DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DE LOS MÓDULOS DE EJECUCIÓN, SEGUIMIENTO Y CONTROL, CONSULTAS E INFORMES GENERALES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS

Marco Antonio Ruiz Rueda
Pedro Antonio Otero Prada

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

DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DE LOS MÓDULOS DE EJECUCIÓN, SEGUIMIENTO Y CONTROL, CONSULTAS E INFORMES GENERALES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS

Marco Antonio Ruiz Rueda

Pedro Antonio Otero Prada

Proyecto de grado presentado como requisito parcial para optar el título de Ingeniería de Sistemas

Director
Emilio Justiniano Cárcamo Troconis
Codirector
Enrique Torres López

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

AGRADECIMIENTOS MARCO RUIZ

A Dios por la salud, el bienestar y por dejarme crecer en una familia llena de valores y costumbres puramente sanas.

A mis amados padres que se encargaron de apoyarme incondicionalmente, a mis hermanos por acompañarme uno a uno en cada una de mis metas y oportunidades, y a las personas que con su cariño sirvieron de bastón en el largo camino al éxito.

Agradezco enormemente a la Universidad Industrial de Santander por darme la oportunidad de aprender un montón de detalles de la vida, y por enseñarme que una universidad no solo nos llena de conceptos sino de experiencias que conducen hacia la madurez.

Un agradecimiento especial a mis compañeros de proyecto y a los profesionales de la División de Servicios de Información de la universidad que con paciencia y cariño me educaron y guiaron por el camino que llevó a la culminación de una etapa de aprendizaje definitiva en mi carrera: los ingenieros Emilio Cárcamo, Enrique Torres, Humberto Ruiz y también otros que ejerciendo sus funciones ayudaron a superar contratiempos que surgieron en la etapa de construcción del proyecto.

AGRADECIMIENTOS PEDRO OTERO

Quiero hacer un agradecimiento especial a mis padres, hermano y hermanas por todo el apoyo recibido hasta hoy y el que sé que seguiré recibiendo constantemente de aquí en adelante.

A la Universidad Industrial de Santander por todos los buenos y malos momentos pasados acá, lo cual incluye por supuesto a la Escuela de Ingeniería de Sistemas e Informática, sus profesores y empleados, los compañeros de estudio y mis amigos.

Finalmente, a la División de Servicios de Información y todo su equipo de trabajo por la invaluable experiencia y conocimiento adquiridos, y a los ingenieros Enrique Torres López y Emilio Cárcamo por su constante acompañamiento en el desarrollo de este importante proyecto para la Universidad y el crecer profesional de quienes estuvimos involucrados en él.

CONTENIDO

LISTADO DE FIGURAS

RESUMEN

TITULO: DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DE LOS MÓDULOS DE EJECUCIÓN, SEGUIMIENTO Y CONTROL, CONSULTAS E INFORMES GENERALES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS[1].

AUTORES: RUIZ RUEDA, Marco Antonio[2], OTERO PRADA, Pedro Antonio[3]

PALABRAS CLAVE: Gestión de Proyectos, Planificación, JavaEE5, Control de cambios.

DESCRIPCIÓN: La Universidad Industrial de Santander (UIS) ha venido consolidando políticas para el manejo de sus recursos de inversión y proporcionando elementos a través del Banco de Programas y Proyectos de Inversión, fomentando de esta forma una cultura de trabajo más saludable, estructurada y orientada a proyectos. La gestión de proyectos, teniendo en cuenta la información que requiere y las unidades académico administrativas que existen en la universidad, implica que existan herramientas informáticas robustas, modernas y fáciles de usar para facilitar estos procesos. Formular un proyecto requiere la planificación cuidadosa de todas las fases, actividades y entregables que deben cumplirse, entre otros aspectos. Aun así es muy probable que la ejecución del proyecto no se dé como fue planeado en esas etapas iniciales porque la organización que emprende los proyectos está expuesta a factores que no puede predecir ni controlar. Es necesario que las técnicas de gestión de proyectos incluyan formas de manejar estos factores, para determinar la necesidad de efectuar cambios que no afecten negativamente al proyecto. O al menos, que las consecuencias de estos cambios sean debidamente analizadas y acordadas. Todo lo anterior hace necesario que un sistema de gestión de proyectos cuente con un Módulo de Control de Cambios que facilite esta tarea.

---

[1] Proyecto de grado en la modalidad de Investigación.

[2,3] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Ing. Emilio Justiniano Cárcamo Troconis. Codirector: Ing. Enrique Torres López.

ABSTRACT

TITLE: DESIGN, IMPLEMENTATION AND COMMISSIONING OF MODULES EXECUTION, MONITORING AND CONTROL, QUERIES AND GENERAL REPORTS FOR THE PROJECT MANAGEMENT SYSTEM OF UIS[4].

AUTHORS: RUIZ RUEDA, Marco Antonio [5], OTERO PRADA, Pedro Antonio [6]

KEY WORDS: Project Management, Planning, JavaEE5, Change control

DESCRIPTION: Universidad Industrial de Santander (UIS) has been consolidating policies for the management of its investment resources and facilitating elements, through the Bank of Programs and Investment Projects, promoting a more structured and healthy working culture which is project oriented. Since project managements requires a considerable amount of information and the university has a lot of academic administrative units, it need to count on robust, modern and easy to use information tools which makes this processes easier. Formulating a project implies planning carefully all the phases, activities and deliverable products that must be done, among others. Nevertheless, it is very likely that the execution of the project does not run exactly as it was planned on such initial stages. The reason for this is that the organization that takes on the projects is exposed to factors it cannot predict nor control. Thus, project management techniques need for ways to handle this factors, to help to decide if changes have to be made and that they do not impact the project negatively. Or at least, that the consequences of these changes are analyzed and agreed properly. All the exposed above suggests that a project management system must have an Change Control Module that makes this task easy.

---

[4] Degree Project in the modality Research

[5,6]Faculty of Physical-Mechanical Engineering. Systems and Computer Engineering.
Director: Ing. Emilio Justiniano Cárcamo Troconis. Codirector: Ing. Enrique Torres López.

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, a través del 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, las cuales deben ser apoyadas con un sistema de información debidamente desarrollado para tal fin.

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 dé un adecuado soporte.

Es por esto que desde la Oficina de Planeación con la colaboración de la División de Servicios de Información, ambas dependencias de la UIS, han tenido la oportunidad de liderar 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 y con calidad de nivel empresarial, para el Sistema de Gestión de Programas y Proyectos de la UIS (SGPUIS), que permita gestionar los cambios que sucedan durante la ejecución de un proyecto. También dotar a dicho sistema de un módulo de consultas que permita manejar el gran volumen de proyectos que pueden generarse con los años, permitiendo búsquedas por distintos criterios.

CAPÍTULO 1

1 CONTEXTO GENERAL

1.1 ESPECIFICACIONES DEL PROYECTO

1.1.1 Título

DISEÑO, IMPLEMENTACIÓN Y PUESTA EN FUNCIONAMIENTO DE LOS MÓDULOS DE EJECUCIÓN, SEGUIMIENTO Y CONTROL, CONSULTAS E INFORMES GENERALES DEL SISTEMA DE GESTIÓN DE PROYECTOS UIS.

Director del proyecto:

NOMBRE: Ing. Emilio Cárcamo Troconis
INSTITUCIÓN: Universidad Industrial de Santander
CARGO: Profesional adscrito a la División de Servicios de Información.

Codirector del proyecto:

NOMBRE: Ing. Enrique Torres López
INSTITUCIÓN: Universidad Industrial de Santander
CARGO: Profesional adscrito a la División de Servicios de Información.

Autores:

Nombre: Pedro Antonio Otero Prada
Código: 2050251

Carrera: Ingeniería de Sistemas e Informática

Nombre: Marco Antonio Ruiz Rueda
Código: 2043701
Carrera: Ingeniería de Sistemas e Informática

Entidades interesadas en el proyecto:

División de Servicios de Información
Universidad Industrial de Santander

Oficina de Planeación
Universidad Industrial de Santander

1.1.2 Objetivos

1.1.2.1 Objetivo General

Realizar el diseño, implementación y puesta en funcionamiento de los módulos de ejecución, seguimiento y control, consultas e informes generales, del software soporte del Sistema de Gestión de Proyectos de la Universidad Industrial de Santander SGPUIS el cual es administrado por la Oficina de Planeación de esta institución, mediante el uso de últimas tecnologías de programación web.

1.1.2.2 Objetivos Específicos

Realizar el análisis, diseño, implementación y puesta en funcionamiento de los módulos de ejecución, seguimiento y control, consultas e informes generales,

enmarcados en el Sistema de Gestión de Proyectos UIS SGPUIS, de tal forma que permitan:

- Soportar el proceso del control integrado de los cambios efectuados durante la ejecución del proyecto, lo que implica:

  - Registrar las solicitudes de cambio identificadas.
  - Evaluar los cambios implementados.
  - Realizar el seguimiento y control de los cambios: conocer el flujo de la solicitud y su estado actual.

- Supervisar el cronograma del proyecto, de tal forma que se pueda:

  - Determinar el estado actual del cronograma del proyecto.
  - Determinar que el cronograma del proyecto se ha modificado, para gestionar los cambios reales a medida que suceden.
  - Procesar los cambios solicitados en línea, que sirven como base para la modificación del cronograma y su impacto en el desarrollo del proyecto.

- Desarrollar el módulo de consultas y de informes que permita conocer información de los proyectos de manera general o parcial por:

  - Por tipo de proyecto.
  - Por unidades académico administrativas responsables.
  - Por estado del proyecto.
  - Nivel de avance del proyecto.
  - Ejecución presupuestal del proyecto.

  El contenido de estas consultas e informes será definido de común Acuerdo con la Oficina de Planeación.

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, hace 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 se utiliza en la UIS, no satisface las expectativas actuales, 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 en los que contribuye 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.

- Contribuir con mejoras operativas, minimización de procedimientos y confiabilidad de la información, mediante la integración del SGPUIS con los sistemas de Recursos Humanos, Financiero y Académico.

- Efectuar consultas e informes generales según lo requiera el sistema de Gestión de Proyectos.

Viabilidad

- Técnica

  Para el desarrollo del proyecto, se cuenta con el recurso humano suficiente, apoyado por profesionales de la División de Servicios de Información con alta experiencia en la dirección de proyectos y en el manejo de las herramientas de modelado y desarrollo de aplicaciones. Adicionalmente, existe una infraestructura tecnológica conformada por software, equipos de desarrollo, servidores de base de datos, prueba y producción que garantizan la óptima ejecución del proyecto.

- Económica

  Como se ha mencionado, la División de Servicios de Información cuenta con todas las herramientas para la elaboración del proyecto, entre ellas se destacan las licencias de software de modelado y programación, los recursos necesarios para la capacitación y el personal de apoyo, que estarán disponibles a lo largo del desarrollo del proyecto.

- Social

  El proyecto satisface las necesidades planteadas por la División de Servicios de Información y la oficina de Planeación, referentes a la gestión de los proyectos de la universidad. Se le considera de alto impacto porque beneficia a gran parte de la comunidad universitaria y en especial:

  A la Dirección de la Universidad, a la cual le permitirá orientar los recursos de inversión hacia las estrategias de desarrollo plasmadas en el Plan de Desarrollo Institucional PDI.

A las unidades Académico-Administrativas le permitirá contar con una herramienta y elementos procedimentales para el desarrollo de los proyectos, que garanticen la coherencia en la maduración y desarrollo de sus propuestas.

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 facilite el uso eficiente y eficaz de los recursos de la institución. 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.

Previos al sistema del cual es objeto este plan de proyecto, se pueden identificar dos herramientas antecesoras:

El primero fue desarrollado en 1997 como el proyecto de grado de Melva Janeth Hernández Ariza. Utilizaba el manejador de bases de datos Access II y estuvo disponible para toda la comunidad universitaria. Sin embargo este sistema de información no estaba tan ligado al proceso de inversión como las directivas de la

Universidad requerían, además de que no seguía los procedimientos y el reglamento adecuado para cumplir su tarea. Estas inconveniencias acabaron por reducir su utilidad notoriamente; Sin embargo, sirvió como referencia a futuros desarrollos y como experiencia para la mejora de la gestión de los proyectos en la UIS.

Cuatro años después los estudiantes Edwin Ramírez y Silvia Suárez, trabajando conjuntamente con la Oficina de Planeación y la División de Servicios de Información, se proponen como proyecto de grado trabajar en una herramienta que supere las falencias de la anterior, disponible también en la intranet de la universidad. Esta alianza influyó en que el nuevo sistema fuera más coherente con la realización de todas las tareas involucradas en la gestión de proyectos, abarcando incluso las etapas más tempranas. El avance tecnológico también favoreció esta nueva versión, pues se contaba con la Arquitectura DNA de Microsoft y ASP. Era además un sistema más seguro, requiriendo la autenticación de los usuarios.

Por otro lado, el mantenimiento del software estaba a cargo de la División de Servicios de Información, y las asesorías a cargo de la Oficina de Planeación, de manera que la nueva herramienta tendría un soporte técnico que la haría gozar de mayor utilidad en el tiempo.

Sin embargo, con el cambio de las tecnologías usadas en la Universidad (Java) y las modificaciones que se realizaron a los modelos para la gestión de proyectos exigían el desarrollo de otra herramienta más moderna que se adaptara a la naturaleza dinámica de la universidad y los proyectos que ella gestiona, cuyo volumen de información iba en aumento. Por otro lado, la seguridad tomó un papel más importante y se hizo necesario hacerla más estricta y fácil de administrar.

De esta forma la Oficina de Planeación y la División de Servicios de Información unen esfuerzos nuevamente para desarrollar una herramienta moderna, dinámica, con mayores facilidades de uso y que maneje las políticas actuales; todo esto envuelto en un proyecto macro que comprende otros proyectos enfocados a satisfacer las necesidades de áreas críticas de la gestión.

2.2 Gestión de Proyectos

2.2.1 ¿Qué es un Proyecto?

Se define como proyecto según la guía *Project Management Body of Knowledge* (PMBOK) como:

"…un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único."[7]

Se dice que son temporales aunque esto no implica explícitamente una duración corta, se espera que una vez el producto o servicio sea puesto en funcionamiento dure por un periodo de tiempo mayor, al menos mientras sea necesario.

