SISTEMA DE INFORMACIÓN ORIENTADO A LA WEB SOPORTE PARA EL MANEJO DE BENEFICIOS QUE OFRECE EL BIENESTAR UNIVERSTARIO DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

LUZ KARINE FLÓREZ DAZA
LUDWING ALBERTO GARCÍA

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

SISTEMA DE INFORMACIÓN ORIENTADO A LA WEB SOPORTE PARA EL MANEJO DE BENEFICIOS QUE OFRECE EL BIENESTAR UNIVERSTARIO DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

LUZ KARINE FLÓREZ DAZA
LUDWING ALBERTO GARCÍA PARRA

Proyecto de Grado presentado como requisito para optar al título de Ingenieros de Sistemas

DIRECTOR
Ing. JACKSSON SONNY GONZALES.

CODIRECTOR
Ing. ENRIQUE TORRES LÓPEZ.

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

AGRADECIMIENTOS

A Dios por haberme permitido tener la fuerza y la perseverancia para llegar hoy hasta esta estancia de mi vida.
A mi papa por ser el motorcito que hace marchar mi vida y llenarla de paz.
A mi mama por confiar siempre en mí y brindarme su tiempo para escucharme y comprenderme. A mis hermanas por apoyarme tanto y ser las personitas que dan sentido a mi vida.

A Ramiro Mejia mi novio por darme tantas ganas y apoyo en esta etapa de mi vida.

A mi compañero Ludwing Garcia por ser una buena fórmula de trabajo y excelente apoyo.
Al Ingeniero Enrique Torres por su constancia y aporte de conocimientos que ha dado a nuestras vidas.
Al Ingeniero Jacksson Sonny por acompañarnos en este proceso.

A la escuela de Ingenieria de Sistemas e Informatica de la Universidad Industrial de Santander por los conocimientos aportados en mi vida.
A la División de Servicios de Información (DSI) de la Universidad Industrial de Santander por hacer de este proyecto un hecho.

A la Universidad Industrial de Santander, por ser el lugar donde he pasado una parte importante de mi vida y en la cual no crecí solamente como profesional sino como persona ya que he conocido gente maravillosa que me ha aportado gran conocimiento.

Luz Karine Flórez Daza

AGRADECIMIENTOS

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

A mi familia, especialmente a mis padres quiénes siempre me ha brindado su apoyo incondicional durante toda la carrera

A mi compañera Luz Karine Flórez Daza por su constante apoyo y por compartir conmigo este logro.

A nuestro director, el ingeniero Jacksson Sonny por su ayuda y constante apoyo durante la elaboración del proyecto.

A nuestro codirector, el ingeniero Enrique Torres López por su colaboración y respaldo para la ejecución del proyecto.

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

A mis compañeros de las practicas DSI quienes de una u otro forma fueron de gran ayuda en momentos críticos del desarrollo del proyecto y con los cuales compartimos momentos felices y difíciles durante la realización del proyecto.

A la escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander por los conocimiento que se adquirieron un ella durante el transcurso de la carrera.

A la División de Servicios de Información (DSI) de la Universidad Industrial de Santander por permitirme realizar uno de sus proyectos como mi trabajo de grado.

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

Ludwing Alberto García Parra

TABLA DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE FIGURAS

RESUMEN

1. **TITULO:** SISTEMA DE INFORMACIÓN ORIENTADO A LA WEB SOPORTE PARA EL MANEJO DE BENEFICIOS QUE OFRECE EL BIENESTAR UNIVERSTARIO DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER[1]

2. **AUTOR:** LUZ KARINE FLÓREZ DAZA[2]
   LUDWING ALBERTO GARCIA[3]

3. **PALABRAS CLAVES:** Sistema de información, Beneficios, Bienestar Universitario, Aplicaciones cliente-servidor, JAVA, Modelo Construcción de prototipos.

4. **DESCRIPCIÓN:**

La División de Bienestar Universitario de la Universidad Industrial de Santander en cumplimiento de sus funciones misionales ofrece y mantiene servicios orientados a apoyar económicamente a los estudiantes de bajos recursos, para contribuir al mejoramiento de su calidad de vida, entre los cuales se encuentran: residencias estudiantiles, auxiliatura estudiantil de sostenimiento femenino, auxilio fondo patrimonial y becas a hijos y cónyuges de servidores.

Ante la necesidad de hacer más eficientes, seguros y confiables los procesos necesarios para el manejo de dichos beneficios (divulgación, inscripción, verificación de requisitos, selección, adjudicación, entrevistas socioeconómicas, revalidación y control) se vio la necesidad de sistematizar dichos procesos mediante un sistema de información orientado a la web que facilitara y agilizara su realización, así como también promover de esta forma con la modernización e innovación tecnológica en la Universidad.

El sistema de información producto de este proyecto facilita y ofrece comodidades para cada uno de los individuos (estudiantes, administradores y otros) que realizan los procesos necesarios para el normal desarrollo de los beneficios.

Este nuevo producto se realizó bajo la dirección, asesoramiento y con los estándares de calidad y eficiencia de la División de Servicios de Información de la Universidad Industrial de Santander en los desarrollos de los nuevos sistemas de información.

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

ABSTRACT

1. **TITLE:** Information System directed to the Web.Support for the use of benefits offered by Bienestar Universitario from Universidad Industrial de Santander[34]

2. **AUTHOR:** LUZ KARINE FLÓREZ DAZA[5]
LUDWING ALBERTO GARCIA[6]

3. KEY WORDS: Information System, Benefits, Bienestar Universitario, Sever- client Applications, Java, Prototypes Construction Model.

4. DESCRIPTION:

Bienestar Universitario Division from the Universidad Industrial de Santander in performance of its missionary functions offers and keeps services guided to support financially to the low income students, to contribute to the improvement of their quality of life, which between we found: students residence, student auxiliary of female support, patrimony fund auxiliary and scholarships for children and spouse of servants.

Because of the necessity of make more efficient, safe and reliable the process required to the management of those benefits (spreading, inscription, requirements verification, selection, allocation, socioeconomic interviews, validation and control) it was necessary to systematize the process trough an information system guided to the web to make easier and faster its execution, and also to promote this way the modernization and technological innovation on the University.

The information system result of this project facilitates and offers comfort for each one of the persons (students, managers and others) that execute the necessary process of the benefits.This new product was made under direction, advisement and with the best standards on quality and efficiency od teh Information System Division of the Universidad Industrial de Santander in the developments of new information systems.

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

TERMINOS Y DEFINICIONES

Beneficiario: Persona que goza de algún beneficio.

Beneficio: Es una palabra que da cuenta de aquel elemento, producto o servicio que se entrega a una persona para su bien.

Caso Especial: Cuando un estudiante no cumple con algún requisito para la adjudicación del beneficio y la dirección de la universidad decide concederlo previo estudio de las condiciones personales.

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

Contraprestación: Cargue de número horas establecido de trabajo en alguna de las unidades de la Universidad Industrial de Santander a los estudiantes que deseen gozar de algún beneficio y sean seleccionados.

DBU: División de Bienestar Universitario (BU) es la dependencia administrativa de la Universidad Industrial de Santander que brinda apoyo para el buen desarrollo de la actividad académica, la cual constituye una de las funciones misionales de la Universidad, contribuyendo activamente en la formación integral de los estudiantes a través del desarrollo de programas y el ofrecimiento de servicios que propenden por el mejoramiento de su calidad de vida

Entrevistador: Profesional a cargo de las entrevistas, requisito para adjudicar al estudiante un beneficio

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

Funcionario UIS: Personal vinculado a la Universidad mediante contrato de trabajo.

Inscripciones Extemporáneas: Es la inscripción que hace el administrador a los estudiantes que por alguna razón no pudieron inscribirse en el tiempo establecido por Bienestar Universitario.

Interfaz: La idea fundamental en el concepto de interfaz, es el de mediación. La 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 caso 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.

Revalidación: Es el proceso de asignación directa a aquellos estudiantes que en el periodo inmediatamente anterior gozaban de un beneficio y desean seguirlo disfrutando cumpliendo con los requisitos para hacerlo.

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.

INTRODUCCIÓN

La División de Servicios de Información de la Universidad Industrial de Santander tiene la importante tarea de impulsar la innovación tecnológica en la Universidad, y de promover la participación de la comunidad universitaria en la generación de soluciones informáticas de alta calidad, que facilitan el proceso de modernización institucional.

Teniendo en cuenta lo anterior, este proyecto se concentra en la División de Bienestar Universitario, la cual enfrenta una serie de dificultades que podrían ser identificadas y resueltas por la División de Servicios de Información, a través de este trabajo.

Este proyecto pretende dar soporte a los procesos asociados al manejo de los diferentes beneficios que ofrece Bienestar Universitario a los estudiantes de la universidad Industrial de Santander, lo cual ha generado una gran necesidad de soporte tecnológico en las actividades cotidianas para la óptima prestación de estos servicios.

Actualmente el manejo de estos procesos se realiza de forma manual, razón por la cual se hace lento, tedioso, y propenso a cometerse errores en el análisis de los requisitos para la adjudicación de dichos beneficios, motivo por el cual es necesario el desarrollo de un sistema de información que de soporte al manejo de todos los procesos.

CAPITULO 1

1. PRESENTACIÓN DEL PROYECTO

1.1 DESCRIPCIÓN DEL PROYECTO.

1.1.1 Titulo

SISTEMA DE INFORMACION ORIENTADO A LA WEB SOPORTE PARA EL MANEJO DE BENEFICIOS QUE OFRECE EL BIENESTAR UNIVERSITARIO DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER.

1.1.2 Objetivo General

Diseñar, implementar y poner en funcionamiento con una muestra de datos reales un sistema de información orientado a la web, que permita, el manejo de solicitudes, adjudicación, seguimiento y control, de los beneficios estudiantiles de residencias, apoyo y sostenimiento de la mujer, fondo patrimonial e hijos de servidores, que ofrece la Universidad Industrial de Santander.

1.1.3 Objetivos Específicos

El sistema permitirá:

- La inscripción vía web de estudiantes a los diferentes beneficios ofrecidos por Bienestar Universitario.

- Confrontar el cumplimiento de los requisitos necesarios para la inscripción de los usuarios a los beneficios ofrecidos por Bienestar Universitario.

- Mostrar ordenadamente el listado de beneficiarios por beneficios (Residencias estudiantiles, apoyo y sostenimiento a la mujer, fondo patrimonial e hijos de trabajadores) teniendo en cuenta los parámetros de selección definidos por Bienestar Universitario.

- Facilitar el proceso de selección de los beneficiarios.

- Consultar la información de beneficiarios que se encuentran en los sistemas de información académico y financiero para la verificación de requisitos.

- Listar los elementos muebles e inmuebles que serán adjudicados al estudiante durante el tiempo de duración del beneficio, y el cargue de deudas generadas a los estudiantes que hacen parte de estos.

- La generación del reporte de cumplimiento de las contraprestaciones por parte de los beneficiados.

- La programación de fecha y hora de entrevista socioeconómica.

- Seguir las políticas de seguridad establecidos para los sistemas de información de la universidad Industrial de Santander, teniendo en cuenta:

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

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

1.2

1.3 JUSTIFICACIÓN

1.3.1 Antecedentes y descripción del problema.

La división de Bienestar Universitario de la Universidad Industrial de Santander ofrece entre otros los beneficios de residencias, apoyo y sostenimiento de la mujer, fondo patrimonial e hijos de servidores; tales beneficios están encaminados a apoyar el sostenimiento de estudiantes de bajos recursos económicos. Actualmente los procesos de solicitud, selección, adjudicación, seguimiento y control de dichos beneficios se hace de manera manual, lo cual causa las siguientes dificultades:

- Bajo nivel de divulgación de los beneficios ofrecidos y de sus fechas de inscripción: En la actualidad los únicos medios de divulgación de los beneficios y de las fechas de inscripción son carteleras colocadas especialmente en el edificio de Bienestar Universitario, lo cual causa que beneficiarios potenciales no se den por enterados de estos beneficios ni de las fechas de inscripción.

- Aumento en costos de tiempo y recursos: Los procesos llevados a cabo manualmente tienden a ser más lentos en su realización, así como también por el sólo hecho de ser manuales necesitan de más recursos que los procesos sistematizados.

- Perdida de información: Al realizar los procesos manualmente aumenta considerablemente el riesgo de pérdida de información, no se cuenta con información oportuna y su calidad puede verse afectada.

- Falta de integración con otros módulos de beneficios como auxiliaturas docentes y administrativas, residencias, entre otros: Los beneficios ofrecidos por la Universidad a cada estudiante son limitados, por eso es necesario verificar que las personas favorecidas no tengan otros beneficios con excepción de algunos casos previamente establecidos.

- Poca disponibilidad de información actualizada que permita la generación de informes y estadísticas necesarias para la toma de decisiones: La información en papel es difícil de mantener actualizada y difícil de organizar específicamente para obtener informes y estadísticas que sirvan para la toma de decisiones.

- Poca o nula integración con sistemas de información institucional: Al manejar los procesos manualmente, la integración con otros sistemas de información de la institución es nula.

Por lo anterior, la División de Servicios de Información de la Universidad Industrial de Santander en su afán de generar soluciones informáticas de la más alta calidad técnica que faciliten el proceso de modernización institucional y que le permitan a la comunidad universitaria el acceso a la información, ve de vital importancia, contar con un sistema de información orientado a la web que permita mejorar y agilizar los procesos asociados al manejo de los beneficios ofrecidos por Bienestar Universitario de la Universidad Industrial de Santander, integrado con los sistemas de información de la institución y realizado con tecnologías avanzadas cumpliendo los estándares de desarrollo establecidos por la universidad para tal fin.

El nuevo sistema fue desarrollado como trabajo de grado por estudiantes de la escuela de Ingeniería de Sistemas e Informática basados en los conocimientos adquiridos en está, durante su formación profesional y también apoyados en las capacitaciones ofrecidas por la División de Servicios de Información.

El sistema fue elaborado con un diseño orientado a objetos con UML y JAVA 5, estándares utilizados por la División de Servicios de Información para el desarrollo de las nuevas versiones de los sistemas de información institucionales.

1.3.2 Impacto

- **Técnico**

El proyecto, que fue implementado en la nueva plataforma JAVA 5 EE, contribuye a la integración y actualización de los sistemas de la universidad; y a la implementación de nuevas metodologías (como el desarrollo basado en componentes) el cual es un objetivo de la División de Servicios de Información.

- **Económico**

El sistema de información desarrollado disminuye los tiempos y los recursos requeridos para la realización de los procesos asociados al manejo de beneficios, y definitivamente mejorara la productividad logrando disminuir el tiempo de las operaciones y mejorar la calidad del servicio. Reduce el tiempo dedicado por los funcionarios de Bienestar Universitario a las actividades asociadas al manejo de los beneficios y facilita la elaboración de informes ejecutivos para la Dirección de la Universidad.

- **Social**

Sin duda, la elaboración del proyecto facilita el proceso de inscripción a los aspirantes de los beneficios, ya que la inscripción se realizará vía

web y al mismo tiempo facilita a los funcionarios del Bienestar Universitario el manejo de los procesos de recibo de solicitudes, selección, adjudicación, seguimiento y control de los beneficios ofrecidos.

1.3.3 Viabilidad

- **Técnica**

Para el desarrollo del proyecto, además de contar con el recurso humano suficiente, se tuvo el apoyo del personal de la División de Servicios de Información de la Universidad Industrial de Santander, toda la infraestructura tecnológica de la universidad (equipos, servidores, software), y el conocimiento necesario para la ejecución del mismo.

