SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LAS ACTIVIDADES DEL PROGRAMA DE ASESORÍA PARA EL MEJORAMIENTO DEL RENDIMIENTO ACADÉMICO (PAMRA) DE BIENESTAR UNIVERSITARIO

ANDREA KATHERINE MARTÍNEZ ROSAS
JEFFERSON CÁCERES MARTÍNEZ
PAULO CESAR RAMÍREZ PRADA

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

SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LAS ACTIVIDADES DEL PROGRAMA DE ASESORÍA PARA EL MEJORAMIENTO DEL RENDIMIENTO ACADÉMICO (PAMRA) DE BIENESTAR UNIVERSITARIO

ANDREA KATHERINE MARTÍNEZ ROSAS
JEFFERSON CÁCERES MARTÍNEZ
PAULO CESAR RAMÍREZ PRADA

Trabajo de grado para optar el título de Ingeniero de Sistemas

Director
ING. ENRIQUE TORRES LÓPEZ
PROFESIONAL DIVISIÓN DE SERVICIOS DE INFORMACIÓN
UNIVERSIDAD INDUSTRIAL DE SANTANDER

Codirector
T.S. DEISY ROCIO LIZARAZO VELASCO
COORDINADORA PAMRA
UNIVERSIDAD INDUSTRIAL DE SANTANDER

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

AGRADECIMIENTOS

Los autores de este proyecto expresan su agradecimiento:

A la División de Servicios de Información de la Universidad Industrial de Santander por darnos la oportunidad de participar en un proyecto de desarrollo de software institucional fortaleciendo con esto nuestros conocimientos y capacidades así como la experiencia de trabajar en un ambiente real de desarrollo.

Al ingeniero Enrique Torres López, director de este proyecto, por su confianza, su invaluable colaboración y guía a lo largo del desarrollo. Al ingeniero Jacksson Sonny Gonzales Bayona por su colaboración y apoyo a los sistemas de Bienestar Universitario. A los ingenieros Danny Felipe Vergel y Robinson Delgado encargados de los servicios de información académicos. Al ingeniero Elkin Suarez por su guía e instrucción en las tecnologías Java; así como a los demás profesionales de la división por su labor y trabajo constante para darle a la Universidad software de calidad.

A la División de Bienestar Universitario, representados por el Doctor Isnardo Ardila, la Doctora María Claudia Caballero, Doctora Elida Jácome, quienes se involucraron para hacer posible este proyecto y lo adoptaron como parte del mejoramiento continuo en los servicios que Bienestar Universitario presta a la comunidad

Al Programa de Asesoría para el Mejoramiento del Rendimiento Académico, a sus coordinadoras las trabajadoras sociales Gilma Puentes de Contreras y Deisy Lizarazo Velasco por su apoyo y respaldo para que el sistema fuera una solución a las necesidades del Programa.

A la Escuela de Ingeniería de Sistemas e Informática, por la formación recibida durante toda nuestra carrera, a los profesores Fabio Reyes, Javier Hernández, Sergio Castillo por compartir sus conocimiento y experiencias e incentivar en nosotros la profundización en lo referente al desarrollo de software.

DEDICATORIA

A Dios por permitirme llegar hasta este punto de mi vida.

A mi madre Luz Marina Rosas, por ser mi guía, mi apoyo, mi ejemplo a seguir, por brindarme su cariño y su comprensión para hacer de mí una mejor persona.

A mi padre Rubén Darío Martínez, por comprenderme y apoyarme en todo momento.

A mi hermana Leidy Viviana Martínez, por su confianza, sus concejos, sus buenos deseos y por ser no solo una hermana sino una amiga.

A mi novio Jefferson Cáceres, por todos los momentos felices que hemos compartido durante toda la carrera, por su especial amor y cariño en todo momento.

A mis compañeros de proyecto Jefferson Cáceres y Paulo Ramírez por que sin su ayuda la culminación de este proyecto no hubiera sido posible.

A mis jefes y amigas Gilma Puentes de Contreras y Deisy Lizarazo, por su constante apoyo y por depositar su confianza en mí.

A todos mis amigos y compañeros por estar siempre disponibles cuando los necesite, y por acompañarme en mi recorrido por la universidad.

ANDREA KATHERINE MARTÍNEZ ROSAS

DEDICATORIA

A mi familia; mis padres, Álvaro Cáceres y María Isabel Martínez, que siempre han estado apoyándome, motivándome para que alcance todas mis metas, y me han dado el empujoncito para salir de mi letargo cuando lo he necesitado. Mi nona, Alicia Carreño, que con su cariño y paciencia me ha apoyado en todas las fases de mi vida. A mi hermano, Jhonattan Cáceres, que siempre me ha brindado su apoyo incondicional, no solo como familia, si no como el mejor de los amigos.

A mis compañeros de proyecto; Andrea Martínez y Paulo Ramírez, sin los cuales no habría sido posible esta meta. Con los que aprendí, me divertí, alegué, razoné, y se convirtieron en los mejores compañeros de trabajo que pude haber pedido.

A mis jefes; Doña Gilma Puentes de Contreras y Deisy Lizarazo, que me brindaron la oportunidad de hacer parte de un familia como lo es PAMRA, que han enriquecido mi formación, y que pueden contar conmigo así como yo siempre cuento con ellas.

A mi novia, Andrea Martínez, porque con su cariño, apoyo, humor, comprensión, inocencia, calidez, entre otras, ha estado junto a mí en etapas decisivas de mi vida y siempre he podido contar con ella.

A mis amigos/parceros, que siempre me han acompañado en los buenos y en los malos momentos, que han sido los compañeros de carrera y que serán los profesionales que estimaré el resto de la vida.

JEFFERSON CÁCERES MARTÍNEZ

DEDICATORIA

A Dios por darnos la oportunidad de estar vivos, y la capacidad para hacer de este mundo un lugar mejor.

A mis padres Néstor Ramírez y Martha Prada, por inculcarme el deseo de aprender y mejorar cada día, por su gran apoyo en momentos difíciles y por enseñarme que sólo dando lo mejor de nosotros mismos podremos obtener grandes resultados.

A toda mi familia, especialmente a mi hermana Natalia Lizeth, por su cariño, comprensión y apoyo, pues siempre serán parte importante de mi vida y mi desarrollo como persona.

A mis compañeros de proyecto Jefferson Cáceres y Andrea Martínez, por su dedicación y esfuerzo, pues sin ellos esta obra no estaría completa.

A Natalia Bravo, por su constante apoyo incondicional a lo largo del desarrollo de este proyecto y especialmente por compartir conmigo su tiempo y su cariño.

A mis amigos, que a lo largo de mi vida siempre han estado presentes para ayudarme a cumplir mis metas y celebrar mis triunfos.

PAULO CESAR RAMÍREZ PRADA

TABLA DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE FIGURAS

RESUMEN

TITULO: SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LAS ACTIVIDADES DEL PROGRAMA DE ASESORÍA PARA EL MEJORAMIENTO DEL RENDIMIENTO ACADÉMICO (PAMRA) DE BIENESTAR UNIVERSITARIO. *

AUTOR: CÁCERES MARTÍNEZ Jefferson, MARTÍNEZ ROSAS Andrea Katherine, RAMÍREZ PRADA Paulo César**

PALABRAS CLAVE: Sistema de Información Web, Tutorías, Beneficiario, Tutor, PAMRA, Mejoramiento Académico, Aplicaciones Cliente Servidor, Java, Bienestar Universitario.

DESCRIPCIÓN:

El Programa de Asesoría para el Mejoramiento del Rendimiento Académico (PAMRA), adscrito a la División de Bienestar Universitario, tiene como principal objetivo desarrollar estrategias metodológicas y educativas que apoyen el proceso de formación profesional de los estudiantes de pregrado Universidad Industrial de Santander, contribuyendo a la disminución de dificultades relacionadas con el bajo rendimiento académico mediante el trabajo entre pares (la tutoría); que es la herramienta metodológica escogida y aplicada, contando con los estudiantes como el recurso humano.

La herramienta virtual desarrollada con el fin de suplir satisfactoriamente las necesidades del programa, está enfocada en brindar opciones y funcionalidades que permitan el correcto seguimiento de las tutorías y demás eventos realizados por el programa, facilitar los procesos que conciernen a la ampliación de la cobertura del programa, la programación, soporte y seguimiento de actividades realizadas; además de la generación de informes y estadísticas basadas en el correcto manejo de la información. Para ello fue necesario que el sistema se encontrara integrado con los demás sistemas de información con los que cuenta la universidad.

Este proyecto se realizó siguiendo los estándares que la División de Servicios de Información de la Universidad Industrial de Santander definió, con el fin de garantizar su calidad y eficiencia, y brindar la flexibilidad necesaria que posibilite adicionar nuevas funcionalidades o desarrollar nuevas versiones cuando sean necesarias.

---

* Trabajo de Investigación

** Facultad de Ingenierías Físico – Mecánicas, Escuela de Ingeniería de Sistemas e Informática. Director: Ingeniero, Enrique Torres López. Co-Director: Trabajadora Social, Deisy Lizarazo

SUMMARY

TITLE: INFORMATION SYSTEM TO SUPPORT THE ACTIVITIES OF THE ADVISORY PROGRAM FOR IMPROVEMENT OF ACADEMIC PERFORMANCE OF THE UNIVERSITY WELFARE DIVISION [*]

AUTHOR CACERES MARTINEZ Jefferson, MARTINEZ ROSAS Andrea Katherine, RAMIREZ PRADA Paulo César [**]

KEYWORDS: Web Information System, PAMRA, Client Serving Applications, Java, Improving Academic Achievement, Mentoring, Beneficiary, Tutor, Welfare University.

DESCRIPTION:

The Advisory Program for Improving of Academic Performance (PAMRA), ascribed to the University Welfare Division aims to develop methodologies and educational strategies to support the vocational training process of the Universidad Industrial de Santander students, contributing effectively to the reduction of problems related to low academic performance through peer work (tutoring), which is the applied methodological tool, with the students as human resources.

The virtual tool developed with the aim of solve successfully the needs of the program is focused on providing options and functions that assist, the right pursuit of the mentoring, and other events done by the program. It also facilitates the processes related to the extension of the coverage of the program, programming, support and pursuit of the activities; besides the report and the statistics generation based on the proper management of information. For this purpose, it was necessary the system was integrated with the other information system of the University.

This project was developed taking into account the Standards of the Information Service Division of the Universidad Industrial de Santander (UIS), in order to guarantee its quality and efficiency, and give the necessary flexibility that makes possible to add new functions or develop new versions when it is considered necessary.

---

[*] Research Paper

[**] Physic-mechanical Faculty of Engineering, School of Systems Engineering and Computer Systems. Director: Engineer Enrique Torres López. Co-Director: Social Worker, Deisy Lizarazo

TÉRMINOS Y DEFINICIONES

PROGRAMA DE ASESORÍA PARA EL MEJORAMIENTO DEL RENDIMIENTO ACADÉMICO:

Actividad: Mínima unidad establecida para la descripción de la programación de un evento.

Asesoría: Acompañamiento formativo integral enfocado a fortalecer de manera preventiva y correctiva las posibles falencias en el área de conocimiento tratada.

Beneficiario: Persona de la comunidad estudiantil que participa del programa recibiendo asesoría.

Estudiante UIS: Persona que se encuentra matriculada en algún programa académico de la UIS.

Evento: Grupo de Actividades de carácter lúdico, académico, y/o formativo, encabezadas por uno o varios coordinadores, enfocadas a los integrantes de la comunidad PAMRA.

Grupo Tutoría: Relación existente entre los tutores y los beneficiarios en la que el deseo de recibir/dar asesoría en una asignatura especifica, representa la conformación de un grupo de trabajo.

PAMRA: Programa de Asesoría para el Mejoramiento del Rendimiento Académico.

Portal del Profesor: Plataforma virtual al servicio de la comunidad docente, que permite la administración de recursos académicos para apoyar las actividades académicas. Este espacio es asignado por los administradores de la plataforma al docente que lo solicite, y es éste el encargo del mantenimiento de su sitio.

Portal de PAMRA: Espacio asignado al PAMRA con el fin de mantener y administrar los recursos didácticos que los tutores aporten para el apoyo en las actividades académicas que se realicen.

Registro Tutoría: Control de las actividades académicas realizadas por los tutores durante los diferentes periodos académicos.

Tutor: Persona de la comunidad universitaria que brinda su apoyo académico mediante las actividades del programa.

Tutoría: Herramienta pedagógica que permite el acompañamiento entre pares (iguales) a través del fomento de la actividad académica.

ENTORNO DE DESARROLLO:

Clase: Una clase de objetos describe un grupo de objetos con propiedades similares, con relaciones comunes y con una semántica común.

Interfaz: Es lo que “media”, lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano y una máquina como el computador. Esto implica, además, que se trata de un sistema de traducción, ya que los dos se comunican con lenguajes diferentes: verbo-icónico en el lado del hombre y binario en el caso de la máquina.

Java: Lenguaje de Programación que se caracteriza por tener una arquitectura que permite que el código escrito funcione en multitud de sistemas operativos sin ser modificado.

Método: Es una operación que define como se comporta un objeto.

Módulo: Se utiliza como sinónimo de Subsistema.

Objeto: Entidad provista de un conjunto de atributos y de métodos que reaccionan a eventos en comunes. Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema.

Sistema de Información: Aplicación comercial para el computador. Está constituida por la base de datos, los programas de aplicación, los procedimientos manuales y automatizados, e incluye los sistemas computacionales que realizan procesamiento.

UML: Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar sistemas de software; está respaldado por el OMG (Object Management Group).

INTRODUCCIÓN

Un sistema de información es un conjunto de procedimientos ordenados que permiten el almacenamiento y procesamiento de la información que sirva como apoyo a la toma de decisión y el control de una organización. La información se define como una entidad tangible o intangible que permite reducir la incertidumbre acerca de algún estado o suceso.

En la última década la aplicación de sistemas de información en las organizaciones ha sido de vital importancia, pues permiten acortar tiempos a la hora de tomar decisiones, gracias a que la información se encuentra organizada, actualizada y completa; optimizar costos al realizar los procesos, permitiendo que las organizaciones inviertan mejor sus recursos, mejorando con esto la calidad de sus productos o servicios y detectando síntomas de problemas a futuro y prevenirlos.

Dada la importancia de los sistemas de información es de vital valor la capacitación del personal directivo y administrativo en el uso del sistema en toda entidad que desee beneficiarse de sus capacidades, además de contar con un mantenimiento regular para conservar su efectividad.

PARTE I. FUNDAMENTOS

1 PRESENTACIÓN DEL PROYECTO

1.1 DESCRIPCIÓN DEL PROYECTO

1.1.1 Objetivo General

Analizar, Diseñar, implementar y poner en funcionamiento el Sistema de Información para el programa PAMRA de la División de Bienestar Universitario que permita su difusión, ampliación de cobertura a las sedes, programación y seguimiento de actividades, asignación de responsables y beneficiarios, evaluación de las actividades que se realizan y del personal que en ellas participan, generación de informes y estadísticas, orientado hacia la web, utilizando los estándares que la división de Servicios de Información tiene para el desarrollo de software y debidamente integrado con los demás sistemas de información con los que cuenta la universidad.

1.1.2 Objetivos Específicos

El sistema de información del grupo PAMRA tiene como objetivos específicos:

- Permitir el control y seguimiento de los procesos que se realizan como apoyo de las actividades de asesoramiento académico brindadas por el grupo PAMRA dando soporte a las actividades administrativas del grupo para su control y administración.

- Crear un sistema de información orientado a la web que permita su acceso y utilización desde cualquier punto de internet con el fin de descongestionar las funciones administrativas del grupo PAMRA permitiendo consultas en menor tiempo y con información más completa.

- Soportar el proceso de selección de asignaturas para las cuales se va a ofrecer asesoramiento académico, tutores asociados a cada asignatura, horarios de asesorías, estudiantes interesados y actividades a realizar.

- Facilitar el acceso a contenidos académicos administrados por la comunidad PAMRA utilizando la estructura establecida para el “Portal del Profesor UIS”, para su utilización en las actividades de asesoramiento académico.

- Facilitar el proceso de registro de usuarios tutores del grupo PAMRA, su evaluación y observaciones por parte de los beneficiarios del programa en búsqueda de un mejoramiento permanente..

- Permitir la creación, control, seguimiento, notificación y evaluación de actividades académicas o lúdicas por parte de la comunidad PAMRA.

- Desarrollar el sistema de información bajo los estándares de la División de Servicios de Información con el fin de garantizar su correcta integración con los demás sistemas de la Universidad Industrial de Santander.

- Implementar el sistema de seguridad basado en roles y usuarios que la División de Servicios de Información ha desarrollado para el sistema de información institucional.

- Realizar las pruebas necesarias que logren garantizar el correcto funcionamiento del sistema de información del grupo PAMRA.

- Elaborar la documentación de los casos de uso que representan el sistema, de tal manera que el usuario conozca los alcances de este sistema.

1.2 JUSTIFICACIÓN

1.2.1 Antecedentes y Descripción del Problema

En la actualidad el grupo PAMRA cuenta con alrededor de 140 tutores y colaboradores que realizan las labores de impartir tutorías, talleres y demás actividades que el grupo cumple en su función de programa preventivo, dentro de los servicios que se ofrecen a la comunidad universitaria. Aproximadamente 800 estudiantes se ven beneficiados actualmente de este servicio. Sin embargo, las actividades realizadas por estos actores como lo es: La difusión de los beneficios de este programa, la programación y control de grupos de tutorías, control de tutores y beneficiarios, asistencia a talleres, evaluación de desempeño de tutores y beneficiarios, entre otros; no cuentan con el soporte informático adecuado que permita su control y manejo de información de una manera eficiente, haciendo que su organización y almacenamiento se convierta en largas horas de procesamiento manual, cruzando grandes volúmenes de datos para generar cualquier informe solicitado. Aunado a esta situación, desde antes de empezar con la recolección de datos, se presentan demoras en la entrega de información institucional necesaria (datos personales de miembros de la comunidad universitaria).