Los proyectos se formulan ante necesidades de empresas e instituciones como ofrecer nuevos servicios o productos, obras de infraestructura física, entre otras. Por esta naturaleza su ejecución implica muchos aspectos de la organización, como los recursos humanos y económicos. Semejante complejidad amerita un esfuerzo de administración y dirección que aplique los conocimientos y habilidades adquiridos por la organización en las actividades relacionadas, tales que el manejo

[7] Project Management Institute. Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK®). Tercera Edición. 2004. Four Campus Boulevard, Newtown Square, PA 19073-3299 EE.UU. P. 5.

de estos recursos sea óptimo y facilite el alcance de los objetivos perseguidos por el proyecto. El alcance de estos objetivos requiere que se especifiquen correctamente los requisitos para que el proyecto sea fiel a las necesidades de los interesados. Además también es muy importante lograr un balance ideal entre los planteamientos del alcance, los costos, la calidad, los recursos y los riesgos.

Aunque un proyecto podría ser llevado a cabo de muchas maneras, existen prácticas reconocidas globalmente como buenas prácticas. Se ha llegado al acuerdo general de que estás prácticas son recomendables dado que se ha comprobado que su aplicación a los proyectos aumenta sustancialmente las posibilidades de éxito de los mismos. El PMBOK recopila estas prácticas y las pone a disposición desde 1987, en forma de una guía para iniciar, planificar, ejecutar, supervisar, controlar y cerrar un proyecto. PMBOK es consistente con estándares como ISO 9000 y CMMI.

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

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
- Gestión de adquisiciones
- Adquirir y desarrollar el Equipo del Proyecto
- Distribución de 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

En este proyecto se ha considerando el proceso de control integrado de cambios como un flujo que se puede resumir en 4 pasos:

- Creación de las solicitudes de cambio: Proceso mediante el cual un solicitante ingresa en el sistema una petición debidamente justificada para efectuar cambios sobre algún aspecto requerido en el proyecto.
- Revisión de las solicitudes: Una vez diligenciada una solicitud de cambio, un evaluador la revisa y decide si los cambios propuestos son factibles o justificables y emite un concepto del cual se notifica al solicitante.

- Ejecución de los cambios propuestos: El solicitante, de haberse aprobado los cambios de la solicitud que creó, procede a ejecutarlos.
- Revisión de los cambios ejecutados: El evaluador observa los cambios ejecutados por el solicitante, teniendo en cuenta si estos cambios obedecen a la justificación de la solicitud (paso 1) y al concepto que emitió previamente el evaluador (paso 2), este último aprueba o rechaza estos 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 designado para tal fin.

- 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 Control Integrado de Cambios

El control de cambios es un proceso mediante el cual se asegura la no realización de variaciones que afecten el éxito del proyecto, y que aquellos implementados sean analizados, negociados, planeados e implementados de una manera adecuada y controlada. El proceso Realizar el Control Integrado de Cambios interviene desde el inicio de la ejecución del proyecto hasta su terminación.

En la gestión de proyectos es necesario llevar a cabo tareas que de forma coordinada dirijan el camino hacia el éxito del proyecto. Al imaginarse la fase de

planificación se tiende a creer que el proyecto no va a permitir ningún tipo de edición porque va a andar en línea recta hacia la meta, y eso sería ideal, pero, en la vida real un proyecto por estar bien planificado tiende a necesitar algunos cambios, estos no deben ser rechazados o permitidos sin antes ser analizados y evaluados. Todo esto sugiere un control sobre todas las fases del proyecto para que esos cambios no sean parte de un problema que desborde hacia el incumplimiento de los objetivos planteados, a estos controles se les llama en la gestión de proyectos: Control Integrado de cambios.

Como es importante asegurarse que el control de los cambios esté presente durante el proyecto en todas sus fases, una vez que las líneas base del plan (cronograma, desempeño, costos, alcance, entre otras) hayan sido definidas para la dirección del proyecto, estas sólo pueden cambiarse tras la generación y aprobación de una solicitud de cambio por medio de la ejecución del proceso de Realizar el Control Integrado de Cambios.

Para realizar el control integrado de cambios se deben revisar todas las solicitudes de cambios, aprobar o rechazarlos y negociarlos, de manera tal que se asegure que sólo los cambios aprobados se incorporen a una línea base revisada.

Este proceso comprende las siguientes actividades:

- Influir en los factores que eluden el control integrado de cambios, de modo que se implementen únicamente los cambios aprobados.
- Hacer el análisis, la revisión y la aprobación o rechazo oportuno de las solicitudes realizadas.
- Administrar los cambios aprobados.
- Dirigir hacia el plan de la dirección de proyectos solo los cambios aprobados, para así mantener la integridad de las líneas bases.

- Examinar y aprobar o rechazar todas las acciones de prevención y corrección recomendadas.
- Regular los cambios solicitados y efectuados a través de todas las fases del proyecto.

Si se trabaja con un sistema de información veraz, siempre los cambios deben registrarse en el módulo de gestión de cambios y/o al sistema de gestión de la configuración, según como se tramite el control de cambios. Las solicitudes de cambio están sujetas a los procesos especificados en los sistemas de control de cambios.

Estos procesos de solicitud de cambios pueden requerir información sobre los impactos en el tiempo y costo estimados.

Cada solicitud de cambio planteada debe ser aprobada o rechazada por un personal autorizado que pertenezca al equipo de dirección del proyecto o a una organización externa.

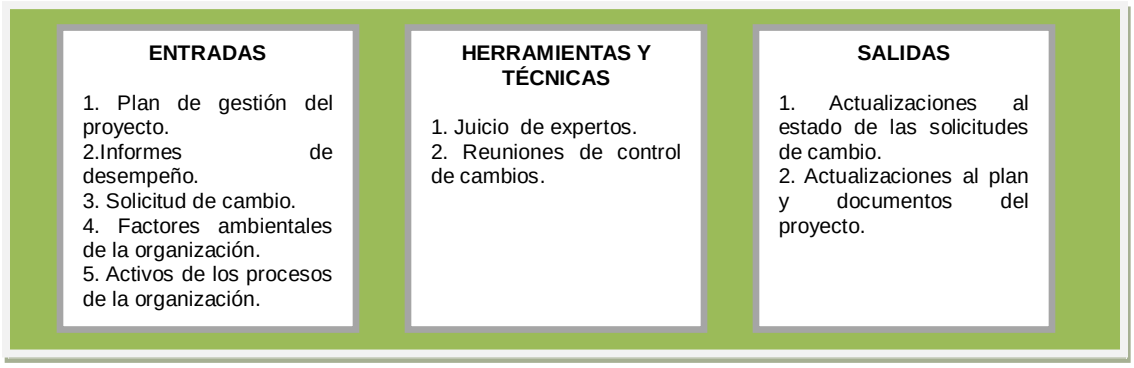
En muchos proyectos, se otorga al director del proyecto la autoridad para aprobar cierto tipo de solicitudes de cambio, según se define en los documentos del proyecto que describen los roles y responsabilidades, y según la oficina de Planeación (unidad encargada de la gestión de proyectos dentro de la Universidad Industrial de Santander) lo indique. Siempre que se requiera, en el proceso Realizar el Control Integrado de Cambios se tendrá un comité de control de cambios (CCB) que será responsable de aprobar o rechazar las solicitudes de cambio.

El alcance del proyecto, el plan de gestión, y otros entregables, deben estar siempre actualizados. Esto se logra administrando rigurosa y cuidadosamente los

cambios solicitados y efectuados. Esta gestión minuciosa permite mantener un control específico sobre el cronograma, el presupuesto, el alcance y la calidad del proyecto.

Es importante sobreponerse a la complejidad del proceso y mantener actualizado el sistema, puesto que hay cambios que afectan puntos importantes del proyecto, como por ejemplo cuando el presupuesto se modifica negativamente, con esto es probable que ciertas tareas puntuales no puedan cumplirse ante la falta del presupuesto. Es de suma importancia definir con anterioridad cuales formatos se pueden sujetar a cambios, y estipular fechas para que se puedan efectuar, de modo tal que no todo en el proyecto sea modificable y que se lleve un control del tiempo en el cual el solicitante debe actuar y el comité de cambios analizar y evaluar la solicitud o el cambio. En cuanto a los atributos específicos aprobados para cambiar, tendrán también una vigencia para la ejecución del cambio. El equipo del proyecto programa la implementación de las solicitudes de cambio aprobadas.

2.2.5.1 Esquema de entradas, herramientas/técnicas y salidas involucradas[8]



Entradas:

[8] Tomado de: http://www.zanzivar.com/zzc/index.php?option=com_content&view=article&id=35:gestion-de-la-integracion-realizar-el-control-integrado-de-cambios&catid=7:blog-administracion-de-proyectos&Itemid=18

1. **Plan para la dirección de proyectos**
2. **Informes de desempeño**
3. **Solicitudes de cambio**: Pueden incluir acciones correctivas, preventivas y reparación de defectos, sin embargo las acciones correctivas y preventivas no afectan las líneas bases del proyecto sino únicamente el desempeño.
4. **Factores Ambientales de la empresa**:
    a. Sistema de información para la administración de proyectos
5. **Activos de los procesos de la Organización**:
    a. Procedimientos de control de cambios
    b. Procedimientos para aprobar y emitir autorizaciones de cambio
    c. Bases de datos para la medición de procesos
    d. Activos del proyecto (líneas bases del alcance, costo y cronograma)
    e. Bases de conocimiento de la gestión de la configuración

Herramientas y técnicas:

1. **El juicio de expertos**, además del juicio de expertos del equipo de dirección de proyectos se puede considerar también consultores, interesados, asociaciones profesionales, grupos industriales, expertos en la materia, oficina de dirección de proyectos.
2. **Reuniones de control de cambios**, todas las decisiones del comité se documentan y comunican a los interesados para su información e implementación de acciones de seguimiento.

Salidas:

1. Actualizaciones al estado de las solicitudes de cambio
2. Actualizaciones al plan para la dirección de proyectos
3. Actualizaciones a los documentos del proyecto

2.2.6 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 institucional.

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"[9].

Con este acuerdo, se logró conferir solidez a la estructura de todos los procedimientos involucrados en las diferentes responsabilidades de las Directivas de la Institución.

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 apuntaran al crecimiento de la Universidad"[10]. 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.

---

[9] 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.
[10] Ibíd. P 23.

2.2.7 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";[11] 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:

2.2.7.1 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.

2.2.7.2 Componente de Capacitación

La capacitación tiene el objetivo de fomentar el uso de las Metodologías propuestas por el DNP y también, la divulgación del uso de técnicas concernientes

[11]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)

a la evaluación de proyectos para finalmente lograr una cultura de proyectos en todo el territorio nacional.

2.2.7.3 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.

2.2.8 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

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la cual basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran

peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas “guerras de métodos” que se dieron a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

En la División de Servicios de Información (DSI) se han establecido estándares concernientes al diseño de sistemas, los cuales deben ser desarrollados por módulos y deben 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

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 se diseña 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 diagrama de casos de uso se identifican tres elementos básicos:

- ***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:

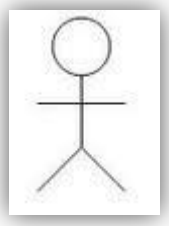


Figura 2: Actor

- ***Caso de Uso:*** Un caso de uso describe una funcionalidad del sistema que produce un resultado perceptible para el usuario, sin entrar en detalles sobre *cómo* la realiza. Su comportamiento puede especificarse como un flujo de eventos en texto formal, o en pseudocódigo. En su conjunto, los casos de uso son los que describen todas las funcionalidades del sistema. 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, es decir, cuando el Actor comete errores al momento de interactuar con el sistema.

- ***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:

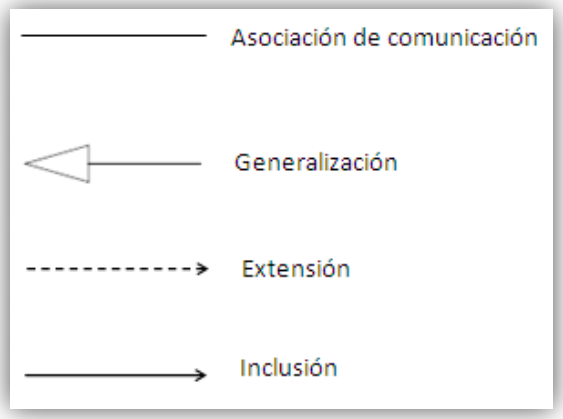


Figura 4: Tipos de Relaciones

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

Una clase es una construcción que modela un objeto perteneciente a un sistema. Comprende una serie de atributos para representar el estado de los objetos que pertenezcan a ella.

Estas clases presentan ciertas características:

- *Estado:* Es definido por los atributos que contiene la clase y por sus posibles relaciones con otras.
- *Comportamiento:* Explica la funcionalidad de una clase, señalando todas las operaciones que esta puede realizar.

Una clase es representada en la figura 5:

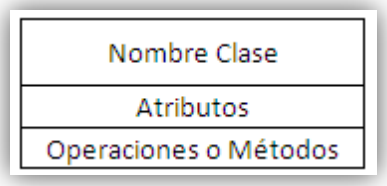


Figura 5: Representación de clase

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:

- ***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:

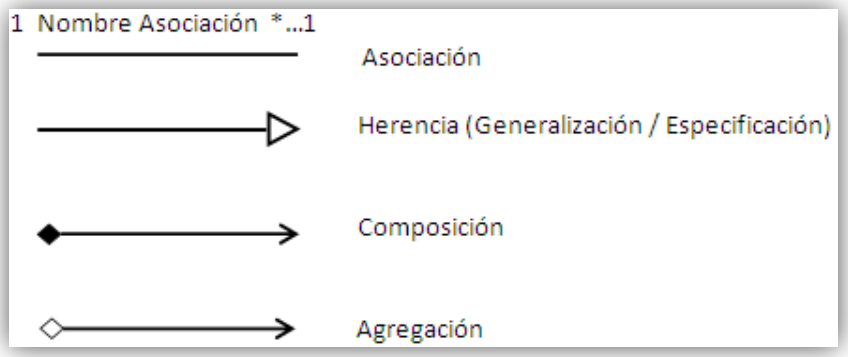


Figura 6: Tipos de Relaciones entre Clases

2.4 Tecnologías de Desarrollo de Aplicaciones Web

2.4.1 JAVA EE5

En los últimos años la gran mayoría de servicios y aplicaciones informáticas han migrado hacia los entornos web en pro de la ejecución en múltiples plataformas con diferentes capacidades de procesamiento e incluso ampliando su cobertura hacia aplicaciones móviles, buscando la reducción de costos y optimización de los procesos. Sin importar la diversidad de las aplicaciones, los desarrolladores reconocen puntos comunes imprescindibles en ellas: Distribuidas, seguras, robustas, con alta capacidad transaccional y eficientes. La respuesta de Java para estas necesidades es Java Enterprise Edition, que busca agilizar el proceso de desarrollo brindándole a los desarrolladores toda una variada oferta de APIs, frameworks y tecnologías para responder a distintas necesidades.