- **Económica**

La División de Servicios de Información (DSI) cuenta con todas las herramientas para la elaboración del proyecto, dentro de las cuales se encuentran las licencias de software, los recursos necesarios para la capacitación, el recurso humano suficiente y el apoyo a lo largo del proyecto.

- **Social**

El proyecto es totalmente viable. No existió ningún obstáculo por parte de la comunidad, ni se perjudicó a ningún sector de la misma. Al

contrario, la comunidad universitaria se verá beneficiada como ya se mencionó en el impacto del proyecto en lo social; el reflejo de esto es la alta prioridad que tiene este proyecto debido a las necesidades planteadas entre el Bienestar Universitario y la División de Servicios de Información (DSI).

CAPITULO 2

2. MARCO TEORICO

2.1 BIENESTAR UNIVERSITARIO

2.1.1 Misión.

Promover y contribuir al desarrollo integral de las personas que conforman la comunidad universitaria UIS y al mejoramiento de su calidad de vida, mediante el desarrollo de Proyectos, Programas y Servicios orientados al desarrollo humano, la protección de la salud y el apoyo social y económico de los grupos vulnerables, con énfasis en la comunidad estudiantil.

2.1.2 Surgimiento de Bienestar Universitario

Desde la década de los 60, se han dado pasos significativos en la UIS, tendientes a mejorar cada día la vida en comunidad; es así como en un comienzo aparece “EL CENTRO DE BIENESTAR UNIVERSITARIO” (CBU), una dependencia de la Dirección de Servicios Universitarios encargada de prestar servicios y realizar aquellas actividades no académicas y contractuales de la UIS, tendientes a satisfacer algunas necesidades en las áreas de salud, y socioeconómica de la población universitaria. Estas necesidades se cubrían inicialmente la prestación de servicios odontológicos, médicos, de farmacia, comedores, cafetería, becas, préstamos, residencias y consejería.

Para el año de 1968, se inauguró el edificio donde actualmente funciona la División de Bienestar Universitario con los “Servicios Médico-Asistenciales”, y los “Servicios de Comedores y Cafetería”; y se iniciaron las prácticas docente-

asistenciales de los programas de Fisioterapia, Nutrición y Dietética y Trabajo Social de la Universidad. Para el año 1973, se creó el servicio psiquiátrico estudiantil de la UIS, dependiente éste del CBU con el propósito fundamental de ofrecer los servicios de asistencia siquiátrica a los estudiantes, y asesoría a los directivos.

Mediante el Acuerdo 090 de 1984, el Consejo Superior Universitario aprueba el reglamento para la prestación de servicios ofrecidos a estudiantes por la Sección de Bienestar Universitario, relacionado con las normas generales, servicios de salud, comedores y cafetería, becas trabajo, orientación, consulta Psicosocial y residencias; y posteriormente, la reforma organizacional de la Universidad propone que Bienestar Universitario sea una dependencia que se derive de la División de Servicios Universitarios y ésta a su vez de la Vicerrectoría Administrativa.

Por su parte, las actividades enmarcadas dentro de la función de promoción de la salud y prevención de la enfermedad comienzan a realizarse en 1988, con campañas de prevención de fármaco-dependencia, información y orientación en el área de planificación familiar y sexualidad humana individualizada y gradual. Lo anterior introdujo cambios internos de personal, adquisición de equipos electrónicos y actualización de la estructura de valoración de cargas administrativas.

En los últimos tiempos, se ha ido perfeccionando progresivamente la concepción de Bienestar Universitario, de tal forma que a comienzos de la década de los 90 el ICFES señala que “El concepto de Bienestar Universitario debe partir de políticas encargadas de investigar, promover, estudiar, formular, y fomentar el desarrollo de los elementos constitutivos de la política de bienestar dando además respuesta por medio de soluciones a los requerimientos de una problemática de tipo no académica, ni administrativa de la vida educativa, problemática que tras haber sido investigada, diagnosticada, medida y formulada brinde una respuesta adecuada por medio de los servicios”.

Dentro de esta tarea de formación integral se han desarrollado convenios con la academia por medio de las prácticas docente asistencial con los diferentes programas académicos permitiendo con esto aumentar la capacidad asistencial y establecer lazos de cooperación mutua en el desarrollo de los programas preventivos dirigidos a toda la comunidad universitaria. Es necesario precisar, que no solo se han establecido convenios con los programas académicos impartidos en la Universidad Industrial de Santander, sino también con otras Universidades del Área Metropolitana de Bucaramanga, como es el caso de la UNAB y la UPB con los programas de Psicología.

En los últimos años el Bienestar Universitario ha evolucionado notoriamente en su concepción, gracias a la participación del Estado y de la Comunidad, a tal punto que se ha descrito y ha sido adoptado a nivel de todas las Instituciones de Educación Superior como “eje transversal a la vida universitaria”, bajo los principios de “formación integral, calidad de vida y construcción de comunidad”. Y en este sentido, a partir del 2004, se da inicio al PROYECTO DE MODERNIZACIÓN Y ADECUACIÓN DE LA INFRAESTRUCTURA DE SERVICIO DE LA DIVISIÓN DE BIENESTAR UNIVERSITARIO.

Finalmente, en la medida que se cumplan estos propósitos la Universidad estará causando un impacto favorable en la situación general de desarrollo de la sociedad de forma significativa, a través de la formación de nuevos profesionales que lleven consigo, no sólo la formación técnico científica propia de su disciplina, sino también los cambios de conducta y estilo de vida, logrados con el esfuerzo propio y el apoyo directo de Bienestar Universitario, a lo largo de su vida universitaria, donde el auto cuidado, la tolerancia, el respeto y cuidado del ambiente sean prioridad sobre otros intereses.

2.1.3 Marco Legal[7]

- Ley 30 de 1992 concepto de formación integral. Artículo 117: Impone la obligación del Bienestar Universitario Articulo 118; Se refiere al presupuesto del Bienestar Universitario Articulo 119: Fomento del deporte.

La ley 30 de 1992 dentro de sus objetivos expresa la profundidad que debe poseer la formación integral de los colombianos dentro de las modalidades y calidades de la educación superior, con el fin d capacitarlos en el cumplimiento de sus funciones profesionales, investigativas y de servicio social que requiere el país.

El título V "Régimen Estudiantil", capítulo III, de la ley 30 hace referencia específicamente al Bienestar Universitario mediante la implementación de programas de bienestar entendidos como el conjunto de actividades que se orientan al desarrollo físico, psico-afectivo, espiritual y social de los estudiantes, docentes y personal administrativo.

También el Consejo Nacional de Educación Superior (CESU) determina las políticas de bienestar universitario, y la creación de un fondo de Bienestar Universitario con recursos del Presupuesto Nacional y las entidades territoriales que puedan hacer aportes; este fondo es administrado por el Instituto Colombiano para el Fomento de la Educación Superior (ICFES), cada institución de igual manera destinará por lo menos 2% de su presupuesto para atender las necesidades del área de bienestar.

[7] http://www.fum.edu.co/snies/inst/bien/contque_es.html

2.2 ATENCION SOCIO-ECONOMICA

La División de Bienestar Universitario ofrece y mantiene servicios orientados a apoyar económicamente a los estudiantes de bajos recursos, para contribuir al mejoramiento de calidad de su vida.

Entre los programas que ofrecen y en los cuales está enfocado este proyecto están:

2.2.2 Servicio de Residencia estudiantiles

Este servicio está dirigido a estudiantes (hombres) de bajos recursos, provenientes de regiones apartadas de Santander y otros departamentos que requieran el apoyo de alojamiento y cumplan con los requisitos establecidos. La asignación se efectúa teniendo en cuenta la cantidad de habitaciones disponibles y el cumplimiento de los requisitos exigidos.

2.2.3 Auxiliatura Estudiantil de Sostenimiento Femenino

A partir del segundo semestre de 2007 se aprobó un subsidio de arrendamiento de vivienda dirigido a treinta (30) mujeres estudiantes cuyo lugar de origen y vivienda del núcleo familiar se encuentre fuera del área metropolitana de Bucaramanga. El pago es uno y medio (1½) SMMVL por semestre. Los requisitos son los siguientes: El valor de la matricula no superior a (½) SMMVL estudiante de tiempo completo, haber aprobado 11 créditos, encontrarse a paz y salvo por todo concepto, y no haber sido sancionado disciplinariamente por parte de la Universidad.

2.2.4 Auxilio Fondo Patrimonial

Conforme al acuerdo Superior N° 69 de 1997, la universidad creó el fondo patrimonial por el cual se manejan las donaciones que se recibe, y su rendimiento financiero se dirige a financiar la matricula a estudiantes de bajos recursos y al desarrollo de la educación, la ciencia y la tecnología.

2.2.5 Becas a Hijos y Cónyuges de Servidores

Conforme a disposiciones vigentes el Consejo Superior estableció un sistema de bienestar social para los servidores de la UIS, y sus beneficiarios: cónyuge o compañero(a) permanente, hijos e hijas de servidores de la UIS matriculados en programas presenciales de pregrado, académicos, administrativos y de proyección social de la Universidad. La universidad otorga un subsidio del 90% de la matrícula y como contraprestación, el estudiante se compromete a trabajar un total de 32 horas semestrales en labores académicas o administrativas según la necesidad de la unidad académica administrativa solicitante, sin que por ello se establezca vínculo laboral alguno.

2.3 DIAGRAMACION UML

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

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

Con el propósito de contextualizar las ventajas que ofrece la utilización del UML como herramienta de diseño, específicamente en el desarrollo de aplicaciones Web; es correcto afirmar que el éxito de un proyecto de esta índole depende en

gran medida de la comunicación entre el Analista (persona que se encarga de documentar los problemas o necesidades del cliente) y los desarrolladores (Diseñadores y Programadores). En este punto es evidente la importancia del uso del UML ya que permite dar forma de una manera convencional a los requerimientos del cliente a través de diagramas que representan el lenguaje de comunicación entre diseñador(es) y desarrollador(es).

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

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

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

- Los **Diagramas de Interacción** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado. Ej. diagramas de secuencia y colaboración.[8]

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

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

2.3.1 Diagrama de Casos de Uso

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

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

- **Actores:** Corresponden a los diferentes roles que los usuarios del sistema pueden representar. Cada rol debe ser único, es decir, distinguible de los demás, contando con un nombre específico y responsabilidades particulares al momento de interactuar con el sistema. No obstante, es necesario aclarar que un rol representa a un tipo o categoría de usuarios del sistema e incluso un usuario puede tener asignado más de un rol en el sistema. Un actor es usualmente identificado como se muestra la figura 1:

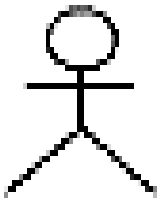

Figura 1. Actor

- **Caso de Uso:** Es una secuencia de transacciones relacionadas, ejecutadas por uno o más actores y el sistema en un diálogo determinado. En su conjunto, los casos de uso son los que describen todas las funcionalidades del sistema y cada uso de ellos está constituido por una secuencia de mensajes. En la figura 2 indica cómo se grafican los casos de uso en el diagrama.

Figura 2. Caso de Uso

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

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

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

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

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

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

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

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

- Relación de Inclusión: Señala cuando el comportamiento normal de un caso de uso por su naturaleza incorpora el comportamiento normal de otro.

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

La siguiente figura 3 ilustra los diferentes tipos de relaciones que pueden presentarse en un diagrama de casos de uso:

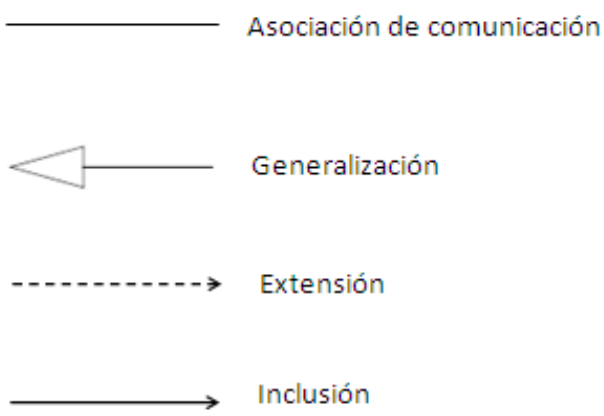

Figura 3. Tipos Relaciones Casos de Uso

2.3.2 Diagrama de Clases

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

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

- **Clases**

  Para entender el concepto de clase es imprescindible tener claridad acerca del concepto de objeto.
  "Un objeto puede ser una abstracción, concepto o cosa con límites bien definidos y con significado en el sistema"[9]
  Adicionalmente un objeto presenta ciertas características:

  - Estado: Es definido por los atributos que contiene el objeto y por sus posibles relaciones con otros objetos.

  - Comportamiento: Explica la funcionalidad de un objeto, señalando todas las operaciones que este puede realizar.

  - Identidad: Implica la unicidad de cada objeto así comparta el mismo estado con otro(s).

"Una clase es una descripción de un conjunto de objetos con las mismas propiedades (atributos), el mismo comportamiento (operaciones), las mismas relaciones entre objetos y la misma semántica"[4]. Los atributos o propiedades hacen referencia a valores concretos que pueden ser numéricos, alfabéticos o alfa-numéricos.
Una clase es representada en la figura 4:

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

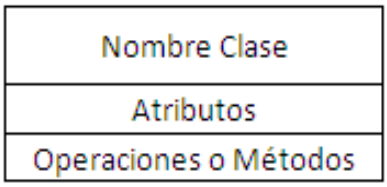

Figura 4. Representación de clase

❖ **Relaciones**

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

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

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

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

La herencia puede darse de dos formas:

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

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

La figura 5 exhibe las relaciones que pueden existir en el diagrama de clases, anteriormente descritas:

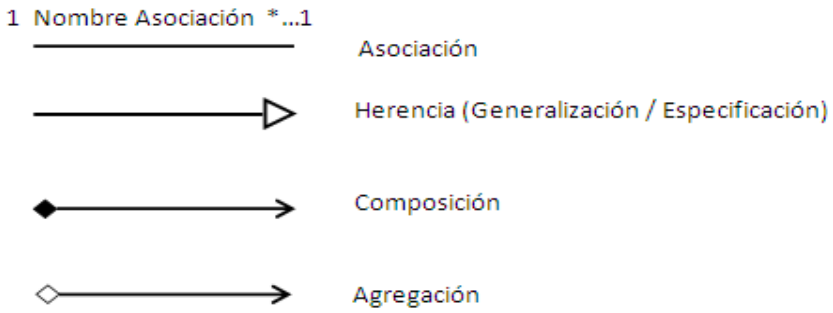

Figura 5. Tipos de Relaciones entre Clases

2.3.3 Diagrama de Secuencia

"Representa la interacción entre clases del modelo de estructuras estáticas, ordenadas temporalmente"[10], donde el Actor es el responsable de iniciarla en el momento que efectúa un mensaje u operación sobre el sistema. El diagrama de secuencia se lee de izquierda a derecha y de arriba abajo; cada caso de uso tiene asociado un diagrama de secuencia, sin embargo, puede tener más de uno dependiendo de los posibles comportamientos alternos por los que el caso de uso deba optar.