Con la implementación de un sistema de información óptimo, se facilitará esta labor, ya que la información se almacenará a medida que es generada, se tendrá acceso directo a las bases de datos de la Universidad y la creación de informes se realizará mediante una simple consulta a una base de datos, correctamente normalizada y optimizada.

El grupo PAMRA crece a medida que más personas se vinculan y sacan provecho de este, por lo que el servicio que prestaba inicialmente en el campus central de la UIS se ha ido expandiendo a las demás sedes de la Universidad. La forma como se obtiene esta información que está afuera de la sede principal es de manera periódica, trayendo gran cantidad de información en documentos que debe ser digitalizada por el administrador del sistema. En la actualidad un sistema de información está en la capacidad de facilitar esta tarea haciendo uso de las tecnologías web y de esta manera seria algo irrelevante la sede en la que se encuentre el emisor de esta información en cuanto a tiempo y eficiencia en el proceso, pues se estaría generando en tiempo real desde cualquier acceso a Internet.

Dentro de las políticas a futuro de la división de Servicios de Información de la Universidad se tiene la de integrar de manera lógica los servicios de Bienestar Universitario, por lo tanto el sistema de información que se desee desarrollar para el grupo PAMRA debe realizarse de acuerdo a los estándares establecidos por la división de Servicios de Información para facilitar su integración con el sistema de información institucional.

Por otra parte el desarrollo de un sistema de información efectivo para el grupo PAMRA y todos los beneficios que contempla su implementación es provechoso y se consideran como un gran proceso de mejoramiento del grupo.

1.2.2 Impacto

Este sistema facilitará el acceso a los servicios del grupo PAMRA desde cualquier punto de acceso a internet lo cual descongestionara las oficinas administrativas permitiendo una fácil ampliación del programa a las demás sedes de la Universidad, agilizando los procesos de inscripción y registro de tutores, tutorías, eventos y actividades realizados y orientados a la comunidad en general.

El sistema será usado por todo tipo de usuarios interesados en los servicios que ofrece el grupo principalmente estudiantes de la Universidad Industrial de Santander quienes son la población a la cual va dirigido el programa, organizando la información concerniente a la actividad misional del grupo.

1.2.3 Viabilidad

Para la realización de éste proyecto se cuenta con herramientas de libre distribución como el Jboss Developer Studio 3.0 G.A basado en Eclipse, el servidor de aplicaciones Jboss-seam ambos productos patrocinados por la compañía Red Hat, la cual tiene un contrato de soporte con la Universidad Industrial de Santander. Dichas herramientas presentan ventajas para el desarrollo pues cuentan con grupos de colaboración en todo el mundo a través de internet que garantiza software de altos estándares de calidad y seguridad así como amplia información y capacitación para su utilización lo cual brinda un mayor soporte que las herramientas de distribución privadas.

Aquellas herramientas que no son de libre distribución ya han sido adquiridos y licenciados por la Universidad Industrial de Santander.

En cuanto a hardware se cuenta con instalaciones adecuadas, con los equipos requeridos y el soporte tecnológico necesario para el desarrollo del mismo. Además se tiene la supervisión por parte del director del proyecto y la colaboración del equipo de trabajo de la División de Servicios de información, organismo encargado de la generación y el uso de soluciones informáticas de la más alta calidad técnica que facilitan el proceso de modernización institucional.

2 MARCO TEÓRICO

2.1 GENERALIDADES DEL PROGRAMA DE ASESORÍA PARA EL MEJORAMIENTO DEL RENDIMIENTO ACADÉMICO[1]

2.1.1 Surgimiento del Primer Programa

Mediante una labor investigativa realizada durante el segundo semestre del año 1992, se detectan las áreas y materias que representan mayor dificultad para el estudiante, concluyéndose la necesidad de un apoyo o refuerzo a nivel general.

Con el fin de determinar la efectividad del refuerzo brindado al estudiante-UIS, se toma una muestra piloto conformada por 5 estudiantes en calidad de tutores, con muy buen rendimiento académico, y se seleccionan a 25 estudiantes con serias dificultades académicas como beneficiarios, recibiendo ambos grupos fortalezas para su crecimiento personal, además de mejorar los hábitos de estudio, para un aprendizaje eficaz.

En el año de 1994 se inician las labores, pero sin un presupuesto previo, existiendo un amplio sentido de solidaridad y una gran voluntad hacia el trabajo comunitario. Es en el año de 1995 cuando se cuenta con una sistematización limitada, sin grandes avances, pero a partir del segundo semestre, se da una relativa viabilidad y consistencia a las tareas y a los logros alcanzados, dándose un manejo un poco más ágil a los resultados presentados.

[1] PUENTES DE CONTRERAS, Gilma. El acompañamiento académico y su discernir histórico durante los 15 años de funcionamiento del Programa de Asesoría para el Mejoramiento del Rendimiento Académico “PAMRA” en la Universidad Industrial de Santander. Bucaramanga. 2010.

2.1.2 Bases Fundamentales en la Estructuración del Programa

El Programa de Asesoría para el Mejoramiento del Rendimiento Académico (PAMRA) se ha constituido en una estrategia de apoyo educativo, que busca dar respuesta a las necesidades académicas de los estudiantes a nivel de pregrado.

Entre las causas específicas por las cuales se estructuró el programa se encontró:

- La alta consulta de estudiantes con dificultades de aprendizaje y problemas relacionados con el rendimiento académico
- Los altos índices de condicionalidad y P.F.U. especialmente de los estudiantes del ciclo básico.
- El aumento de los factores tensiónales cuando el estudiante repite materias por segunda, tercera y cuarta vez.
- La conveniencia y necesidad de generar estrategias que permitan dar una solución constructiva a esta delicada problemática
- Inadecuada adaptación de los estudiantes ante el sistema evaluativo de la universidad en comparación al del colegio (calificación numérica vs. calificación cualitativa).

El programa se concibe como una estrategia de apoyo y asesoría pedagógica entre pares, que busca brindar al estudiante una oportunidad para mejorar su rendimiento académico y solucionar las dificultades que se le puedan presentar en cuanto a:

- Inadecuados hábitos de estudio
- Problemas de aprendizaje
- Desorientación y desadaptación bajo rendimiento académico
- Desarrollo de habilidades sociales

2.1.3 Acción Fundamental del Programa

La asesoría y la formación del tutor y del beneficiario se desarrollan teniendo en cuenta los intereses y necesidades de los participantes.

La capacitación para el estudiante tutor, atienda a áreas específicas, mediante la programación semestral de talleres, conversatorios, foros, paneles, simposios y otros.

Estas entre otras, son las temáticas tenidas en cuenta:

- Formación pedagógica en el manejo de las tutorías: cómo enseñar las matemáticas, la química y otras.
- La creatividad en el proceso enseñanza-aprendizaje.
- Conformar de grupos de trabajo.
- El liderazgo y el manejo de grupos.
- Desarrollo de habilidades comunicativas y cognitivas.
- Manejo y control de las emociones.
- Aprestamiento académico (estilos, hábitos, métodos, estrategias de aprendizaje y desarrollo de operaciones mentales)

2.1.4 Beneficios que Brinda el Programa

El programa busca brindar los siguientes beneficios a los estudiantes que se inscriben, ya sea en calidad de estudiante beneficiario o estudiante tutor.

- Mejoramiento del proceso de aprendizaje y por ende el rendimiento académico.
- Desarrollo de aptitudes y actitudes que posibiliten la formación y crecimiento personal y profesional.

- Incremento del desarrollo de adecuadas habilidades sociales.
- Generación de espacios de auto reflexión que permite una postura crítica frente al propio proyecto de vida y la actividad académica de cada estudiante.

2.2 GENERALIDADES DEL ENTORNO DE DESARROLLO

2.2.1 Diseños con el Estándar UML[2]

2.2.1.1 Introducción a 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 concebido 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 que 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 a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE (Object-oriented software engineering). UML ha puesto fin a las llamadas “guerras de métodos” que se habían mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones

[2] Ferré Grau Xavier, Sánchez S. María Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática - UPM

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.

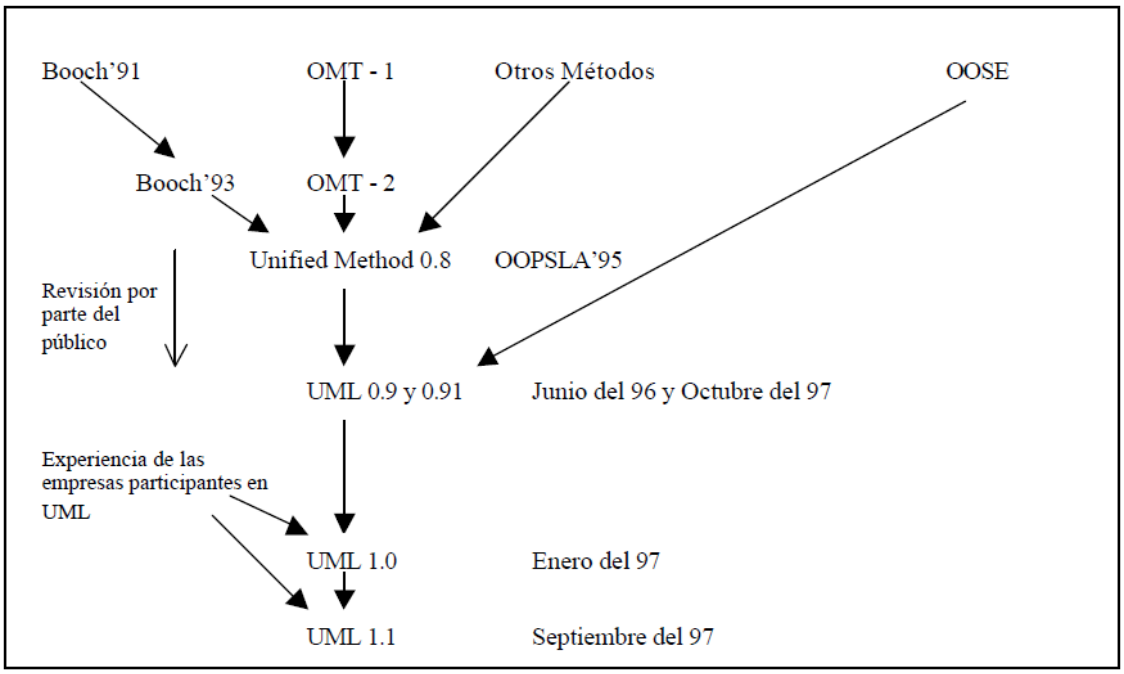

Figura 1. Historia de UML.

Elementos comunes en todos los diagramas

Nota

Una nota sirve para añadir cualquier tipo de comentario a un diagrama o a un elemento de un diagrama. Es un modo de indicar información en un formato libre, cuando la notación del diagrama en cuestión no permite expresar dicha información de manera adecuada.

Una nota se representa como un rectángulo con una esquina doblada con texto en su interior.

Puede aparecer en un diagrama unida a un elemento por medio de una línea discontinua. Puede contener restricciones, comentarios, el cuerpo de un procedimiento, etc.

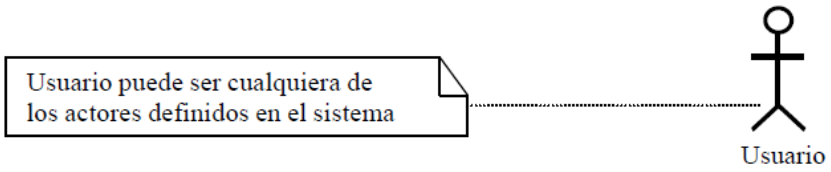

Figura 2. Ejemplo de una Nota UML.

Dependencias

La relación de dependencia entre dos elementos de un diagrama significa que un cambio en el elemento destino puede implicar un cambio en el elemento origen (por tanto, si cambia el elemento destino habría que revisar el elemento origen).

Una dependencia se representa por medio de una línea de trazo discontinuo entre los dos elementos con una flecha en su extremo. El elemento dependiente es el origen de la flecha y el elemento del que depende es el destino (junto a él está la flecha).

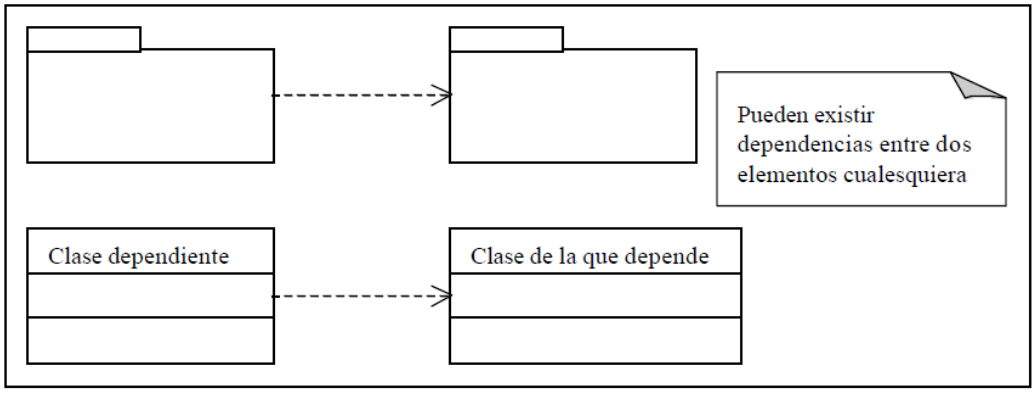

Figura 3. Ejemplo de una Dependencia UML.

2.2.1.2 Diagramas de Casos de Uso

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Elementos del diagrama de casos de uso

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

Actores

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo.

Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).

Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre Casos de Uso

Entre dos casos de uso puede haber las siguientes relaciones:

- Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.
- Incluye: Cuando un caso requiere de otro para su funcionamiento

Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extiende>> o <<incluye>> según sea el tipo de relación.

En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

En la Figura se muestra un ejemplo de Diagrama de Casos de Uso para un cajero automático.

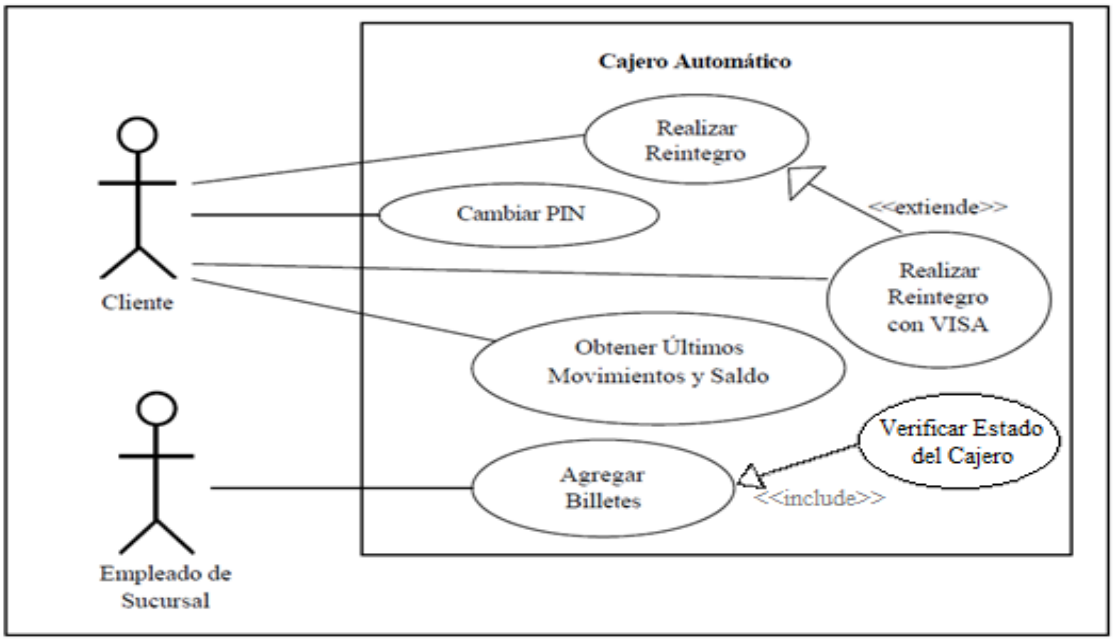

Figura 4. Ejemplo de Diagrama de Caso de Uso

2.2.1.3 Diagramas de Secuencia

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

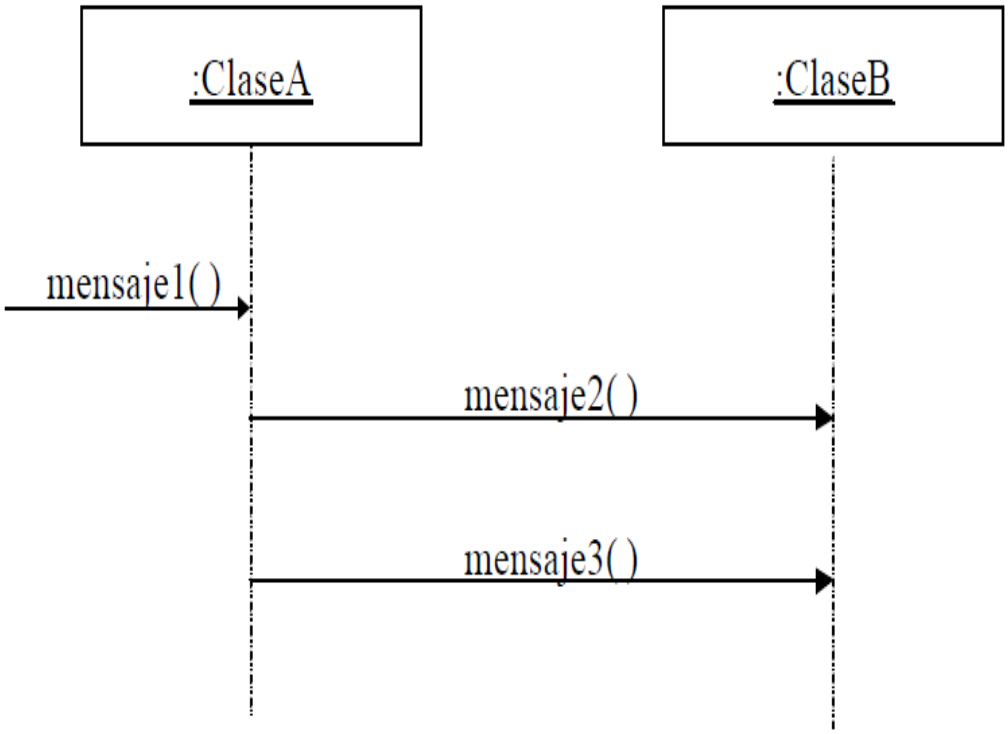

Figura 5. Ejemplo de Diagrama de Secuencia

2.2.1.4 Diagrama de Clases[3]

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. Los diagramas de clases cuentan con:

- Propiedades también llamados atributos o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.
- Operaciones comúnmente llamados métodos, son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar.
- Interfaz es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo.
- Herencia se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede especializarse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además cada uno tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

[3] Fuente: http://es.wikipedia.org/wiki/Diagrama_de_clases

Diagrama de Clases

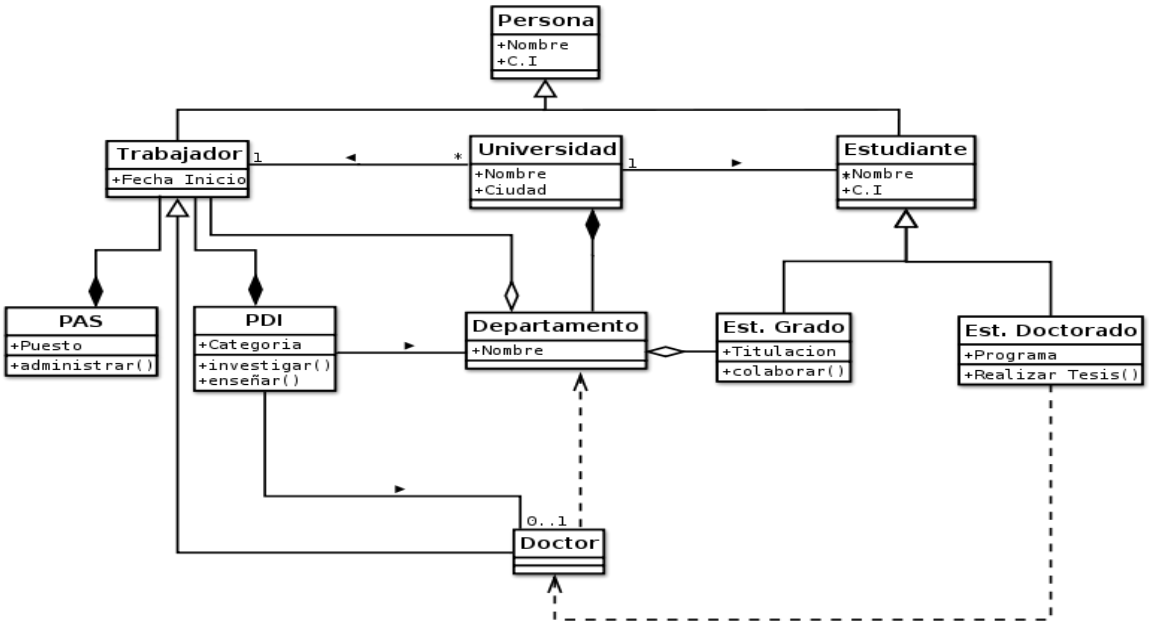

Figura 6. Ejemplo de Diagrama de Clases.

Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como una persona, un transporte, un documento o un paquete. Estos ejemplos de clases de objetos reales, es sobre lo que un sistema se diseña. Durante el proceso del diseño de las clases se toman las propiedades que identifican como único al objeto y otras propiedades adicionales como datos que corresponden al objeto.

2.2.2 Diagrama Entidad – Relación[4]

Un diagrama o modelo entidad-relación (a veces denominado por su siglas, E-R "Entity relationship", o, "DER" Diagrama de Entidad Relación) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades.

[4] Fuente: http://es.wikipedia.org/wiki/Diagrama_entidad-relaci%C3%B3n

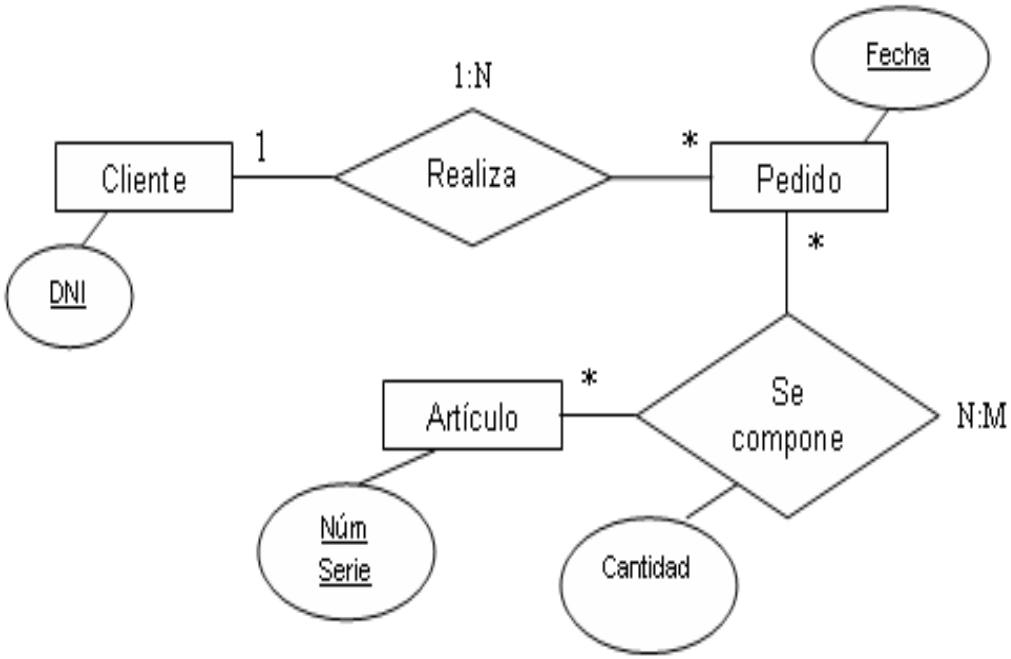

Figura 7. Ejemplo Diagrama Entidad-Relación.

2.2.2.1 El Modelo Entidad-Relación

- Se elabora el diagrama (o diagramas) entidad-relación.
- Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

Dado lo complejo de esta técnica se necesita cierto entrenamiento y experiencia para lograr buenos modelos de datos.

El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr un modelo directamente implementable en una base de datos.

Brevemente:

- Transformación de relaciones múltiples en binarias.
- Normalización de una base de datos de relaciones (algunas relaciones pueden transformarse en atributos y viceversa).
- Conversión en tablas (en caso de utilizar una base de datos relacional).

2.2.3 Tecnologías de Desarrollo de Aplicaciones Web

2.2.3.1 La Web como Sistema de Información[5]

La evolución de Internet como red de comunicación global y el surgimiento y desarrollo de la Web como servicio imprescindible para compartir información, creó un excelente espacio para la interacción del hombre con la información hipertextual, a la vez que sentó las bases para el desarrollo de una herramienta integradora de los servicios existentes en Internet. Los sitios Web, como expresión de sistemas de información, deben poseer los siguientes componentes:

- Usuarios.
- Mecanismos de entrada y salida de la información.
- Almacenes de datos, información y conocimiento.
- Mecanismos de recuperación de información.

Se pudiese definir sistema de información como el conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su posterior uso, generados para cubrir una necesidad (objetivo). Dichos elementos formarán parte de alguna de estas categorías:

- Personas
- Datos
- Actividades o técnicas de trabajo.
- Recursos materiales en general (típicamente recursos informáticos y de comunicación, aunque no tienen por qué ser de este tipo obligatoriamente).

Todos estos elementos interactúan entre sí para procesar los datos (incluyendo procesos manuales y automáticos) dando lugar a información más elaborada y

[5] Rodríguez Perojo Keilyn, Ronda León Rodrigo. El Web como Sistema de Información. http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm

distribuyéndola de la manera más adecuada posible en una determinada organización en función de sus objetivos.

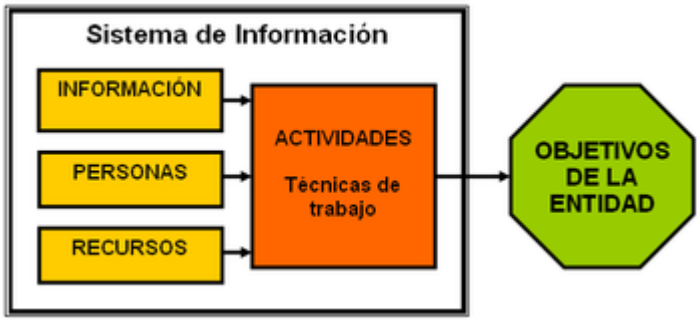

Figura 8. Elementos de un Sistema de Información[6]

Actualmente, los sistemas de información se encuentran al alcance de las grandes masas de usuarios por medio de Internet; así se crean las bases de un nuevo modelo, en el que los usuarios interactúan directamente con los sistemas de información para satisfacer sus necesidades de información.

2.2.3.2 Java Enterprise Edition 5.0 (JEE 5)

Introducción a Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

[6] Fuente: http://es.wikipedia.org/wiki/Archivo:Esquema_sistema_de_informacion.png

Java EE5

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. Ofrece funciones como Java Server Faces (JSF) y la tecnología de servicios web API, Java EE 5 hace que la codificación sea más simple y directa, pero mantiene el poder que ha establecido a Java EE como la primera plataforma para servicios web y desarrollo de aplicaciones empresariales.

El SDK de Java EE 5 y Java Application Platform SDK prestan apoyo a las especificaciones Java EE 5, y el SDK características adicionales, tales como tiempo de ejecución de Open ESB, Portlet Container, y Sun Java System Access Manager.

2.2.3.3 Entorno de Desarrollo

Java Server Faces (JSF)

Java Server Faces es una solución integral al problema de proveer una experiencia de usuario rica y a la vez sencilla en aplicaciones web. Para los desarrolladores de software, JSF provee una API estandarizada, fácil de usar y orientada a objetos, permitiendo crear interfaces de usuario con componentes reutilizables. Esto también repercute en la consistencia de tales interfaces, brindándole al usuario una experiencia de máxima calidad.

Su principal ventaja consiste en que el desarrollador sólo debe aprender el modelo de interfaces de usuario de JSF una sola vez, lo que le permitirá usar cualquier componente que cumpla los estándares de este modelo, aún si tales componentes provienen de terceros.

2.2.3.4 Object Relational Mapping / 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.

2.2.3.5 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. Facilita la implementación de la persistencia por medio del api JPA.

2.2.3.6 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 muchos usuarios, han permitido que JBoss AS (Application Server) se popularice

ampliamente y sea común en los currículos de muchos desarrolladores. Encuestas recientes muestran que es el servidor de aplicaciones más popular actualmente.

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. JBoss no sólo marca la pauta en la adopción de estándares con su servidor de aplicaciones, sino en la imposición de los mismos. Recientemente se le eligió para hacer parte del Java Community Process (JCP). Además en los últimos años ha estado a la cabeza del desarrollo de Java Enterprise llegando a establecerse en todas las especificaciones de requerimientos de Java (Java Specification Requests, JSRs).

2.2.3.7 Enterprise Architect

Enterprise Architect es una herramienta desarrollada por Sparx Systems que ofrece la capacidad de realizar el modelado de un proyecto y apoyar el desarrollo del mismo durante todo su ciclo de vida. Enterprise Architect logra esto usando UML. También puede generar código en varios lenguajes de acuerdo a algunos de estos diagramas.

Su valor como herramienta radica en la capacidad de permitir a los ingenieros desarrolladores comunicar sus ideas y su visión sobre los proyectos facilitando la administración y la redistribución de esta información.

2.2.3.8 Construcción de Prototipos de Software.

Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y, enterarse más acerca del problema y sus posibles soluciones. Un prototipo de software apoya a dos actividades del proceso de ingeniería de requerimientos:

- Obtención de requerimientos: les permite adquirir nuevas ideas para los requerimientos y encontrar áreas fuertes y débiles del software.
- Validación de requerimientos. El prototipo puede revelar errores y omisiones en los requerimientos. La construcción de prototipos puede utilizarse como un análisis de riesgo y una técnica de reducción. Un modelo iterativo del proceso, como el desarrollo incremental, se utiliza junto con un lenguaje diseñado para el desarrollo rápido de aplicaciones. Por lo tanto, las técnicas utilizadas para desarrollar un prototipo para validar los requerimientos también se utilizan para desarrollar el sistema de software mismo.

Ventajas:

- Al mostrar las funciones del sistema se identifican las discrepancias entre los desarrolladores del software y los usuarios, para las cuales se debe buscar un acercamiento.
- Durante el desarrollo del prototipo el personal del desarrollo de software puede darse cuenta de que los requerimientos son inconsistentes y/o están incompletos.
- Se dispone rápidamente de un sistema que funciona y demuestra la factibilidad y usabilidad de la aplicación a administrar.

En sistemas grandes y complejos una forma de resolver la dificultad de evaluación es utilizar un enfoque evolutivo para el desarrollo de sistemas. Esto significa proporcionar al usuario un sistema incompleto y después modificarlo y aumentarlo en el momento en que los requerimientos del usuario sean claros.

El enfoque de construcción de prototipos desechables es para ayudar a refinar y clasificar la especificación del sistema. El prototipo se escribe, evalúa y modifica. La evaluación del prototipo informa del desarrollo de la especificación detallada del sistema que se incluye en el documento de requerimientos de este. Una vez que se ha redactado la especificación, el prototipo ya no es útil y se desecha.

2.2.3.9 Prototipo No funcional [7]

Es un modelo no funcional a escala configurado para probar ciertos aspectos de diseño. Un ejemplo de este enfoque es un modelo a escala completa de un automóvil que se usa para pruebas en un túnel de viento. El tamaño y forma del automóvil son precisos, pero el automóvil no es funcional. En este caso solo se incluyen las características del automóvil que son fundamentales para la prueba en el túnel de viento.

Un modelo no funcional a escala de un sistema de información podría producirse cuando la codificación requerida por las aplicaciones es demasiado extensa para incluirse en el prototipo, pero se puede conseguir una idea útil del sistema a través de la elaboración de un prototipo de la entrada y la salida. En este caso, el procesamiento, debido al excesivo costo y el tiempo requerido, no podría incluirse en el prototipo. Sin embargo, aún se podrían tomar algunas decisiones sobre la utilidad del sistema con base en la entrada y la salida incluidas en el prototipo.

[7] Fuente: Kendall, Julie E. Analisis y Diseño de sistemas. Capitulo 6. Página 153

3 METODOLOGÍA DE DESARROLLO

3.1 CICLO DE VIDA DEL PROYECTO.

A continuación se hace una descripción de las diferentes actividades que se llevaron a cabo durante el transcurso del proyecto, buscando conceptualizar la metodología que se aplicó en el desarrollo del nuevo sistema de información para el programa PAMRA de la Universidad Industrial de Santander.

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.

Todo el proceso de análisis de requerimientos se realizo con las coordinadoras del programa, donde se especificó las funcionalidades y comportamiento del sistema, además de definirse la interfaz y el esquema de navegación entre las páginas.

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

En esta etapa se hicieron reuniones periódicas con las coordinadoras del programa, para que fuesen ellas los que definan las características del software que se desea implantar.

3.1.2 Diseño

El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos de programa: 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 efectuó, mediante modelos UML (Lenguaje de Modelado Unificado) dónde se incluyeron los Diagramas de Casos de Uso, Diagramas de Clases, y de Secuencia, utilizando la herramienta Enterprise Architect licenciada por la Universidad Industrial de Santander a través de División de Servicios de Información.

3.1.3 Implementación de la Aplicación

El diseño se debe traducir en una forma legible por la máquina. El paso de generación de código se llevó a cabo en esta tarea.

En esta etapa se procede a generar el software que se ha diseñado. Se realizó teniendo en cuenta los parámetros establecidos por la División de Servicios de Información en cuanto a los estándares técnicos y de calidad que caracterizan las aplicaciones que son generadas para el servicio de la Universidad, teniendo como base el Lenguaje de programación JAVA 5, frameworks como: seam, java server faces (JSF), Enterprise Java Beans (EJB 3.0) e Informix como motor de base de datos.

3.1.4 Pruebas del software

Son los procesos que permiten verificar y revelar la calidad de un producto software.

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

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

Estas pruebas se aplicaron de forma permanente a lo largo del desarrollo del proyecto por parte del equipo de trabajo, y se abrió un espacio dónde usuarios de prueba interactuaron con la aplicación con el objetivo que detectaran posibles fallos que hayan sido omitidos en el momento del desarrollo.

3.1.5 Ajustes

Es el proceso de mejora del sistema tomando como base las sugerencias y observaciones que se plantearon en el periodo de pruebas, aquí también se incluyen las correcciones a los posibles fallos detectados, mejoras en la interfaz y el diseño de la misma.

Este proceso está incluido dentro del proceso iterativo de refinamiento que se le dio al sistema, para adaptarse plenamente a las necesidades del cliente.

