prototipo

❖ Módulo Entregables

Ampliando el segundo prototipo, se desarrollaron las acciones que permiten llevar a cabo el seguimiento de un entregable.

- Crear seguimiento al entregable seleccionado.
- Ver el detalle de los seguimientos que se han realizado.
- Modificar los registros del último seguimiento realizado a un entregable.

Además se incluyeron las funcionalidades de consulta tanto de entregables como de objetivos.

- Listar todos los entregables creados en el proyecto.
- Consultar los objetivos asociados a un entregable seleccionado.
- Listar todos los objetivos existentes en el proyecto.
- Consultar los entregables asociados a un específico.

❖ Módulo Riesgos

En este módulo se adicionaron al prototipo dos las siguientes funcionalidades, relativas al registro de los riesgos que se hayan presentado:

- Acción que hace posible registrar los riesgos presentados, identificando los entregables afectados por el mismo.
- Modificar las principales características de los riesgos presentados.
- Eliminar un registro de riesgo presentado.

❖ Módulo Actividades

Se adicionaron al módulo actividades las siguientes funcionalidades:

- Registrar los seguimientos realizados a las actividades.
- Ver detalle de los seguimientos realizados a las actividades
- Editar los seguimientos registrados.

En la siguiente imagen se aprecia la interfaz correspondiente al registro de los seguimientos realizados a los entregables del proyecto, en la parte superior se evidencia una de las validaciones realizadas en este prototipo.