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

Los principales elementos que involucra un diagrama de secuencia se

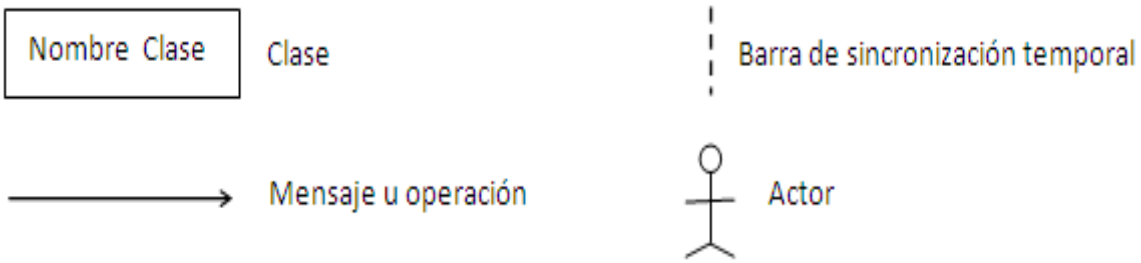

Figura 6. Elementos del Diagrama de Secuencia

muestran en la figura 6:

2.4 TECNOLOGIAS DE DESARROLLO DE APLICACIONES WEB

2.4.1 JAVA EE5

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

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

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

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

2.4.2 Seguridad

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

2.4.3 Componentes Java EE

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

- Aplicaciones cliente y Applets: Estos componentes se ejecutan en el cliente.

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

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

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

Comunicaciones del servidor Java EE

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

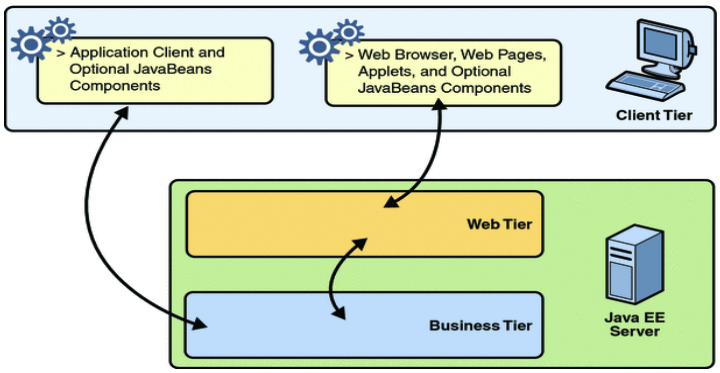

Figura 7. Comunicación del servidor

En caso de que la aplicación Java EE haga uso de un navegador o una aplicación cliente pesada se debe tener cuidado en la decisión de cuál de los dos utilizar para equilibrar la funcionalidad en el cliente y acercarse al usuario

para cargarle al servidor toda la funcionalidad que sea posible, esto facilita la distribución, despliegue y manejo de la aplicación.

2.4.4 Contenedores Java EE

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

2.4.5 Soporte para servicios web

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

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

La traducción de los datos a un flujo de datos estandarizado basado en XML es lo que hace que los servicios web y clientes escritos con las APIs de Java EE

puedan operar entre ellos totalmente. Esto no necesariamente significa que los datos transportados incluyan etiquetas XML porque el transporte de los datos puede ser texto plano, datos XML o cualquier tipo de dato binario como audio, vídeo, mapas, ficheros de programa, documentos CAD o lo que sea requerido.

2.4.6 Ensamblaje y despliegue de una aplicación Java EE

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

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

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

2.4.7 Empaquetado de aplicaciones

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

Un fichero EAR contiene, como muestra la figura, módulos Java EE y descriptores de despliegue.

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

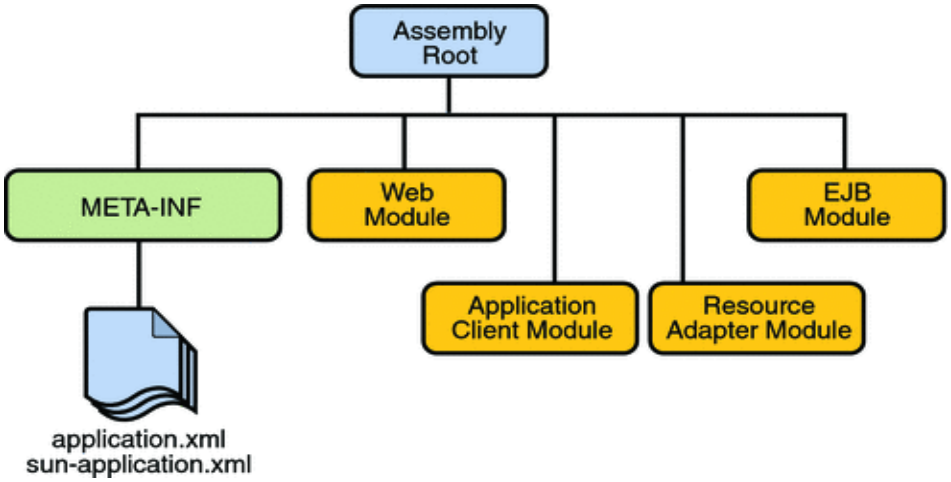

Figura 8. Estructura de un fichero EAR

2.4.8 Aplicaciones web

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

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

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

Desde la introducción de la tecnología de Servlets y JSP, han sido desarrollados marcos de trabajo para construir aplicaciones web interactivas. La figura 9 muestra estas tecnologías y sus relaciones.

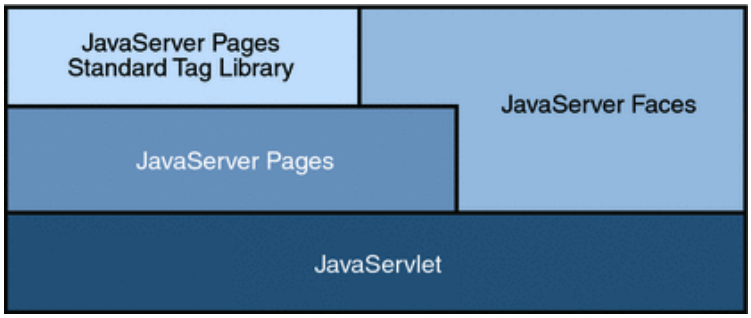

Figura 9. Tecnologías Java para aplicaciones web

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

2.4.8.1 Ciclo de vida de una aplicación web

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

crear y ejecutar una aplicación web es diferente que el utilizado para las clases Java autónomas.
El proceso para crear, desplegar y ejecutar una aplicación web puede resumirse como sigue:

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

2.4.9 Tecnología Servlet Java

2.4.9.1 ¿Qué es un Servlet?

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

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

2.4.9.2 Ciclo de vida de un servlet

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

1. Si una instancia del servlet no existe, el contenedor web:
   - Carga la clase servlet.
   - Crea una instancia de la clase servlet.

2. Inicializa la instancia del servlet llamando al método init.

3. Invoca el método service, pasando los objetos request y response.

4. Si el contenedor necesita quitar el servlet, este finaliza el servlet llamando el método destroy.

5. Inicializando un Servlet

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

2.4.9.3 Manteniendo el estado del cliente

Muchas aplicaciones necesitan que una serie de solicitudes de un cliente sean asociadas con otro. Las aplicaciones basadas en Web son responsables por

mantener este estado, llamado sesión, ya que HTTP no tiene estado. Para soportar las aplicaciones que necesitan mantener el estado, la tecnología Servlet de Java proporciona una API para manejo de sesiones y permite varios mecanismos para implementar sesiones.

- ❖ Accediendo a una sesión

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

2.4.9.4 Finalizando un servlet

Cuando un contenedor de servlet determina que un servlet debe ser eliminado del servicio, el contenedor llama al método destroy de la interface Servlet. En este método, usted libera todos los recursos que el servlet está utilizando y guarda cualquier estado persistente.
Todos los métodos de un servicio dado por un servlet deben ser completados cuando un servlet es eliminado. El servidor trata de asegurarse de esto llamando el método destroy solo luego de que todas las solicitudes de servicio han retornado o después de un período de gracia específico del servidor, lo que suceda primero. Si su servlet tiene operaciones que toman un tiempo largo en ejecutar, las operaciones pueden seguir ejecutándose luego de que el método destroy es llamado. Es preciso asegurar que todos los hilos manejan las solicitudes de clientes de forma completa realizando los siguientes procesos:

- Mantener la cuenta de cuantos hilos siguen ejecutando el método service.

- Proveer un apagado limpio con el método destroy notificando a los hilos que tardan mucho tiempo en ejecutase de la baja y esperar a que estos completen.

- Hacer un sondeo periódico de los métodos que tardan mucho en ejecutar y si es necesario, detener el trabajo, hacer limpieza y retornar

2.4.10 JAVA SERVER FACES

2.4.10.1 Características principales

La tecnología JavaServer Faces constituye un marco de trabajo (framework) de interfaces de usuario del lado de servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).
Los principales componentes de la tecnología JavaServer Faces son:

- Una API y una implementación de referencia para:

  - ✓ Representar componentes de interfaz de usuario (UI-User Interface) y manejar su estado.

  - ✓ Manejar eventos, validar en el lado del servidor y convertir datos.

  - ✓ Definir la navegación entre páginas.

  - ✓ Soportar internacionalización y accesibilidad.

  - ✓ Proporcionar extensibilidad para todas estas características.

1. Una librería de etiquetas JavaServerPages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

Este modelo de programación bien definido junto con la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado servidor. Con un mínimo esfuerzo, es posible:

- Conectar eventos generados en el cliente a código de la aplicación en el lado del servidor.

- Mapear componentes UI a una página de datos en el lado servidor.

- Construir una interfaz de usuario con componentes reutilizables y extensibles.

Como se puede apreciar en la figura, la interfaz de usuario que se crea con la tecnología JavaServer Faces se ejecuta en el servidor y se renderiza en el cliente.

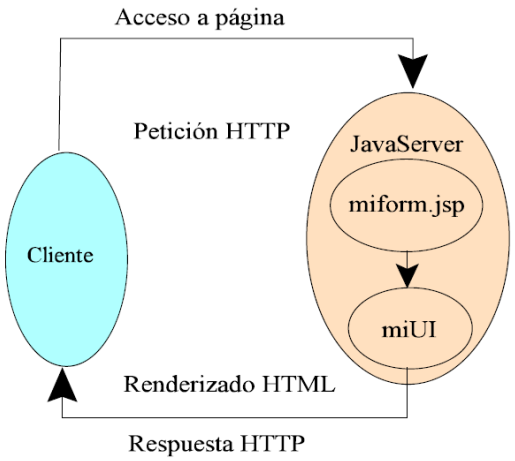

Figura 10. Diagrama de una aplicación jsf

En la figura 10, la página JSP (**miform.jsp**), especifica los componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por la tecnología JavaServer Faces. La UI de la aplicación web (representada por miUI en la figura 12) maneja los objetos referenciados por la JSP, que pueden ser de los siguientes tipos:

• Objetos componentes que mapean las etiquetas sobre la página JSP.

• Oyentes de eventos, validadores y conversores registrados y asociados a los componentes.

• Objetos del modelo que encapsulan los datos y las funcionalidades de los componentes específicos de la aplicación (lógica de negocio).

2.4.10.2 Beneficios de la Tecnología JavaServer Faces

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

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

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

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

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

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

2.4.10.3 ¿Qué es una aplicación JavaServer Faces?

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

• Componentes JavaBeans conteniendo datos y funcionalidades específicas de la aplicación.

• Oyentes de Eventos.

• Páginas, (principalmente páginas JSP).

• Clases de utilidad del lado del servidor, como beans para acceder a las bases de datos.

Además de estos ítems, una aplicación JavaServer Faces también tiene:

• Una librería de etiquetas personalizadas para implementar componentes UI en una página.

• Una librería de etiquetas personalizadas para representar manejadores de eventos, validadores y otras acciones.

• Componentes UI representados como objetos con estado en el servidor.

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

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

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

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

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

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

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

2.4.11 MODELO VISTA CONTROLADOR EN JSF

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

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

En la figura 11 se muestra un esquema del patrón de diseño MVC:

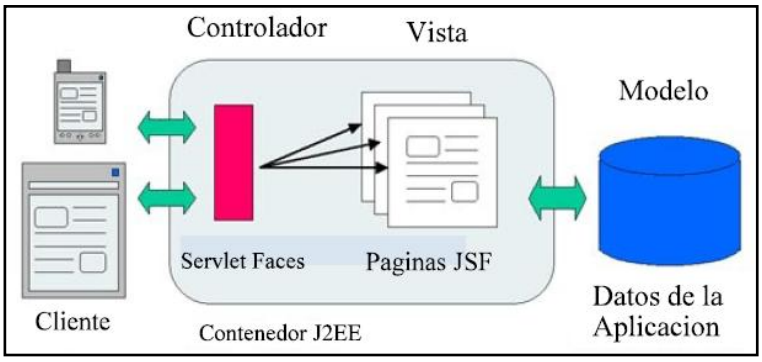

Figura 11. Patrón de diseño MVC

Este modelo de arquitectura presenta otras importantes ventajas:

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

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

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

2.4.11.1 Modelo

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

2.4.11.2 Vista

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

2.4.11.3 Controlador

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Entra en acción cuando se realiza alguna operación, ya sea un cambio en la información del modelo o una interacción sobre la Vista. Se comunica con el modelo y la vista a través de una referencia al propio modelo.

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

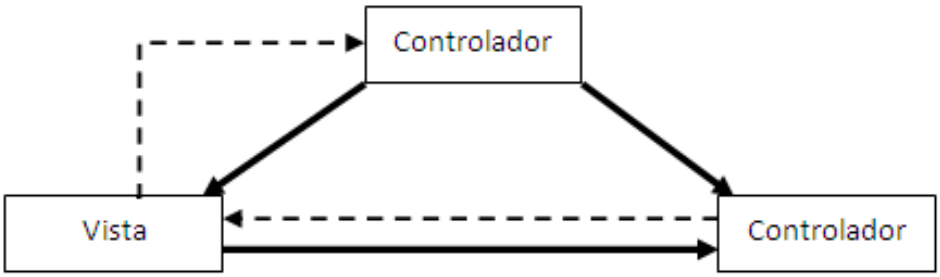

Figura 12. Modelo Vista-Controlador

En la figura 13 se puede observar las relaciones entre el Modelo, la Vista y el Controlador. Cabe destacar en la figura las líneas continuas que significan una relación directa como también las líneas discontinuas que implican una relación indirecta. También es importante tener en cuenta que aunque puede haber

cierta “referencia indirecta” entre el Modelo y la Vista, el primero sigue sin saber nada del segundo.