3.2 METODOLOGÍA DE DESARROLLO DEL PROYECTO

3.2.1 Modelo de Construcción de Prototipos

Teniendo como base las actividades anteriormente descritas, se dispuso como metodología de desarrollo el MODELO DE CONSTRUCCIÓN DE PROTOTIPOS.

Se eligió esta metodología debido a que es muy frecuente que los usuarios que están solicitando el sistema, 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 de 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 tanto funcionales como no funcionales permitirán la familiarización del usuario con el sistema que se está desarrollando.

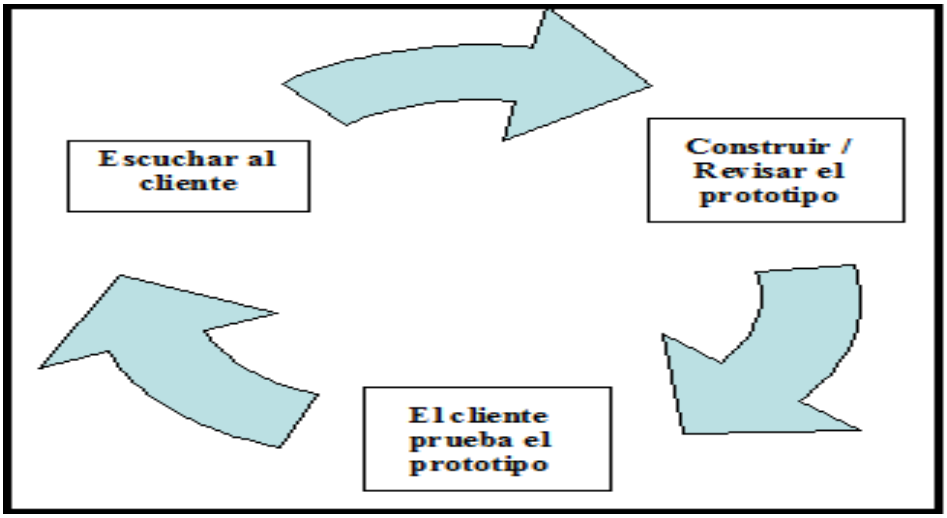

Figura 9. Modelo de Construcción de Prototipos

3.2.2 Estructura del modelo de construcción de prototipos

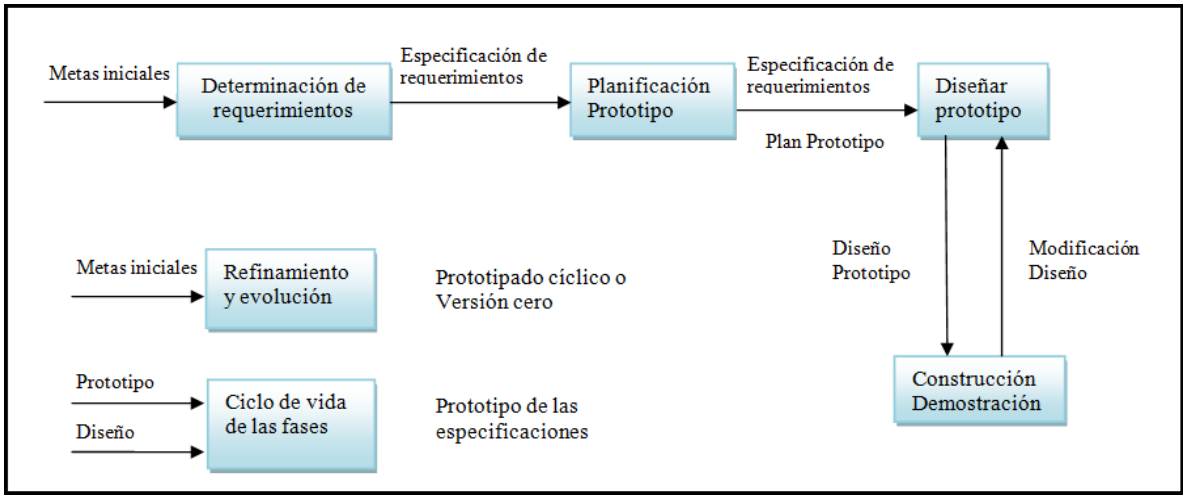

Figura 10. Estructura del Modelo de Construcción de Prototipos

Ésta metodología planteada, permitió que el proyecto se llevara a cabo de manera eficiente, debido a:

- En la creación de los prototipos iniciales se trabajó con unas ideas aproximadas de lo que pretendía alcanzar el programa PAMRA, estas ideas permitieron obtener un producto o prototipo inicial, el cual evolucionó dando como resultado un prototipo más maduro, que cumple con todos los requerimientos del usuario.
- Con el uso del modelo de prototipos se dio la facilidad de mejorar, de manera temprana los prototipos, teniendo en cuenta las sugerencias del usuario solicitante del proyecto, de manera que se pudieran cubrir a cabalidad las necesidades por las cuales se desarrolló el proyecto.
- Es importante conocer de forma temprana si los diferentes prototipos de los servicios, cumplen con los requerimientos expuestos por las coordinadoras del programa.
- En este sistema de desarrollo no se libera un prototipo sin haber realizado todas las pruebas necesarias del mismo, pero los inconvenientes resultan inevitables y se da la facilidad de detectarlos y corregirlos de manera oportuna.

3.2.3 Procedimiento para la metodología planteada

a. La construcción de prototipos comienza con la recolección de los requisitos. El Programa de Asesoría para el Mejoramiento del Rendimiento Académico (PAMRA) de la Universidad Industrial de Santander (quienes son los usuarios administradores del sistema) cuentan de manera verbal lo que desean alcanzar con la nueva versión y los problemas de la actual versión que inevitablemente deben ser subsanados.
b. El desarrollador y usuario se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesaria una mayor definición.
c. Teniendo la abstracción del sistema por parte del desarrollador, se realiza todo el diseño de los objetos mediante los diagramas UML: Casos de uso, de clases, de secuencia y entidad – relación.
d. Luego se produce el Diseño del Prototipo que se enfoca sobre la representación de los aspectos del software visibles al usuario (por ejemplo, métodos de entrada y formatos de salida), en ésta etapa se presenta al usuario el diseño tanto de los diagramas UML como el prototipo NO funcional, para que éste lo apruebe y se adecúe plenamente a su solicitud.
e. Se procede con la construcción del prototipo que se ha propuesto.
f. El prototipo es evaluado por el usuario y se utiliza para refinar los requisitos de tal forma que el usuario identifique aspectos que deban ser replanteados o mejorados.
g. Se produce un proceso iterativo en el que el prototipo es “afinado” (Refinamiento del prototipo) para que satisfaga las necesidades del usuario, al mismo tiempo que facilita al desarrollador una mejor comprensión de lo que hay que hacer y poder entregar el producto final requerido.

PARTE II. DESARROLLO DEL SISTEMA.

4 APLICACIÓN DE LA METODOLOGÍA

4.1 LEVANTAMIENTO DE REQUERIMIENTOS

Los requerimientos para el desarrollo del sistema PAMRA empiezan a redactarse durante las primeras reuniones con las directivas del programa, donde se pudo obtener una idea principal de sus actividades, que sirvió como base para las etapas iniciales del análisis y diseño, determinado las prestaciones que se ofrecerán.

4.2 DESCRIPCIÓN GENERAL

El sistema en su función de dar soporte a las actividades del grupo PAMRA presenta dos grandes módulos, uno orientado al soporte de tutorías y el otro que respalda la programación de eventos. Todo esto enmarcado dentro de la formación integral ha la cual está encaminado el programa PAMRA.

4.2.1 Soporte de Tutorías

- Gestión de los beneficiarios y tutores facilitando el proceso de inscripción identificándolos claramente como dos grupos con diferentes responsabilidades y beneficios.
- Control de los grupos de asesoría, permitiendo incluir, modificar, o eliminarlos, administrando de forma adecuada los horarios y los beneficiarios inscritos. Restringiendo las modificaciones y la eliminación de tal forma que no se

comprometa la integridad de los datos almacenados, en lo que corresponde a las evaluaciones, registros de tutoría, nuevas inscripciones, etc.

- Administración de la participación de los beneficiarios en el programa desde la perspectiva de los grupos, permitiéndole la inclusión de asesorías, cambios o cancelación de las mismas, manteniendo la integridad de las bases de datos y el control de los cambios realizados para futuras consultas.
- Registrar las actividades académicas desarrolladas por los tutores validándolas contra los horarios previamente establecidos para cada grupo de tutoría, haciendo de este registro una fuente fiable del desempeño de los tutores y de la participación de los beneficiarios.
- Acceso al material pedagógico alojado en espacios asignados dentro del proyecto del “Portal del Profesor”, en los cuales, mediante la clasificación y la distribución de dicho material, sea usado como soporte de las actividades académicas realizadas.
- Elaboración y registro de evaluaciones, del Tutor a sus Beneficiarios y viceversa. Al igual que la evaluación realizada al programa por parte de la comunidad participante y beneficiaria de sus servicios.
- Consultar los datos recopilados de las actividades académicas tanto del periodo vigente como los históricos y permitir la generación de informes y estadísticas que puedan ser utilizados en la evaluación del de la efectividad del programa así como para analizar la posibilidad de futuras modificaciones.

4.2.2 Administración de Eventos

- Gestión de los eventos y las actividades del grupo PAMRA dirigidas a la comunidad, facilitando la creación y planeación de estos.
- Control de los eventos en forma adecuada permitiendo llevar de manera práctica los cronogramas de actividades que componen dichos eventos

- Administración de los asistentes a eventos y actividades, brindando información completa de los mismos. Así como dar la posibilidad de realizar un registro en línea de los asistentes en los eventos que así lo requieran.
- Administración de los coordinadores de eventos y actividades, asignando funciones y responsabilidades, en el caso de los coordinadores tutores dichas funciones justificarían beneficios definidos con anterioridad y que complementan su labor dentro del grupo PAMRA
- Control de participación bien sea designando a un usuario en particular para que realice un registro en línea de los eventos y actividades en curso o bien permitiendo al administrado realizar la gestión de estos registros de manera completa y en cualquier momento.
- Consultar el historial de participaciones de un determinado usuario mostrando información completa sobre su estado dentro del evento.
- Consultar el historial de eventos, mostrando información estadística de participaciones. De igual forma permitir la creación de certificaciones totales o individuales si así se requiriera.

4.3 DIAGRAMAS UML

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

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

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

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

4.3.1 Diagrama de Casos de Uso

El caso de uso seleccionado como ejemplo, corresponde al módulo de Administrar Grupos del usuario Tutor, en él se describe la forma en el cual se lleva a cabo la administración de los grupos de asesoría.

En este módulo realizaran las siguientes acciones:

- Creación, edición y eliminación de grupos.
- Consulta de beneficiarios inscritos o retirados.

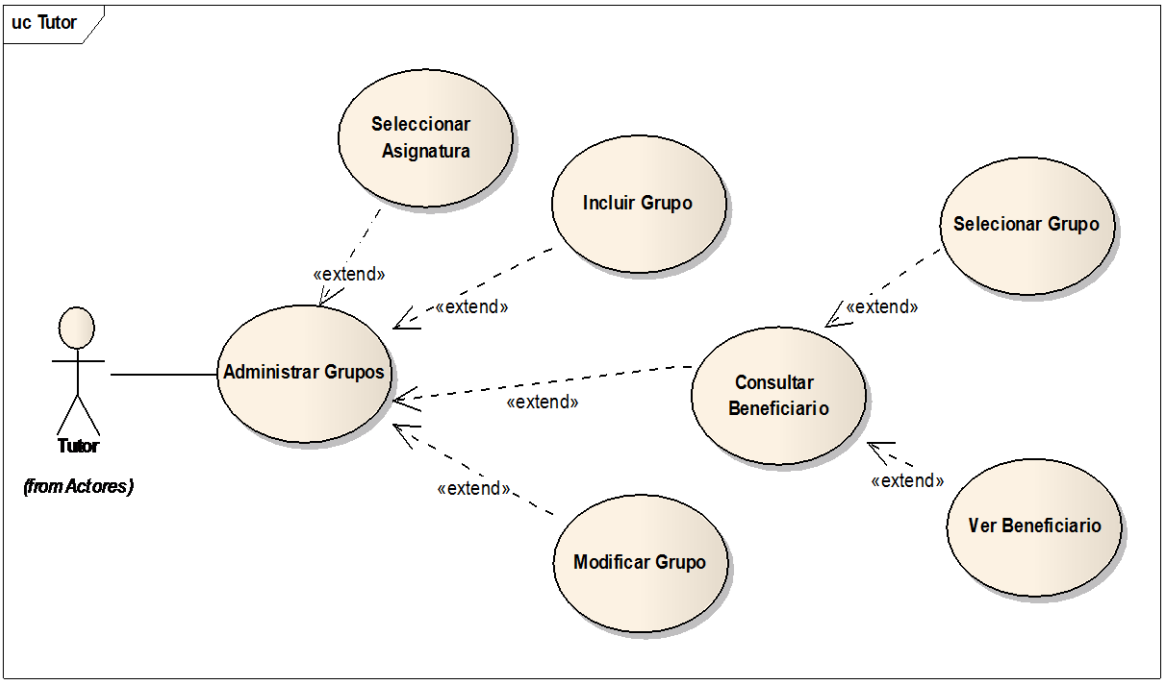

Figura 1. Diagrama Casos de Uso Administrar Grupos

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

Tabla 1. Administrar Grupos

Administrar Grupos	
Actor	Tutor
Propósito	El tutor puede gestionar libremente sus grupos de tutoría, sin salirse de los requisitos establecidos por PAMRA
Resumen	La administración de grupos contempla: Seleccionar asignatura. Incluir grupo. Consultar beneficiario. Modificar grupo.
Flujo Principal	El usuario elige “Administrar Grupos” y se decide por alguna de las siguientes opciones: Seleccionar asignatura. Incluir grupo. Consultar beneficiario. Modificar grupo. **SELECCIÓN** El Tutor: Decide ver los detalles de alguna de sus asignaturas el cual invoca el caso de uso Seleccionar Asignaturas. Decide incluir un grupo el cual invoca el caso de uso Incluir

	Grupo. Decide realizar una consulta de sus beneficiarios el cual invoca el caso de uso Consultas Beneficiario. Desea realizar una modificación del grupo el cual invoca el caso de uso modificar Grupo
Precondición	El usuario debe estar inscrito como Tutor.

Tabla 2. Seleccionar Asignatura

Seleccionar Asignatura	
Actor	Tutor
Propósito	Ver el detalle de los grupos que tiene inscritos.
Resumen	El Tutor selecciona la asignatura del cual desea ver el detalle
Flujo Principal	El tutor selecciona “Detalle” lo que implica: Se muestra un resumen de los detalles del grupo y su horario. En este pantalla puede acceder a otras opciones como: • Modificar Horario • Ver o imprimir beneficiarios
Precondición	El usuario debe estar inscrito como Tutor y tener por lo menos un grupo activo.

Tabla 3. Incluir Grupo

Incluir Grupo	
Actor	Tutor
Propósito	Realizar la inclusión de un nuevo grupo de asesoría.
Resumen	El Tutor selecciona la asignatura en la que desea brindar asesoría.
Flujo Principal	El tutor selecciona “Nuevo Grupo” y se despliega en pantalla una lista de las asignaturas que cumplen con los requisitos exigidos por PAMRA para brindar asesoría. Luego elige la asignatura de la cual desea conformar un grupo.
Precondición	El usuario debe estar inscrito como Tutor.

Tabla 4. Consultar Beneficiario

Consultar Beneficiario	
Actor	Tutor
Propósito	Ver los beneficiarios que se encuentran inscritos con él o los que se retiraron del grupo.
Resumen	El tutor selecciona el grupo del cual quiere ver los beneficiarios activos o retirados.
Flujo Principal	El tutor selecciona el grupo al que quiere ver cuáles son sus beneficiarios. Luego selecciona la opción de “Ver Beneficiarios” y se despliega en pantalla dos tipos de listas, en una esta los

	beneficiarios activos en el grupo y en la otra se encuentran los beneficiarios que se han retirado del grupo. En este punto el tutor puede realizar las siguientes opciones: • Generar un documento .pdf con las listas de los beneficiarios. • Ver los datos de localización del beneficiario incluyendo una foto del mismo.
Precondición	El usuario debe estar inscrito como Tutor, tener por lo menos un grupo activo y beneficiarios inscritos en el.

Tabla 5. Modificar Grupo

Modificar Grupo	
Actor	Tutor
Propósito	Modificar aspectos generales del grupo de tutoría.
Resumen	Realizar las modificaciones que el tutor desea a su grupo de tutoría.
Flujo Principal	El tutor selecciona el grupo al que desea realizar las modificaciones y puede decidir entre las siguientes opciones: • “Edición”, esta despliega un panel modal donde se puede realizar la modificación del cupo. • “Modificar Horario”, esta lo lleva a otra pantalla donde puede realizar la modificación del horario
Precondición	El usuario debe estar inscrito como Tutor

4.3.2 Diagramas de Clases

El diagrama de clases seleccionado como ejemplo, corresponde al módulo de Administrar Grupo, dicho diagrama se aprecia en la figura 2.