Java Enterprise Edition 5 (Java EE 5) se centra en hacer más fácil el desarrollo. Sin embargo, conserva la riqueza de la plataforma J2EE 1.4. Reduce enormemente la necesidad de archivos XML de configuración (configuración del despliegue de las aplicaciones), lo que repercute en rapidez en el desarrollo ante

la eliminación de código repetitivo. Ofrece la tecnología de servicios web API, haciendo que la codificación sea más simple y directa, pero manteniendo el poder que ha establecido a Java EE como la primera plataforma para servicios web y desarrollo de aplicaciones empresariales.

2.4.1.1 EL MODELO DE APLICACIÓN JAVA EE

La plataforma empresarial de Java fue diseñada para soportar aplicaciones que brinden servicios empresariales para clientes, empleados, proveedores, socios y demás personas o entidades que interactúan con la empresa.

El modelo de aplicación Java EE define una arquitectura para implementar servicios como aplicaciones de múltiples capas que distribuyen la escalabilidad, accesibilidad y facilidad de manejo que se necesita para aplicaciones empresariales.

2.4.1.2 APLICACIONES DE MÚLTIPLES CAPAS DISTRIBUIDAS

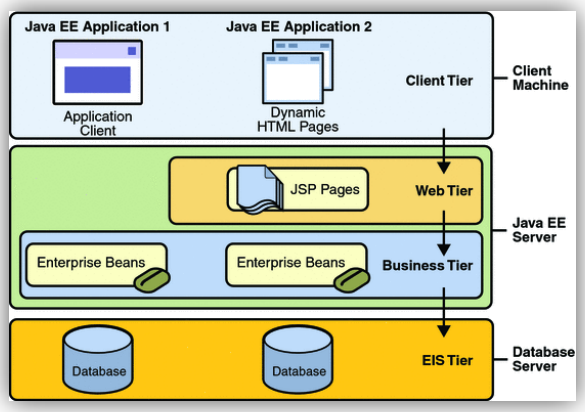


Figura 7: Aplicaciones de múltiples capas[12]

12 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

La lógica de la aplicación se distribuye según su función en componentes, los cuales son instalados en diferentes máquinas de acuerdo a la capa en el ambiente de múltiples capas de Java EE a la cual pertenece el componente de aplicación.

La figura anterior muestra dos aplicaciones Java EE de capa múltiple dividida en sus capas las cuales están descritas en la siguiente lista.

- Los componentes de la capa de cliente se ejecutan en la máquina del cliente.
- Los componentes de la capa Web se ejecutan en el servidor Java EE.
- Los componentes de la capa de negocios se ejecutan en el servidor Java EE.

Una aplicación Java EE puede consistir en tres o cuatro capas, como se muestra en la figura, sin embargo las aplicaciones de múltiples capas de Java EE son consideradas de tres capas porque se distribuyen en tres ubicaciones: máquinas clientes, la máquina servidor Java EE y la base de datos. Estas aplicaciones extienden el modelo cliente servidor, ubicando un servidor de aplicación entre la capa cliente y la de almacenamiento.

2.4.2 Servidor de Aplicaciones – Jboss

El servidor de aplicaciones JBoss, es una herramienta certificada para el desarrollo de aplicaciones empresariales Java. Su madurez y el esfuerzo de muchos desarrolladores, e incluso las sugerencias que han realizado sus usuarios, han permitido que JBoss AS (Application Server) se popularice ampliamente y sea común en los currículos de desarrolladores.

Es reconocido por soportar los estándares más recientes. De hecho, es el primer servidor de aplicaciones en alcanzar la certificación J2EE 1.4 cuando salió su versión 4.0. Pero JBoss no sólo marca la pauta en la adopción de estándares con

su servidor de aplicaciones, sino en la imposición de los mismos. Hace parte del *Java Community Process* (JCP).

2.4.3 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.

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

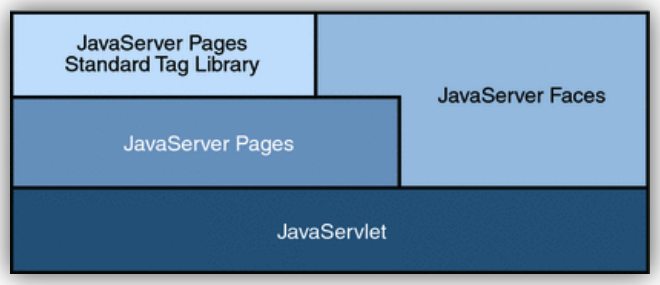


Figura 8: Tecnologías Java para aplicaciones web[13]

13 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

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 sí misma es más fácil de mantener, de escalar y más robusta.

Los componentes web son soportados por los servicios de una plataforma en tiempo de ejecución llamada contenedor web. Un contenedor web proporciona los servicios como distribuir una petición, seguridad, concurrencia y manejo de ciclos de vida. También les da a los componentes web acceso a las APIs para nombrado, transacciones y correo electrónico.

2.4.3.1 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 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.

2.4.3.2 Módulos web

En la arquitectura Java EE, los componentes web y los ficheros con contenido estático como imágenes son llamados recursos web. Un módulo web es la unidad más pequeña de un recurso web que se puede utilizar y desplegar. Un módulo web Java EE corresponde con una aplicación web como se define en la especificación de Java Servlet. (numeral 2.4.4)

Además de los componentes web y los recursos web, un módulo web puede contener otros ficheros:

- Clases utilitarias del lado del servidor (beans para bases de datos, carritos de compras y demás). A menudo estas clases cumplen con la arquitectura JavaBeans.
- Clases del lado del cliente (applets y clases utilitarias).

Un módulo web tiene una estructura específica. El directorio más alto de la jerarquía de directorios de un módulo web es la raíz de documento de la aplicación. Es donde las páginas JSP, clases y archivos del lado del cliente y los recursos estáticos son almacenados.

La raíz de documentos contiene un subdirectorio llamado WEB-INF, que contiene los siguientes ficheros y directorios:

- web.xml: El descriptor de despliegue de aplicación.
- Los ficheros descriptores de las librerías de etiquetas.
- classes: Un directorio que contiene las clases del lado del servidor: componentes Servlets, clases utilitarias y JavaBean.

- tags: Un directorio que contiene ficheros de etiquetas, que son implementaciones de librerías de etiquetas.
- lib: Un directorio que contiene los archivos JAR de las librerías llamadas por las clases del lado del servidor.

Se pueden crear subdirectorios específicos de la aplicación (esto es, directorios de paquete) tanto en la raíz de documentos como en el directorio WEB-INF/classes/.
Un módulo web puede ser desplegado como una estructura de ficheros sin empaquetar o puede ser empaquetado en un fichero JAR conocido como un archivo web (WAR). Dado que el contenido y uso de los ficheros WAR difieren de aquellos ficheros JAR, el nombre del fichero WAR utiliza una extensión .war. El módulo web descrito es portátil, se puede desplegar en cualquier contenedor web que cumpla con la especificación Java Servlet.

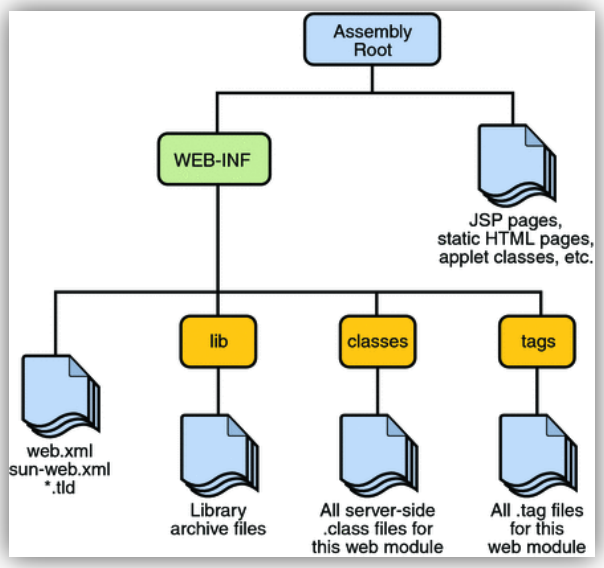


Figura 9: Estructura de un módulo web[14]

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/acadad.html

2.4.4 Tecnología Servlet Java

2.4.4.1 ¿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 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.

2.4.5 JAVA SERVER FACES

2.4.5.1 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.

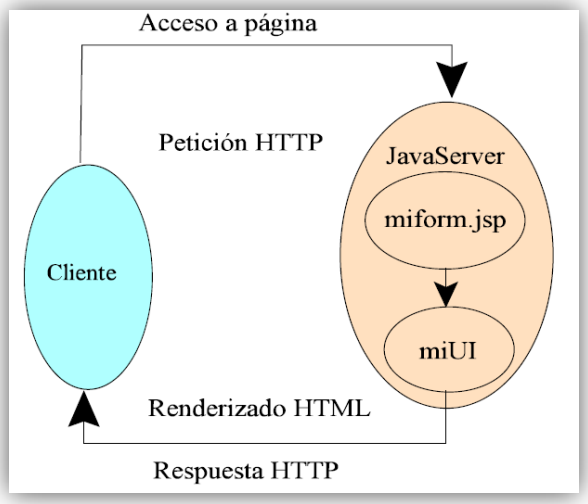


Figura 10: Diagrama de una aplicación JSF[15]

[15] 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, la página JSP (**miform.jsp**), especifica los componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por la tecnología 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 los componentes.
- Objetos del modelo que encapsulan los datos y las funcionalidades de los componentes específicos de la aplicación (lógica de negocio).

2.4.5.2 Beneficios de la Tecnología JavaServer Faces (JSF)

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 permitir la selección del 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 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 JavaServer Faces 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.

2.4.5.3 ¿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 beans para 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 Sun Microsystems, 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.

Otra ventaja importante de las aplicaciones JavaServer Faces es que los componentes UI de la página están representados en el servidor como objetos con estado. Esto permite a la aplicación manipular el estado del componente y conectar los eventos generados por el cliente en el lado del servidor.

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.

Debido a la división de labores que permite el diseño de la tecnología JavaServer Faces, el desarrollo y mantenimiento de una aplicación JavaServer Faces se puede realizar muy rápida y fácilmente.

2.4.6 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 interactuar 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), entre otras ventajas. 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 se muestra un esquema del patrón de diseño MVC:

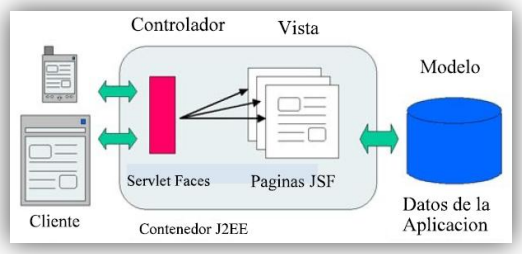


Figura 11: Modelo vista controlador en JSF[16]

[16] 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

Este modelo de arquitectura presenta otras importantes ventajas:

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

- 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.

2.4.6.1 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.

2.4.6.2 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 interactúa con el modelo a través de una referencia al propio modelo.

2.4.6.3 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.

El siguiente diagrama muestra la relación existente entre el modelo, la vista y el controlador cada vez que un usuario realiza operaciones sobre la página:

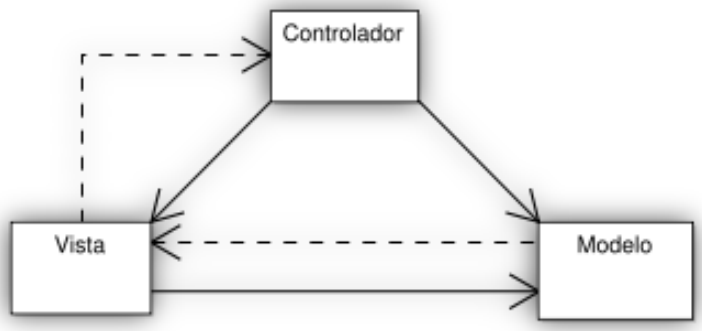


Figura 12: Modelo Vista-Controlador

En la figura 12 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 puede describir a continuación:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
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. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. 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.

2.4.7 SEAM

Seam es un framework de aplicaciones de Java Enterprise Edition inspirado 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 tener un 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 incorporó 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 el flujo de páginas de la capa de presentación utilizando el mismo lenguaje que jBPM utiliza para la definición de procesos de negocio.

- Declarativa administración del estado

Tradicionalmente, las aplicaciones J2EE implementan la administración de estado de forma manual. Este enfoque es la fuente de muchos errores y pérdidas de memoria cuando las aplicaciones no pueden limpiar los atributos de sesión, o cuando los datos de sesión asociados a diferentes flujos de trabajo chocan en una aplicación de múltiples ventanas. Seam tiene el potencial de eliminar casi por completo esta clase de errores.

La declarativa de administración del estado se ha logrado gracias a que Seam extiende el modelo de contexto definido por la especificación de servlets (petición, sesión, aplicación) con dos nuevos contextos (conversación y procesos de negocio).

- Biyección

La biyección es dinámica y bidireccional, se puede 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, lo que 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.

- 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 asincronías, 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.

Seam funciona en cualquier servidor de aplicaciones Java EE y también en Tomcat. Si el IDE no admite EJB 3.0 puede utilizar Seam incorporado en la gestión de transacciones con JPA o Hibernate3 para la persistencia. También se puede utilizar JBoss embebido en Tomcat, y tener un soporte completo para EJB 3.0.