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

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

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

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

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

2.4.11.4 Ciclo de vida de una página JavaServer Faces

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

El cliente hace una petición HTTP de la página y el servidor responde con la página traducida a HTML. Sin embargo, debido a las características extras que ofrece la tecnología JavaServer Faces, el ciclo de vida proporciona algunos servicios adicionales mediante la ejecución de algunos pasos extras.
Los pasos del ciclo de vida se ejecutan dependiendo de si la petición se originó o no desde una aplicación JavaServer Faces y si la respuesta es o no generada con la fase de renderizado del ciclo de vida de JavaServer Faces. En la figura 13 se visualiza todo el ciclo de vida de una petición-respuesta de una página JSF:

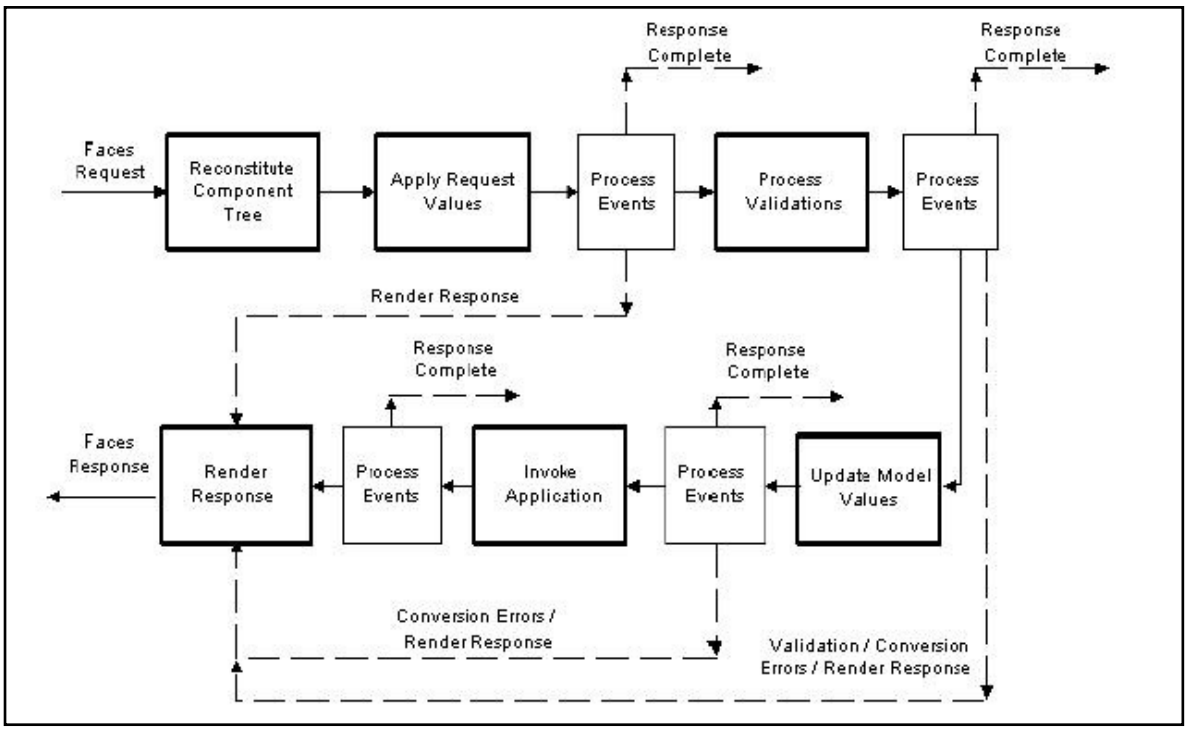

Figura 13. . Ciclo de vida petición-respuesta de una página JSF

2.4.11.5 Ciclo de Vida Estándar de Procesamiento de Peticiones

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

- **Reconstituir el árbol de componentes**

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

  - **Aplicar valores de la petición**

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

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

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

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

- **Procesar validaciones**

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

- **Actualizar los valores del modelo**

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

página mostrando los errores, similar a lo que sucede con los errores de validación.

- **Invocar Aplicación**

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

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

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

- **Renderizar la Respuesta**

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

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

2.4.12 SEAM

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

- **Un tipo de cosas**

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

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

- **Integrar JSF con EJB 3.0**

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

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

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

- **Integrar AJAX**

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

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

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

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

- **Biyección**

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

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

- **Preferir anotaciones a XML**

La comunidad java ha estado confundida sobre el tipo de meta información con la que cuenta la configuración; las anotaciones de java han logrado cambiar esto.
EJB 3.0 abarca las anotaciones y la configuración de excepción como la forma más fácil de proporcionar información al contenedor en forma declarativa. SEAM extiende las anotaciones de EJb3 con una serie de anotaciones para la administración del estado declarativo y demarcación del contexto declarativo.

- **La prueba de integración es fácil**

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

- **Las especificaciones técnicas no son perfectas**

Existen limitaciones en el ciclo de vida de JSF para las solicitudes GET que fija seam. Los autores de seam actualmente trabajan con expertos

JCP para asegurarse de que las correcciones pertinentes sean realizadas para la próxima versión.

- **Hay más de una aplicación web que sirve páginas HTML**

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

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

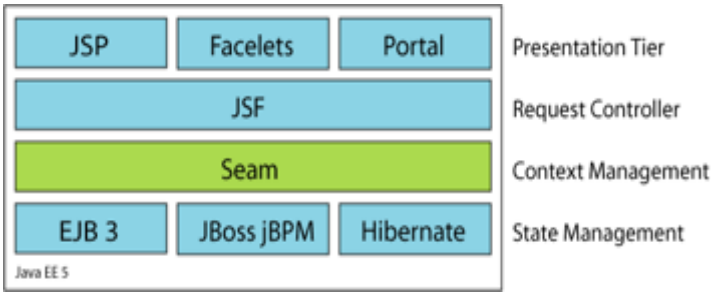

Figura 14. Framework Seam

2.4.12.1 EL MODELO DE COMPONENTES CONTEXTUALES

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

2.4.12.1.1 *Contextos de Seam*

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

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

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

- **Contexto de Página**: Este contexto permite asociar el estado de un componente con la instancia del llamado de una página, es decir, Seam crea el contexto al momento de direccionar una página. Es muy útil al momento de requerir listas de hacer clic (Combo Box) donde cada lista es devuelta por el cambio en los datos en el servidor. Los componentes perduran mientras se permanezca en la misma página.

- **Contexto de Conversación**: Es posiblemente el lugar ideal para guardar el estado de una aplicación ya que permite al desarrollador implementar casos de uso relativamente extensos o realizar transacciones relativamente largas. Una "conversación" es una unidad de trabajo desde el punto de vista del usuario, lo que abarca varias

peticiones e incluso varias transacciones con la base datos. La conversación mantiene su estado asociándolo a cada ventana en forma individual con el fin de evitar posibles colisiones entre conversaciones, ya que un usuario puede estar manejando múltiples conversaciones al mismo tiempo (por ejemplo en el caso de tener la misma página en más de una instancia del navegador). Las conversaciones también se conocen como “tareas”, en consecuencia, una tarea es una conversación significativa en términos de un proceso de negocio extenso y tiene el potencial para disparar una transición del estado de un proceso de negocio cuando este es satisfactoriamente culminado. Seam controla el estado de las conversaciones mediante un tiempo de espera que se puede configurar con el propósito de evitar el incremento de conversaciones inactivas de un usuario en sesión; dado el caso que un usuario abandone la conversación. Otra característica particular del contexto de conversación es la posibilidad de tener conversaciones anidadas; en otras palabras una conversación contenida dentro de otra mayor.

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

- **Contexto de Proceso de Negocio**: Se encuentra asociado con una ejecución de proceso de negocio largo que abarca múltiples interacciones entre múltiples usuarios lo que implica que el estado sea compartido, manejado y persistido por el motor BPM (Business Process Management). Seam carece de anotaciones para realizar la

demarcación de este contexto por lo que se debe manejar en forma externa utilizando un Lenguaje de Definición de Procesos.

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

2.4.12.1.2 Prioridad de búsqueda de los Contextos Seam

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

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

2.4.12.1.3 Componentes Seam:

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

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

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

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

Beans de Entidad: Los Beans de Entidad pueden ser ligados a una variable de contexto o a una función como componentes Seam. Como las entidades tienen una identidad de persistencia adicional a su identidad contextual, las instancias de entidad están explícitamente ligadas al código Java, en lugar de ser creadas implícitamente por el framework.

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

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

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

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

2.4.12.1.4 Modelo de Concurrencia

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

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

2.4.12.2 Eventos, interceptores y el manejo de excepciones

2.4.12.2.1 Eventos seam

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

- JSF eventos
- jBPM eventos de transición
- Seam acciones de página
- Seam impulsado en componentes eventos
- Seam contextuales eventos

2.4.12.2.2 Página de acciones

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

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

2.4.12.2.3 Página de parámetros

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

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

- No es posible especificar parámetros de la petición usada para redirigir la página.
- No es posible iniciar o finalizar las conversaciones de una regla.
- En la evaluación del método de retorno no es posible evaluar una expresión EL arbitraria.

2.4.12.2.4 Eventos impulsados por componentes

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

- **Eventos Contextuales**

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

- org.jboss.seam.validationFailed: invocada cuando falla la validación JSF.

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

❖ **Interceptores Seam**

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

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

❖ **Administrar excepciones**

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

2.4.12.2.5 Conversaciones y gestión de espacio de trabajo

- **Modelo de conversación seam**

Seam se propaga de forma transparente en el contexto de conversación mediante las devoluciones de datos JSF y las redirecciones. Si no se hace nada especial o no realiza una petición el contexto de la conversación no se propaga y se procesa en una nueva conversación temporal; este es el comportamiento deseado.
Si desea propagar la conversación seam a través de una solicitud non-faces se debe codificar explícitamente la conversación seam identificando el parámetro de la petición.

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

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

- **Conversaciones anidadas**

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

- La anidación de una conversación se inicia en un contexto dentro de la conversación externa, esta es considerada como la conversación padre.

- Los objetos cargados directamente en el contexto de la conversación anidada no afectan los objetos accesibles en la conversación padre.
- La inyección en un contexto de búsqueda en la primera conversación, busca el valor en el contexto de la conversación en curso, si no encuentra ningún valor continúa por la conversación en pila si la conversación es anidada.

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

  Las conversaciones pueden ser anidadas a cualquier profundidad arbitraria.

❖ **Iniciando conversaciones con la solicitud GET**

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

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

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

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

❖ **Una conversación de larga duración**

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

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

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

- Un evento contextual llamado org.jboss.seam.noConversationes levantado.

- Se registra un mensaje de alerta con el conjunto de claves org.jboss.seam.NoConversation.

- Si se define el usuario es redirigido a una página alternativa.

2.5 NAVEGACION EN SEAM

❖ **Modelo de navegación Stateless**

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

El modelo Stateless define un conjunto de salidas lógicas y salidas de nombre de un evento directamente a la página resultante de la vista. Las reglas de navegación son totalmente ajenas a cualquier estado en poder de la aplicación aparte de la página fuente del evento. Esto implica que

los métodos de acción oyente (actionlistenermethods) deben en algunos escenarios tomar decisión con respecto al flujo de página, ya que sólo tienen acceso al estado actual de la aplicación.

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

Una ventaja de implementar este tipo de navegación radica en utilizar reglas simples, de ser flexible, y además, de permitir el retorno de IDs de vista desde los métodos de acción oyente. No obstante, la paginación Stateless no soporta procesos de negocios complejos.

- **Modelo de Navegación Stateful**

Define un conjunto de transiciones entre un conjunto de estados lógicos y estados de nombres de la aplicación. Este modelo está orientado a los procesos de negocio en el que es posible establecer el flujo producido por cualquier interacción del usuario, en su totalidad, con la definición del flujo de página de JPDL e incluso escribir métodos de acción oyentes que desconocen por completo el flujo de la interacción.

JPDL es un lenguaje xml simple y de fácil lectura, el cual es instaurado por el Motor de Gestión de procesos de Negocios de Jboss (JBPM). JPDL logra resolver los siguientes problemas:

- Definición del flujo de página involucrado en complejas interacciones de usuario (definir el flujo de página de una conversación en particular).

- Definición de los procesos de negocio globales (cuando los procesos de negocios involucran múltiples conversaciones con múltiples usuarios simultáneamente).

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

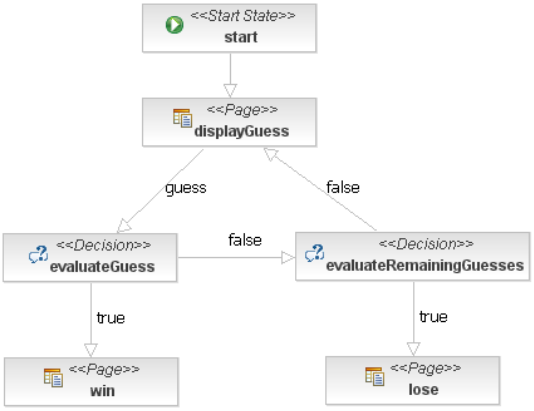

Figura 15. Modelo de Navegación Stateful

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

2.6 SEAM Y EL MAPEO OBJETO-RELACIONAL

Seam provee un extenso soporte para las Arquitecturas de Persistencia más importantes de Java: (a) Hibernate 3 y (b) Java Persistence API presentada por EJB 3.0. Entender cuál es la relación de Hibernate 3 y EJB 3.0 con Seam, o

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

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

- **Transacciones gestionadas por Seam**

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

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

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

  - Usar el contexto de persistencia extendido en el ámbito de la conversación, en lugar de asociarlo a la transacción.

- Usar dos transacciones por solicitud: la primera abarca desde la restauración de la fase de vista hasta el final de la invocación de la fase de aplicación; y la segunda transacción abarca la fase de devolución de la respuesta.

❖ **Sincronización de la Transacción**

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

❖ **Contextos de Persistencia gestionados por Seam**

Cuando se utiliza Seam en otro ambiente diferente a Java EE5 no se puede confiar el manejo del ciclo de vida del contexto de persistencia al contenedor, e incluso, aun trabajando con la plataforma Java EE5 la propagación del contexto de persistencia entre los componentes es difícil y propensa a errores.
De cualquier manera se necesita un “Contexto de persistencia administrado” (especificado por JPA) o una “Sesión administrada” (de Hibernate) en los componentes.

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

Los contextos de persistencia que son gestionados por Seam son extremadamente eficientes en ambientes distribuidos, pues Seam es capaz

de realizar una optimización de la especificación EJB 3.0 y es que los contenedores se puedan usar para administrar contextos de persistencia extendidos. Seam también puede hacerse cargo de los errores en el contexto de persistencia extendido con la ventaja de llevar a cabo esta tarea en forma transparente en donde no existe la necesidad de replicar cualquier estado contexto de persistencia entre los nodos.

2.7 EL MARCO DE APLICACIONES SEAM

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

- **Objetos Principales**

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

- **Controlador de objetos**

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

2.8 ALMACENAMIENTO EN CACHÉ DE SEAM

En la mayoría de las aplicaciones empresariales, la base de datos es el principal cuello de botella como también es la capa menos escalable en el

entorno de ejecución. En conclusión es casi imposible que una aplicación sea escalable mientras la interacción con la base de datos sea ostensible. Por lo tanto la escalabilidad de cualquier aplicación se puede favorecer si se disminuyen las transacciones directas con la base de datos, y esa es la principal misión de la caché.

- **Almacenamiento en Caché multi-capa**

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

- **Caché de la Base de Datos**

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

- **Caché de segundo nivel**

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

- **Caché a nivel de Contexto**

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

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

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

- **Caché a nivel de JSF**

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

2.9 Hibernate y ORM

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

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

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

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

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

2.10 JPA

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

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

2.11 JPQL

Java PersistenceQueryLanguage el cual es el lenguaje de consultas que se usará dentro de la aplicación. Con él, el desarrollador jamás se refiere a las entidades directamente al momento de hacer las consultas, sino a los objetos mismos que fueron mapeados. Si el diseño de la BD cambiara, sólo habría que modificar las anotaciones de las entidades. El resto del código quedaría intacto

y seguiría funcionando tal y como antes. Esta facilidad permite optimizar las bases de datos cuando se considere necesario, migrar a bases diferentes, o sencillamente corregir diseños defectuosos con un esfuerzo mínimo.

CAPITULO 3

3. METODOLOGIA DE DESARROLLO

3.1 METODOLOGÍA DE DESARROLLOCICLO 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 de para el Manejo de Beneficios que Ofrece el Bienestar Universitario de la Universidad Industrial de Santander.

- **Análisis de Requerimientos**

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

En el análisis de requerimientos se especificó junto con los funcionarios del Bienestar Universitario, la función y comportamiento que deben tener los programas, se indicó la interfaz con otros elementos del sistema y se establecieron los estándares de diseño que debe cumplir el sistema.

El análisis de requerimientos permitió la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministró los medios para valorar la calidad de los programas.

Es en esta etapa, donde se realizaron reuniones periódicas con los funcionarios de la división de Bienestar Universitario, para que fueran ellos los

que definieran las características del software que se desarrolló y se adaptó plenamente a las exigencias de los procesos que realizan.