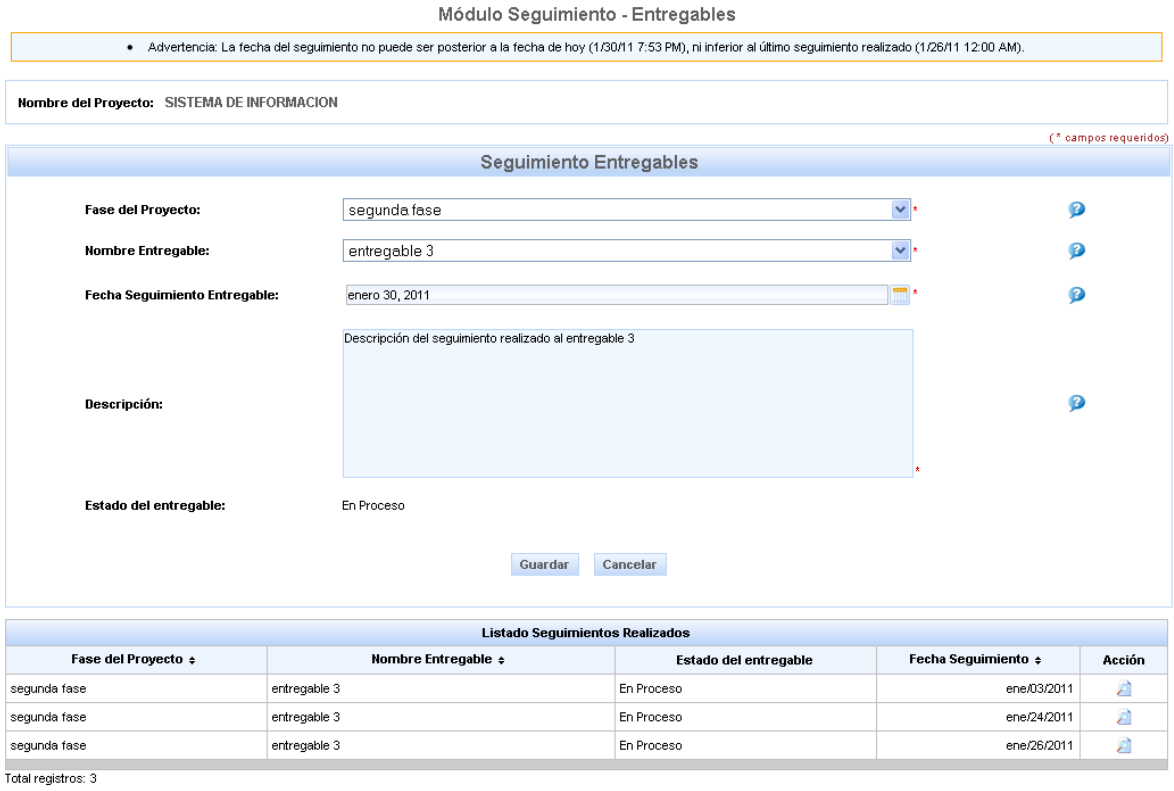


**Figura 27. Formato Seguimiento Entregables**

### 3.4.4 Cuarto Prototipo

- Módulo entregables

En este prototipo se realizaron las validaciones necesarias para este módulo (verificación del estado del entregable, restricciones en la modificación del seguimiento de los entregables).

Adicionalmente se desarrolló la interfaz que facilita al usuario la creación de actas de entrega, que una vez hayan sido revisadas y firmadas deben ser ingresadas al sistema para así formalizar la finalización de un entregable.
Las actas subidas al sistema pueden ser eliminadas, indicando al usuario que al realizar esta acción el entregable correspondiente vuelve al estado en proceso.

- Módulo Riesgos

Se desarrolló la interfaz que permite consultar los riesgos presentados, permitiendo filtrar los resultados por fases y/o entregables afectados; si el usuario lo desea puede también consultar todos los riesgos que se hayan presentado a lo largo del proyecto.

Estas funcionalidades tienen en cuenta la validación de la información.

La imagen que se muestra a continuación ilustra la GUI que permite la generación de actas de entrega.