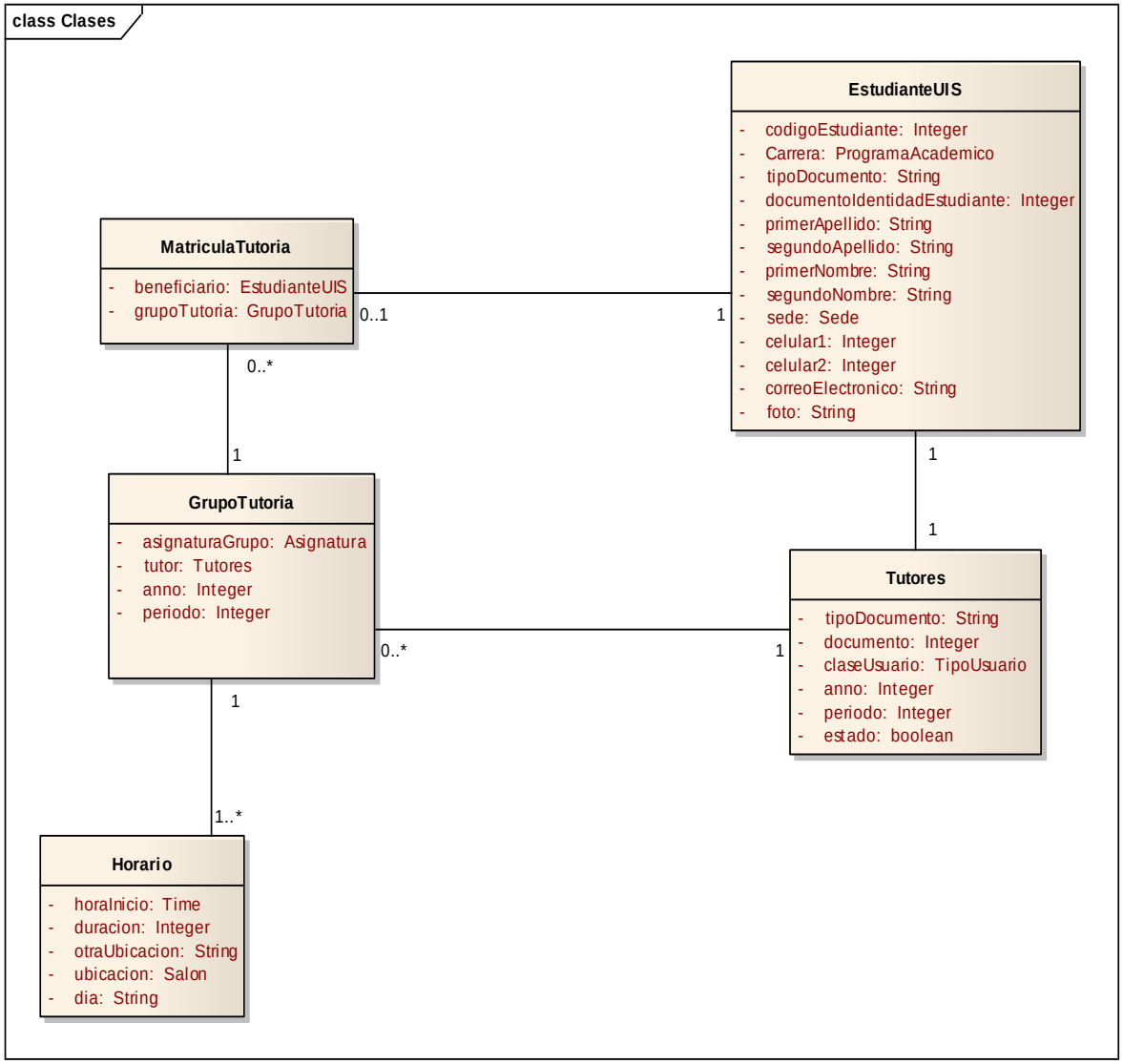

Figura 2. Diagrama de Clases Grupos Tutorías

Clase Tutores: Esta clase contiene los atributos que proporciona una descripción completa de los tutores

Tabla 6. Atributos de la Clase Tutores

Nombre	Tipo de dato	Descripción
tipoDocumento	String	Tipo de identificación del tutor
Documento	Integer	Número de identificación del tutor
claseUsuario	Integer	Clase de usuario del tutor: estudiante, empleado-UIS o contratación externa
anno	Integer	Año académico vigente
periodo	Integer	Periodo académico vigente
Estado	Boolean	Estado del tutor: activo o inactivo

Clase matriculaTutoria: Esta entidad contiene información que relaciona al beneficiario con el grupo al que se inscribe para recibir asesoría académica

Tabla 7. Atributos de la clase matriculaTutoria

Nombre	Tipo de datos	Descripción
Beneficiario	Integer	Usuario de rol beneficiario que se encuentra inscrito a un grupo en particular.
grupoTutoria	Integer	Grupo de asesoría al cual se encuentra inscrito el beneficiario

Clase GrupoTutoría: Esta clase representa la asociación entre Tutor y Beneficiario, por lo tanto contiene atributos que proporcionan una descripción completa de los grupos.

Tabla 8. Atributos de la Clase GrupoTutooria

Nombre	**Tipo de datos**	**Descripción**
asignaturaGrupo	Integer	Asignatura asociada a cada grupo
tutor	integer	Tutor que dirige el grupo
anno	Integer	Año académico vigente
periodo	Integer	Periodo académico vigente

4.3.3 Diagrama de Secuencias

El diagrama de secuencia que se presenta a continuación da una visión más amplia acerca del funcionamiento y la interacción del sistema con el usuario, en el caso de uso que corresponde a la administración de grupos desde la perspectiva del tutor.

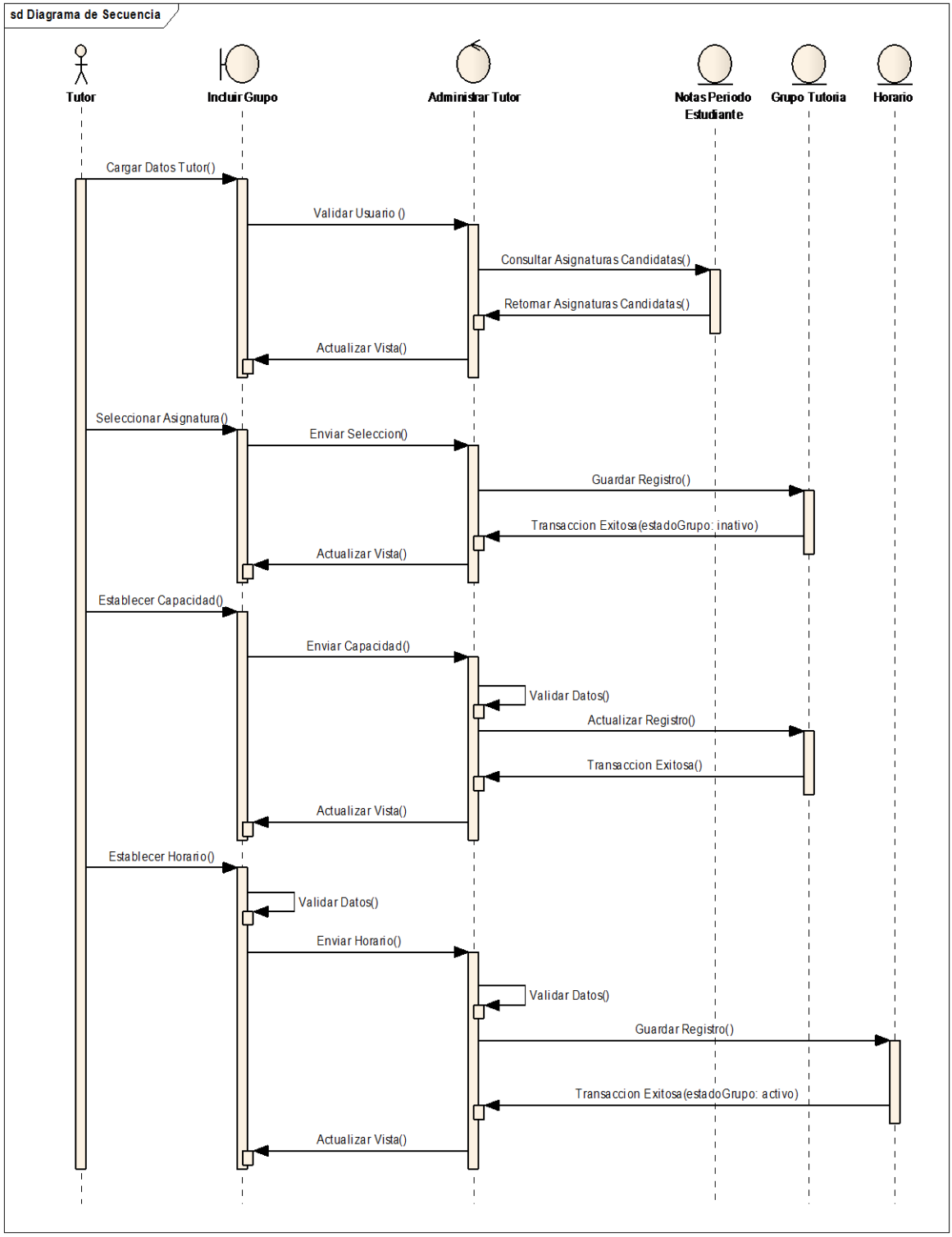

Figura 3. Diagrama de Secuencias de Administrar Grupos

4.4 ESTÁNDARES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN[8]

4.4.1 Aspectos Generales

4.4.1.1 Interfaz de Desarrollo

El IDE de desarrollo a utilizar es el JBoss Developer Studio, el cual debe ser instalado en la carpeta por defecto del instalador.

Este y todos los programas necesarios pueden ser descargados por el personal autorizado, del equipo establecido para tal fin.

4.4.1.2 Servidor de Aplicaciones

En cuanto al servidor de aplicaciones de desarrollo se debe utilizar el mismo que se encuentra en los servidores de producción y desarrollo, el cual debe ser instalado en la carpeta C:\jboss-5.0.0.GA.

4.4.1.3 JAVA

La versión del compilador de JAVA debe ser la 1.6, la cual debe ser instalada en el directorio C:\java1.6.

4.4.1.4 SEAM

La versión del seam a utilizar es la 2.1.2.GA. (jboss-seam-2.1.2.GA.zip).

[8] Fuente: Estándares de la División de Servicios de Información de la Universidad Industrial de Santander.

4.4.1.5 Espacio de trabajo

El espacio de trabajo se debe crear en C:\workspace.

El nombre del proyecto para los JPA debe estar conformado de la siguiente manera:

[Sistema]Entidades

Por ejemplo: AcademicoEntidades (La primera letra de cada palabra en mayúscula).

El repositorio para los JPA debe estar conformado de la siguiente manera:

[Sistema]JPA

Por ejemplo: AcademicoJPA (La primera letra de cada palabra en mayúscula).

El Server name en el IDE de desarrollo se debe llamar JBoss 5 y el nombre del JBoss Runtime Enviroment se debe llamar JBoss 5.

4.4.1.6 Servidor de versiones

Para su configuración se debe instalar en el JBoss Developer Studio los siguientes paquetes:

- subclipse-site-1.4.7
- ajdt_1.6.1a_for_eclipse_3.4
- org.tmatesoft.svn_1.2.1.eclipse

Una vez instalados se debe solicitar el nombre de usuario y la contraseña al Ingeniero encargado.

Cualquier inquietud acerca de la instalación y configuración del servidor de versiones, dirigirse con el Ingeniero encargado.

También se debe instalar el siguiente programa: TortoiseSVN-1.5.8.15348-win32-svn-1.5.5.msi para conectarse con el servidor de versiones de la documentación.

4.4.1.7 Plantillas, estilos, imágenes y formateador

Descargar del servidor de versiones de documentación las carpetas Estilos, Plantillas e Imágenes y copiarlas en la careta view de la aplicación.

4.4.1.8 Patrones a utilizar

Los patrones a utilizar corresponden a cada una de las capas implementadas:

- Capa de presentación: Modelo Vista Controlador, el cual es implementado por Java Server Faces (JSF).
- Capa de lógica de negocio: Session Façade
- Capa de persistencia: Entity Access Object (EAO)

4.4.1.9 Rich Faces

La versión a utilizar es la incluida en el jboss-seam-2.1.2.

4.4.1.10 Código HTML

Está prohibido el uso de etiquetas HTML en las páginas.

4.4.1.11 Anotaciones en la entidad

Las anotaciones en la entidad se deben colocar antes del método get correspondiente y no en la declaración del atributo.

4.4.1.12 Llaves compuestas

Se debe utilizar la anotación EmbeddedId, la cual obliga a declara un objeto del tipo de la llave dentro del EJB de entidad.

4.4.1.13 HashCode e Equals en EJB de entidad

Se debe utilizar únicamente los campos que conforman la llave primaria. En el caso de las llaves compuestas, el atributo que hace referencia a ésta.

4.4.1.14 JPA externos

Para utilizar los JPA externos (academico.jar, recursos-humanos.jar, etc), éstos deben copiarse en la carpeta raíz. En el caso de Windows C:\, para Linux \).

En cada uno de los archivos persistence.xml de su aplicación, se debe utilizar <jar-file>file:/jpa.jar</jar-file>.

Por ejemplo:

<jar-file>file:/academico.jar</jar-file>

<jar-file>file:/recursos-humanos.jar</jar-file>

<jar-file>file:/general-UIS.jar</jar-file>

4.4.1.15 Servicios

Para crear un servicio se debe tener en cuenta las siguientes reglas:

- La interfaz debe contener la anotación Remote

- La implementación de la interfaz debe contener la anotación RemoteBinding con su respectivo nombre jndi (igual al utilizado por la anotación Name). Por ejemplo:

@RemoteBinding(jndiBinding = "ConsultarGenerales")

Para utilizar el servicio se debe utilizar la anotación EJB indicando el nombre jndi. De acuerdo al ejemplo anterior sería:

@EJB(mappedName = "ConsultarGenerales")

4.4.2 Documentación de los Diagramas de Diseño

4.4.2.1 Casos de Uso

Para el desarrollo del modelo de casos de uso, se debe realizar un diagrama de casos de usos por módulo del sistema a implementar siguiendo el estándar propuesto por el Lenguaje Unificado de Modelado 2.1 (UML). Se deben tener en cuenta los siguientes los siguientes puntos:

4.4.2.2 Identificación de Actores

Se identifican con el rol que desempeñan en el sistema.

4.4.2.3 Diagrama de Casos de Uso

El diagrama de casos de uso se tiene que realizar con la herramienta Enterprise Architect. Cada caso de uso constituye un flujo completo de eventos especificando la interacción que toma lugar entre el actor y el sistema.

4.4.2.4 Casos de Uso

Se deben identificar con una acción. Los verbos que se pueden utilizar se encuentran al final del documento.

La información mínima requerida por cada caso de uso es la siguiente:

- Descripción completa
- Precondiciones y postcondiciones
- Descripción del escenario básico y alternos
- Diagrama de clases
- Si el caso de uso es complejo se debe incluir:
- Diagrama de secuencia y/o diagrama de actividades

4.4.3 Sintaxis de nombres en java

4.4.3.1 Reglas de sintaxis generales

- En esta sección se especifican las reglas de sintaxis generales para todos los identificadores (nombres de variables, clases, métodos, etc.)
- Todos los nombres de los identificadores deben estar en español.
- Siempre se deben utilizar nombres que sean claros, concretos y libres de ambigüedades. Usando palabras completas evitando acrónimos y abreviaturas.
- Los nombres deben estar definidos sin espacios en blanco, sin guiones (_ , -), ni comillas (“,’), sin operadores (+, - , / , *), sin tildes, utilizar la n en vez de la ñ y sin caracteres especiales.
- No se debe utilizar la mayúscula para diferenciar entre identificadores distintos. Ejemplo: contador y Contador.

- No se deben diferenciar dos identificadores solo con numerales en cualquier posición. Ejemplo: contador1, contador2, 1contador, 2contador.
- Las siguientes partículas están prohibidas en la declaración de los nombres de identificadores: artículos (el, la, los, unos, unas, un), determinantes demostrativos (este, ese, aquel, aquellos), cardinales (uno, dos, etc.), pronombres de cualquier tipo (yo, tu, el, me, te, se, este, ese, mi, tu, su, etc.).
- Se deben utilizar máximo 5 palabras por nombre, las 3 primeras palabras van completas, a partir de la cuarta palabra se quitan las vocales a la palabra exceptuando la última vocal y la primera si la palabra empieza por vocal. No se utiliza ningún separador entre las palabras, se separa cada palabra utilizando su primera letra en mayúscula. Ejemplo: hacerMantenimientoConsultaUsros, hacerMantenimientoAsignaturasCntxto.

4.4.3.2 Clases

- Los nombres de las clases deben iniciar siempre en mayúscula, deben ser simples y descriptivos.
- Los nombres de las clases de Entidad deben ser sustantivos en singular.
- Para las clases de Entidad siempre se debe definir la anotación @Table que indica la tabla de la base de datos relacionada para su persistencia. Además se debe utilizar el parámetro name de la anotación @Entity para identificar el EJB (@Entity(name = “xxx”)). El nombre de la clase puede ser distinto al nombre de la tabla y debe seguir el estándar de identificadores mencionado en el numeral anterior.
- Los nombres de los EJB utilizados por el patrón Session Façade está conformado por un verbo autorizado (Ver anexo de verbos). Además debe incluir al final las letras “EJB”. Ejemplo: RegistrarMatriculaEstudianteEJB.
- La interfaz asociada al EJB debe llevar el mismo nombre del EJB sin el sufijo “EJB”. Ejemplo: RegistrarMatriculaEstudiante.

- Para los EJB de entidad, cuando la clase representa una relación entre dos entidades, el nombre se forma uniendo los nombres de las entidades involucradas.
- Los nombres de las clases que representan excepciones terminaran siempre con el sufijo “EXCEPCION”.
- El nombre de la clase no contendrá detalles sobre la implementación interna de la misma. Por ejemplo ArrayEstudiantes no es nombre válido.

4.4.3.3 Métodos

- Se deben utilizar los verbos autorizados para su codificación.
- El nombre de los métodos debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.
- La primera palabra del nombre de los métodos debe ser un verbo en infinitivo y debe representar una acción o comportamiento de la clase.
- El nombre del método debe describir claramente el comportamiento del mismo.
- En lo posible no se deben usar verbos genéricos aplicables a todo como: procesar, gestionar, manejar. Ejemplo: procesarEstudiante(), gestionarCliente(), en este caso el verbo no aclara el cometido real del método.