3.2 Diseño

El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo).

En esta etapa de diseño se hizo una traducción de los requisitos a una representación del software donde se pudo evaluar su calidad antes de que comenzara la codificación.

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

3.3 Implementación de la Aplicación:

En esta etapa se procedió a generar el software que se había 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 y la plataforma Informix como motor de base de datos.

3.4 Pruebas del software

Las pruebas del software son los procesos que permitieron verificar la calidad de un producto software y el cumplimiento de los requerimientos establecidos en la fase de análisis de requerimientos.

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

Una vez generado el código, comenzamos 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 fueran probadas y asegurando que la entrada definida produce los resultados esperados.

Las pruebas fueron de carácter permanente a lo largo del desarrollo del proyecto por parte del equipo de trabajo, y se hizo un periodo de tiempo para que los usuarios finales interactuaran con la aplicación y detectaran posibles ajustes.

- **Ajustes**

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

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

3.5 Modelo de construcción por prototipos.

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

A continuación se observa la estructura del MODELO:

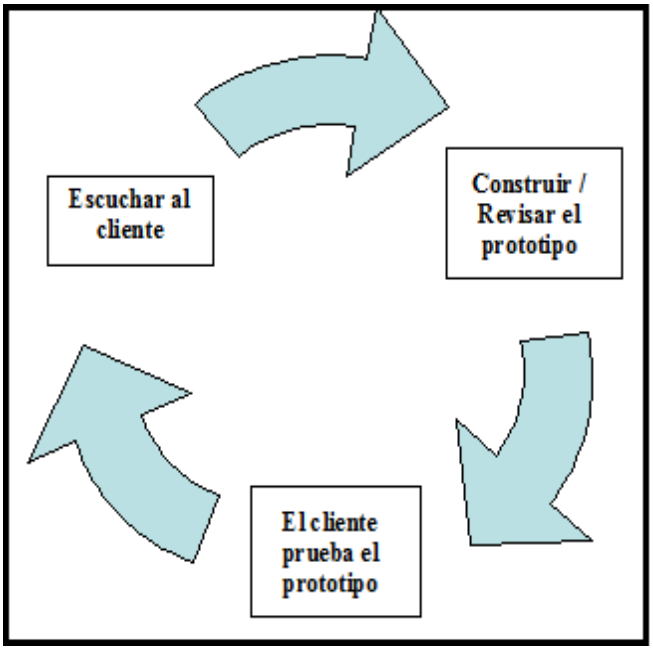

Figura 16. Modelo Construcción Por Prototipos

3.6 Estructura

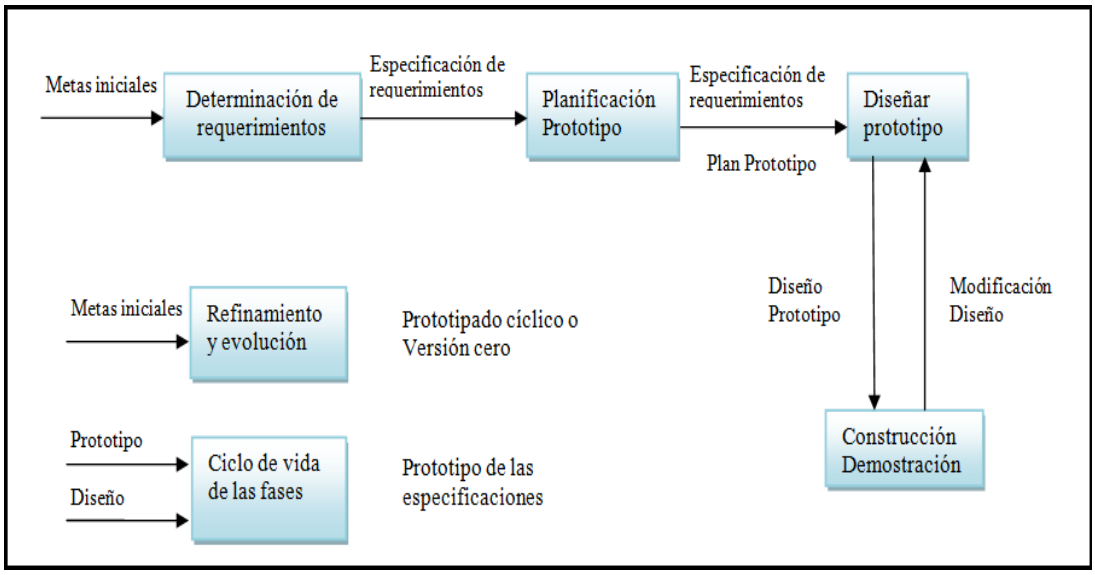

Figura 17. Estructura

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

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

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

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

3.7 Aplicación de la metodología

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

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

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

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

3.8 LEVANTAMIENTO DE REQUERIMIENTOS

Se presentan los requerimientos generados después de las primeras reuniones con el cliente, en donde se pudo abstraer las ideas principales de lo que se quiere alcanzar durante el desarrollo del proyecto, que sirvió como base para las primeras etapas del análisis y diseño del nuevo sistema de información para el manejo de los beneficios ofrecidos por Bienestar.

3.8.1 Descripción General.

El sistema de Información de Beneficios permite:

1. Definición de los beneficios juntos con los requisitos que deben cumplir los estudiantes que desean ser beneficiarios

2. Inscripción a los beneficios ofrecidos por la Universidad a través de BU. Realizando una corroboración del cumplimiento de los requisitos exigidos para poder acceder a ellos.

3. Inscripción extemporánea, realizada por el rol de administrador del sistema de Información.

4. Cada Beneficio podrá ser modificado, en cada uno de los parámetros que lo componen, tales como requisitos, tipo de contraprestación, etc..

5. Realizar una preselección de los estudiantes a los cuales se les podría asignar el beneficio solicitado, teniendo en cuenta los requisitos exigidos y la prioridad asignada a dichos requisitos.

6. Los estudiantes preseleccionados pueden programar la fecha y hora de la entrevista socioeconómica de acuerdo a la programación establecida por BU.

Actualmente se han definido para el Sistema de Información 4 roles de acuerdo a las funcionalidades implementadas. (Administrador, Estudiante, Jefe de Unidad y entrevistador).

3.9 Diagramas UML

Dentro del proceso de análisis y diseño del sistema de información se generaron los diagramas UML, que se presentarán a continuación, es de destacar que durante el transcurso del proyecto el diseño se fue adaptando a medida que se fue perfeccionando el prototipo inicial.

Aquí se presentarán los esquemas básicos de cada uno de los diagramas que se generaron, producto del diseño, con el fin de ilustrar el fondo y la forma como fueron producidos.

3.9.1 Diagrama de Casos de Uso.

- **Identificación de los Actores**

  - **Estudiante:** Estudiante de pregrado que se encuentra activo en la universidad y está interesado en obtener algún beneficio ofrecido por BU.

  - **Administrador:** El administrador del sistema es la persona autorizada por la División de Sistema de Información y Bienestar Universitario para el manejo del sistema.

  - **Entrevistador:** El profesional a cargo de las entrevistas.

  - **Jefe de Unidad Académico- administrativas**: Es la persona a cargo de la unidad o unidades tanto académicas como administrativas de la Universidad.

- **Casos de uso por Actor**

Actor: Estudiante

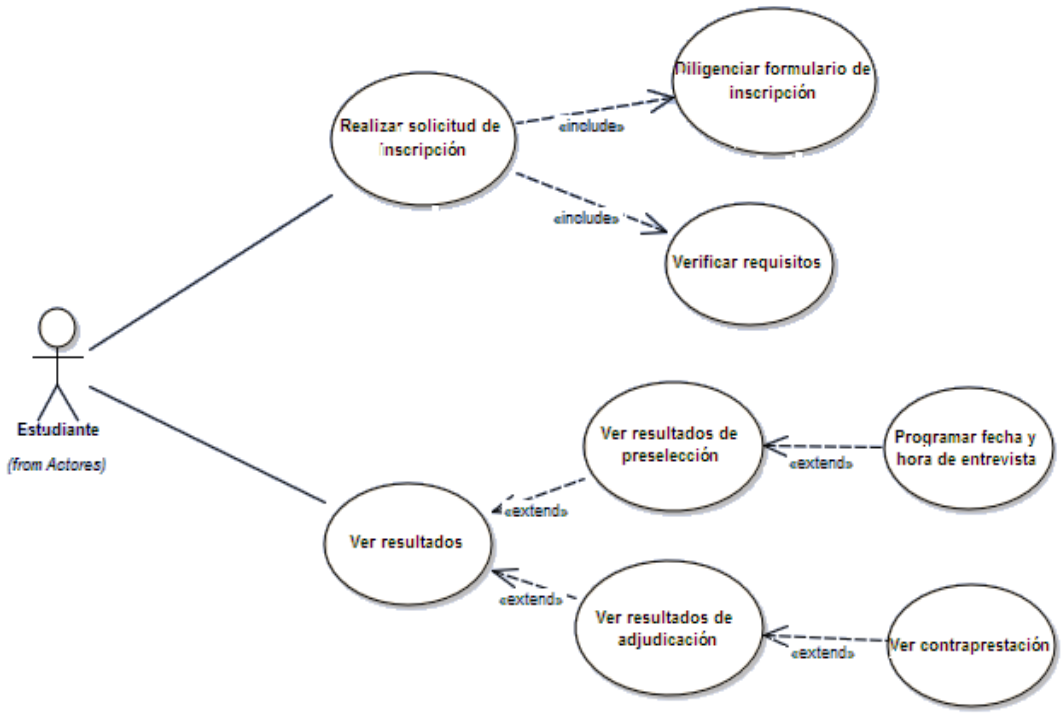

Figura 18. Caso de Uso asociado con Estudiante

Se identifican dos casos de uso asociados a estudiante

- Realizar solicitud de inscripción
- Ver resultados

- **Caso de uso:** Realizar solicitud inscripción

El sistema de información permite al estudiante realizar la inscripción en línea a cualquiera de los beneficios ofrecidos por BU, facilitando a los estudiantes preseleccionados la escogencia de las fechas y horas para la realización de la entrevista socioeconómica.

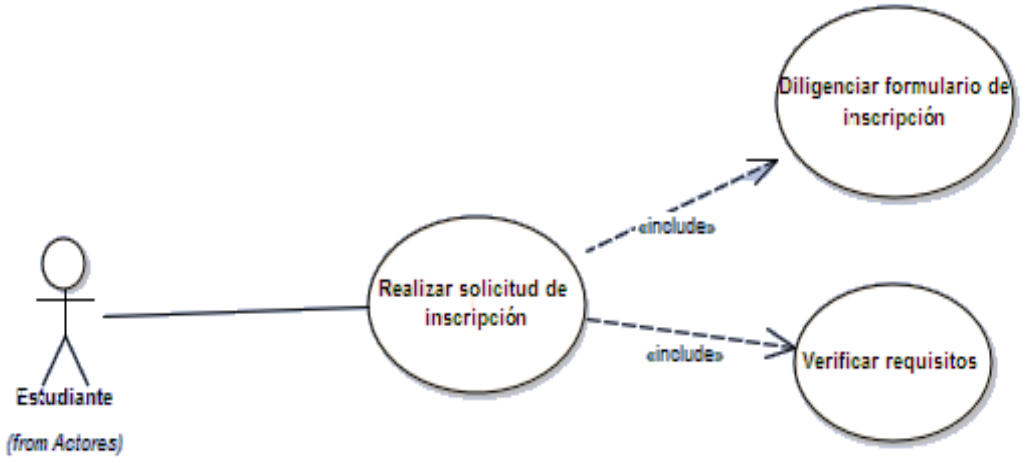

Figura 19. Caso de Uso: Registrar Solicitud de Inscripción

Tabla 1 Solicitud Inscripción

CASO DE USO	**REGISTRAR SOLICITUD DE INSCRIPCIÓN**	
Objetivo	El estudiante solicita la inscripción a uno o varios beneficios ofrecidos por Bienestar Universitario	
Actores	Estudiante	
Pre condiciones	Ser estudiante activo de la Universidad Industrial de Santander, en caso de cumplir con los requisitos, realizar la inscripción	
Post Condiciones	Cumplir con los requisitos para ser inscrito al Beneficio	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresa al Sistema desde la página de la universidad con un usuario y una contraseña	
		Le direcciona a la página donde se ofrece la inscripción al beneficio

	Escoge el beneficio o los beneficios a los cuales desea inscribirse con su respectivo orden de prioridad	
		Presenta la opción de Registrar o Cancelar
	Registra la Solicitud	
		Si el estudiante cumple con los requisitos necesarios para ser aspirante al beneficio, actualiza la información para la entrevista o visita por parte del profesional a cargo del beneficio, en caso contrario le muestra los requisitos que no cumple para ser beneficiario.
	Guardar en el caso de que si cumpla con los requisitos, o regresar en el caso contrario	
		Se registra la inscripción y se actualizan los registros en la base de datos.
Variaciones	En el caso que el estudiante no cumpla con los requisitos necesarios para ser inscrito al beneficio el sistema le indicará cuales son aquellos que no cumple	
Extensiones	El sistema accede a la base de datos, y trae los datos necesarios para corroborar si el estudiante cumple o no con los requisitos establecidos	

- **Caso de uso:** Ver resultados

El estudiante puede ver los resultados tanto de la preselección como de la adjudicación del beneficio.

Figura 20. Caso de Uso: Ver resultados

Posterior a la inscripción al beneficio, si el estudiante ha cumplido con los requisitos necesarios para ser preseleccionado y está entre los cupos que se encuentran disponibles, el estudiante podrá programar la fecha y hora de la entrevista socioeconómica si esta es requisito para la adjudicación al beneficio. Después de realizadas las entrevistas se darán los resultados de los estudiantes que serán beneficiarios, los estudiantes inscritos podrán ver los resultados del proceso de selección. Si el estudiante ha sido seleccionado podrá mirar la respectiva contraprestación del beneficio al cual se inscribió.