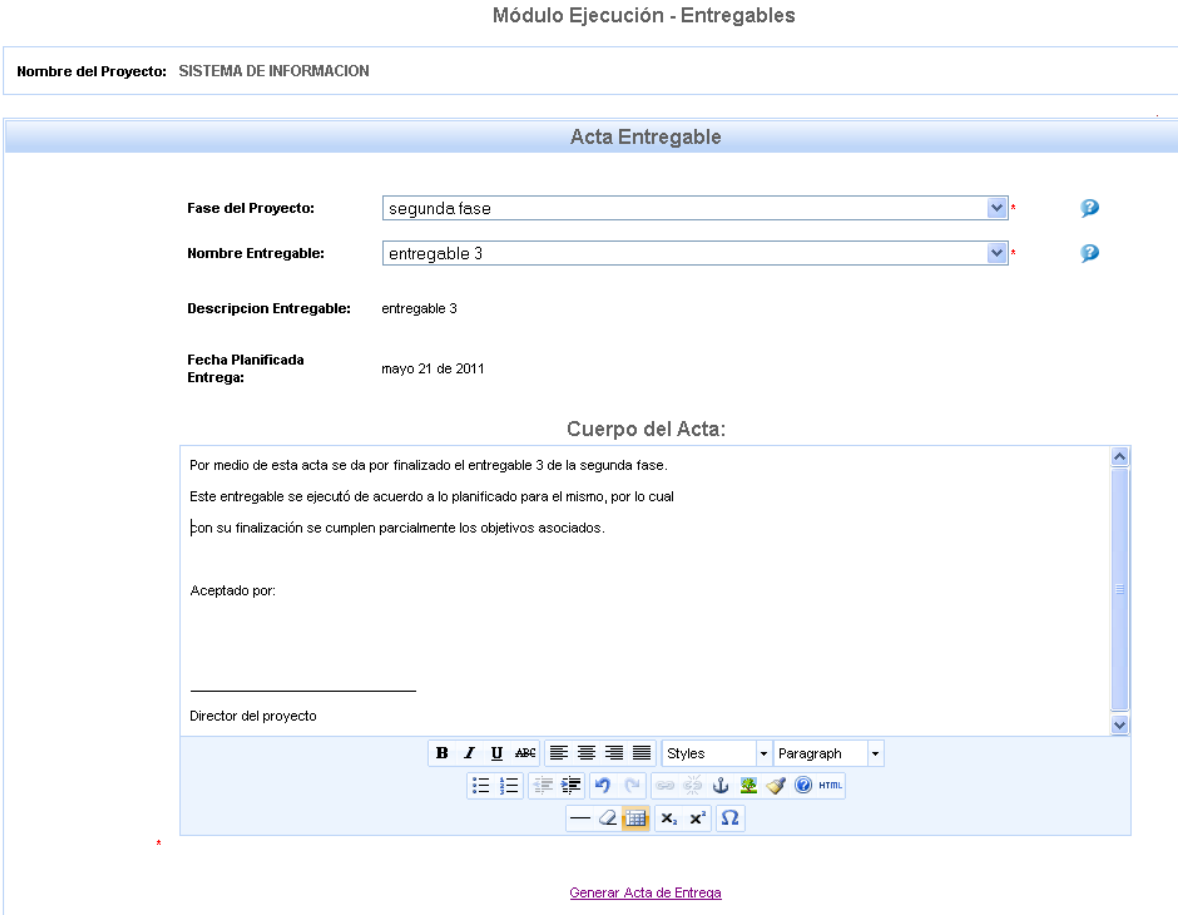


**Figura 28. Formato Generación Actas Entrega**

### 3.4.5 Quinto Prototipo

❖ Módulo Alarmas Actividades / Entregables

Este módulo comprende dos escenarios:

- El primero de ellos consiste en la generación de alarmas automáticas tanto para actividades como para entregables, cada 24 horas el sistema realiza una revisión en todos los proyectos que se encuentren en ejecución, detectando los entregables y/o actividades que estén próximos a vencer; una vez identificados

el sistema envía un correo electrónico al administrador del banco de proyectos y al responsable del proyecto informándoles en que proyecto existen entregables y/o actividades próximos a vencer.

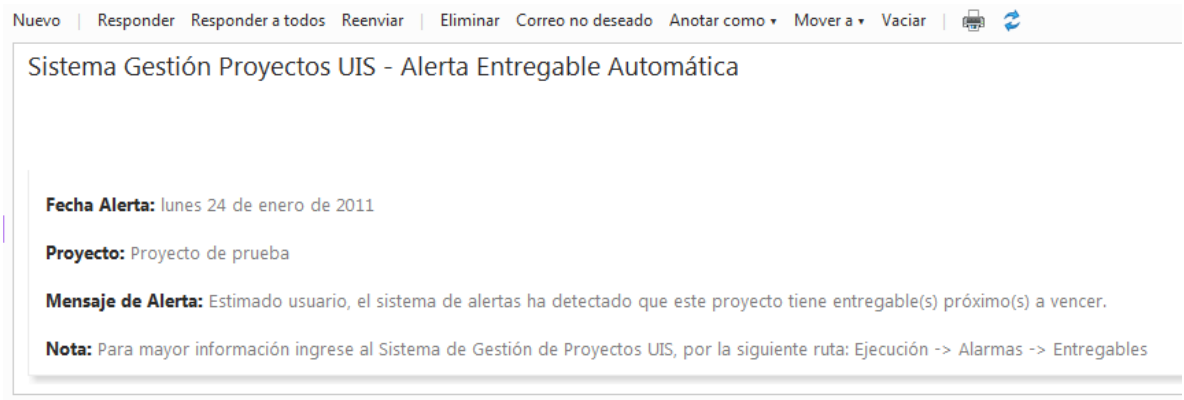


**Figura 29. Alerta Automática Entregable**

- El segundo escenario corresponde a la interfaz en que el usuario puede consultar el estado de las actividades y los entregables, con sus respectivas fechas de vencimiento.