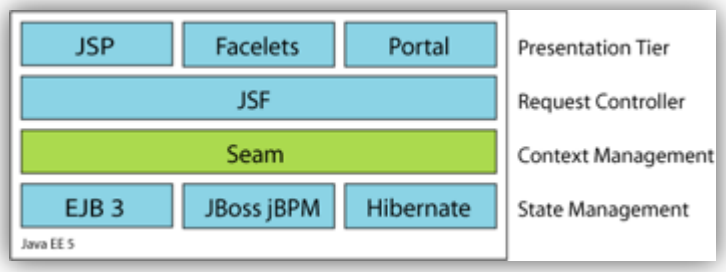


Figura 13: Arquitectura de Seam[17]

2.4.7.1 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.

Seam no sólo fue el primer framework que introdujo el concepto de Biyección de componentes sino que también es considerado como un Meta-Contenedor (Contenedor de Contenedores) que define y administra el ciclo de vida de cada componente inyectado o eyectado en un Contenedor o Contexto.

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

---

17 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

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 siempre y cuando no haya un redireccionamiento a otra 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, 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). 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.

- *Variables de Contexto*

Corresponden al conjunto de objetos contenidos en el espacio de nombres definido por el Contexto. A una variable de contexto se le puede asociar cualquier valor deseado pero usualmente se realiza un enlace de las instancias de Componentes Seam a las variables de Contexto; de esta manera una instancia de un componente es identificada por su nombre dentro de un contexto.

❖ *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 contextos 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 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 soportan 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.

2.4.7.2 Eventos, interceptores y el manejo de excepciones

❖ *Eventos Seam*

El modelo de componentes Seam fue desarrollado para su uso con aplicaciones orientadas a eventos. Los eventos en Seam son de varios tipos:

- Eventos JSF
- Eventos de transición jBPM
- Acciones de página de Seam
- Eventos de Seam orientados a componentes
- Eventos contextuales de Seam

❖ *Acciones de página*

Una acción de página Seam es un evento que ocurre antes de renderizar 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 hacia una vista.

❖ *Parámetros de página*

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*

Es posible utilizar las reglas estándar de navegación JSF definidas en el faces-config.xml en una aplicación de Seam. Sin embargo, las reglas de navegación estándar de JSF tienen una serie de limitaciones:

- No es posible especificar parámetros de la petición usada para ser usados al momento de redirigir a una página.
- No es posible iniciar o finalizar las conversaciones a partir de una regla de navegación.
- Las reglas de navegación funcionan evaluando el valor de retorno de un método; no es posible evaluar una expresión EL arbitraria.

❖ *Eventos orientados a componentes*

Los componentes de Seam pueden interactuar tan sólo llamando a sus métodos entre ellos. Los componentes con estado pueden incluso poner en práctica el modelo observador/observable. Pero para permitir que los componentes interactúen de una manera más débilmente acoplada de lo que es posible cuando

los componentes llaman a los métodos de los otros directamente, Seam proporciona eventos impulsados por componentes.

❖ *Eventos Contextuales*

Seam define una serie de eventos 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 que el contexto (sesión, conversación, etc) 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. Para los interceptores que no necesitan tener un estado, Seam deja optimizar el rendimiento mediante una anotación.

Mucha de la funcionalidad de Seam está implementada como un conjunto de interceptores de Seam predefinidos. No es necesario especificarlos explícitamente para todos los componentes que se desarrollen; estos interceptores existen para todos los componentes de Seam que son interceptables.

❖ *Administrar excepciones*

JSF es limitado en el manejo de excepciones. Por esta razón Seam permite especificar como ciertas clases de excepciones son tratadas bien sea con anotaciones o mediante declaraciones de estas clases en un archivo .xml.

2.4.7.3 Conversaciones y gestión de espacio de trabajo

❖ *Modelo de conversación Seam*

Seam propaga automáticamente el contexto de la conversación a través de los *postbacks* JSF y los redireccionamientos. Si no se hace nada especial, una solicitud tipo GET, por ejemplo, no propagará el contexto de la conversación y será procesada en una nueva conversación temporal. Por lo general este es el comportamiento deseado.

Si se quiere propagar una conversación Seam a través de una solicitud non-faces (no dirigida al árbol de componentes de JSF), se requiere especificar la identificación de la conversación como un parámetro de la petición.

El modelo de conversación de Seam facilita crear aplicaciones que se comporten correctamente con múltiples ventanas. Por lo general esto es suficiente salvo ciertos requerimientos adicionales de algunas aplicaciones como:

- 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 que se desee.

❖ *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.

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.

2.4.7.4 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 consistir de 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).

El siguiente diagrama representa en forma básica un ejemplo de paginación Stateful:

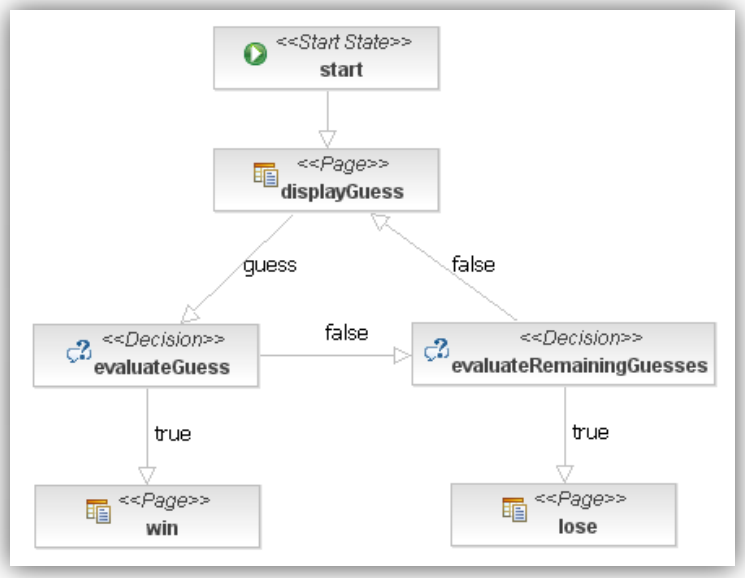


Figura 14: Modelo de Navegación Stateful[18]

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.

2.4.7.5 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 cómo se integran estos ORM con el Framework es de vital importancia para conocer las ventajas más importantes de Seam.

---

18 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

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.

Muy pronto los desarrolladores que utilizaron tecnologías anteriores a Seam comenzaron a discrepar con el diseño del contexto de persistencia asociado a una transacción atómica, y es ahí donde Seam y EJB 3.0 surgen como una alternativa para solucionar este problema. Lo que se proponía era cambiar el concepto de Transacción Atómica por el de Transacción Optimista (una transacción que soporta más de una solicitud sin perder el contexto de persistencia vigente).

❖ *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, aún 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.

2.4.7.6 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 framework Seam es la clase del controlador y sus subclases EntityControllerHibernateEntityController y BusinessProcessController,

que ofrecen algunos métodos de conveniencia para el acceso a los componentes integrados.

2.4.7.7 SEGURIDAD SEAM

La seguridad del API de Seam abarca las siguientes áreas:

- Autenticación: esta capa se basa en JAAS que permite a los usuarios autenticarse en cualquier proveedor de seguridad.

- Administrador de identidad: un API para la gestión de usuarios y sus funciones en tiempo de ejecución.

- Autorización: esta capa incluye apoyo a las funciones de usuario, permisos persistentes basados en reglas y una resolución de autorización para implementar fácilmente la lógica de seguridad.

- Permiso de gestión: conjunto de componentes integrados para facilitar la gestión de la política de seguridad de una aplicación.

- CAPTCHA: para ayudar en la prevención de software automatizado.

❖ *Desactivación De La Seguridad*

En caso que se desee implementar su propio enfoque de seguridad o por cualquier otro motivo se desee desactivar la seguridad que proporciona Seam simplemente se debe llamar al método estático Identity.setSecurityEnabled (false), que deshabilita la infraestructura de seguridad.

Otra opción para configurar la aplicación es controlando estos parámetros en el archivo components.xml:

- Entidad de seguridad
- Interceptor de seguridad Hibernate
- Interceptor de seguridad Seam
- Página de restricciones
- Servlet API de integración de seguridad

❖ *Autenticación*

La autenticación basada en JAAS (Java Authentication and Authorization Service) proporciona un API robusto y altamente configurable para manejar la autenticación de usuario. Si se necesita una autenticación menos compleja Seam ofrece un método simplificado de autenticación que oculta la complejidad de JAAS.

- *Configurar un componente autenticador*

Seam incorpora un módulo JAAS de entrada, SeamLoginModule, que delega la autenticación a los propios componentes de Seam; este módulo ya viene configurado y no requiere ningún archivo de configuración adicional. Permite escribir un método de autenticación utilizando las entidades de su aplicación, o, para autenticar con algún proveedor.
La configuración simplificada requiere el componente de identidad que se configura en el archivo components.xml.

- *Escribir un método de autenticación*

Este método no toma parámetros, y devuelve un booleano que indica si la autenticación fue realizada con éxito o no.

Cualquier rol que corresponda al usuario debe ser asignado mediante Identity.addRole ().

Para evitar que el método de autenticación sea invocado varias veces durante una única solicitud, cualquier código que deba ejecutarse en una autenticación debe ser escrito implementando un observador de eventos.

- *Identity.addRole ()*

Este método se comporta de manera diferente dependiendo de si las sesiones actuales se autentican o no. El siguiente diagrama de secuencia representa la lista de roles pre autenticados como un objeto de primera clase, para mostrar más claramente el proceso de autenticación.

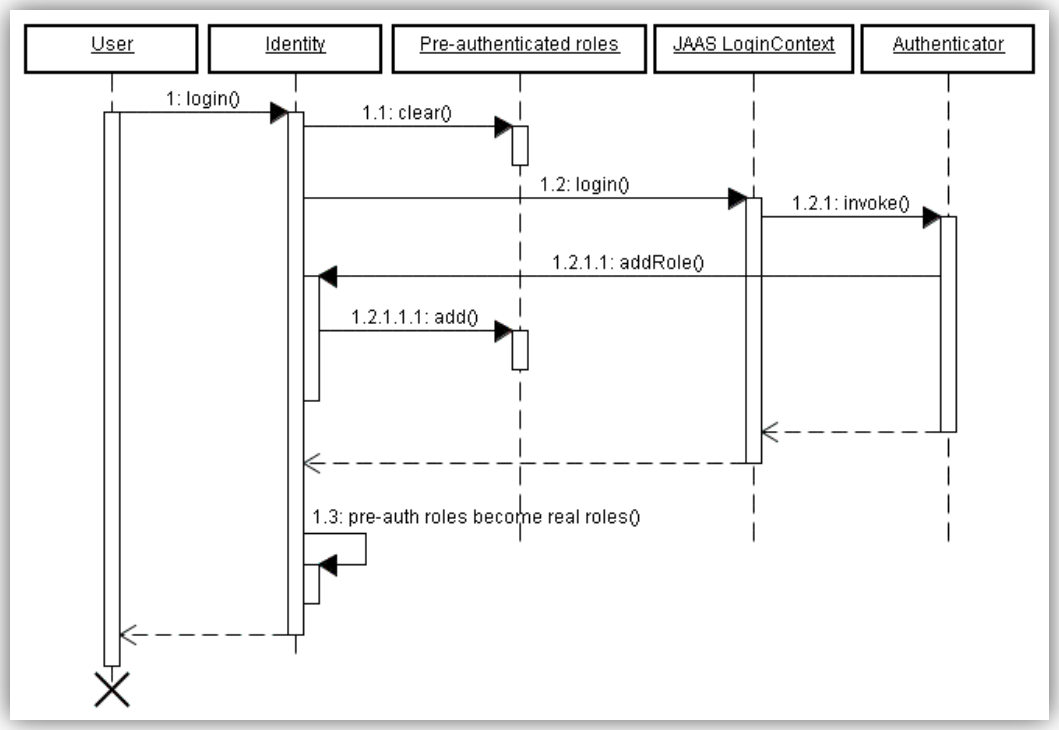


Figura 15: Diagrama de Secuencias Autenticación[19]

- *Manejo de excepciones de seguridad*

Con el fin de evitar que los usuarios vean la página de error por defecto, se recomienda que en pages.xml se configure para redirigir los errores de seguridad a una página especial. Los principales tipos de excepciones producidas por el API de seguridad son:

- NotLoggedInException: se produce cuando el usuario intenta acceder a una acción restringida o cuando no se ha autenticado.

19 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/security.html#d0e8695

- AuthorizationException: se produce cuando el usuario ya está conectado y ha intentado acceder a una acción restringida para su rol.

- *Redirección de entrada*

Si un usuario no autenticado intenta acceder a una acción restringida, Seam puede configurarse para redirigirlo a una página de inicio de sesión y una vez el usuario se haya autenticado lo devuelve automáticamente a la página que intentaba acceder.

- *Autenticación HTTP*

Su uso no es recomendado a menos que sea absolutamente necesario, Seam proporciona los medios para la autenticación utilizando HTTP básico o HTTP Digest (RFC 2617). Para cualquier forma de autenticación el filtro de autenticación debe estar habilitado en components.xml.

- *Escribir un autenticadorDigest*

Si usa la autenticación implícita, la clase autenticador debe ampliar la clase abstracta org.jboss.seam.security.digest.DigestAuthenticatory, y el uso de validatePassword () para validar la contraseña del usuario.

- *Características avanzadas de autenticación*

Estas características son proporcionadas por el API de seguridad para abordar requisitos de seguridad más complejos.

- *Configuración del contenedor JAAS*

Si delega la seguridad en el sistema por defecto, proporciona una propiedad JAAS-config-nombre en components.xml. Por ejemplo, si está utilizando JBoss AS y desea utilizar la política de otros (que utiliza el módulo de entrada UsersRolesLoginModule proporcionada por JBoss AS), la entrada en components.xml se vería así:

<security:identity jaas-config-name="other"/>

- *Gestión de identidad*

Seam proporciona un estándar API para la gestión de usuarios de acuerdo al rol que desempeñe cada uno, el centro de gestión de identidad es el componente identityManager, que proporciona los métodos para crear, modificar y eliminar usuarios, concediendo y eliminando funciones.

2.4.7.8 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. 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 en 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 clúster.

- *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 inhabilitado cuando los datos cambian, así que su invalidación se debe ser explícita en el código de la aplicación o establecer las políticas de expiración apropiadas.

2.4.8 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-relational mapping).

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.