4.4.3.4 Paquetes

- El nombre de los paquetes debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.
- La estructura de los paquetes es la siguiente:

  co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]

Ejemplo:
co.edu.uis.financiero.egresos.contratacion.ordenarPrestacionServicio.

- La estructura para el paquete donde van a estar las entidades comunes para todos los sistemas es el siguiente:

  co.edu.uis.sistema.entidades

- La estructura para el paquete donde van a estar los servicios comunes para todos los sistemas es el siguiente:

  co.edu.uis.sistema.servicios

4.4.3.5 Variables

- Para el nombre de las variables utilizar palabras completas en singular (máximo tres palabras). El nombre deber ir en plural cuando la variable representa una lista o un conjunto de elementos.
- Si las palabras no son suficientes para la descripción, se debe hacer un comentario, al frente de la variable.
- Las constantes (final) van en mayúscula sostenida separando las palabras por guión de piso (_).
- Para los argumentos, deben iniciar siempre con la letra “a” y posteriormente el nombre según lo establecido anteriormente.
- Para las instancias de las clases, las entidades llevan el mismo nombre de la clase, solo que la palabra inicial va en minúscula. Si se necesita más de una instancia, para diferenciarlas se debe adicionar una palabra que identifique el rol que desempeña. Ejemplo: Estudiante estudiantePregrado, Estudiante estudiantePostgrado.
- La variable del EntityManager se debe llamar em.
- La variable de tipo FacesMessages se debe llamar facesMessages.

4.4.3.6 Librerías (JAR)

El nombre de las librerías no sigue el mismo estándar de las clases. Éstas se deben escribir en minúscula, separando cada palabra con un guión (-). Ejemplo: recursos-humanos.jar.

4.4.3.7 Nombres de archivos

Se sigue el mismo estándar descrito en las reglas de sintaxis generales.

4.4.4 Documentación

La documentación relacionada con el diseño del sistema reside en la base de datos de Enterpirse Architect.

La documentación del código fuente se debe hacer en cada una de las clases, siguiendo el estándar de JAVADOC y documentando la definición de la clase, descripción de los métodos get y set y descripción de los parámetros de entrada y salida de cada uno de los métodos que componen la clase.

4.4.5 Capa de Presentación

4.4.5.1 Plantilla principal

La plantilla principal de una página es la siguiente:

Figura 11. Plantilla Principal División de Servicios de Información

4.4.5.2 Contenido

Esta sección se refiere al caso de uso implementado. La estructura de esta sección es:

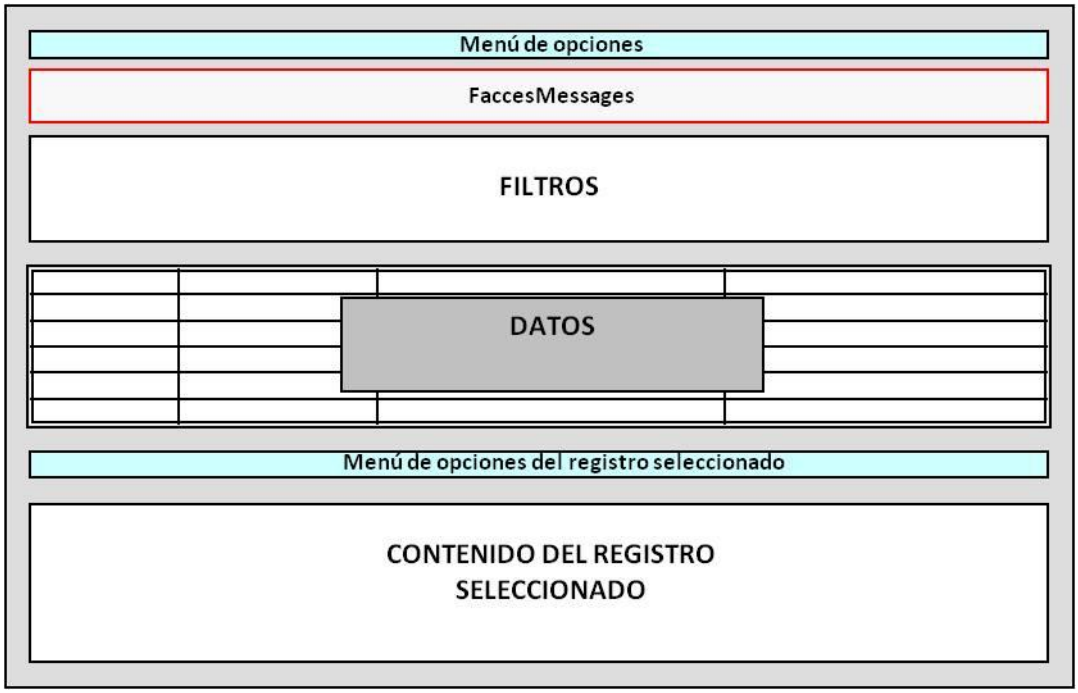

Figura 12. Plantilla de Contenido División de Servicios de Información.

Para las páginas que muestran formularios:

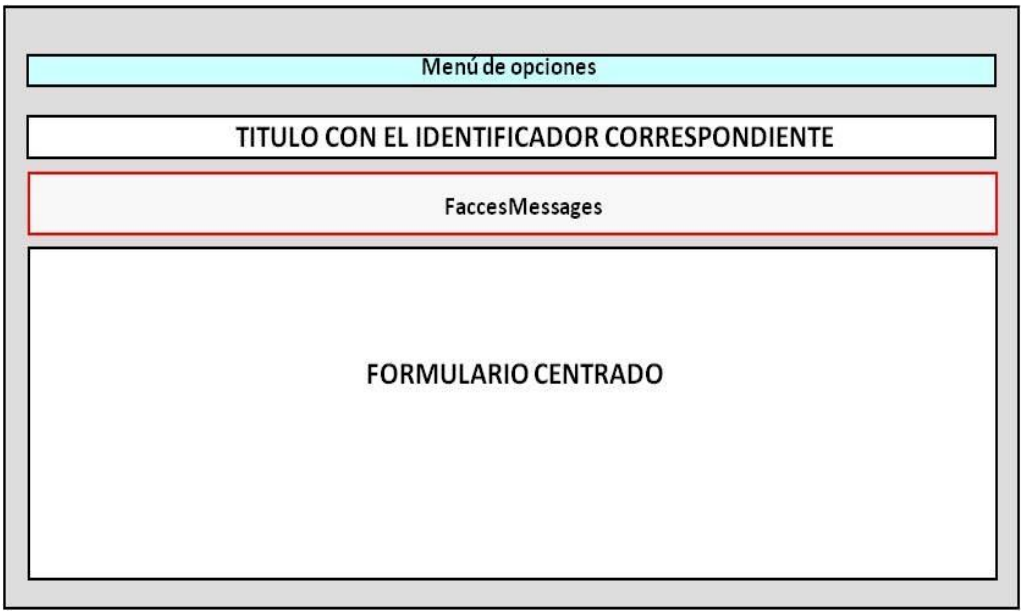

Figura 13. Plantilla de Contenido con Formulario. División de Servicios de Información.

4.4.5.3 Paginación

Para la paginación se debe tener en cuenta el tipo de consulta que se va a realizar y la memoria consumida por ésta en el servidor de base de datos. De acuerdo a esto la aplicación decide el número de registros máximos que debe retornar la consulta.

Esto indica que cuando se implemente el método de consulta en JPQL, el parámetro setMaxResults debe tener el valor de 100, para que los resultados de las consultas no superen este valor

4.4.5.4 Texto

Existen cinco tipos de plantillas para mostrar texto en la página. Éstas se encuentran dentro de la carpeta Plantillas.

- etiqueta.xhtml: Texto o datos.
- etiquetaColumna.xhtml: El título de la columna en una tabla
- titulo.xhtml: Titulos
- etiquetaNavegacion.xhtml: Utilizado en la barra de navegación
- mostrar.xhtml: Permite visualizar dos etiquetas (etiqueta, dato) al mismo tiempo. Su objetivo es la de visualizar datos como si fuera un formulario.

La sintaxis para utilizar las plantillas son:

```
<s:decorate template="../Plantillas/[nombrePlantilla]">
     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
</s:decorate>
```

Donde nombrePlantilla puede ser etiqueta.xhtml, etiquetaColumna.xhtml, etiquetaTitulo.xhtml o etiquetaNavegacion.xhtml.

Para la plantilla mostrar.xhtml:

```
<s:decorate template="../Plantillas/mostrar.xhtml">
     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
     <h:outputText
     value="#{formatoWeb.usuario.primerNombre}"/>
</s:decorate>
```

4.4.5.5 Edición

Para capturar cualquier tipo de dato en una caja de texto se debe utilizar la plantilla edicion.xhtml de la siguiente manera:

```
<s:decorate id="dcr[identificador]"
     template="/Plantillas/edicion.xhtml">

     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>

     <h:inputText id="txt[nombreCajaDeTexto]" value="[valor]"
          required="true">

          <a:support event="onblur"
     reRender="dcr[identificador]"/>
     </h:inputText>

</s:decorate>
```

dcr[identificador] es el nombre del elemento decorate. Por ejemplo: dcrPrimerNombre.

txt[nombreCajaDeTexto] es el nombre de la caja de texto. Por ejemplo: txtPrimerNombre.

4.4.5.6 Tablas estáticas

Se debe usar el control panelGrid de Java Server Faces.

4.4.5.7 Tablas dinámicas

Se debe usar el control dataTable de Rich Faces, agregando las siguientes líneas de programación en su definición:

onRowMouseOver="this.style.backgroundColor='#E1E1E1'"

onRowMouseOut="this.style.backgroundColor='#{a4jSkin.tableBackgroundColor}'

Las cuales indican el color de la fila cuando el puntero del mouse esta sobre ésta.

4.4.5.8 Listas desplegables

Para cualquier tipo de lista se debe utilizar el control de Java Server Faces.

4.4.5.9 Mensajes del sistema

Los mensajes del sistema son textos enviados por los EJB de sesión, ya sea de confirmación de una acción o errores en el procedimiento. Dichos errores se deben mostrar en la etiqueta FacesMessages la cual debe estar definida al comienzo de la página después del menú si existiera éste. El código es el siguiente:

```
<h:messages globalOnly="true" styleClass="message"/>
```

Para mostrar errores de validación en los formularios, éstos deben aparecer al frente de cada control y no globalmente.

4.5 PROTOTIPOS

De igual forma que con los diagramas antes presentados, no se incluye la totalidad del prototipo, se limita a la presentación de la sección que representa al diseño que se ha plasmado en este documento.

4.5.1 Primer prototipo

El primer prototipo elaborado consistió en el desarrollo de la interfaz de usuario de forma no funcional, es decir, a pesar de contar con la estructura de la interface del usuario, no era posible realizar transacciones a base datos.

En el modulo de tutores y en el de beneficiarios se contemplo la inscripción al programa como el primer proceso a seguir. Luego de que el estudiante se inscribiera accedía a las opciones de la administración de grupos, en las que dependiendo del rol podía inscribir grupos, cancelarlos, modificarlos, incluir grupos, cancelar inscripciones y realizar cambios de grupo.

Para el control de las actividades de asesoría académica, el tutor, mediante un listado de sus grupos y de los beneficiarios inscritos, registraba la asistencia a las tutorías, las cuales eran verificadas por los beneficiarios desde su propio acceso.

La evaluación de los usuarios y del programa se realiza mediante los formatos de evaluación que se diseñaron en base a los formatos aprobados por la universidad para que los usuarios realizaran este proceso.

En el modulo de los eventos se estableció una cartelera de eventos, en la cual los usuarios podía examinar que actividades grupales estaban propuestas contando con la información básica de las mismas. Además, se presentaron los formatos de la formulación de eventos para la posterior programación de los mismos por parte del administrador.

El modulo de administración dotaba al administrador de todas las opciones de funcionamiento que se le asignaron a los tutores y a los beneficiarios. A demás del acceso al modulo de consultas e informes, así como a la administración de eventos.

Como ejemplo, a continuación presentamos una sección del prototipo no funcional, correspondiente a la administración de grupos por parte del Tutor:

Módulo de Administración de Grupos

La siguiente imagen pertenece al formato de las GUI (Graphical User Interface) de la creación de nuevos grupos de asesoría, seleccionando la(s) asignatura(s) en la cual(es) se desee brindar asesoría, determinando las características del grupo de trabajo:

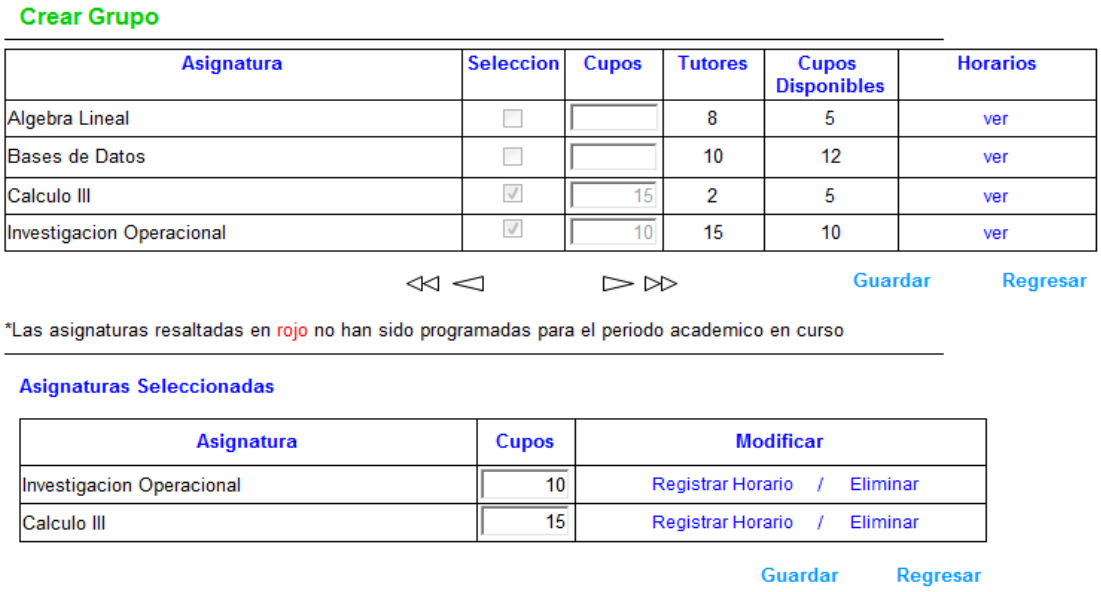

Figura 4. Formato Creación Nuevo Grupo Tutoría

Los grupos creados con éxito pueden ser modificados por el usuario al que le pertenecen, a continuación se muestra el formulario con las opciones de edición de cada grupo.

Figura 5. Formato Detalle y Edición Grupo Tutoría

4.5.2 Segundo prototipo

Como segundo prototipo se elaboró la interfaz del usuario con sus respectivas acciones implementando los templates ofrecidos por el framework RichFaces de Java Server Faces.

Para la elaboración de este prototipo se comenzó por el modulo de mantenimiento de las tablas de soporte, y a continuación se paso al modulo de evaluaciones debido a que de este era el que con menos formatos físicos contaba, permitiéndonos proponer varias opciones del manejo de los formularios que finalmente podrían ser reutilizados en otros módulos.

En la administración de eventos se diseñaron los formularios de propuesta, programación, validación y registro de asistencia a evento.

El modulo de tutoría se enfoco para obtener formularios que fueran similares a los que estaban acostumbrados los usuarios y que brindaran la posibilidad de que el sistema cubriera las necesidades en cuanto a la programación, seguimiento y control de las actividades académicas de asesoría.

Para todo los formularios diseñados se tuvo encunara la validación de los datos ingresados en la capa de presentación, ya que estas condiciones están estipuladas como parte de los estándares de desarrollo.

La siguiente figura muestra el formato inicial de la administración de grupos, donde se puede observar la validación en la que se verifica que todos los datos deben ser registrados en el formulario antes de poder guardar el registro, y que cada uno tiene un rango de acciones disponibles en el sistema

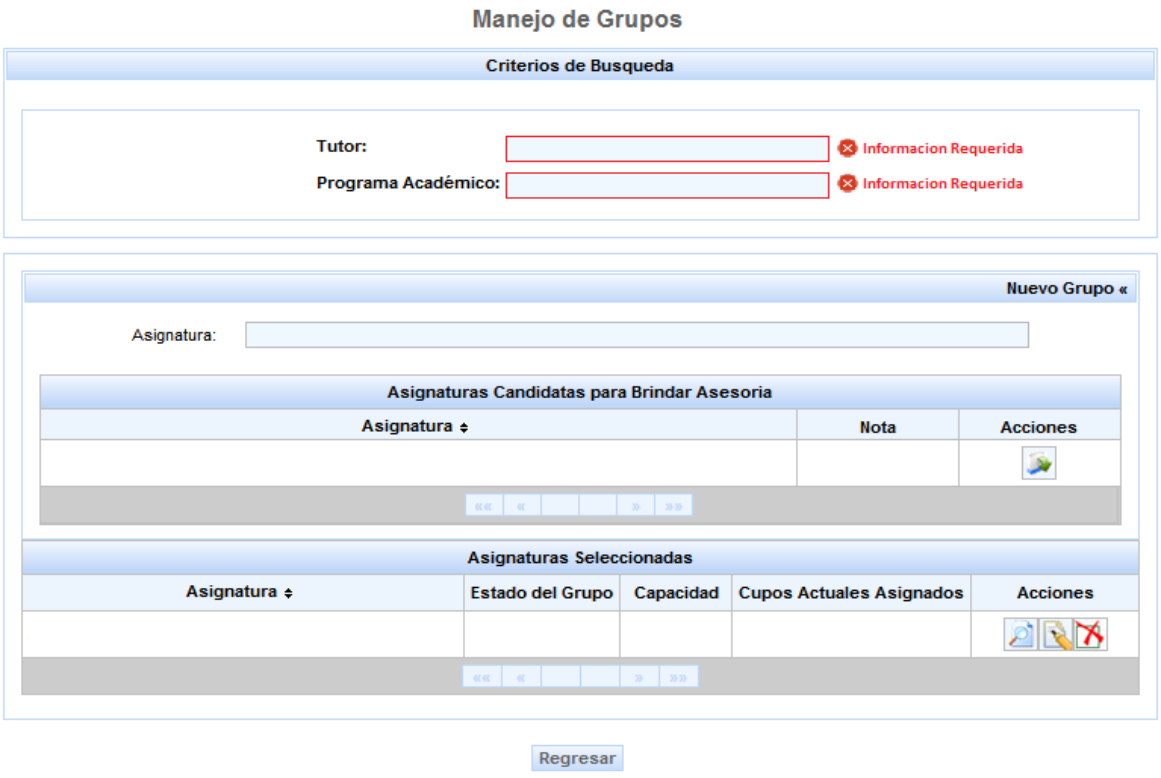

Figura 6. Formato Creación y Edición Grupo Tutoría

La siguiente figura ilustra la interfaz para la modificación los atributos de grupo.