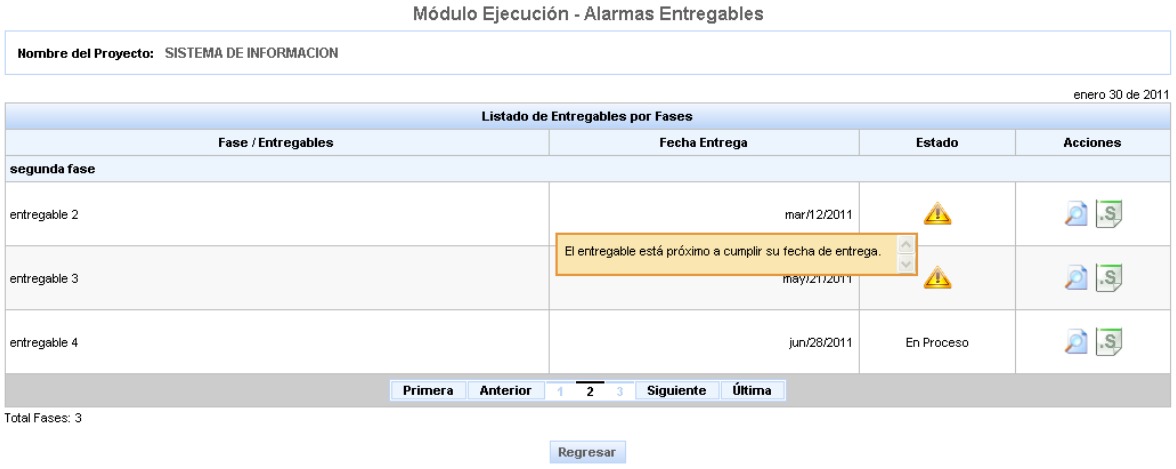


**Figura 30. Formato Alarmas Entregables**

❖ Módulo Auditorias

Se construyó un módulo para la auditoría de los proyectos el cual consiste en consultas que permiten ver los cambios realizados tanto a las fases como a las actividades y entregables desde el momento en el que un proyecto de cualquier índole inicia su ejecución.

### 3.4.6 Prototipo Final

Este prototipo cumple a cabalidad con todos los objetivos estipulados en el plan del proyecto, incluyendo las políticas de seguridad instauradas por la DSI de la Universidad Industrial de Santander.

El esquema de seguridad está basado en la estructura de roles usuarios, este esquema se define ampliamente en el numeral 3.4.7.

Actualmente los sistemas de información desarrollados en la DSI no sólo deben cumplir con los estándares de diseño y programación, sino también deben implementar la seguridad que administra el control de acceso a cada sistema de información que apoya la gestión de la Universidad Industrial de Santander.

A continuación se realiza una breve descripción acerca del proceso de implementación del sistema de seguridad. En el siguiente numeral se tratará el tema con más detalle.

Para que todos los programas desarrollados puedan implementar el sistema de seguridad, básicamente es necesario identificar los roles asociados a los usuarios potenciales del Sistema de Información de Gestión de Proyectos.

Una vez establecidos los roles se procede a establecer los permisos y menús de cada rol, tareas que posteriormente debe realizar el administrador del sistema de seguridad.

Cuando la información descrita en el párrafo anterior se encuentra registrada en la base de datos del sistema de seguridad se procede a configurar la aplicación de gestión de proyectos con el fin de adaptarla al esquema de seguridad definido específicamente para dicha aplicación.

Para el sistema de gestión de proyectos de la UIS existen dos roles tipos de usuarios principales:

- ***Administrador del Banco de Proyectos:*** persona encargada de supervisar y acompañar a los proyectos que son propuestos en la Universidad durante todo el ciclo de vida de los mismos.

- ***Director del Proyecto (Project Manager):*** persona encargada de proponer y ejecutar uno o más proyectos.

❖ Módulo Entregables

El prototipo definitivo abarca las siguientes funcionalidades:

- Administración completa de los productos entregables resultantes de cada proyecto, esto implica la formulación de los mismos, el seguimiento y control y su finalización, que comprende la generación de actas de entrega.