Actor: Administrador

Figura 21. Casos de Uso asociados con Administrador

Sin duda alguna éste es el actor que mayor interacción tiene con el sistema ya que es el encargado directo de la administración del beneficio que incluye: la creación del beneficio, los requisitos establecidos para el proceso de selección,

proceso de selección y publicación de resultados entre otros. Se identifican los siguientes casos de uso:

- Administrar Beneficio
- Preseleccionar Beneficiarios
- Entrevista socioeconómica
- Adjudicar beneficio
- Controlar beneficios y beneficiados
- Generar consulta

- **Caso se uso: Administrar beneficio**

El administrador puede crear, listar o hacer mantenimiento a cualquiera de los beneficios ofrecidos por Bienestar Universitario

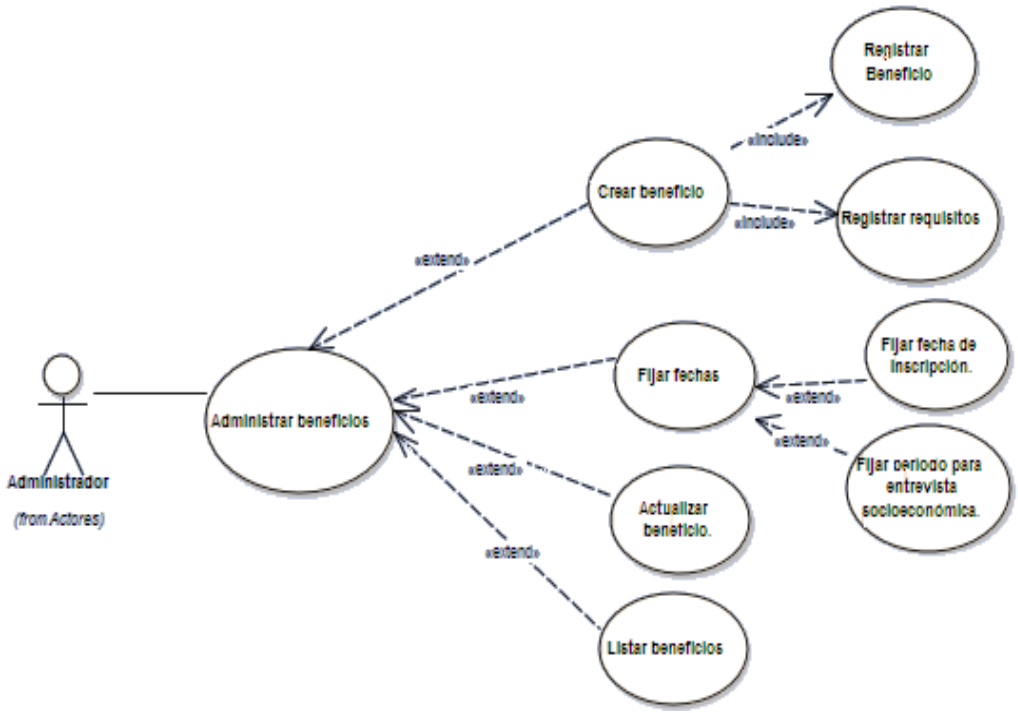

Figura 22. Caso de Uso: Administrar Beneficios

Tabla 2 Administrar Beneficio

CASO DE USO	**ADMINISTRAR BENEFICIO**	
Objetivo	Crear, actualizar los beneficios y fijar las fechas correspondientes de inscripción y entrevista de cada uno.	
Actores	Administrador	
Pre condiciones	Validarse como Administrador del Sistema	
Post Condiciones	Tener en cuenta las diferentes resoluciones relacionadas con cada beneficio, publicadas y actualizadas por los entes encargados de su regulación	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresar al módulo de Administración	
		Le presenta el menú con las diferentes acciones para la creación, modificación o eliminación del beneficio y sus respectivos requisitos, cupos y contraprestaciones relacionadas a los beneficios
	Escoge un ítem del menú	
		Le presenta la opción de Crear registros , Editar o Eliminar los datos existentes.
	Escoge la opción que desea realizar, según las presentadas por el sistema	
		Se actualizan los registros en la base de datos y muestra los cambios realizados en pantalla
Variaciones	La creación así como la modificación de un beneficio está regulada por parámetros como el periodo y el año según el calendario actual además de la duración de la entrevista.	
Extensiones	El sistema no permite eliminar el beneficio si este tiene datos asociados.	

- **Caso se uso: Preseleccionar beneficiados**

El administrador realiza las inscripciones extemporáneas si las hay y posterior a esto efectúa la preselección de los inscritos al beneficio

Figura 23. Caso de Uso: Preseleccionar beneficiados

Tabla 3 Preseleccionar Beneficiados

CASO DE USO	PRESELECCIONAR BENEFICIADOS	
Objetivo	Realizar la preselección de los inscritos al beneficio como candidatos a ser beneficiarios	
Actores	Administrador	
Pre condiciones	Validarse como Administrador del Sistema	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresar al módulo de Administración	
		Le presenta el menú con las diferentes acciones: realizar inscripción extemporánea o efectuar el proceso de preselección
	Escoge un ítem del	

	menú	
		Inicialmente en la inscripción extemporánea se inscribe los estudiantes que no alcanzaron en el período determinado por el administrador para la inscripción en línea, posterior a esto se efectúa la preselección donde aparece un listado en el cual se señalan los estudiantes que cumplen los requisitos y están dentro de los cupos establecidos.
	Se le da la opción de imprimir preseleccionados	
Extensiones	La lista de los preseleccionados está ordenada de acuerdo a la prioridad de los requisitos establecidos para cada beneficio	

- **Caso de uso: Entrevista socioeconómica**

El administrador puede ver las fechas y horas de las entrevistas escogidas por los preseleccionados, asignar los entrevistadores correspondientes al período actual

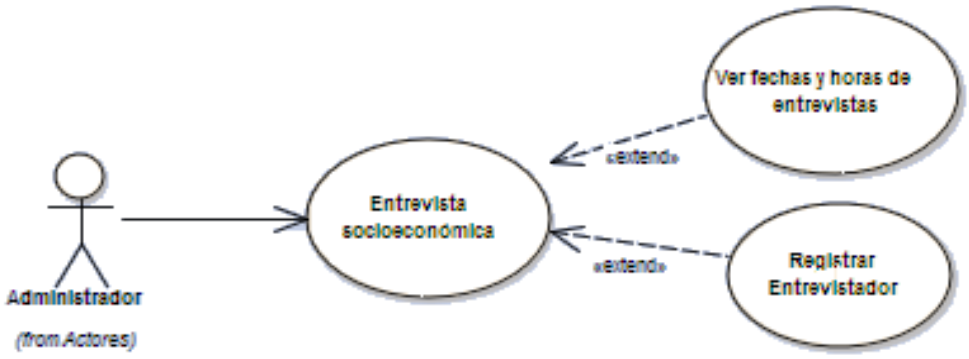

Figura 24. Caso de Uso: Entrevista socioeconómica

- **Caso se uso: Adjudicar el beneficio**

El administrador puede adjudicar el beneficio a los estudiantes de acuerdo a:

- Estudiantes que ya gozaban de este beneficio

- Estudiantes inscritos que serán los nuevos beneficiados

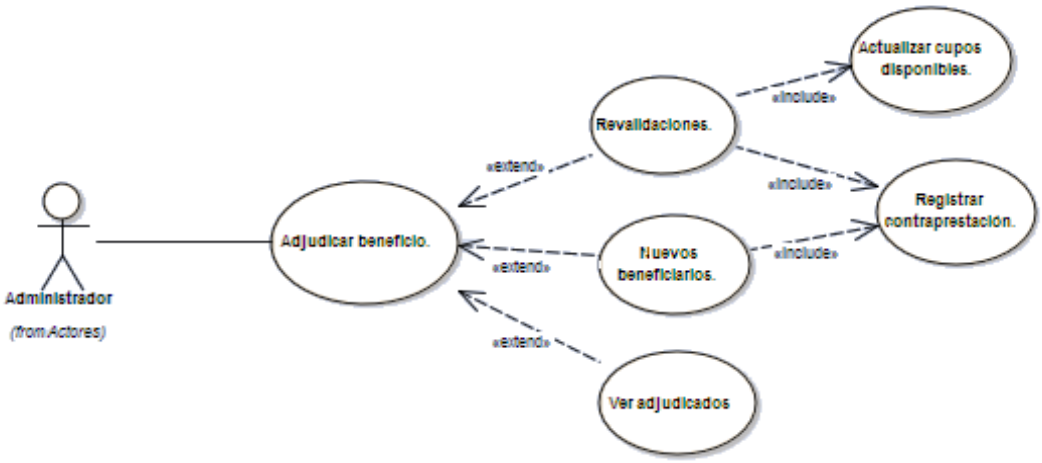

Figura 25. Caso de Uso: Adjudicar beneficio

Tabla 4 Adjudicar Beneficio

CASO DE USO	ADJUDICAR BENEFICIO
Objetivo	Adjudicar Beneficios a: • Realizar las revalidaciones correspondientes • Adjudicar el beneficio a los estudiantes seleccionados • Casos especiales • Asignación de la contraprestación relacionada con cada beneficio
Actores	Administrador

Pre condiciones	Validarse como Administrador del Sistema	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresar al módulo de Administración	
		Le presenta el menú con las diferentes acciones de adjudicación, revalidación y caso especial
	Escoge un ítem del menú	
		En el caso de adjudicación le muestra al administrador las lista con los beneficiados indicando el tipo de adjudicación
	Escoge la opción que desea realizar, según las presentadas por el sistema adjudicar o cancelar	
	Se actualizan los registros en la base de datos.	

- **Actor: Entrevistador**

Es el profesional a cargo de las entrevistas. Se identifican los siguientes casos de uso:

- Consultar entrevista
- Realizar entrevista

El entrevistador puede consultar las fechas con su respectivo horario en las cuales tiene programada las entrevistas que están a su cargo, realizará la

entrevista, emitirá un concepto y validará la información socioeconómica necesaria para adjudicar el beneficio.

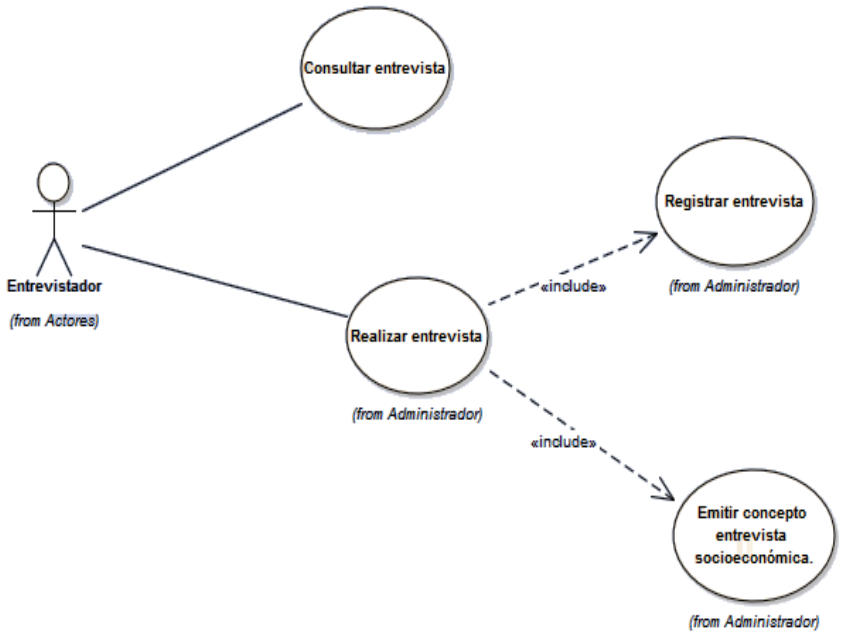

Figura 26. Casos de Uso asociados con Entrevistador

- **Actor: Jefe de Unidad**

Es la persona a cargo de una unidad ya sea académica o administrativa. Se identifican los siguientes casos de uso:

- Solicitar auxiliares
- Registrar Cumplimiento Contraprestación

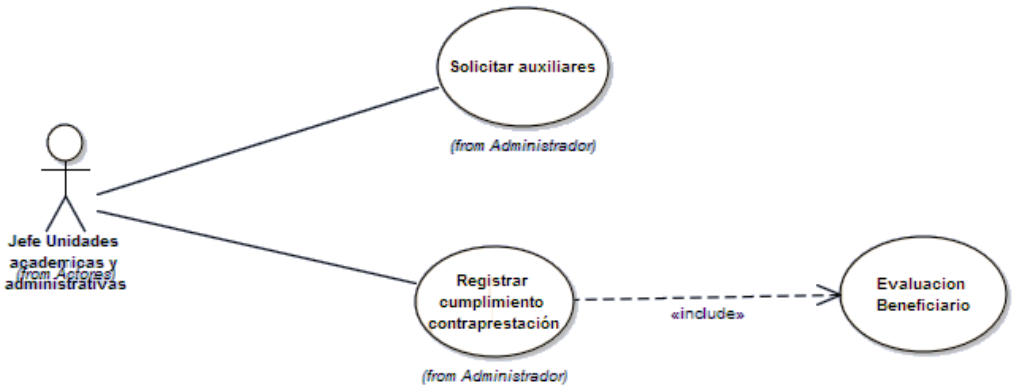

Figura 27. Casos de Uso asociados con Jefe de Unidad

3.9.2 Diagramas de Clases

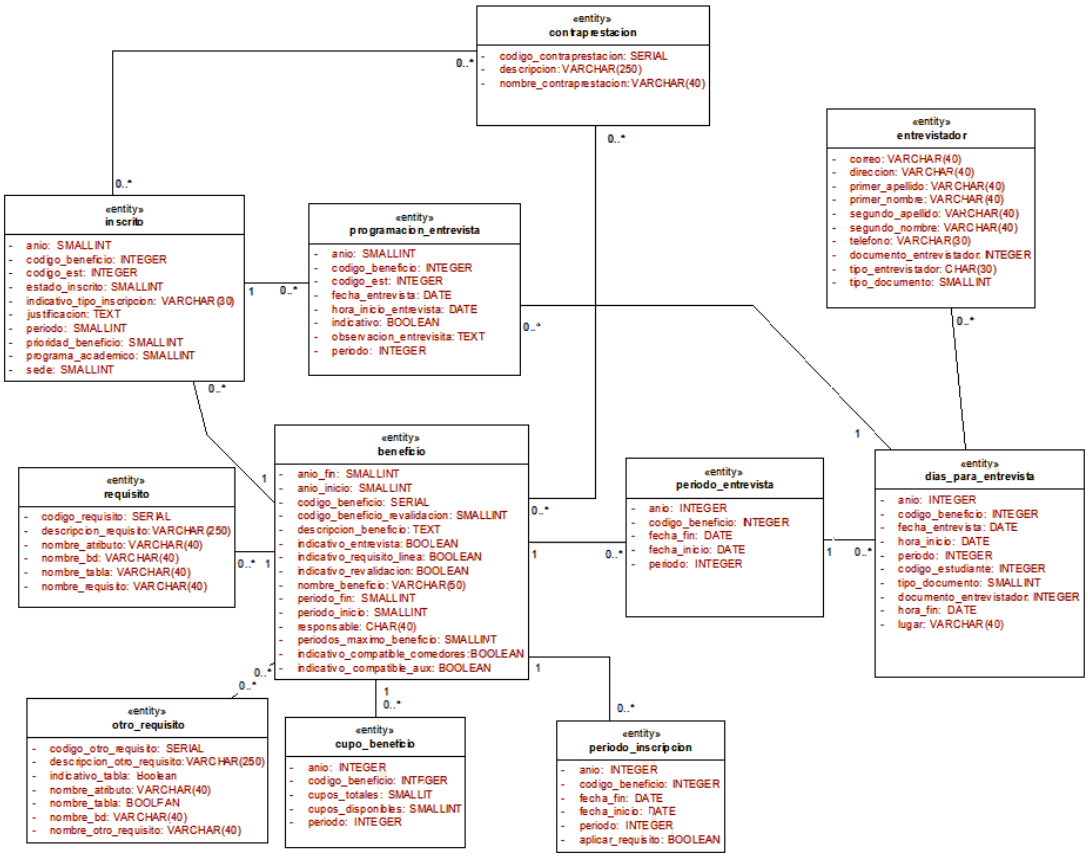

Figura 28. Diagrama de Clases

- **Clase beneficio:**

Relaciona los beneficios ofrecidos por parte de Bienestar Universitario para los estudiantes activos

Tabla 5 Atributos de la Clase Beneficio

Nombre	Tipo de datos	Descripción
anio_fin	Integer	Fecha de Fin del Beneficio.
anio_inicio	Integer	Fecha de Inicio del Beneficio.
codigo_beneficio	Serial	Código del Beneficio ofrecido por Bienestar Universitario para los estudiantes activos.
codigo_beneficio_revalidacion	Smallint	Código del beneficio con el cual se revalida este beneficio. Este beneficio que revalida debe tener requisitos de menor exigencia que el beneficio principal
descripcion_beneficio	Text	Descripción del Beneficio ofrecido por Bienestar Universitario para los estudiantes activos
indicativo_entrevista	Boolean	Indica si el beneficio tiene como requisito la entrevista socioeconómica
indicativo_requisito_linea	Boolean	Indica si al beneficio se le aplicará los requisitos en línea o no
indicativo_revalidacion	Boolean	Indica si a el beneficio aplica la revalidación o no
nombre_beneficio	Varchar	Nombre del beneficio ofrecido por Bienestar Universitario para los estudiantes activos
periodo_inicio	Smallint	Inicio de vigencia del período del beneficio
periodo_fin	Smallint	Fin de vigencia del período del beneficio
indicativo_ compatible_comedores	Boolean	Indica si el beneficio es compatible con el de comedores
Responsable	Char	Responsable a cargo del Beneficio ofrecido por Bienestar

		Universitario
indicativo_ compatible_aux	Boolean	Indica si el beneficio es compatible con una auxiliatura

- **Clase inscrito:**

Relaciona los estudiantes preseleccionados posteriormente a la inscripción y aplicación de los requisitos exigidos por cada beneficio.