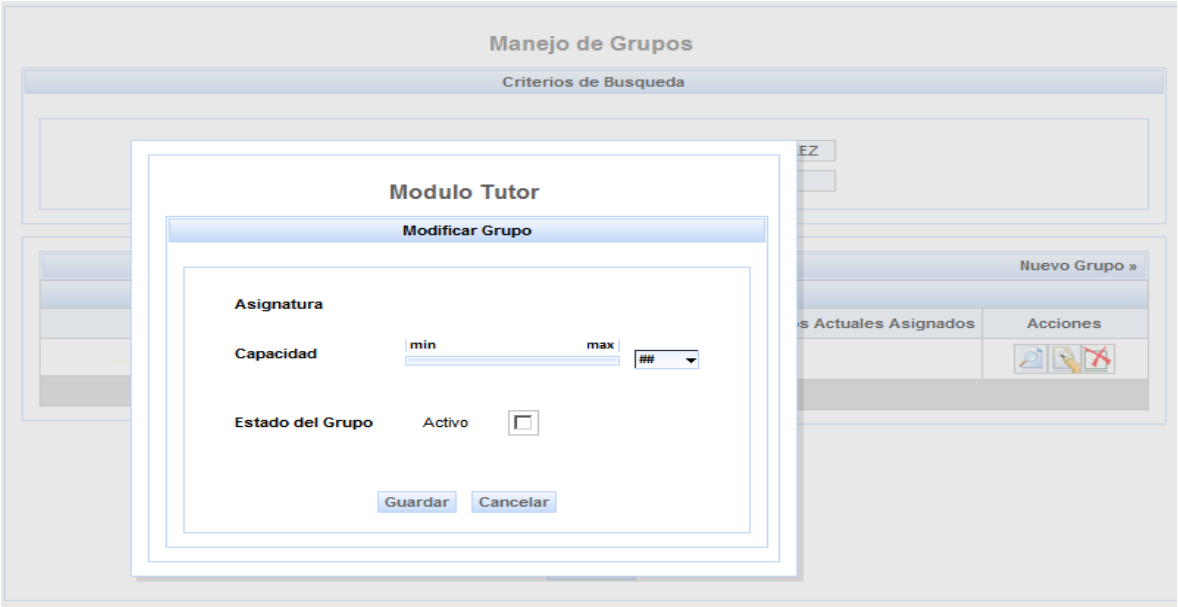

Figura 7. Formato Edición Grupo Tutoría

En la figura que se muestra a continuación se puede observar el formulario para la asignación de horarios, los cuales corresponden a la programación semanal de las actividades de acompañamiento.

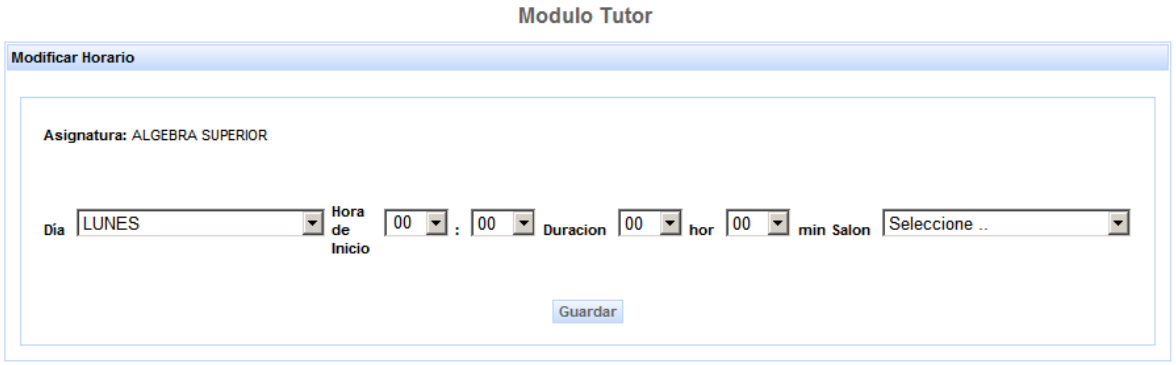

Figura 8. Formato Asignación de Horarios

4.5.3 Prototipo Final

Este prototipo cumple con todos los objetivos estipulados para el sistema, incluyendo las políticas de seguridad provistas por la División de Servicios de Información de la Universidad Industrial de Santander.

La estructura del sistema permite diferenciar tres grandes módulos que a su vez encapsula varios sub-módulos. En estos encontramos la Administración de Tutores, que contempla la inscripción, para la cual se requiere de la actualización de datos, creación de al menos un grupo de tutoría con todas sus características de funcionamiento. La Administración de los Grupos de Tutoría permite la modificación de las características de los grupos inscritos, sus horarios, capacidad, asignatura, todo dentro de los parámetros de validación del sistema, limitando las opciones de eliminación con el fin de mantener un control y seguimiento preciso. Junto a la administración de grupos se encuentra el registro de tutorías, que a pesar de ser una funcionalidad separada, parte del mismo esquema de navegación, para brindar una interface amigable.

La Administración de Beneficiarios permite la inscripción al programa, actualización de datos, selección de grupo de tutoría basado en la información de los tutores que se presentan. La administración de grupos desde la prospectiva del beneficiario brinda las opciones de inclusión, cambio de grupo, y cancelación de inscripción, todo esto sin afectar la información previamente registrada en el sistema. El registro de tutorías se presenta en un formato similar al que el tutor usa para realizar el registro, de esta forma es posible revisar todas las actividades que están registradas por parte del tutor en las que el beneficiario haya participado.

El modulo de evaluaciones permite registrar las apreciaciones de la comunidad PAMRA. Dependiendo del tipo de evaluación se cargan los criterios seleccionados por el administrador como más conveniente para cada situación. De igual forma se cuenta con escalas para cada criterio evaluado.

En el modulo de Eventos se cuenta con la programación, aprobación, modificación, registro de asistencia para cada evento y a su vez de cada actividad que lo compone. En el calendario de eventos se presenta la información básica

necesaria, filtrándolos por su estado, el cual permite tener un control y un seguimiento de cada actividad realizada.

El sistema cuenta también con la posibilidad de generar certificados de las actividades que lo ameriten, así como listados de beneficiarios, tutores y actividades realizadas.

Todos los módulos cuentan con la opción de realizar consultas y generar informes. Esta funcionalidad está ligada directamente al modulo en el cual se accede permitiendo al usuario contar con una vasta información en cuanto al seguimiento y control de todas las actividades soportadas por el sistema, incluyendo los registros históricos almacenados.

A continuacion se presentan los formularos que comprenden la administración de grupos desde la perspectiva del tutor, la creación, modificación y eliminación.

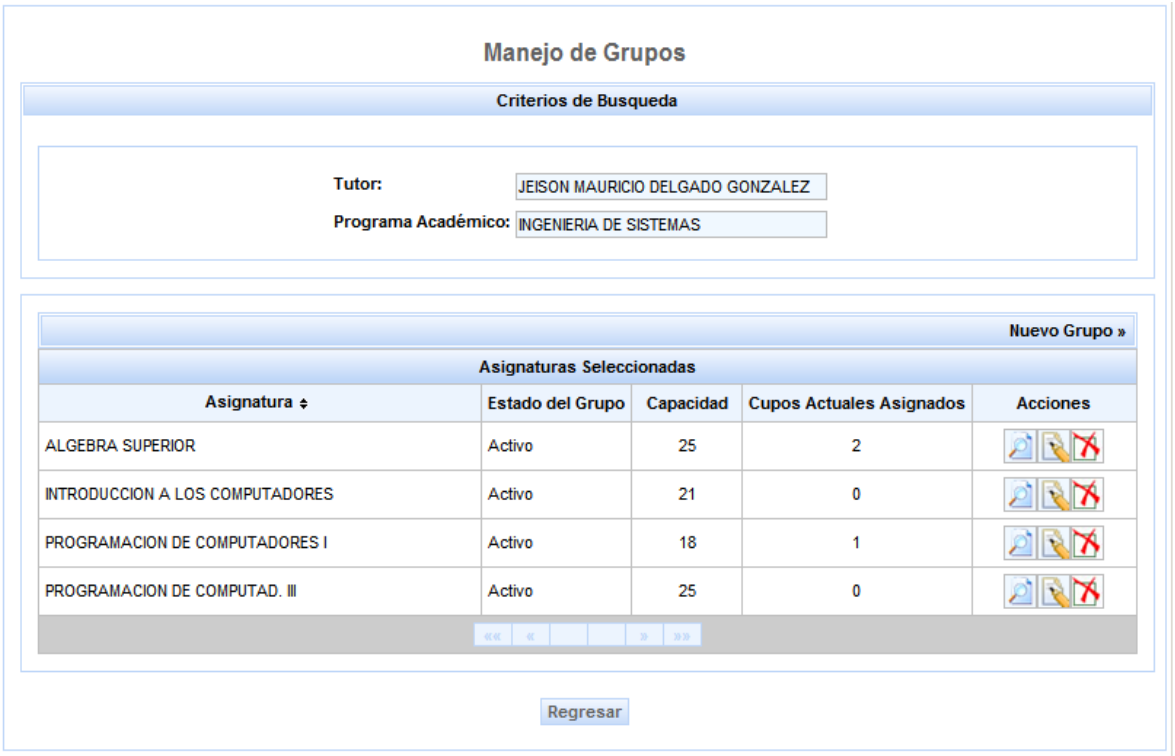

Figura 9. Formato Consulta Grupos Tutorías

Cuando el usuario accede al sistema identificado como tutor, en el modulo de administración de Grupos Tutorías, se puede observar el resultado de la consulta de los grupos creados por el usuario, dándole la opción de ver los detalles de cualquiera de los registrados, además de la posibilidad de editar o modificar los grupos que cumpla las condiciones para estas acciones.

En la siguiente figura se presenta el detalle de un grupo específico, en el cual se dan los accesos para la modificación de horarios y ver los beneficiarios inscritos.

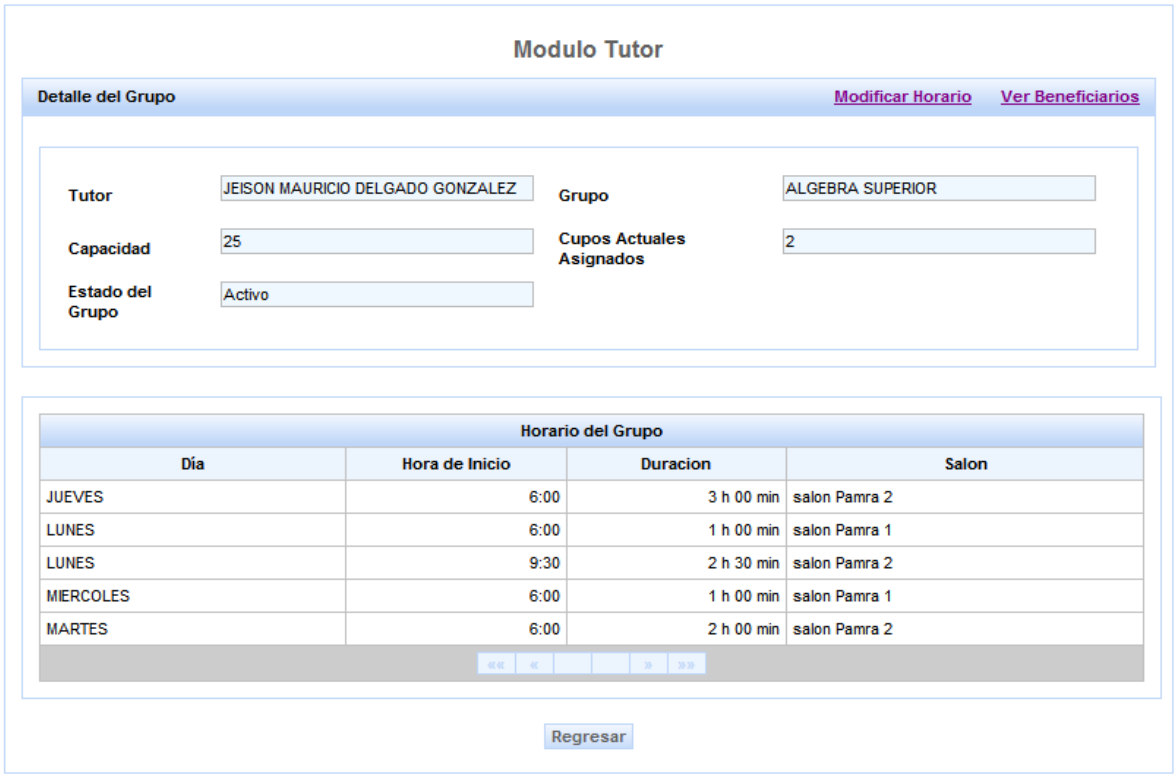

Figura 10. Formato Detalle Grupo Tutoría

La determinación del horario para cada grupo se realiza en el formulario presentado a continuación.

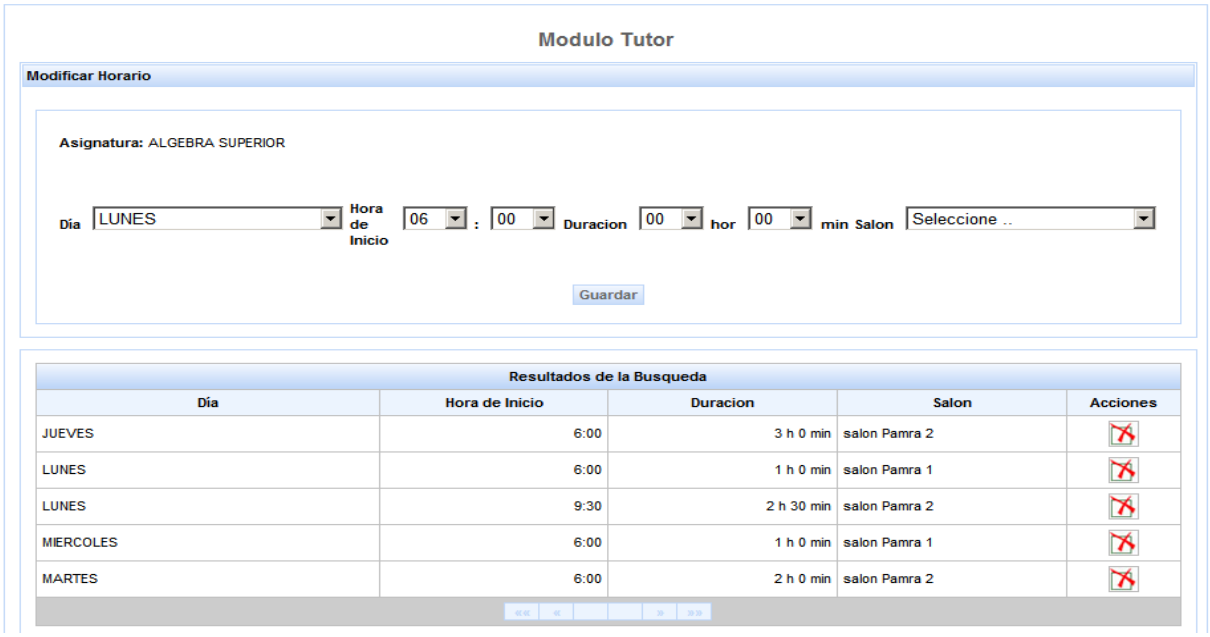

Figura 11. Formato Registro y Modificación Horarios Grupo Tutoría

El listado de los beneficiarios inscritos está separado por el estado de la inscripción, detallando para los inscritos activos su fecha de inscripción y para los inactivos la de su retiro.