- Gestión de alertas de entregables por medio de consultas o de envío de correos masivos en forma automática.

- Validación de toda la información en el proceso de formulación y en el manejo de pistas de auditoría que permiten visualizar los cambios en los entregables registrados, en proyectos que se estén ejecutando.

❖ Módulo Cronograma

Está asociado con las fases y actividades definidas para los proyectos y permite formalizar los siguientes requerimientos:

- Gestión integral del cronograma de los proyectos que se extiende desde la formulación de las fases y actividades hasta el seguimiento, control y culminación del cronograma programado para dichos proyectos.

- Implementación del sistema de alertas consistente en consultas y envío de correos electrónicos automáticos para contribuir al control de los tiempos establecidos en el cronograma de actividades.

- Auditoría en los cambios que se hayan realizado al cronograma cuando los proyectos se encuentran en ejecución.

- Validación de la información asegurando la consistencia en las fechas establecidas en el cronograma.

❖ Módulo Riesgos

La estimación de los riesgos en la aplicación de Gestión de Proyectos consta de las siguientes características:

- Registro de todos los riesgos que pueden impactar durante el ciclo de vida del proyecto y junto con el respectivo plan de contingencia para cada uno de ellos.

- Seguimiento de los riesgos dejando constancia de su impacto en cualquiera de los entregables del proyecto.

- Validación de la información registrada en el proceso de formulación.

- Almacenamiento de pistas de auditoría en los cambios realizados en el seguimiento de los riesgos.

### 3.4.7 Esquema de seguridad Universidad Industrial de Santander

Para este proyecto se utiliza el esquema de seguridad definido por la División de Servicios de Información para los diferentes sistemas de información que apoyan la gestión de la Universidad Industrial de Santander, el cual está basado en la estructura de roles – usuarios.

Los roles se establecen en cada una de las unidades académico administrativas, UAA, responsables de cada sistema, de acuerdo a las actividades que realizan. A cada uno de los roles definidos se le asocian los usuarios de acuerdo a las funciones que desempeñen.

### *3.4.7.1 Estructura de la Base de Datos soporte*

La base de datos que soporta el esquema de seguridad contempla básicamente las siguientes tablas:

**Sistema:** Contiene información de los sistemas de información de la universidad. Para cada sistema se específica: Nombre, descripción del sistema, fecha y hora de creación en la base de datos, fecha y hora de inicio de vigencia del sistema, fecha y hora de cierre de vigencia del sistema.

**Rol:** contiene información de los diferentes roles definidos para cada sistema de información, como: Nombre asignado al rol, descripción del rol, fecha y hora de creación, fecha y hora de inicio de vigencia del rol, fecha y hora de cierre de vigencia del rol.

**Usuario:** Contiene información de los posibles usuarios de los sistemas de información. Entre esta información está: tipo y número de documento de identidad del usuario, fecha y hora de creación del usuario, fecha y hora de inicio de vigencia del usuario, fecha y hora de cierre de vigencia del usuario.

**Sistema–rol:** Contiene los roles definidos para cada uno de los sistemas de información, indicando: rol, sistema, fecha y hora de creación del rol – sistema, fecha y hora de inicio de vigencia del rol en el sistema, fecha y hora de cierre de vigencia del rol en el sistema.

**Rol–usuario:** Contempla los usuarios asociados a cada uno de los roles definidos, considerando: Rol, usuario, fecha y hora de creación del rol – usuario, fecha y hora de inicio de vigencia del usuario en el rol, fecha y hora de cierre de vigencia del usuario en el rol.

**Menú–rol–sistema:** Contiene los menús asociados a los roles en los distintos sistema de información, contemplando: Sistema de información, nombre del menú, descripción del menú, fecha y hora de creación del menú, fecha y hora de inicio de vigencia del menú asociado al rol, fecha y hora de cierre de vigencia del menú asociado al rol.