2.4.9 EJB 3.0

Enterprise JavaBeans (EJB) es una arquitectura de componentes para la construcción de aplicaciones empresariales ejecutadas en servidores. Tiene por propósito proveer una forma estándar de implementar este tipo de aplicaciones, haciéndose cargo de aspectos comunes y repetitivos como la persistencia, la integridad transaccional y la seguridad, permitiendo que el desarrollador pase a preocuparse exclusivamente por la lógica del negocio en sí.

JBoss AS fue de los primeros servidores de aplicaciones en adoptar las especificaciones de EJB 3.0. Este modelo de EJB simplifica el desarrollo eliminando la necesidad de una interfaz “Home” y descriptores de despliegue (reemplazándolos por anotaciones), y facilita la implementación de la persistencia de una nueva manera por medio de JPA.

2.4.10 JPA

Más conocida por su sigla **JPA**, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares.

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.

2.4.11 JPQL

Java Persistence Query Language es el lenguaje de consultas que se usará en el desarrollo de este trabajo. 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. 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.

El análisis de requerimientos de este proyecto se especificó teniendo en cuenta la metodología propuesta por el PMI (Project Management Institute) en el PMBOK (Project Management Body of Knowledge). También, la DSI y la oficina de Planeación acordaron conjuntamente los requerimientos para adaptar correctamente la metodología propuesta por el PMI a las necesidades específicas de la universidad.

Se hizo un seguimiento continuo a los prototipos desarrollados para acercarlos cada vez más a los requerimientos y para que la DSI y la oficina de Planeación tuvieran un acercamiento al producto final y pudieran especificar cualquier requisito que no estuviera siendo cumplido o que pudiera refinarse.

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 modelado Enterprise Architect. Teniendo en cuenta que los diagramas de secuencia describen en el tiempo la serie de pasos para un caso de uso, y que para este desarrollo se harán prototipos no funcionales o de funcionalidad básica, se decidió omitir los diagramas de secuencia. Los prototipos ofrecen la misma visión de comportamiento del sistema, y son útiles para el desarrollo posterior.

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 desarrolladas para el servicio de la Universidad, teniendo como base la Arquitectura de Desarrollo de Aplicaciones de JAVA EE5 y la plataforma IBM 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 terminada la codificación, 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 tenidos 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:

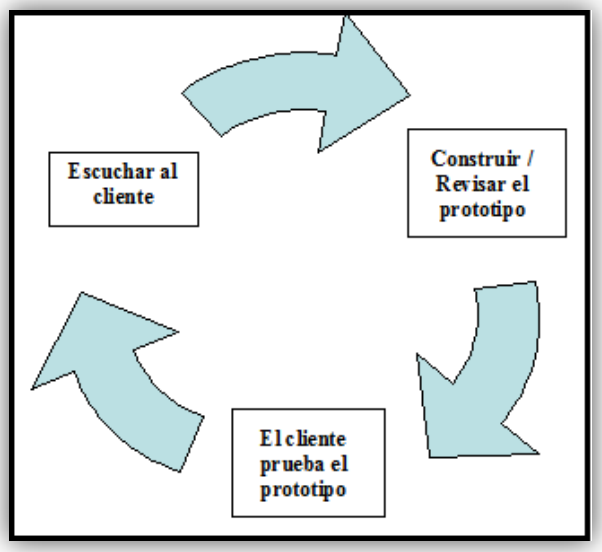


Figura 16: Modelo Construcción Por Prototipos

3.2.1.1 Estructura

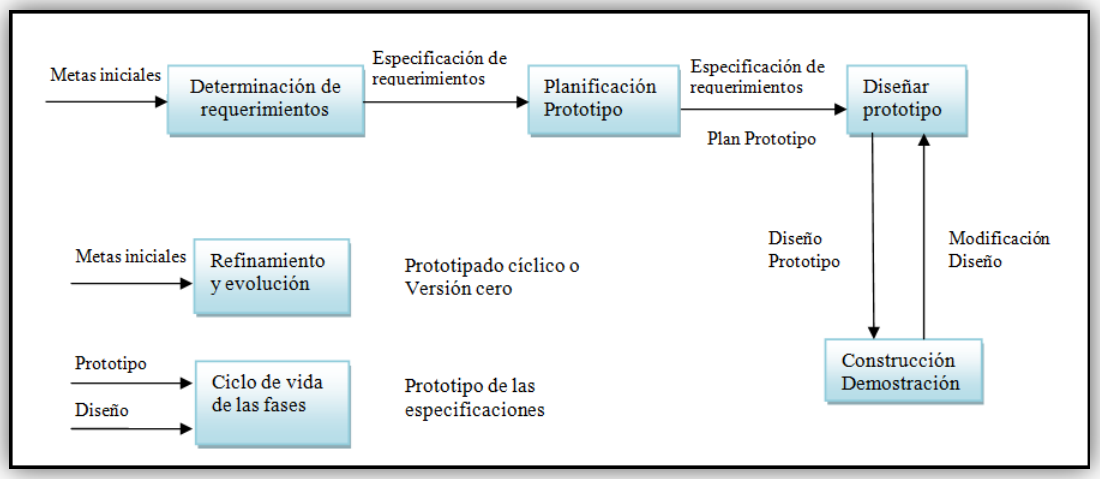


Figura 17: Estructura del modelo

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

A continuación se ilustra un ejemplo de cómo se desarrollaron los diagramas UML. Los diagramas completos están en medio digital que se adjunta a este documento. Se presentará una imagen con todas las clases empleadas directamente en el desarrollo de este proyecto, pero de allí se extraerán las más significativas para así, entrar en detalle. En el medio digital se encuentran los esquemas completos y las especificaciones de cada una de las clases y sus atributos.

Para los casos de uso se hará exactamente lo mismo, de modo que si se desea entrar en detalles de todo el diagrama, en el medio digital se tiene una guía completa para entender y familiarizarse con las características específicas y las funciones del sistema.

.

3.3.1.1 Diagramas de Clases

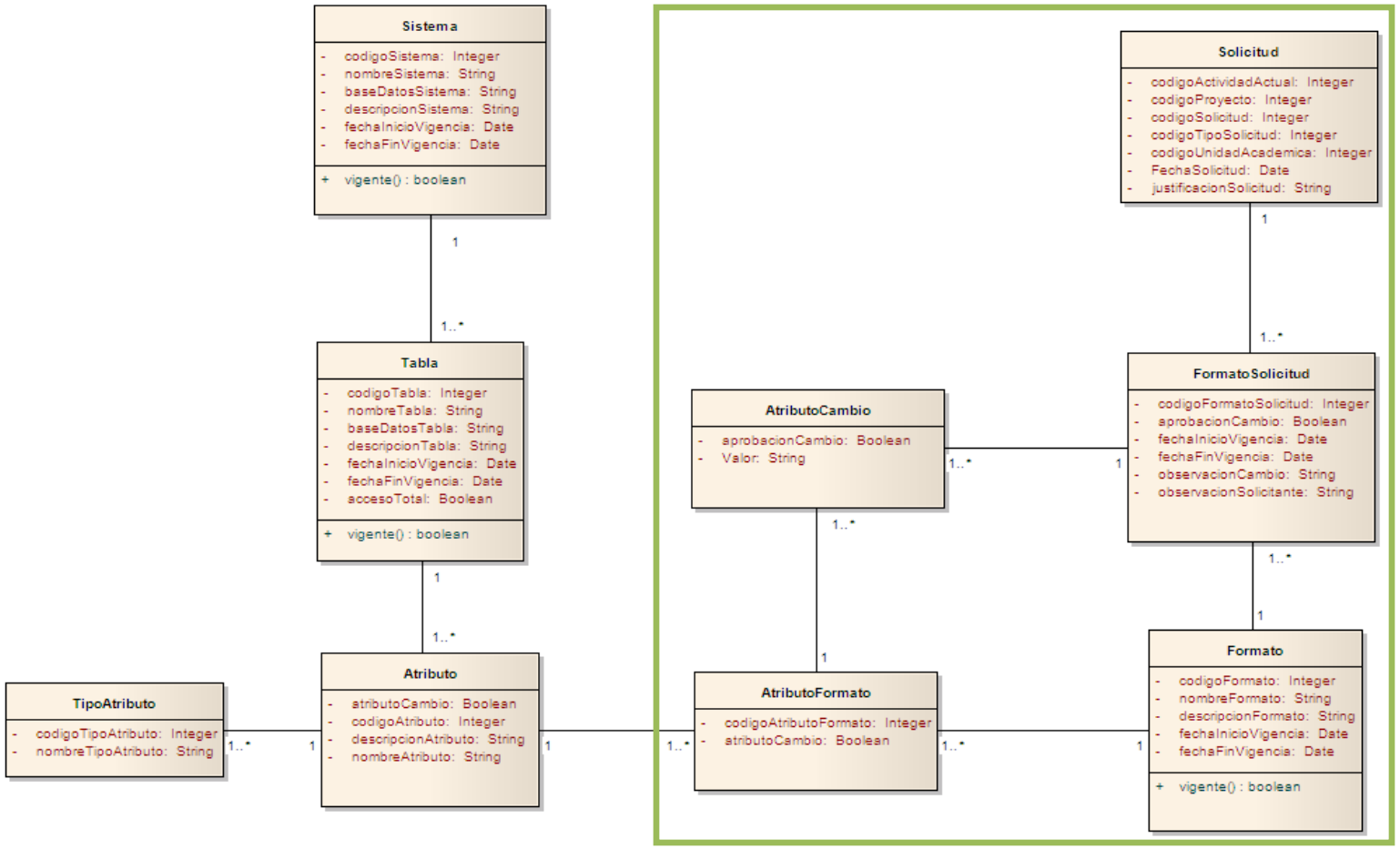


Figura 18: Diagrama de clases

Para describir este diagrama y aclarar el proceso de Control de Cambios según lo efectúa el módulo desarrollado en este proyecto para SGPUIS, se han escogido las clases resaltadas en la sección derecha del diagrama. De estas clases se hará una breve explicación. La documentación de las demás clases (entre otros diagramas y aspectos) está contenida en el anexo en medio digital.

Las clases *Solicitud, FormatoSolicitud, Formato, AtributoCambio* y *AtributoFormato* comprenden las solicitudes formuladas, los formatos asociados, los atributos que estos afectan y los cambios que han sido ejecutados por las solicitudes a través de los formatos.

Clase Formato

La clase Formato representa los formatos existentes en el sistema para el control y las solicitudes de cambios. La metodología para la gestión de proyectos no especifica en ningún momento el uso de formatos, pero es una herramienta creada en este desarrollo para agrupar atributos de un proyecto a los que se les puedan efectuar cambios.

Miembros		
Nombre	**Tipo de datos**	**Descripción**
codigoFormato	Entero	Identificador del formato.
nombreFormato	Cadena (String)	Nombre natural del formato.
descripcionFormato	Cadena (String)	Miembro que contiene un texto informativo sobre el formato.
fechaInicioVigencia	Fecha (Date)	Esta fecha marca el inicio del periodo a partir del cual el formato puede sufrir modificaciones. Se utiliza junto con el atributo siguiente por motivos de seguridad, para impedir los cambios no deseados por

		cualquier usuario.
fechaFinVigencia	Fecha (Date)	Por la misma razón de ser del atributo fechaFinVigencia, 'fechaInicioVigencia' marca el fin del periodo durante el cual son admitidas las modificaciones al formato.
List<AtributoFormato> atributoFormatos	Colección de AtributoFormatos	Lista de tipo AtributoFormato.
List<FormatoSolicitud> formatoSolicitudes	Colección de FormatoSolicitudes	Lista de tipo FormatoSolicitud.

Métodos		
Nombre	**Devuelve**	**Finalidad**
Vigente	Booleano (Verdadero o Falso)	Devuelve 'true' si el formato está vigente. 'false' en el caso contrario. Este método permite obtener el valor de vigencia del formato sin necesidad de realizar estos cálculos en un EJB o en la página.

Clase AtributoFormato

Define la relación entre los formatos y sus atributos asociados. Contiene la información de vigencia de la asociación Atributo-Formato, indicando si es posible que el formato asociado afecte el atributo asociado.

Miembros		
Nombre	**Tipo de datos**	**Descripción**
Formato	Formato	Formato al que corresponde la asociación.
Atributo	Atributo	Atributo al que corresponde la asociación.

codigoAtributoFormato	Entero	Identificación de la asociación.
atributoCambio	Booleano	Indica si el atributo dentro del formato se puede sujetar a cambios.

Clase Solicitud

Esta clase representa las solicitudes de cambio propuestas.

Miembros		
Nombre	**Tipo de datos**	**Descripción**
proyecto	Proyecto	Proyecto asociado a la solicitud.
codigoSolicitud	Entero	Identificador de la solicitud.
tipoSolicitud	TipoSolicitud	Tipo de solicitud asociado a la solicitud.
codigoUnidadAcademica	Entero	Identificador de la unidad académica que está desarrollando la actividad en la cual se encuentra la solicitud.
fechaSolicitud	Fecha (Date)	Fecha de origen de la solicitud.
justificacionSolicitud	Cadena (String)	Descripción y justificación de los cambios propuestos por la solicitud.
List<ActividadSolicitud> actividadSolicitudes	Colección de ActividadSolicitudes	Lista de tipo ActividadSolicitud.
List<FormatoSolicitud> formatoSolicitudes	Colección de FormatoSolicitudes	Lista de tipo FormatoSolicitud.
Auditoría		

Clase FormatoSolicitud

Esta clase representa una asociación entre un formato y una solicitud. A cada solicitud se le asocian una serie de formatos que comprenden los atributos que la solicitud pretende modificar.

Miembros		
Nombre	**Tipo de datos**	**Descripción**
codigoFormatoSolicitud	Entero	Identificador de la asociación.
aprobacionCambio	Booleano	Indica si la propuesta de cambios por la solicitud sobre este formato es aprobada o no.
formato	Formato	Formato asociado.
solicitud	Solicitud	Solicitud asociada.
fechaInicioVigencia	Fecha (Date)	Fecha a partir de la cual se pueden hacer los cambios, de haber sido aprobados.
fechaFinVigencia	Fecha (Date)	Fecha hasta la cual se pueden hacer los cambios, de haber sido aprobados.
observacionCambio	String	Observaciones del cambio solicitado por el director del proyecto. Son realizadas por el comité de cambios cuando aprueba o rechaza los cambios propuestos.
observacionSolicitante	String	Justificación del solicitante sobre los cambios que pretende realizar con la solicitud.