Tabla 6 Atributos de la Clase inscritos

Nombre	Tipo de datos	Descripción
Anio	Smallint	Año actual
Periodo	Smallint	Período actual del calendario académico en el que se encuentra el beneficio
codigo_beneficio	Integer	Código del Beneficio al cual el estudiante esta inscrito.
codigo_est	Integer	Código del estudiante preseleccionado
estado_inscrito	Smallint	1 Preseleccionado, 2 revalidado, 3 no cumple
indicativo_tipo_inscripcion	Boolean	Indica si el estudiante fue preseleccionado para el beneficio por proceso normal o caso especial
Justificación	Text	Se debe registrar las causas por las cuales es preseleccionado por caso especial
prioridad_beneficio	Smallint	Indica la prioridad que el estudiante le asigna a los beneficios solicitados
programa_academico	Smallint	Programa académico en el que actualmente esta activo el estudiante preseleccionado
Sede	Smallint	Sede de la Universidad Industrial de Santander donde se encuentra activo el estudiante preseleccionado

Clase programacion_entrevista:

Relaciona la programación de las fechas establecidas por los preseleccionados

Tabla 7 Atributos de la Clase programación_entrevista

Nombre	Tipo de datos	Descripción
Anio	Smallint	Año actual
Periodo	Smallint	Período actual del calendario académico en el que se encuentra el beneficio
codigo_beneficio	Integer	Código del Beneficio al cual el estudiante esta inscrito.
codigo_est	Integer	Código del estudiante preseleccionado
fecha_entrevista	Date	Fecha para la cual el estudiante preseleccionado programó su entrevista
hora_inicio_entrevista	Date	hora de inicio de la entrevista programada por cada uno de los estudiantes preselecionados
Indicativo	Boolean	Indica si el estudiante preseleccionado es apto o no para que se le sea adjudicado el beneficio
observacion_entrevista	Text	Observación por parte del entrevistador

- **Clase requisito:**

Relaciona el tipo de requisito con el beneficio, es decir, si es de tipo financiero o académico.

Tabla 8 Atributos de la Clase requisito

Nombre	Tipo de datos	Descripción
codigo_requisito	Serial	Código del requisito, si es académico o financiero
descripcion_requisito	Varchar	Describe con más detalle el tipo de requisito relacionado con cada beneficio
nombre_atributo	Varchar	Nombre que relaciona el requisito con beneficio
nombre_bd	Varchar	Nombre con el cual se encuentra el requisito en la base de datos
nombre_tabla	Varchar	Relaciona la tabla que se encuentra en la base de datos del requisito de cada beneficio
nombre_requisito	Varchar	nombre del tipo requisito, si es academico o financiero

- **Clase otro_requisito:**

Requisitos que no son ni académicos ni financieros de cada Beneficio

Tabla 9 Atributos de la Clase otro_requisito

Nombre	Tipo de datos	Descripción
codigo_otro_requisito	Serial	Requisitos que no son ni académicos ni financieros de cada Beneficio
descripcion_otro_requisito	Varchar	Describe el requisito booleano exigido para realizar la inscripción al beneficio
nombre_atributo	Varchar	Nombre que relaciona el requisito con beneficio
nombre_bd	Varchar	Nombre con el cual se encuentra el requisito en la base de datos
nombre_tabla	Varchar	Relaciona la tabla que se encuentra en la base de datos del requisito de cada beneficio

nombre_requisito	Varchar	nombre del tipo requisito, si es académico o financiero
indicativo_tabla	Boolean	Indica si el estudiante aparece o no, en la BD con deudas, o sanciones, etc...

- **Clase cupo_beneficio:**

Relaciona el número de cupos habilitables para la adjudicación del Beneficio

Tabla 10 Atributos de la Clase cupo_beneficio

Nombre	Tipo de datos	Descripción
Anio	Smallint	Año actual
Periodo	Smallint	Período actual del calendario académico en el que se encuentra el beneficio
codigo_beneficio	Integer	Código del Beneficio
cupos_disponibles	Smallint	Número de los cupos disponibles de cada Beneficio
cupos_totales	Smallint	Número de los cupos habilitables dependiendo a cada Beneficio

- **Clase periodo_inscripcion:**

Relaciona el periodo de tiempo el cual el administrador habilita para realizar la inscripción a cada beneficio perteneciente al calendario académico actual.

Tabla 11 Atributos de la Clase perido_inscripcion

Nombre	Tipo de datos	Descripción
Anio	Smallint	Año actual
Periodo	Smallint	Período actual del calendario académico en el que se encuentra

		el beneficio
codigo_beneficio	Integer	Código del Beneficio
fecha_inicio	Date	Fecha de inicialización en la cual el estudiante podrá inscribirse al beneficio
fecha_fin	Date	Fecha de finalización en la cual el estudiante podrá inscribirse al beneficio

Clase periodo_entrevista:

Relaciona el periodo de tiempo el cual el administrador habilita para realizar la entrevista a cada estudiante preseleccionado a algún beneficio, tiempo perteneciente al calendario académico actual.

Tabla 12 Atributos de la Clase periodo_entrevista

Nombre	**Tipo de datos**	**Descripción**
Anio	Smallint	Año actual
Periodo	Smallint	Período actual del calendario académico en el que se encuentra el beneficio
codigo_beneficio	Integer	Código del Beneficio
fecha_inicio	Date	Fecha inicio del período de entrevistas para los estudiantes preseleccionados
fecha_fin	Date	Fecha fin del período de entrevistas para los estudiantes preseleccionados

- **Clase días_para_entrevista:**

Días hábiles en los cuales el estudiante preseleccionado puede escoger para que se le realice la entrevista.

Tabla 13 Atributos de la Clase días_para_entrevista

Nombre	Tipo de datos	Descripción
Anio	Smallint	Año actual
Periodo	Smallint	Período actual del calendario académico en el que se encuentra el beneficio
codigo_beneficio	Integer	Código del Beneficio al cual el estudiante está inscrito.
codigo_estudiante	Integer	Código del estudiante inscrito y preseleccionado
fecha_entrevista	Date	Fecha de la entrevista en la cual se realizará la entrevista a los estudiantes preseleccionados
hora_ inicio	Date	Hora de inicio de la entrevista
hora_fin	Date	Hora de fin de la entrevista
tipo_documento	Smallint	Tipo de documento del profesional a cargo de la entrevista
documento_entrevistador	Integer	Documento del profesional a cargo de la entrevista
Lugar	Varchar	Lugar donde se realizará la entrevista

- **Clase entrevistador:**

Relaciona los datos del profesional a cargo de entrevistas socioeconómicas

Tabla 14 Atributos de la Clase entrevistador

Nombre	Tipo de datos	Descripción
Correo	Varchar	Correo electrónico del profesional a cargo de las entrevistas socioeconómicas
Dirección	Varchar	Dirección del profesional a cargo de las entrevistas socioeconómicas
primer_apellido	Varchar	Primer apellido del profesional a cargo de la entrevista
segundo_apellido	Varchar	Segundo apellido del profesional a cargo de la entrevista
primer_nombre	Varchar	Primer nombre del profesional a cargo de la entrevista
segundo_nombre	Varchar	Segundo nombre del profesional a cargo de la entrevista
tipo_entrevistador	Char	Si el profesional a cargo de entrevistas es externo o interno
tipo_documento	Smallint	Tipo de documento del profesional a cargo de las entrevistas
Documento	Integer	Número del documento del profesional a cargo de las entrevistas
Teléfono	Smallint	Número del teléfono del profesional a cargo de las entrevista

- **Clase contraprestación:**

Relaciona la carga horaria de servicio en cualquier división de la universidad como contraprestación según el beneficio adquirido por el estudiante.

Tabla 15 Atributos de la Clase contraprestacion

Nombre	Tipo de datos	Descripción
codigo_contraprestacion	Serial	Código de la contraprestación relacionada según cada beneficio
nombre_contraprestacion	Varchar	Período actual del calendario

		académico en el que se encuentra el beneficio
Descripción	Varchar	Descripción de la contraprestación

3.9.3 Diagrama de Secuencias

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

El diagrama de secuencia que se presenta a continuación da una visión más amplia acerca del funcionamiento y la interacción del sistema con el usuario, en el caso de uso de la inscripción de un Estudiante a un Beneficio.

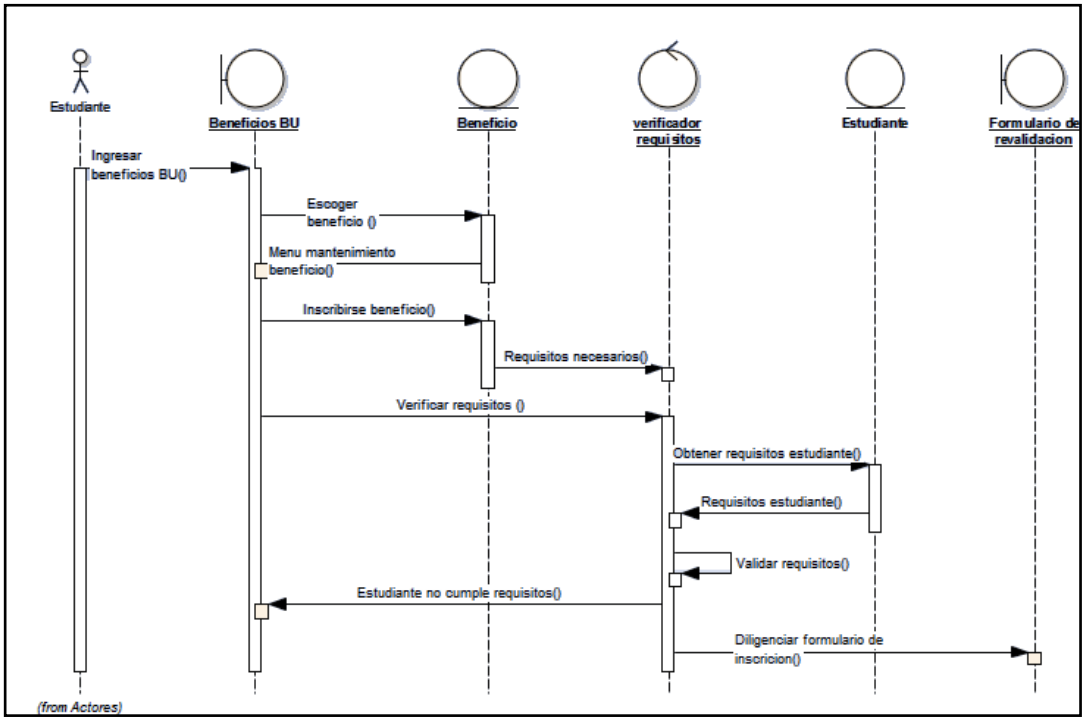

Figura 29. Diagrama de Secuencias Inscripción Beneficio

3.10 Prototipos

3.10.1 Prototipo Inicial

En el primer prototipo, se dejó reflejado la mayor parte de la funcionalidades del sistema de Información, ya que era de suma importancia que los usuarios tuvieran una percepción más clara, mediante las interfaces graficas del sistema de Información que se pretendía realizar. A continuación se muestra uno de los módulos más significativos de este primer prototipo:

- **Beneficio BU**

En este primer modulo se mostraba como seria la navegación del sistema para los procesos de:

- **Administrar beneficios**: Proceso encargado de la creación, edición y eliminación de beneficios.

Figura 30. Formato Crear Nuevo Beneficio

- **Asignar requisitos**: Proceso encargado de la asignación, edición y eliminación de cada uno de los requisitos tanto académicos - financieros como otros requisitos que sería asociados a cada uno de los beneficios.

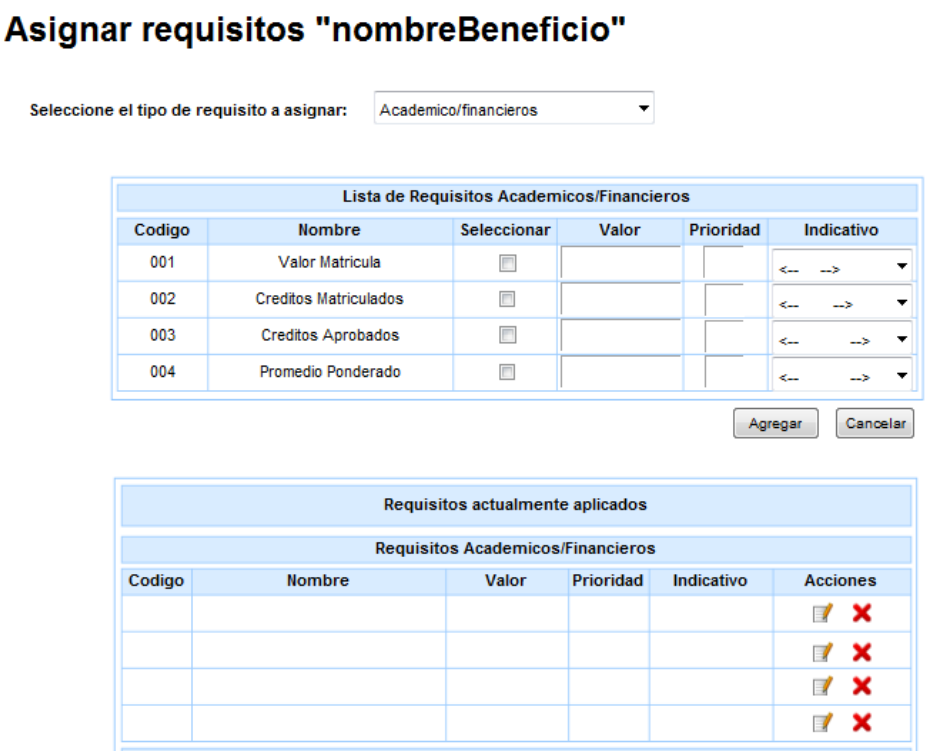

Figura 31. Formato Asignar Requisitos

- **Asignar cupos**: Proceso encargado de asignación, edición y eliminación del número de cupos que un beneficio iba a tener para determinado año y periodo académico.

- **Establecer contraprestaciones**: Proceso encargado de la asignación y eliminación de las contraprestaciones que tendría asociado cada beneficio.

- **Listar beneficios**: Proceso encargado de mostrar la lista de beneficios que actualmente se tienen creados.

3.10.2 Prototipo final

Basados en las sugerencias y conclusiones planteadas por el personal de BU en el momento de la creación del primer prototipo se dio origen al desarrollo del sistema actual, dada la complejidad del sistema a continuación se describe los procedimientos de dos de los módulos de mayor importancia:

- **Beneficio BU**

Este módulo permite la administración y la actualización de los beneficios, lo componen los siguientes procesos:

- **Administrar beneficios**: Proceso permite la creación, edición y eliminación de beneficios.