**Opción–menú–rol:** Contempla las opciones definidas para cada una de los posibles menús establecidos para cada sistema de información. Contiene: Nombre de la opción, descripción de la opción, nombre del menú superior, nombre del menú que contiene la opción, nombre del programa a ejecutar cuando la opción es la de más bajo nivel, fecha y hora de creación de la opción del menú, fecha y hora de inicio de vigencia de la opción, fecha y hora de cierre de la opción.

**Tabla–sistema:** Contiene información de las tablas que conforman la base de datos que soporta cada uno de los sistemas de información. Considera: Sistema de información, nombre de la tabla, descripción de la tabla.

**Tipo–permiso:** Establece para cada tabla de un sistema de información, los roles que tienen permisos para incluir registros, para modificar registros o para eliminar registros en ella. Contiene: Sistema de información, nombre de la tabla, clase de permiso (inclusión, modificación, eliminación de registros), fecha y hora de creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

**Acceso–tabla:** Define para las tablas de un sistema de información si un rol tiene permiso sobre toda la información de la tabla o sobre una parte de esta. Considera: Sistema, nombre de la tabla, clase de acceso (total, parcial), fecha y hora de creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

**Atributo–tabla:** Establece los atributos sobre los cuales se debe controlar el acceso a una tabla, cuando a un rol se le concede permiso para hacer uso parcial de la información existente en una tabla. Contiene: Sistema de información, nombre de la tabla, nombre del atributo sobre el cual se controla el acceso a la tabla, descripción del atributo, fecha y hora de creación del atributo, fecha y hora de inicio de vigencia del atributo, fecha y hora de fin de vigencia del atributo.

**Valor–atributo-proceso:** Contiene los valores que deben tener los atributos definidos en cada tabla en la tabla atributo – tabla que permiten el acceso a la información asociada a estos valores. Específica: Sistema de información, nombre de la tabla, nombre del atributo, valor del atributo, descripción, fecha y hora de creación del valor del atributo, fecha y hora de inicio de vigencia del valor del atributo, fecha y hora de fin de vigencia del valor del atributo.

**Acceso-sistema:** Contempla el histórico de acceso que un usuario ha realizado a un sistema, identificando las opciones que ha seleccionado. Contiene: Login de usuario, rol, identificación de la sesión, sistema, opción seleccionada, fecha y hora de ingreso, fecha y hora de salida.

#### *3.4.7.2 Entorno de Navegación*

Para cada sistema de información, la UAA responsable define los roles necesarios para el adecuado uso del sistema de información de acuerdo a las funciones que realice y establece los usuarios asociados a cada uno de ellos.

Para cada rol se define el menú de inicio, el cual le permite a cada usuario que hace parte de este rol, empezar la navegación por las distintas opciones que le ofrece el sistema, hasta llegar al nivel más bajo en el cual se ejecuta el proceso que soporta la actividad que desea realizar.

Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, menú-rol, opción-menú rol, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

#### *3.4.7.3 Entorno de Control de Datos*

Para los roles definidos en cada uno de los sistemas de información se especifican las tablas a las cuales puede acceder, el tipo de transacción que puede realizar sobre estas tablas (inclusión, modificación o eliminación de registros), si tiene acceso total o parcial a la información que contiene la tabla.
Para el acceso a la información de la tabla de manera parcial, se debe establecer el atributo o atributos seleccionados, los valores que estos atributos deben tener para autorizar el acceso solicitado.

Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, tabla–sistema, tipo-permiso, acceso-tabla, atributo-tabla, valor atributo proceso, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

#### *3.4.7.4 Auditoría*

Todas las tablas que conforman la base de datos soporte del esquema de seguridad tienen el historial de las transacciones realizadas sobre cada una de ellas.

El historial de las transacciones de cada tabla contiene información de los registros incluidos en la tabla, de los registros modificados y de los registros eliminados. Adicionalmente, en cada transacción se especifica: Fecha de la transacción, hora de la transacción, tipo de transacción (I/U/D), tipo y número de

documento de identidad del usuario que realizó la transacción, login, rol asociado, dirección IP y MAC del equipo desde el cual llevó a cabo la transacción.