Clase AtributoCambio

Esta clase representa las modificaciones hechas a un atributo mediante un formato en determinada solicitud.

Miembros		
Nombre	**Tipo de datos**	**Descripción**
aprobacionCambio	Booleano	Indica si el cambio propuesto es aprobado.
atributoFormato	AtributoFormato	Asociación atributo-formato que permite el cambio.
formatoSolicitud	FormatoSolicitud	Asociación solicitud-formato mediante la cual se solicita el cambio.
Valor	String	Valor propuesto por el cambio.
Auditoría		

3.3.1.2 Diagrama de Casos de Uso

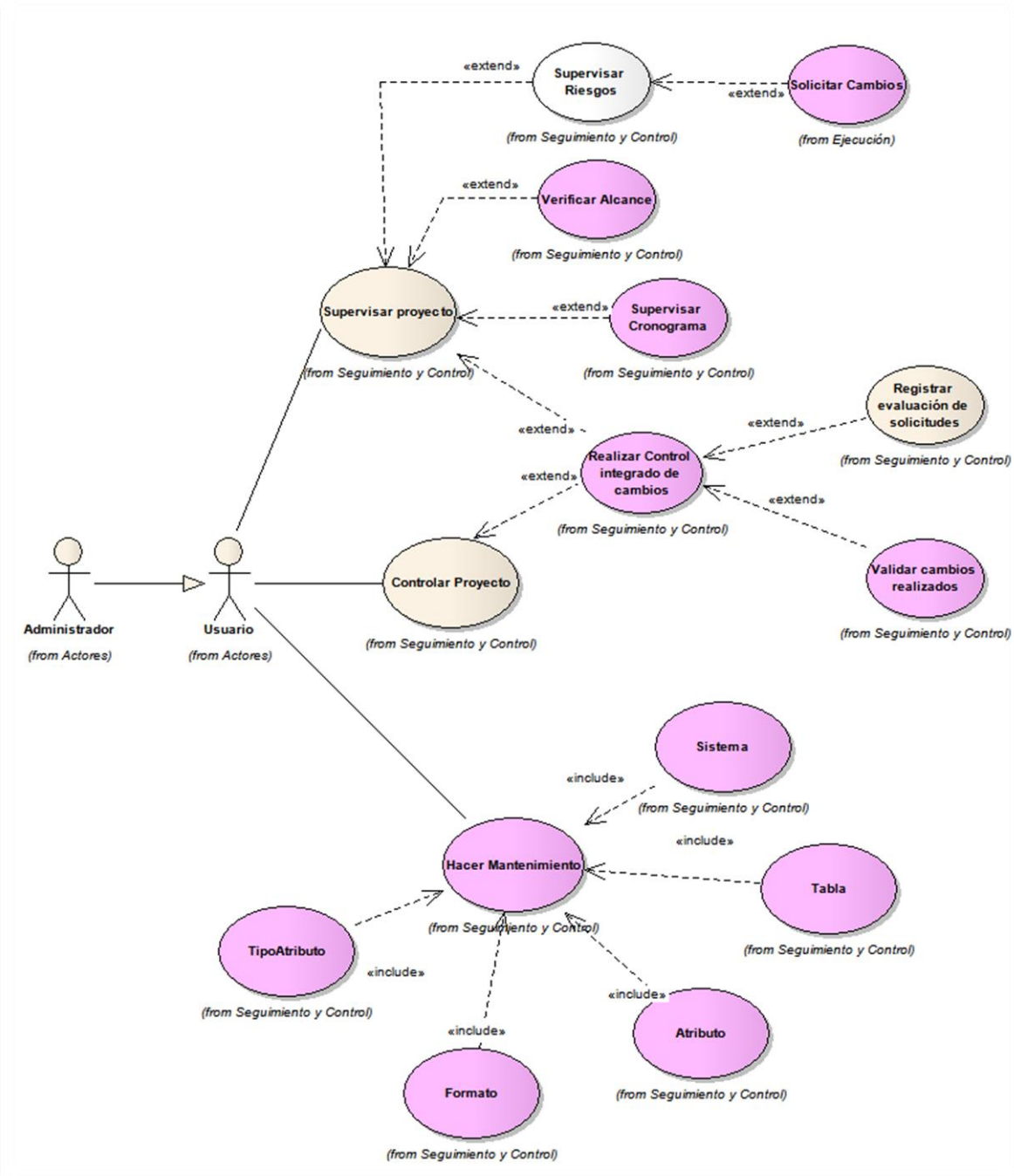


Figura19: Caso de uso

De la misma manera que con el diagrama de clases, se han escogido 3 casos de uso representativos para ser explicados acá. Los demás casos de uso pueden ser consultados en el anexo digital incluido con el proyecto. Los casos de uso escogidos son *Solicitar cambios, Realizar Control Integrado de Cambios* y *Validar cambios realizados.*

Solicitar cambios

El proceso descrito por este caso de uso es el que se realiza cuando un solicitante determina que ciertos atributos del proyecto deben actualizarse. Entonces decide formular una solicitud de cambio especificando qué cambios desea hacer y cuál es el motivo de los mismos.

Solicitar Cambios	
Actor	**Usuario**
Propósito	Los cambios solicitados son para ampliar o reducir el alcance del proyecto, para modificar políticas o procedimientos, para modificar el coste o el presupuesto del proyecto, o para revisar el cronograma del proyecto; son identificados mientras se ejecuta el proyecto.
Resumen	El usuario ingresa por el menú correspondiente e ingresa los datos, y luego tiene la opción de salir o solicitar más cambios.
Flujo Principal	El usuario selecciona "Realizar control de cambios " y realiza las siguientes acciones en este orden: - Escoger Proyecto. - Creación de una Solicitud de cambio. - Asociar Formato(s). El sistema regula y controla las vigencias de cambio y los permisos. La solicitud y los formatos asociados deben ser debidamente justificados. El usuario sale del sistema o procede a crear otra solicitud.
Precondición	El proyecto seleccionado debe estar en el estado de ejecución.

Realizar control integrado de cambios

En este caso de uso es cuando un evaluador se dispone a observar los cambios propuestos por un solicitante. De estos cambios, el evaluador debe emitir un concepto de aprobación. Para más claridad sobre el proceso de control cambios según lo maneja SGPUIS, consultar *2.2.5 Control de cambios.*

Realizar control integrado de cambios	
Actor	**Usuario**
Propósito	• Diligenciar solicitudes de cambio. • Analizar acciones preventivas recomendadas. • Analizar acciones correctivas recomendadas.
Resumen	El usuario ingresa por el menú correspondiente y escoge proyecto, selecciona la solicitud de cambio y verifica si el formato debe o no aprobarse. Luego de aprobado el formato, se revisará el atributo que desea cambiar y se aprobará o rechazará el cambio.
Flujo Principal	El usuario ingresa por el menú correspondiente a realizar el control de cambios y: • Selecciona el proyecto • Escoge una solicitud • Evalúa los cambios propuestos por el solicitante • Emite un concepto de aprobación por estos cambios El sistema valida la información ingresada.

	El usuario sale del sistema o procede a realizar la evaluación de otra solicitud de cambio.
Precondición	Usuario debidamente autenticado y con permisos para evaluar solicitudes

Validar cambios realizados

Por validar cambios debe entenderse que el evaluador verifique los cambios ejecutados por el solicitante, para asegurar que estos son fieles a la justificación en base a la cual se dio permiso para hacerlos.

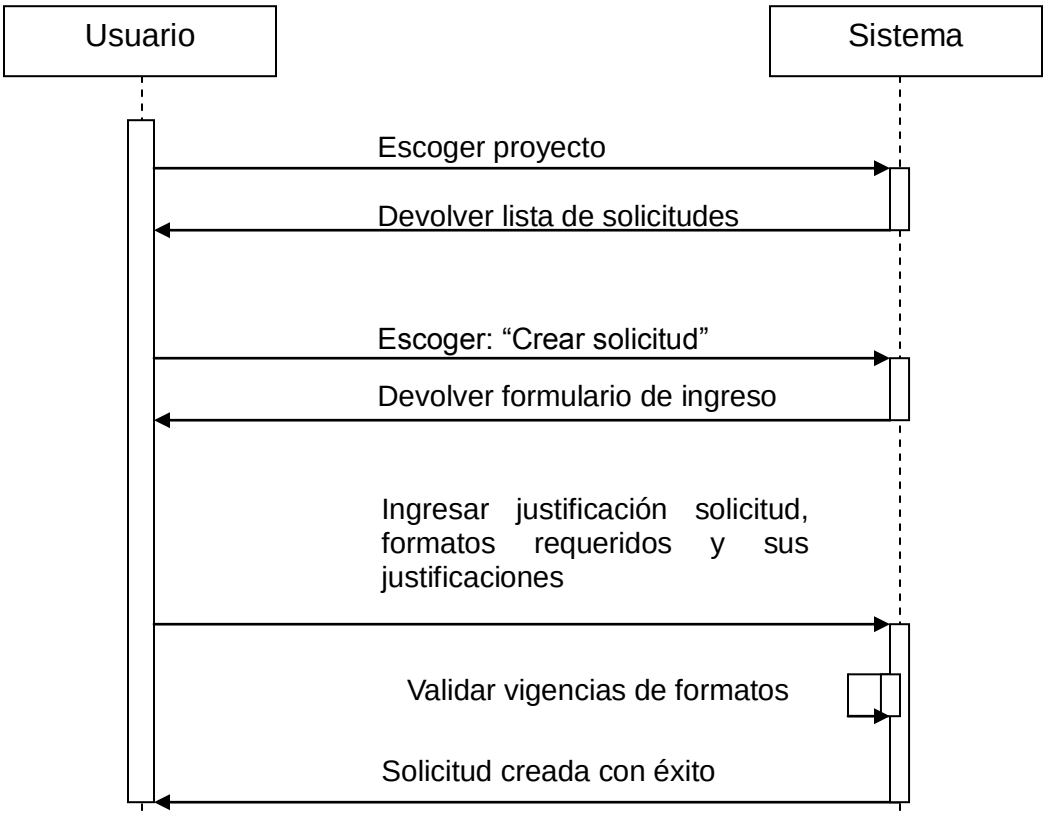
Validar cambios realizados	
Actor	**Usuario**
Propósito	Verificar que los cambios realizados hayan sido efectuados de acuerdo a los requisitos especificados, o si necesitan ser reevaluados.
Resumen	El usuario ingresa por el menú correspondiente e ingresa los datos, y luego tiene la opción de salir o validar otros cambios.
Flujo Principal	El usuario ingresa por el menú correspondiente. Selecciona el proyecto y la solicitud que quiere evaluar. Observa los cambios ejecutados por el solicitante. Aprueba o rechaza los cambios según los criterios de evaluación pertinentes. El usuario confirma los cambios aprobados y rechazados, si

	efectuó alguna de estas acciones. El usuario sale del sistema o procede a realizar la evaluación de otra solicitud de cambio.
Precondición	

3.3.1.3 Diagrama de Secuencias

Solicitar cambios

A continuación el diagrama de secuencias que corresponde al caso de uso *Solicitar cambios.* Es importante recordar que este diagrama se muestra a modo de ejemplo, puesto que se decidió usar los prototipos no funcionales o de funcionalidad básica para describir la serie de pasos para efectuar cada caso de uso, función que también cumplen los diagramas de secuencia.



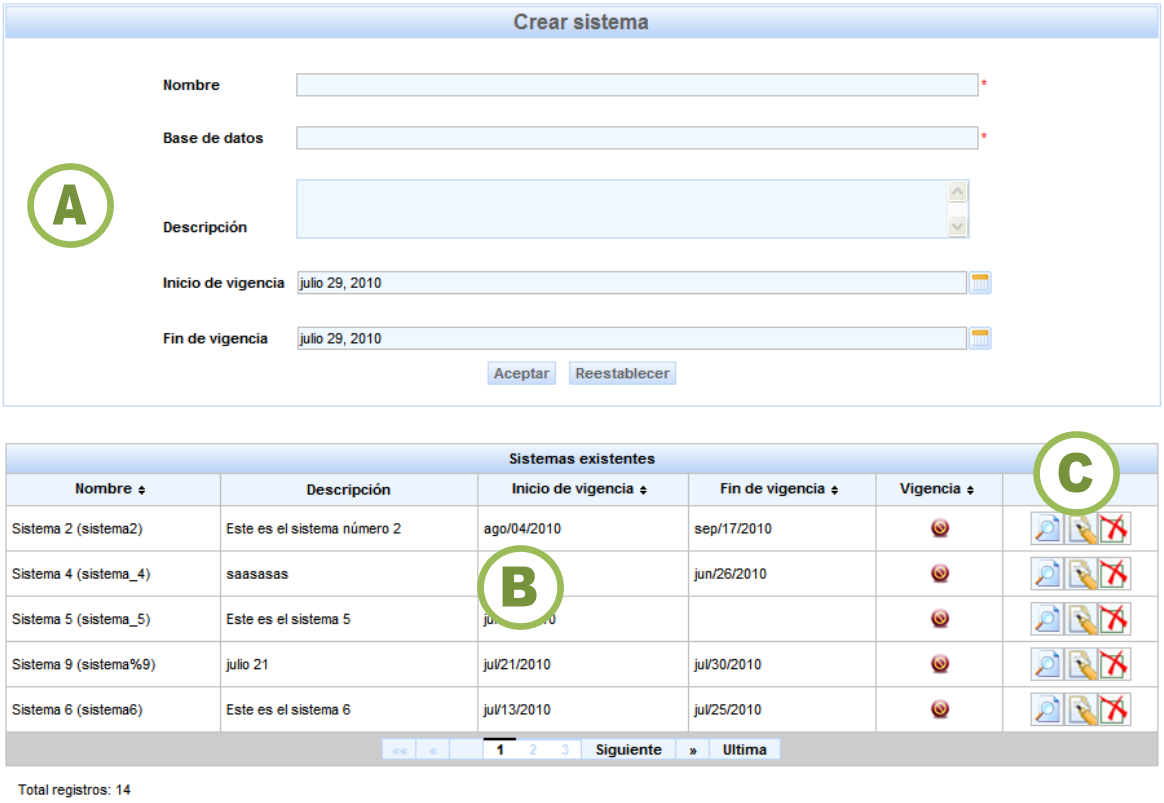
3.4 Prototipos

3.4.1 Primer prototipo

Administración de la estructura de datos: Para el módulo de control integrado de cambios de SGPUIS se desarrolló un prototipo inicial con funcionalidad básica de creación y eliminación de registros. El prototipo consistía de una serie de formularios maestro/detalle que manipulaban las clases del módulo pertenecientes a la jerarquía **Sistema -> Tabla -> Atributo** según se aprecia en el diagrama de clases. También permitía crear tipos de atributo (clase TipoAtributo) para asignar a los atributos (Atributo).