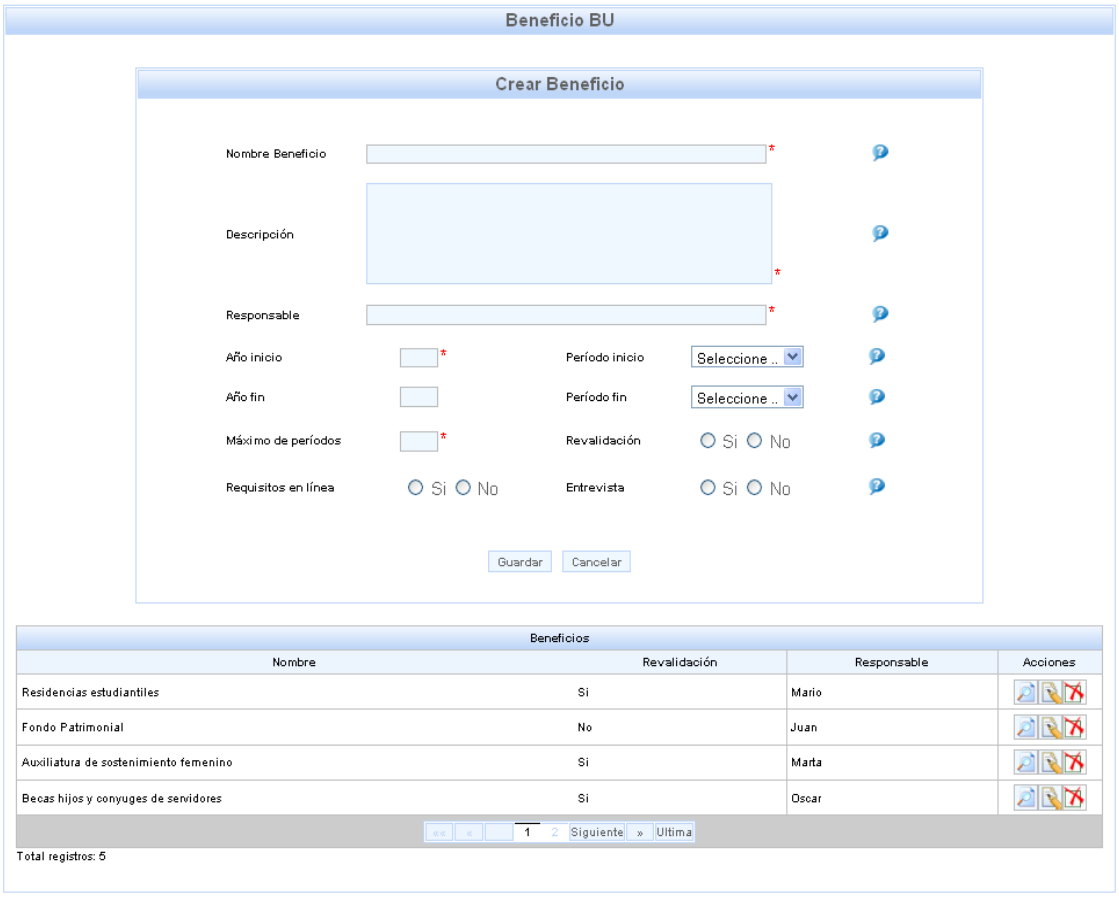

Figura 32. Formato Crear Beneficio

- **Asignar requisitos**: Proceso que permite la asignación, edición y eliminación de cada uno de los requisitos que sean asociados a determinado beneficio.

Figura 33. Formato Asignar Requisito Beneficio

- **Asignar cupos**: Proceso que permite la asignación, edición y eliminación de los cupos que son asignados a cada uno de los beneficios para cada año y periodo académico.

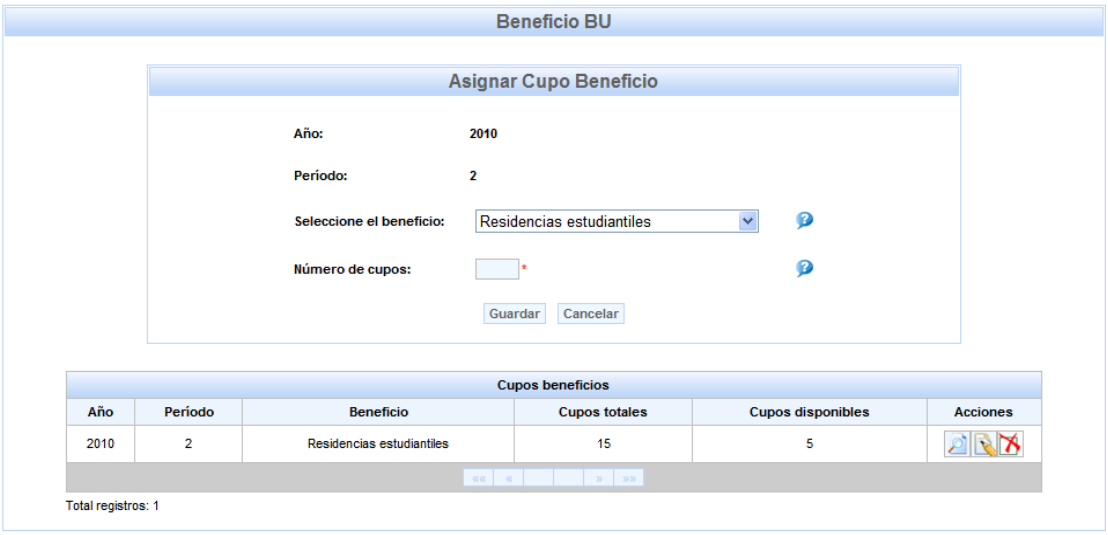

Figura 34. Formato Asignar Cupo Beneficio

- **Asignar contraprestación**: Proceso que permite la asignación y eliminación de las contraprestaciones que le sean asociadas a cada beneficio.

Figura 35. Formato Asignar Contraprestación

- **Listar beneficios**: Este proceso lista todos los beneficios que actualmente se encuentran creados.

Figura 36. Formato Listar Beneficios BU

Inscripciones

Este módulo permite actualizar los periodos de inscripciones y ofrece gran flexibilidad al momento del proceso de inscripción.

A continuación se describen los procedimientos que componen este módulo.

- **Abrir fechas**: Este proceso permite asignar, editar y eliminar la fecha de inicio y la fecha de finalización de las inscripciones a cada uno de los beneficios para cada periodo y año académico.

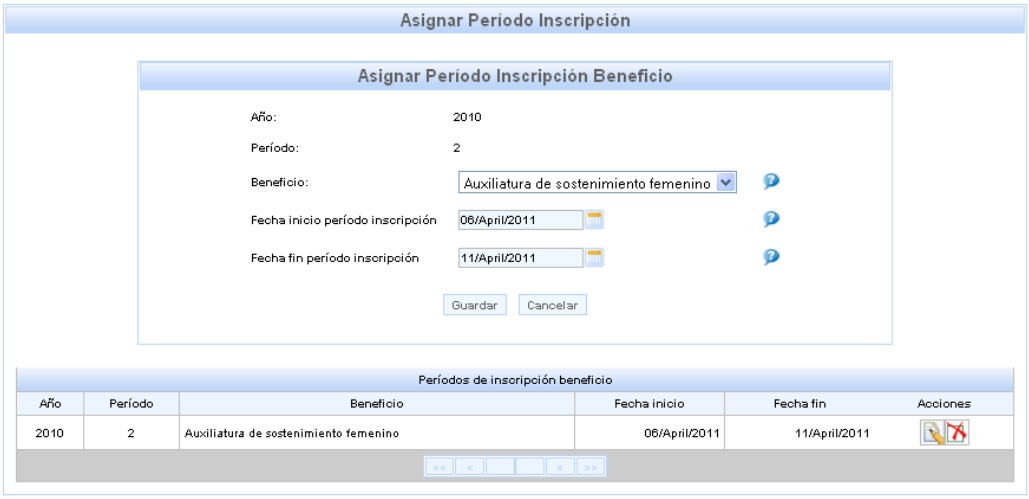

Figura 37. Formato Asignar Periodo Inscripción Beneficio

- **Extemporánea**: Este proceso permite la inscripción de estudiantes fuera de las fechas de inscripción con la previa autorización del comité de beneficios de BU.

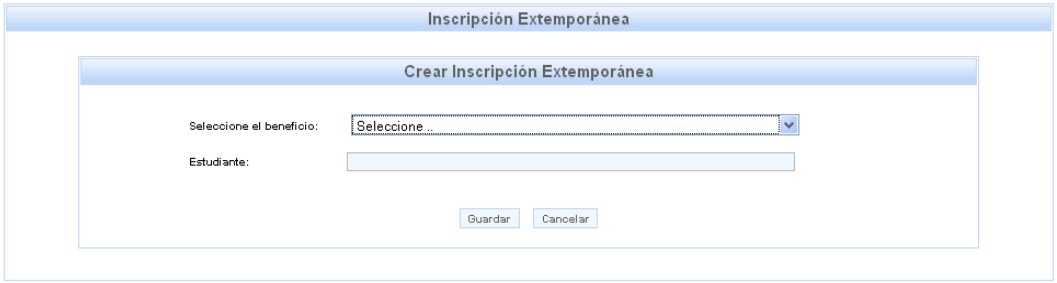

Figura 38. Formato Inscripciones Extra temporáneas

- **Efectuar preselección**: Este proceso realiza un ordenamiento de los inscritos al beneficio de acuerdo a los criterios de selección establecidos por BU resaltando los estudiantes que son seleccionados o preseleccionados para la adjudicación del beneficio.

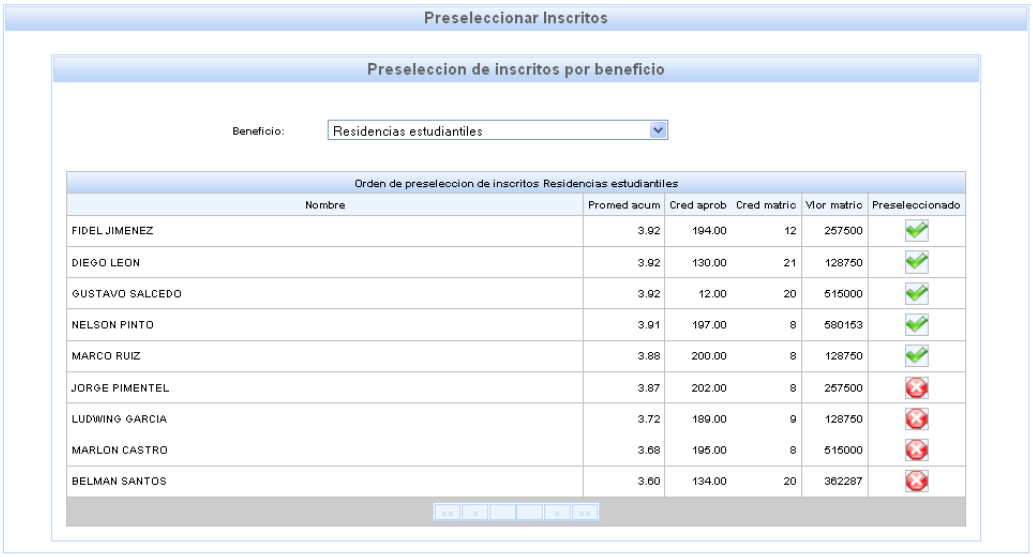

Figura 39. Formato Preseleccionar Inscritos

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

3.11 Esquema de seguridad Universidad Industrial de Santander

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

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

3.11.1 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 sistemas de información, contemplando: Sistema de información, nombre del

menú, descripción del menú, fecha y hora de creación del menú, fecha y hora de inicio de vigencia del menú asociado al rol, fecha y hora de cierre de vigencia del menú asociado al rol.

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

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

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

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

Atributo–tabla: Establece los atributos sobre los cuales se debe controlar el acceso a una tabla, cuando a un rol se le concede permiso para hacer uso parcial de la información existente en una tabla. Contiene: Sistema de

información, nombre de la tabla, nombre del atributo sobre el cual se controla el acceso a la tabla, descripción del atributo, fecha y hora de creación del atributo, fecha y hora de inicio de vigencia del atributo, fecha y hora de fin de vigencia del atributo.

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

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

3.11.2 Entorno de Navegación

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

Para cada rol se define el menú de inicio, el cual le permite a cada usuario que hace parte de este rol, empezar la navegación por las distintas opciones que le ofrece el sistema, hasta llegar al nivel más bajo en el cual se ejecuta el proceso que soporta la actividad que desea realizar.
Este entorno está soportado por las siguientes tablas de las base de datos del esquema de seguridad: Rol, usuario, sistema, sistema-rol, usuario-rol, menú-rol, opción-menú rol, descritas en “ESTRUCTURA DE LA BASE DE DATOS SOPORTE”.

3.11.3 Entorno de Control de Datos

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

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

3.11.4 Auditoría

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

El historial de las transacciones de cada tabla contiene información de los registros incluidos en la tabla, de los registros modificados y de los registros eliminados. Adicionalmente, en cada transacción se especifica: Fecha de la transacción, hora de la transacción, tipo de transacción (I/U/D), tipo y número de documento de identidad del usuario que realizó la transacción, login, rol asociado, dirección IP y MAC del equipo desde el cual llevó a cabo la transacción

CAPITULO 4

4. CONCLUSIONES

- El sistema de información permite ampliar la cobertura al sector estudiantil para que todos los estudiantes activos tengan conocimiento de los beneficios ofrecidos por BU y quienes debido a sus condiciones económicas más lo requieran puedan hacer uso de ellos.

- El diseño realizado para el desarrollo de este software permitió la construcción de un software dinámico y flexible que puede adaptarse a los cambios que se den en el proceso de manejo de beneficios. Entre estos cambios se pueden considerar la creación de nuevo beneficios, ampliación de cobertura.

- El Sistema de Información orientado a la web, soporte para el manejo de los Beneficios que ofrece BU cumple con las expectativas de las personas a cargo de la administración de los servicios ofrecidos.

- La complejidad del software se debe a la multi-capa en la arquitectura de la tecnología Java EE5 consecuencia de eso los programadores principiantes se encuentran con muchos conceptos que no conocen y esto esto puede retrasar la codificación de los programas ya que existen muchos conceptos que la mayoría de programadores principiantes no conocen y retrasan la eficiencia en la creación del software.

CAPITULO 5

5. RECOMENDACIONES

- Incluir en la futura versión el módulo de sicología que en este momento no hace parte del proceso para la adjudicación de los beneficios pero que actualmente se están haciendo los análisis respectivos para su implementación.

- Teniendo en cuenta que esta fue una experiencia muy gratificante y beneficiosa para nuestra formación profesional, se recomienda que la escuela de Ingeniería de Sistemas acompañada de la DSI le puedan ofrecer a más estudiantes la posibilidad de tenerla, ampliando así los conocimientos adquiridos a lo largo de la carrera.

- Las tecnologías de información, así como los frameworks de desarrollo, seleccionados para un proyecto de elaboración de software facilitan y garantizan en cierta medida la calidad que sugiere el nivel empresarial. No obstante, expertos Analistas Programadores afirman que el uso de frameworks de desarrollo Seam o Rich Faces no es justificable ni tan eficiente en aplicaciones pequeñas o poco complejas. Por consiguiente es recomendable para futuros desarrollos que lidere la División de Servicios de Información, o cualquier otra dependencia de la UIS, escoger la tecnología de información y herramientas de desarrollo que mejor se adapten a las necesidades del sistema.

CAPITULO 6

6. BIBLIOGRAFIA

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

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

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

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

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

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

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

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

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

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

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

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