# CAPITULO 4

# 4 CONCLUSIONES

- A pesar que el Sistema de Gestión de Proyectos UIS está enfocado en los proyectos de inversión, el Sistema de información cuenta con un diseño lo suficientemente flexible como para dar soporte a la gestión de proyectos de diferente índole.

- Por primera vez el Sistema de Gestión de Proyectos UIS cuenta con una herramienta que apoya íntegramente la administración de los productos entregables y los riesgos de los proyectos, desde la fase de planificación hasta la fase de ejecución, seguimiento y control y cierre.

- La aplicación Web permite elaborar un cronograma de actividades detallado y consistente como también facilita realizar el seguimiento y control sobre cada una de dichas actividades a lo largo del ciclo de vida del proyecto.

- El Sistema de Información incorpora un esquema de seguridad cuya estructura de datos simplifica, restringe y controla la navegación por la aplicación.

- El nuevo Sistema de información facilita los procesos de auditoría necesarios para garantizar la transparencia en la ejecución de los proyectos registrados en el Banco de Proyectos.

- Se identifica un alto grado de complejidad en el desarrollo de software de nivel empresarial debido a la arquitectura multi-capa de la tecnología Java EE5;

esto puede retrasar la codificación de los programas ya que existen muchos conceptos que la mayoría de programadores principiantes no conocen. El uso de frameworks de desarrollo de nivel empresarial, específicamente en este contexto: Seam y Rich Faces contrarrestan en gran medida las dificultades a las que usualmente se exponen los programadores.

- Cuando un software se desarrolla por módulos y en equipo, el trabajo coordinado y sincronizado permite una integración más estable pues minimiza el riesgo de cometer errores.

- Los conocimientos adquiridos en las capacitaciones de Java Básico, Java EE5 y en la ejecución de este proyecto; y con la experiencia en el desarrollo de software utilizando frameworks aportan positivamente a nuestra competitividad en la industria del software.

❖ La revisión exhaustiva del código fuente por parte de expertos en la DSI ayuda a garantizar la calidad del software y ayuda a fomentar buenas prácticas en la codificación

# CAPITULO 5

# 5 RECOMENDACIONES

- El requerimiento del uso de gráficas en la gestión de proyectos se evidencia en la definición de cronogramas, entre otros, ya que un diagrama podría dimensionar con mayor facilidad los tiempos que se establecen en la planificación del proyecto. Por este motivo es preciso recomendar para la segunda versión de este nuevo sistema de información el uso de herramientas que permitan crear aplicativos para la manipulación o visualización de gráficas; sean soluciones software comerciales que puedan suplir esta necesidad como Nectronic (desarrolladores de diagramas de Gantt para aplicaciones web), o también herramientas libres como es el caso de la librería para java con licencia GPL llamada jFreeChart, la cual permite crear aplicativos para la gestión de una gran variedad de gráficas.

- Es necesario que la aplicación pueda proporcionar soporte en el seguimiento y control del cronograma y los productos entregables en caso de que un proyecto deba ser suspendido, que por su complejidad y por el alcance de este proyecto de grado no fue contemplado.

- Se recomienda manejar un servidor de versiones cuando los sistemas a desarrollar involucren integraciones de diferentes segmentos de software.

- Es recomendable realizar la clasificación de los riesgos ya que de esta manera se realizaría una mejor administración del banco de conocimientos y un mejor uso de este servicio por parte de los usuarios.

- Se sugiere fortalecer el vínculo entre la División de Servicios de Información y la Escuela de Ingeniería de Sistemas e Informática para compartir experiencias y conocimientos en el desarrollo de software empresarial.