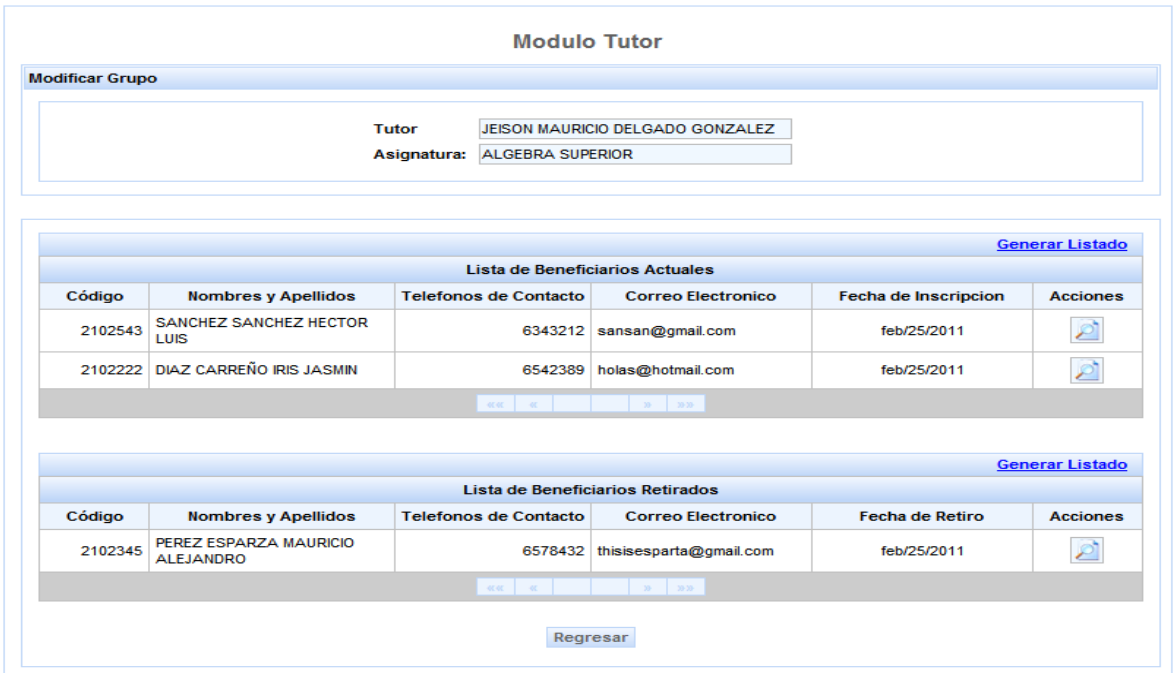

Figura 12. Formato Ver Listado Beneficiarios

4.6 PROYECTO ENMARCADO EN EL ESQUEMA DE SEGURIDAD DE LA UIS.

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.

4.6.1 Estructura de la Base de Datos Soporte

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

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

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

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

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

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

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

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

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

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

creación del permiso, fecha y hora de inicio de vigencia del permiso, fecha y hora de fin de vigencia del permiso.

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

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

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

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

4.6.2 Entorno de Navegación

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

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

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

4.6.3 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”.

4.6.4 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.

4.6.5 Organización de Directorios

4.6.5.1 Consideraciones iniciales

Para cada uno de los sistemas principales se debe crear un proyecto que siga los estándares para la estructura de directorios y nombres. En estos proyectos van las entidades propias de cada sistema, así como los componentes comunes a varias aplicaciones.

Las entidades van, en cada sistema, empaquetadas en un .jar, de tal manera que el desarrollo de una nueva aplicación no implica la implementación de éstas.

En el paquete general van los elementos que son utilizados por todos los módulos.

Las entidades van a estar todas en co.edu.uis.[Sistema].entidades.

Los servicios van a estar todos en co.edu.uis.[Sistema].servicios.

Para la creación de las entidades se toma la estructura de la base de datos. Por lo tanto se hace indispensable comenzar por este paso, seguido de la creación de los componentes para posteriormente desarrollar las demás aplicaciones.

4.6.5.2 Carpetas

Deben comenzar con minúscula y están basados en la estructura de directorios que se genera mediante “Seam-Gen” cuando se crea un nuevo proyecto. Las carpetas que se crean son las siguientes:

- bootstrap
- classes
- dist
- exploded-archives
- lib
- nbproject
- resources
- src
  - hot
  - main
  - test
- test-build
- view

A continuación se hace una breve descripción de las carpetas con las que el desarrollador tiene contacto directo.

Carpeta src: En esta carpeta se guardan los archivos fuentes de las clases del sistema distribuidos en dos subcarpetas de la siguiente manera:

- **Carpeta main:** Contiene las clases de apoyo al proceso. No incluye a las entidades, pues estas estarán disponibles en otro lugar que se describe más adelante en este documento.
- **Carpeta hot:** Contiene los casos de uso reflejados en los EJBs y sus interfaces.

Carpeta resources: Contiene los archivos de configuración de la aplicación.

Carpeta view: Contiene las páginas, imágenes, estilos y todo lo referente a aspecto de la aplicación.

Carpeta dist: Contiene los archivos jar, war y ear de la aplicación. Estos se generan automáticamente.

Carpeta lib: Contiene las librerías necesarias para la ejecución de la aplicación.

4.6.5.3 Nombres de archivos

Se sigue el mismo estándar dado para el nombre de variables descrito en el apartado 4.4.3 de éste documento

4.6.5.4 Organización del código fuente dentro de la estructura de directorios

Paquetes

La composición del nombre de los paquetes se sigue teniendo en cuenta la siguiente sintaxis:

co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]

Ejemplo: co.uis.edu.co. pamra.administracion.mantenimiento.TipoUsuario

En el caso de los proyectos principales (los que empaquetan entidades y componentes para cada sistema) los paquetes serán los siguientes:

Administración (src/action): co.edu.uis.[Sistema].[Aplicación].administracion

Clases generales (src/action): co.edu.uis.[Sistema].[Aplicación].general

Paginas (carpeta view)

Dentro de esta carpeta se tendrán la siguiente estructura:

- Plantillas: Plantillas del sitio
- imágenes
- estilos
- scripts (Si son necesarios. No es obligatorio y se deben evitar en la medida de lo posible).
- Administración: Todo lo referente a la administración
- generales Páginas de uso global en la aplicación
- ayudas
- módulos

4.6.6 Documentación de Programas Fuente[9]

4.6.6.1 Nombre de archivos, variables, constantes, atributos, métodos y parámetros

Los nombres dentro del código fuente siguen los parámetros establecidos en el estándar general de nombres, con las siguientes particularidades:

[9] Fuente: Estándares de la División de Servicios de Información de la Universidad Industrial de Santander

- Los nombre de los parámetros de los métodos empiezan con guión de piso, el resto del nombre sigue el estándar para nombres de variables. Una vez dentro del código fuente, se deben asignar a una nueva variable con el mismo nombre pero sin el guión de piso.
- Los nombres de constantes o variables finally, se escriben en mayúscula sostenida separando las palabras por guiones de piso.

4.6.6.2 Comentarios Java

Los programas en Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación están delimitados por /*.. */ cuando se trata de varias líneas y // cuando se trata de una línea. Los comentarios de documentación (conocidos como "comentarios JavaDoc") son específicos de Java y están delimitados por:

/** ... */. Los comentarios de documentación se pueden extraer a ficheros HTML usando la herramienta javadoc.

Los comentarios de implementación están destinados a comentar el código o para comentarios sobre la implementación en particular, buscan dar una orientación y hacer aclaraciones sobre la implementación a quien observa el código fuente. Los comentarios de documentación están destinados a describir la especificación del código, desde una perspectiva independiente de la implementación, están hechos para ser leídos por desarrolladores que pueden no tener necesariamente el código fuente a mano.

Los comentarios se deben usar para dar una visión general del código y para proporcionar información adicional que no esté disponible fácilmente en el propio código.

4.6.6.3 Orden dentro de los archivos de código fuente:

Cada archivo de código fuente Java debe contener una única clase o interfaz pública y deben seguir el siguiente orden:

- Sentencias package.
- Sentencias import.
- Comentario de documentación de la clase/interfaz.
- Declaraciones de clase e interfaz.
- Comentario de la implementación de la clase/interfaz si fuera necesario.
- Declaración de constantes (solo se declaran dentro de interfaces).
- Declaración de atributos. (la declaración se debe hacer uno por línea y al frente debe ir un comentario de implementación de ser necesario.)
- Declaración de variables. (Según los estándares no se deben declarar variables públicas, para ello se definen los métodos gets() y sets()). Las variables aquí declaradas serán privadas y se utilizarán como indicadores de estado de la clase. Las variables se deben declarar en el siguiente orden:
    1. Variables estáticas: organizadas por ámbito o accesibilidad de la siguiente manera: públicas, protegidas, de paquete (sin modificador) privadas.
    2. Variables de instancia: organizadas de la misma manera que las estáticas.
    3. Al igual que los atributos se debe declarar una por línea para facilitar los comentarios de implementación frente a ellas en caso de necesitarse.
    4. Comentario de documentación sobre cada uno de los constructores (no aplica a entidades).
    5. Declaración de constructores. (Siempre se declarará el constructor sin parámetros, se requiera o no una acción dentro de este. No aplica a entidades).
    6. Comentario de documentación sobre cada uno de los métodos que lo requieran.

7. Declaración de métodos del la lógica del negocio: estos deben ir agrupados por funcionalidad en lugar de por ámbito, siendo el objetivo de esta organización el hacer el código de mas fácil lectura y compresión.
8. Declaración de métodos gets() , sets() e is().
9. Sobre escritura del método equals().
10. Sobre escritura del método hashCode().
11. Sobre escritura del método compare().
12. Sobre escritura del método compareTo().

4.6.6.4 Especificaciones sobre el formato del código fuente

Tabulación

La unidad de tabulación será de 2 espacios.

Longitud de línea

Se deben evitar líneas de 80 o más caracteres ya que algunas herramientas no las manejan bien, además que líneas demasiado extensas hacen difícil la lectura del código.

Ruptura de líneas

- Cuando una expresión no cabe en una única línea, se debe romper de acuerdo a estos principios generales:
  - Romper después de una coma.
  - Romper antes de un operador.

- Preferir las rupturas de alto nivel a las de bajo nivel. (por ejemplo no romper por operador dentro de paréntesis.
- Alinear la nueva línea con el principio de la expresión al mismo nivel que la línea anterior y dar cuatro tabulaciones.

4.6.6.5 Declaraciones y sentencias

Sentencias package e imports

Deben seguir las siguientes reglas de formato.

- Estas sentencias no van tabuladas.
- Se debe declarar una por línea.
- No utilizar comodines.

Variables locales:

Todas las variables locales deben inicializarse en el sitio en donde se declaran.

Las variables deben declararse al inicio de cada método y no esperar a declararlas hasta su primer uso. La única excepción son las variables de bucles que en Java se pueden declarar dentro de la sentencia for. Apropósito de estas variables, se utilizarán las letras de la i a la z en la medida que se vayan necesitando.

Clases, interfaces y métodos

En las declaraciones se deben seguir las siguientes reglas de formato:

- Ningún espacio entre el nombre del método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea que la sentencia de declaración.
- La llave de cierre "}" comienza una línea nueva tabulada para coincidir con su sentencia de apertura correspondiente, excepto cuando es un bloque vacío que la llave de cierre "}" debe aparecer inmediatamente después de la de apertura "{".
- Los métodos están separados por una línea en blanco.
- Las clases e interfaces principales no van tabuladas
- Los bloques de código que pertenecen a una clase o método van tabulados.

Sentencias simples

Se debe escribir solo una sentencia por línea, y no escribir más de una separadas por punto y coma en la misma línea sin importar lo cortas que puedan ser.

Sentencias compuestas

Las sentencias compuestas son sentencias que contienen una lista de sentencias encerradas entre llaves "{" y "}" y deben seguir las siguientes reglas de formato.

- Las sentencias internas deben estar tabuladas un nivel más que la sentencia compuesta.
- La llave de apertura debe estar al final de la línea que comienza la sentencia compuesta; la llave de cierre debe estar en una nueva línea y estar tabulada al nivel del principio de la sentencia compuesta.
- Las llaves se usan en todas las sentencias compuestas, incluidas las sentencias únicas, cuando forman parte de una estructura de control, como

una sentencia if-else o un bucle for. Esto hace más fácil introducir nuevas sentencias sin provocar errores accidentales al olvidarse añadir las llaves.

Líneas en blanco

Las líneas en blanco mejoran la legibilidad resaltando secciones de código que están relacionadas lógicamente.

- En las siguientes circunstancias, siempre se deben usar dos líneas en blanco:
- Entre secciones de un archivo de código fuente.
- Entre definiciones de clases e interfaces.
- En las siguientes circunstancias, siempre se debería usar una línea en blanco:
- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de una sola línea.
- Entre las secciones lógicas de un método, para mejorar la legibilidad.

Espacios en blanco

Los espacios en blanco deberían usarse en las siguientes circunstancias:

- Una palabra reservada seguida por un paréntesis Por ejemplo:

```
while (true) {
        sentencias;
}
```

- En las listas de argumentos, debe haber un espacio después de cada coma.

- Todos los operadores binarios, excepto el operador punto (.) deben estar separados de sus operandos por espacios. Los operadores unarios (incremento ++, decremento --, negativo -) nunca deben estar separados de sus operandos. Por ejemplo:
- Las expresiones de una sentencia for deben estar separadas por espacios en blanco. Por ejemplo:

  for (expr1; expr2; expr3);

- Las conversiones de tipo (cast) deberían estar seguidas de un espacio en blanco. Por ejemplo:

  variableEntera = (int) variableCadena;

5 CONCLUSIONES

- La implementación de JAVA EE 5 permite la creación de componentes software modulares con arquitectura de capas, lo cual facilita el desarrollo de software a gran escala o como en este caso la integración de varios sistemas dentro de un grupo de desarrollo amplio.

- La utilización de componentes java como javabeans, javaServer Pages, permiten de una manera ágil la implementación de características novedosas dentro de los sistemas de información modernos permitiendo implementaciones que sin estos componentes tomarían mucho más tiempo en ser desarrolladas pues cuentan con el soporte de comunidades alrededor del mundo que también trabajan en el campo de las ampliaciones informáticas de gestión

- Al utilizar un servidor de aplicaciones el grupo de desarrollo puede concentrarse más en la lógica de negocio que en las tareas de mantenimiento de bajo nivel como son transacciones, escalabilidad, concurrencia, gestión de componentes desplegados, pues la implementación de los mismos se facilita al contar con dichos servidores

- Al trabajar un proyecto de grado bajo estándares de programación y al ser parte de un equipo de desarrollo como lo es la División de Servicios de Información, se obtiene un alto nivel de aprendizaje y de experiencia importante para nuestro futuro como profesionales en el área de las TIC.

- A través del Sistema de Información desarrollado para el grupo PAMRA se podrán expandir las fronteras del mismo, ya que con las herramientas ofrecidas por los diferentes módulos que lo conforman, se satisfacen las necesidades actuales y

facilita la implementación de nuevas funcionalidades a medida que el programa evoluciona dentro de un plan de mejoramiento continuo.

- Con el uso del Portal de PAMRA, similar al Portal del Profesor, se le brinda a los beneficiarios y en general a todos los estudiantes la oportunidad de contar con material didáctico para reforzar sus conocimientos académicos y culturales.

6 RECOMENDACIONES

- Para lograr que el sistema mantenga un alto grado de estabilidad, se recomienda el soporte de forma permanente por parte de profesionales de la División de Servicios de Información que tengan un vasto conocimiento del sistema, de su desarrollo, así como del actuar del PAMRA, con el fin de que cada actualización o modificación hecha a la herramienta cumpla satisfactoriamente con las demandas del programa así como con las normas y estándares de los sistemas de información con los que cuente la institución.

- Brindar la capacitación necesaria para la correcta administración del Portal de PAMRA, con el fin de sacar el mayor provecho a esta herramienta y contar en un futuro con un recurso académico muy completo que abarque la mayor cantidad posibles de áreas de conocimiento.

- Incentivar el uso del sistema de información del grupo PAMRA para que no caiga en desuso y por el contrario mantenerlo con información actualizada que permita su correcto funcionamiento y pueda brindar el servicio y las facilidades para las cuales fue creado.

- Aprovechar el modulo de administración de eventos con el que cuenta el sistema para la programación, difusión, y seguimiento de las actividades que realiza la División de Bienestar Universitario.

- Dotar a la División de Bienestar Universitario de la estructura tecnológica necesaria para aprovechar todo el potencial que ofrece esta solución en cuanto a cobertura, seguimiento y control de los servicios ofrecidos.

7 BIBLIOGRAFÍA

COBOS, Carlos Alberto; MENDOZA, Martha Eliana. Manual De Informix SQL. Universidad Industrial de Santander, 1998.

DIVISIÓN DE SERVICIOS DE INFORMACIÓN DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER. Estándares de la División de Servicios de Información de la Universidad Industrial de Santander. 2010

FERRÉ GRAU, Xavier. SÁNCHEZ S., María Isabel. Desarrollo Orientado a Objetos con UML. Facultad de Informática UPM.

GROFF, James R. WEINBERG, Paul N. Aplique SQL. Osborne/McGraw-Hill. 1991.

KENDALL, Julie E. Análisis y Diseño de sistemas. Capitulo 6. P 153

PRESSMAN, Roger. Ingeniería del Software. Un enfoque práctico. Quinta Edición. McGraw Hill. España. 2005.

PUENTES DE CONTRERAS, Gilma. El acompañamiento académico y su discernir histórico durante los 15 años de funcionamiento del Programa de Asesoría para el Mejoramiento del Rendimiento Académico “PAMRA” en la Universidad Industrial de Santander. Bucaramanga. 2010.

RODRÍGUEZ PEROJO, Keilyn. RONDA LEÓN, Rodrigo. El Web como Sistema de Información. [En línea]. [Consultado 21 Febrero de 2010]. Disponible en: http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm

Sparx Systems, Pty Ltd. Tutorial UML 2. [En línea]. [Consultado 8 Enero de 2010]. Disponible en: http://www.sparxsystems.com/resources/uml2_tutorial/index.html

SUNMICROSYSTEMS, Inc.The Java EE 5Tutorial. [En línea]. [Consultado 6 Septiembre de 2010]. Disponible en: http://download.oracle.com/javaee/5/tutorial/doc/

WIKIPEDIA La enciclopedia libre. [Web en línea]. [Consultado 16 Noviembre de 2010]. Disponible en: http://es.wikipedia.org/wiki/Diagrama_de_clases
Disponible en: http://es.wikipedia.org/wiki/Diagrama_entidad-relaci%C3%B3n