El prototipo además asociaba atributos con formatos (asociación representada por la clase AtributoFormato), lo cual es clave en el módulo de Control Integrado de Cambios de SGPUIS para determinar con claridad qué aspectos de los proyectos se pretende modificar con una solicitud.

Generalidades del los formularios



Las páginas de los formularios detalle/maestro tienen ciertas generalidades:

A: Panel para ingreso de los datos. Varía según la clase.

B: Tabla para mostrar los registros guardados. Se actualiza inmediatamente se agrega un nuevo registro.

C: Columna de acciones de la tabla de registros creados. Contiene 3 botones que permiten ver el detalle de cada registro, editarlo o eliminarlo:

Acción	Detalles
 Ver detalle	Muestra los datos del registro y los registros asociados. Por ejemplo, en el caso de ver el detalle de un sistema, muestra una tabla con las tablas pertenecientes al sistema.
 Editar registro	Permite modificar todos los datos del registro.
 Eliminar registro	Elimina el registro (mostrando una pantalla de confirmación)

Se resumen las funcionalidades de este primer prototipo en:

- Crear registros (sistemas, tablas, etc)
- Editar registros
- Eliminar registros
- Capturar excepciones de errores transaccionales (manipulación de la base de datos)
- Listar los registros creados
- Encontrar registros creados mediante una tabla paginada y organización alfabética de ciertos criterios relevantes
- Asociar atributos y formatos

El prototipo no se encargaba de otra validación de datos que no produjera errores transaccionales. Es decir, desde que la base de datos aceptara el registro, se podía continuar. No se tenía en cuenta determinar que los datos fueran ilógicos o que no representaran nada, como fechas de inicio y fin de vigencia que no describieran un periodo válido de tiempo.

Otras carencias de este prototipo podrán conocerse más a fondo conociendo las características introducidas en los siguientes prototipos.

Gestión de solicitudes: Para la gestión de solicitudes se creó inicialmente una herramienta maestra que asumiera roles administrador/solicitante, para agilizar la incorporación de funcionalidades al desarrollo. Desde el inicio, esta herramienta controlaba todo el flujo de las solicitudes de cambio, de manera que podía:

- Crear solicitudes nuevas
- Asociar formatos a las solicitudes
- Aprobar/rechazar estos formatos y en caso de aprobación, especificar los datos de la aprobación.
- Ejecutar cambios sobre los formatos aprobados.
- Aprobar/rechazar los cambios ya ejecutados.

3.4.2 Segundo prototipo

Administración de la estructura de datos: El prototipo siguiente del módulo de control integrado de cambios de SGPUIS sufrió un vasto cambio en la capa de la lógica del negocio, mientras que en la capa de presentación permaneció igual,

En la capa de lógica del negocio, fue crítica la inclusión de la validación. Esta validación comprendía:

- Verificar la repetición de nombres
- Verificar los periodos descritos por *fechaInicioVigencia* y *fechaFinVigencia:* Era necesario que la primera fecha fuera anterior a la segunda, de manera que describieran un periodo de tiempo válido.
- Limitar las descripciones de los sistemas, tablas y atributos a textos cortos de 250 caracteres máximo e incluyendo un contador en las páginas.
- Impedir tablas o atributos “sueltos”; es decir, que no pertenecieran a un sistema o tabla.

- Insertar en la capa de presentación mensajes descriptivos acerca del error que no permitió crear, modificar o eliminar el registro
- Impedir el intento de eliminación de registros que tuvieran asociaciones. Ejemplo: Impedir la eliminación de una tabla que tuviera atributos asociados. Hacer esto no es permitido por las bases de datos en pro de mantener la integridad referencial. Si se configurara la base de datos para, en el caso de ejemplo, eliminar todos los atributos asociados, se le daría al administrador del sistema la capacidad de ocasionar una gran pérdida de información. Los atributos también pueden estar asociados a formatos, así que permitir la eliminación en cascada podría traer graves consecuencias al funcionamiento del sistema.

Además se añadían funcionalidades para facilitar la creación y edición de registros como:

- Asociación de una tabla con sistema, al momento de editar un sistema.
- Asociación de un atributo con una tabla, al momento de editar una tabla

Gestión de solicitudes: Para este nuevo prototipo, se simplificaron algunas funciones y se cambió el esquema de navegación.

De aquí en adelante, el usuario se encuentra al acceder a la herramienta con una lista de proyectos existentes. Al seleccionar un proyecto el usuario entra a una vista con todas las solicitudes del proyecto seleccionado, y ahí es donde se le permite crear una solicitud nueva y asociarle formatos inmediatamente.

A partir de esta vista el usuario accede a todas las demás funciones del prototipo anterior como:

- Aprobar/rechazar formatos

- Ejecutar cambios
- Aprobar/rechazar cambios ejecutados

Las vistas de estas funciones fueron modificadas para ajustarse a los estándares de la División de Servicios de Información.

3.4.3 Tercer Prototipo

Tras hacer numerosas pruebas con los prototipos desarrollados, se llegó a la conclusión de que la herramienta administrativa para el módulo de control de cambios de SGPUIS presentaba ciertas falencias referentes a la usabilidad. Dada la necesidad de una herramienta para el manejo de registros que estuviera funcionando rápidamente y permitiera avanzar en el desarrollo de las funcionalidades de creación y gestión de solicitudes de cambio, resultó una herramienta que:

- Comprendía muchas vistas.
- No mostraba correctamente la naturaleza jerárquica de la estructura de datos que se estaba manejando.
- No permitía al usuario ser eficiente con el tiempo para ciertas tareas.

Considerando lo anterior, y tomando como referente los entornos informáticos (software) a los que comúnmente están expuestos los usuarios se decidió implementar una sola vista que manejara toda la estructura de datos del módulo de control de cambios del sistema. Esto repercutiría en:

- Facilidad de uso de la herramienta administrativa.

- Rapidez para realizar tareas sencillas como autorizar o prohibir cambios para distintos aspectos de los proyectos.
- Mayor comprensión por parte del administrador de la dependencia de ciertos aspectos de los proyectos de otros, lo que en la estructura de datos está dado por la jerarquía Sistema – Tabla – Atributo.

Similar a un explorador de archivos, la vista ofrece una columna izquierda que muestra todos los registros existentes y los agrupa en una estructura de árbol. Los detalles de estos se pueden ver en la sección derecha de la pantalla. Esta sección también muestra los respectivos formularios para el ingreso de datos y creación de nuevos registros. Todas las operaciones de validación de esta nueva vista son heredadas de las versiones anteriores de los prototipos, y toda la experiencia de estos desarrollos anteriores hizo que ensamblar esta nueva herramienta fuera relativamente rápido.

El acceso a funcionalidades como ver detalles, editar registros, o asociar y crear registros nuevos se hace mediante menús contextuales sobre el árbol de la sección izquierda.

En todo momento se pueden visualizar qué registros dependen de otros. Por ejemplo, al expandir una tabla, se verán todos los atributos asociados a esta. Permitir o prohibir un cambio para un atributo está a un clic de distancia y el cambio es inmediato ante los ojos del usuario.

Todas estas mejoras de usabilidad son posibles gracias a las capacidades AJAX integradas a la librería RichFaces, que es el estándar en la División de Servicios de Información para los nuevos sistemas de información en la universidad. Esto hace que la ejecución de las operaciones solicitadas por el usuario sea rápida y no exija que se recargue la página innecesariamente, cuando realmente solo se necesita actualizar porciones de la misma.

Gestión de solicitudes: Para esta etapa de madurez del proyecto se ramificó el desarrollo de la herramienta para la gestión de solicitudes de cambio. Esto, además de ser la concepción final de la herramienta, sería ideal para probar más estrictamente el comportamiento del software en varios escenarios según el usuario involucrado.

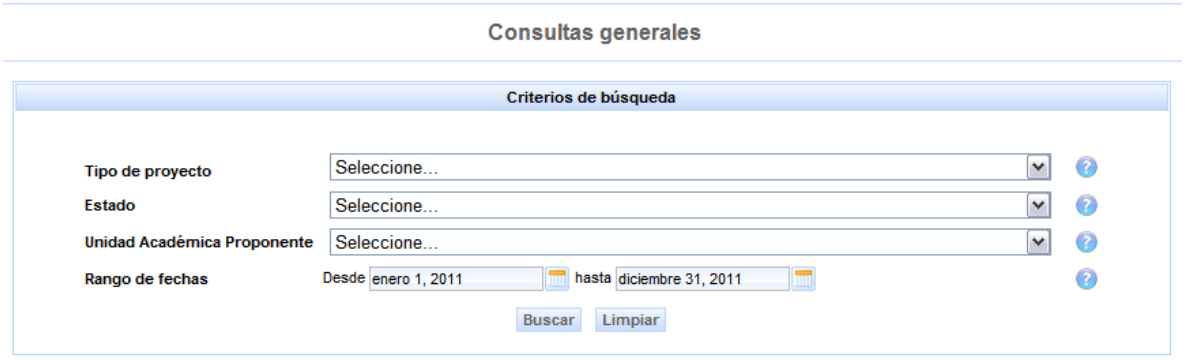
Entonces el flujo de navegación se iniciaría con la misma lista inicial de proyectos, pero según el rol del usuario se le redireccionará a una página que solo le dejaría ejecutar las funciones para las que estuviera autorizado.

Así las cosas, la vista de usuario al ver las solicitudes de un proyecto comprende la función *Ejecutar cambios*, mientras que la del administrador es más compleja y permite *aprobar/rechazar formatos* y *aprobar/rechazar cambios*.

Módulo de consultas: En el tercer prototipo se introdujo el módulo de consultas. Su propósito es servir para la búsqueda de proyectos mediante diferentes criterios.

Se desarrolló una herramienta de búsqueda que filtra los proyectos de acuerdo a los siguientes criterios:

- Tipo de proyecto
- Estado de proyecto
- Unidad académica proponente
- Fecha de formulación



La búsqueda muestra la tabla con la lista de proyectos que cumplen todos los criterios seleccionados. Esta lista puede ser organizada ascendente o descendentemente de acuerdo a los criterios en las columnas de la tabla, para facilitar aún más la búsqueda.

Al seleccionar un proyecto es posible navegar por diversos aspectos de los proyectos como los objetivos, las fases, actividades y entregables.

3.4.4 Prototipo Final

Un producto de software de calidad tiene que responder a muchas expectativas en diferentes aspectos. Sin embargo, estos aspectos no son valorados todos en la misma medida. Esta diferencia radica en el uso del mismo. De la misma forma, es

necesario que los productos resultantes de la labor de desarrollo sean fáciles de mantener.

En el prototipo anterior se observaron ciertos detalles que podrían dificultar la labor de mantenimiento a largo plazo del software. El más importante es la extensión de los archivos de código fuente. Para el prototipo anterior de la vista de administración, este archivo era considerablemente largo, lo cual afectaba su legibilidad y la rapidez de la página resultante en el cliente. Hay que considerar que el uso intensivo de AJAX mediante RichFaces genera una buena cantidad de código JavaScript, el cual se ejecuta en el cliente.

Así pues, era necesario mantener la simplicidad y usabilidad del prototipo anterior sin sacrificar el rendimiento. Por esto, el prototipo final para la vista administrativa del módulo de control de cambios de SGPUIS quedó dividido en 3 partes:

1. Una vista para la gestión de la estructura Sistema – Tabla – Atributo
2. Una vista para gestionar los formatos, los atributos asociados a ellos y los permisos de cambios sobre estos atributos.
3. Otra vista para la tabla de soporte TipoAtributo.

El resultado es una herramienta altamente usable, que permite la manipulación de muchas tablas en la base de datos mediante una interfaz sencilla, con una capa de software controladora compleja; pero todo esto sin ser una carga demasiado pesada para el servidor y las máquinas cliente.

Gestión de solicitudes: Para el prototipo final del módulo se decidió:

- Renovar la vista incluyendo ayudas en campos críticos.

- Eliminar información redundante o demasiado grande en las tablas. Si la información se quitaba por ser muy grande se daba la opción de consultarla mediante un diálogo o en otra página.

Para esta etapa del desarrollo era mucho más clara la visión del usuario al sistema, por lo que el nuevo enfoque del equipo de trabajo en usabilidad motivaba a realizar cambios que impactaran sobre la experiencia de usuario, como los descritos anteriormente.