# CAPITULO 6

# 6 BIBLIOGRAFIA


BERZAL GALIANO, Fernando. Doctor en Informática. Universidad de Granada, España. Relaciones entre clases: Diagramas de clase UML. {En línea}. {Consultado 15Julio de 2010}. Disponible en: http://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf

CADENA RUIZ, Ana María. Antecedentes BPIN. {En línea}. {08 Febrero de 2010}. Disponibleen: (http://www.dnp.gov.co/PortalWeb/Portals/0/archivos/documentos/DIFP/Presupuesto/Antecedentes_Bpin.pdf)

HERNÁNDEZ RAMÍREZ, Edwin, SUAREZ BARÓN, Silvia. Sistema de información para el banco de programas y proyectos de inversión de la Universidad Industrial de Santander. Bucaramanga, 2001, P. 5-7,57-58. Trabajo de Grado (Ingeniería de Sistemas). Universidad Industrial de Santander. Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática.

ICONTEC INTERNATIONAL. EL COMPENDIO DE TESIS Y OTROS TRABAJOS DE GRADO. {En línea}. {Consultado junio 2009}. Disponible en: http://www.ICONTEC.org/BancoConocimiento/C/compendio_de_tesis_y_otros_trabajos_de_grado/compendio_de_tesis_y_otros_trabajos_de_grado.asp?CodIdioma=ESP

KING, Gavin; MUIR, Pete; RICHARDS, Norman; BRYZAK, Shane; YUAN, Michael; OUNGSTROM, Mike; BAUER, Christian; BALUNAS, Jay; ALLEN, Dan; ANDERSEN, Max Rydahl; BERNARD, Emmanuel; KARLSSON, Nicklas; ROTH, Daniel; DREES, Matt; ORSHALICK, Jacob; and NOVOTNY, Marek. Seam a frameworkforenterprise java 2.2.0.GA.{En línea}. {Consultado 12 Julio de 2010}. Disponible en: http://www.seamframework.org/Seam2/Documentation

Project Management Institute. Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK®). TerceraEdición. 2004. Four Campus Boulevard, Newtown Square, PA 19073-3299 EE.UU. P. 5-8, 337-341.

SALINAS, Patricio. Modelo de Clases.Departamento de Ciencias de la Computación Universidad, Universidad de Chile. {En línea}. {Consultado 01Septiembre de 2010}. Disponible en: http://www.dcc.uchile.cl/~psalinas/uml/modelo.html

SCHMULLER, Joseph. Aprendiendo UML en 24 horas. Primera edición. Editorial Prentice Hall, 2001. 5-18p.

SERRANO, Giobani. Plan de proyecto: propuesta de modelo para la Gestión de Proyectos de Inversión en una Institución Pública de Educación Superior en Colombia: una perspectiva desde el Pensamiento Sistémico. Universidad Industrial de Santander. Bucaramanga. 2009. P. 23.

SICUMA: Sistemas de Información Cooperativos Universidad de Málaga. España. Tutorial de JavaServer Faces. {En línea}. {Consultado 27 Agosto de 2010}.Disponible en: http://www.sicuma.uma.es/sicuma/formacion.jsp

Sparx Systems, Pty Ltd. Tutorial UML 2. {En línea}. {Consultado 16 Junio de 2010}. Disponible en: http://www.sparxsystems.com/resources/uml2_tutorial/index.html

SUNMICROSYSTEMS, Inc.The Java EE 5Tutorial. {En línea}. {Consultado 15Junio de 2010}. Disponible en: http://download.oracle.com/javaee/5/tutorial/doc/

DE AMESCUA SECO, Antonio; CUADRADO GALLEGO, Juan José; ERNICA LAFUENTE, Emilio; GACÍA GUZMÁN, Javier; GARCÍA SÁNCHEZ, Luis; MARTÍNEZ FERNÁNDEZ, Paloma; SÁNCHEZ SEGURA Mª Isabel. Análisis y Diseño Estructurado y Orientado a Objetos de Sistemas Informáticos. Quinta Edición. Universidad Carlos III de Madrid, España. McGraw-Hill, 2003. 3-5p, 25-28p, 64p.