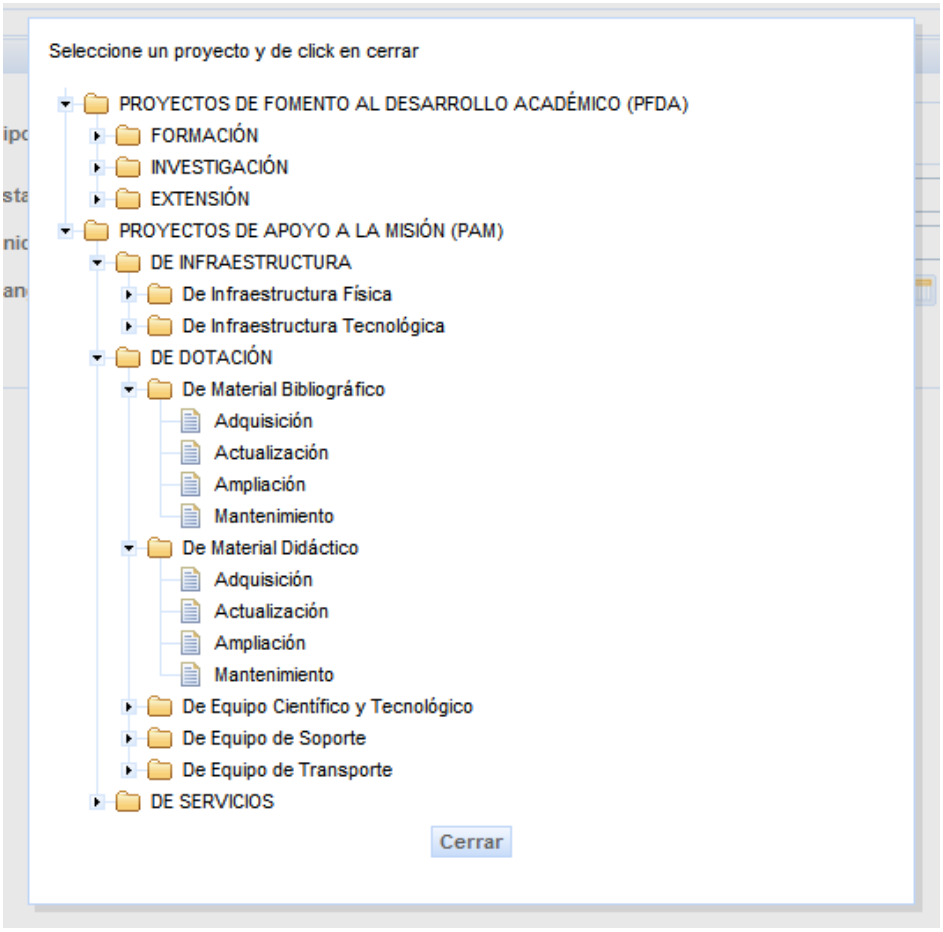
Un detalle importante fue la inclusión de una página del detalle de la solicitud. Esta vista comprendía información referente a los formatos asociados a la solicitud, sus detalles de aprobación si fueron aprobados, los cambios ejecutados y cuáles habían sido aprobados y rechazados.

También se modificó la herramienta para crear solicitudes nuevas. Para el prototipo final esta herramienta envía notificaciones por correo electrónico al administrador del sistema acerca de la creación de nuevas solicitudes, y cuando se efectúen o aprueben/rechacen cambios sobre ellas.

Módulo de consultas: Para el prototipo final el módulo de consultas introduce algunas mejoras enfocadas a la eficiencia y rapidez de las búsquedas.

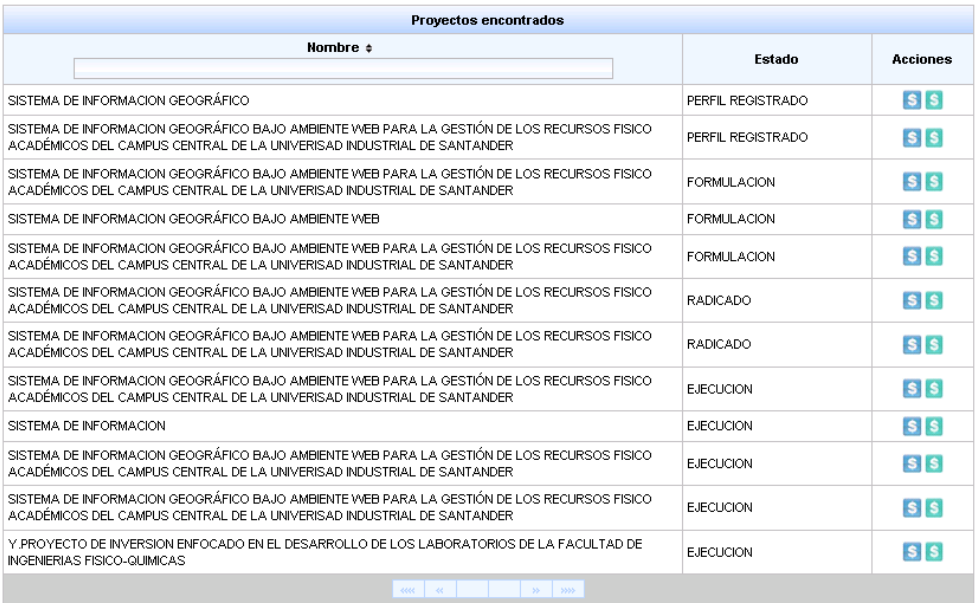
El criterio de búsqueda por fecha se hace obligatorio para no dar la posibilidad a que se haga una búsqueda sin parámetros, lo que traería todos los proyectos formulados.

Dado que los tipos de proyectos son muchos, también se elaboró una ventana que permite navegar entre los tipos de proyectos mediante una estructura de árbol para encontrar el criterio de búsqueda seleccionado más rápidamente:



A la tabla de proyectos resultantes, se le agregó un filtro por el nombre del proyecto.

Se agregó la posibilidad de consultar la ejecución presupuestal del proyecto como se puede ver en la siguiente imagen:



Proyectos encontrados

Nombre	Estado	Acciones
SISTEMA DE INFORMACION GEOGRÁFICO	PERFIL REGISTRADO	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	PERFIL REGISTRADO	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	FORMULACION	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB	FORMULACION	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	FORMULACION	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	RADICADO	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	RADICADO	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	EJECUCION	
SISTEMA DE INFORMACION	EJECUCION	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	EJECUCION	
SISTEMA DE INFORMACION GEOGRÁFICO BAJO AMBIENTE WEB PARA LA GESTIÓN DE LOS RECURSOS FISICO ACADÉMICOS DEL CAMPUS CENTRAL DE LA UNIVERISAD INDUSTRIAL DE SANTANDER	EJECUCION	
Y.PROYECTO DE INVERSION ENFOCADO EN EL DESARROLLO DE LOS LABORATORIOS DE LA FACULTAD DE INGENIERIAS FISICO-QUIMICAS	EJECUCION	

Se pueden observar dos acciones, ambas muestran la ejecución presupuestal a nivel de elementos. La diferencia radica en que la primera sigue el árbol: Fases-Actividades-Rubros para llegar a los elementos, en cambio la segunda muestra directamente los elementos por cada fase del proyecto.

Ambas acciones conllevan a mostrar la ejecución presupuestal del proyecto desglosando el presupuesto de las fases, actividades o rubros.

3.4.5 Esquema de seguridad

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.

3.4.5.1 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. No confundir con la entidad Sistema de la figura 18 en el numeral 3.3.1.1, correspondiente a la estructura de datos del módulo de control de cambios.

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.

3.4.5.2 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-menu rol, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

3.4.5.3 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 accesar, 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”.

3.4.5.4 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.

3.4.6 Documentación de programas fuente

En la labor desarrollo la documentación juega un papel muy importante para el mantenimiento de software. Esto es aun mas cierto en proyectos grandes, en los que pueden estar involucrados demasiados desarrolladores, todos con culturas de desarrollo diferentes. Aunque todos estos desarrollos cuentan con estándares muy definidos, siempre puede ser necesaria la aclaración sobre ciertos fragmentos o secciones de código.

Históricamente todos los lenguajes de programación han contado con facilidades para la inserción de comentarios dentro del código fuente. Estos comentarios no son compilados puesto que no tienen utilidad funcional pero cuando otro desarrollador ve el código fuente, son necesarios para entender partes que no son claras a simple vista. Java heredó de C la forma de hacer tales comentarios:

Para líneas	`// comentario`

Para bloques de comentarios	`/*`
	`Para bloques enteros de comentarios, con varias líneas`
	`*/`

Sin embargo estas herramientas no son necesarias para documentar proyectos largos, como es el caso de las aplicaciones empresariales. Para esto Java ofrece Javadoc, que es un generador de documentación a partir de código Java. El código que Javadoc utiliza es aquel que encuentre entre:

```
/** documentación */
```

El propósito de este modelo de documentación no es desaprobar las formas tradicionales, que siguen siendo válidas sobre secciones pequeñas de código. Lo que busca Javadoc es generar documentación clara y precisa sobre clases, sus miembros y sus métodos. Así pues, los comentarios especificados mediante el formato anterior van en la definición de clases, sus métodos y miembros.

Ejemplo de documentación de método

```
/**
 * Guarda en el cache de hibernate el concepto de aprobación de un
formato
* Estos cambios no son escritos en base de datos hasta que se ejecute
* {@link #aprobarFormatos()}
*/
public void aprobarFormatoSolicitud() throws DSIException {
	try {
		this.em.merge(this.formatoSolicitud);
	} catch (Exception e) {
		log.error(e.toString());
		throw new DSIException(e);
	}
}
```

Antes de la firma del método va una descripción básica de la tarea que cumple el mismo. Pero en esta documentación se pueden especificar otras cosas. Por ejemplo, mediente `{@link #aprobarFormatos()}` la documentación está haciendo referencia a otro método que está en la misma clase. De la misma manera se pueden hacer referencias a otros miembros, y hasta a las excepciones que maneja el método.

CAPITULO 4

4. CONCLUSIONES

1. Es ideal que durante la etapa de ejecución de un proyecto, todo se dé como fue inicialmente planeado y no haya necesidad de realizar ningún ajuste. Sin embargo, esta no es la situación real puesto que un proyecto, sus participantes y la organización u organizaciones involucrados pueden ser afectados por factores externos no predecibles. Ante esta situación se hace necesario un mecanismo de control que permita a los involucrados reaccionar rápidamente ante estos aspectos y efectuar oportunamente los cambios para tener un mejor control sobre los aspectos del proyecto que tales ajustes impactan. El módulo de control de cambios desarrollado en este proyecto para el Sistema de Gestión de Proyectos de la Universidad Industrial de Santander, SPGUIS, es una solución competente para gestionar los cambios que surgen en la etapa de ejecución de cualquier proyecto.
2. El módulo de control de cambios desarrollado en este proyecto para SGPUIS ofrece a dicho sistema y sus usuarios (administradores) completo control sobre los aspectos de los proyectos sobre los que se pueden efectuar cambios, gracias a una estructura de datos segura y completa con atributos de seguridad en todos los niveles de la jerarquía de esta estructura.
3. Java EE proporciona a los desarrolladores una vasta capa de software sobre la cual trabajar, ofreciendo marcos de trabajo y tecnologías para seguridad, interacción con bases de datos relacionales, creación y manipulación de vistas, entre otros; dejando a los desarrolladores la labor de crear software para la tarea específica que se necesita sin preocuparse por capas de software de soporte a estos procesos que implicarían generar

código repetitivo y de volumen considerable. Si se ahorra tiempo en el desarrollo, se obtiene más rápido la solución deseada y se impacta positivamente sobre la organización y la agilidad de sus procesos.

4. El proceso de desarrollo de software implica muchos pasos más allá de la interacción del desarrollador con la máquina para crear el producto. La etapa preliminar a la codificación es vital en el éxito de la solución informática implementada. El módulo de control de cambios y el módulo de consultas de SGPUIS están basados en un diseño y una metodología claros que facilitan la labor misma de la codificación.
5. La metodología propuesta por el Project Management Institute a través del Project Management Body of Knowledge es una metodología detallada y específica sobre la gestión de proyectos. Aunque no es un estándar, es ampliamente reconocida como parte de las buenas prácticas en la gestión de proyectos y su aplicación en esta tarea repercute positivamente sobre los productos y/o servicios resultantes de los proyectos ejecutados.
6. La Universidad Industrial de Santander, gracias a la División de Servicios de Información, tiene siempre a su alcance la posibilidad de contar con las soluciones informáticas que necesita. Al ser la DSI parte de la universidad, esta puede contar con soluciones hechas a la medida que se adapten totalmente a sus necesidades.

CAPITULO 5

5. RECOMENDACIONES

1. La estructura de datos sobre la que está soportada el desarrollo de este proyecto es una estructura compleja que ofrece un control absoluto sobre el Control de Cambios en la gestión de proyectos en la universidad. Dicha estructura, por sus numerosos detalles de seguridad, puede resultar complicada para algunos usuarios. La consecuencia de esta percepción es que las ventajas ofrecidas no sean completamente aprovechadas. Es necesario que la División de Servicios de Información ofrezca soporte a los administradores de SGPUIS para que se familiaricen con esta estructura de datos y las ventajas y desafíos que ofrece.

2. La cultura informática juega un papel importante en la utilidad de los productos de software y en muchos casos, las consecuencias de estas son difíciles de prever y van más allá de los límites del desarrollador. Es importante que la Oficina de Planeación oriente debidamente, y a la vez sea estricta con los lineamientos que se deben seguir a la hora de formular solicitudes de cambio. Estas solicitudes deben ir correctamente detalladas en sus justificaciones y deben hacer uso correcto de las herramientas que ofrece el producto para aplicar formato al texto. En la medida en que estas buenas costumbres sean aplicadas, el proceso de revisión de las solicitudes se hace más claro, y en general la gestión de cambios progresa más rápido.

3. SGPUIS como conjunto (no solo los módulos desarrollados en este proyecto) es un proyecto que maneja información crítica referente a presupuestos y otros recursos de la universidad. Como es un producto de

software que funciona en ambiente web, el desempeño y la estabilidad de la máquina cliente juegan un papel muy importante en que la solución sea altamente usable. Aunque las tecnologías usadas para el desarrollo de este proyecto garantizan la compatibilidad del producto con los navegadores más populares del mercado y esto se comprobó durante la fase de desarrollo, es conveniente que la universidad en general impulse el uso de navegadores diferentes a Internet Explorer para aprovechar las bondades que estos pueden ofrecer en los aplicativos orientados a la web.

CAPITULO 6

6. BIBLIOGRAFIA

BERZAL, Fernando. *Relaciones entre clases: Diagramas de clase UML.* {En línea}. Disponible en: http://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf

Booch, Grady. Rumbaugh, James. Jacobson, Ivar. *El Lenguaje Unificado de Modelado.* Addison Wesley. 1999.

CADENA RUIZ, Ana María. *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

De Amescua Seco, Antonio. Cuadrado Gallego, Juan José. Ernica Lafuente, Emilio *"Análisis Y Diseño Estructurado Y Orientado A Objetos De Sistemas Informáticos"*. Madrid: Editorial Mc Graw Hill, 2003.

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.

JENDROCK, Eric. BALL, Jennifer. CARSON, Debbie. EVANS, Ian. FORDIN, Scott. HAASE, Kim. *The Java EE 5 tutorial*. 2007. {En línea}. Disponible en: http://download.oracle.com/javaee/5/tutorial/doc/javaeetutorial5.pdf

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

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.

VILLALOBOS, Jorge Alberto. CASALLAS, Rubby. *"Fundamentos de programación – Aprendizaje Activo Basado en Casos: Un Enfoque Moderno Usando Java, Uml, Objetos Y Eclipse"*. México: Pearson Educación, 2006.