SISTEMA DE INFORMACIÓN ORIENTADO A LA WEB COMO SOPORTE PARA EL PROCESO DEL MANEJO DE LAS MINUTAS OFRECIDO POR LA SECCIÓN DE CAFETERIA Y COMEDORES DE LA DIVISIÓN DE BIENESTAR UNIVERSITARIO.

ANGELICA MARÍA AMAYA GIRALDO

FRANCY ZULAY CARRILLO VEGA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2011

SISTEMA DE INFORMACIÓN ORIENTADO A LA WEB COMO SOPORTE PARA EL PROCESO DEL MANEJO DE LAS MINUTAS OFRECIDO POR LA SECCIÓN DE CAFETERIA Y COMEDORES DE LA DIVISIÓN DE BIENESTAR UNIVERSITARIO.

ANGELICA MARÍA AMAYA GIRALDO

FRANCY ZULAY CARRILLO VEGA

Trabajo de grado para optar el título de Ingeniero de Sistemas

Director:

Ingeniero Enrique Torres López

Codirectora:

Nutricionista Consuelo Serrano Vega

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2011

AGRADECIMIENTOS

A Dios, porque de él solo recibo bendiciones, y por guiar mis pasos por el sendero del éxito.

A mi familia por su apoyo incondicional porque sé que este también es un triunfo para ellos. Mi madre por ser mi mejor amiga, mi consejera, mi más grande apoyo en cada momento de mi vida, mi padre por creer en mí y darme todo de él para que yo alcanzara mis sueños, mis hermanos por su apoyo y su presencia en mi vida.

A mi compañera Francy Zulay Carrillo, quien con su trabajo y esfuerzo me apoyo para lograr culminar este proyecto que no es solo académico sino de vida.

A todos los profesionales de la División de Servicios de Información, quienes siempre estuvieron allí para orientarnos ante cualquier duda o problema que se presentara.

A nuestra Codirectora la Dra. Consuelo Serrano quien estuvo dispuesta a brindarnos su colaboración cada vez que la necesitamos.

Muy especialmente a nuestro Director el Ingeniero Enrique Torres que con sus consejos, orientación y su confianza nos estuvo acompañando durante todo el proceso, aportándonos a nuestra formación profesional y personal.

A todas las personas que hicieron posible este paso, que es un peldaño para una nueva etapa en mi vida.

Angélica María Amaya Giraldo

AGRADECIMIENTOS

Quiero agradecer a Dios, por permitirme llegar hasta aquí y cumplir esta meta, a pesar de tantos tropiezos, me aportó fortaleza y sabiduría para tomar decisiones.
Gracias a mis padres y mis hermanos, que son mi motor y mi polo a tierra, por la paciencia, apoyo y consejos.

A nuestro Director de Proyecto, el Ingeniero Enrique Torres López, por ser un gran maestro durante todo este proyecto, por la paciencia, la confianza depositada en nosotras y por guiarnos de la mejor manera. A nuestra Codirectora, la Nutricionista Consuelo Serrano Vega y al Ingeniero Jacksson por su constante apoyo durante este proceso.

A los Ingenieros de la DSI, por su constante colaboración durante este proceso de desarrollo del proyecto, especialmente a los Ingenieros Danny Vergel y Humberto Díaz. Y a la División de Servicios de Información en general, por haber hecho parte de mi formación como profesional.

A mis compañeros de equipo en el proyecto, con quienes compartí tantos momentos durante todo este proceso: Jonathan, Ludwing, Paulo, Tatiana, Jefferson, Fidel, Andrea, Marco, Edwing y especialmente a Angélica Amaya, quien no solo ha sido mi coequipera en este largo camino construyendo SIMIN sino durante toda mi vida universitaria, amiga a quien quiero y profesional que admiro.

A mi amigo Jorge Torres y a todos aquellos amigos que siempre han estado aquí conmigo apoyándome y creyendo en mis capacidades, sabían que algún día lo iba a lograr, y me alegra no haberlos defraudado.

Francy Zulay Carrillo Vega

TABLA DE CONTENIDO

INTRODUCCION **20**

PARTE I FUNDAMENTOS **21**

1 DESCRIPCIÓN DEL PROYECTO **21**

1.1 DEFINICIÓN DEL PROBLEMA **21**

1.1 OBEJTIVOS **23**

1.1.1 OBJETIVO GENERAL 23

1.1.2 OBJETIVOS ESPECÍFICOS 23

1.2 JUSTIFICACIÓN **25**

1.3 IMPACTO **26**

1.4 VIABILIDAD **27**

2. MARCO TEORICO **29**

2.1 ESTRUCTURA DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER **29**

2.1.1 ESTRUCTURA ORGANIZACIONAL DE LA UIS 29

2.2 BIENESTAR UNIVERSITARIO **32**

2.2.1 MISIÓN DE BIENESTAR UNIVERSITARIO 34

2.2.2 ORIENTACIÓN FILOSÓFICA 34

2.2.3 OBJETIVOS DE BIENESTAR UNIVERSITARIO 34

2.2.4 POLÍTICAS DE BIENESTAR UNIVERSITARIO 34

2.2.5 ESTRUCTURA ORGANIZACIONAL DE LA DBU 35

2.3 SECCIÓN COMEDORES Y CAFETERÍA **36**

2.3.1 MISIÓN DE LA SECCIÓN DE COMEDORES Y CAFETERÍA DBU 36

2.4 SERVICIO DE NUTRICIÓN 37

2.4.1 PLANEACIÓN 37

2.4.2 BALANCEO NUTRICIONAL 37

2.4.3 CÁLCULOS DE LA DIETA 40

2.5 ESQUEMA DE DESARROLLO 41

2.5.1 APLICACIONES ORIENTADAS A LA WEB 41

2.5.2 DISEÑO CON EL ESTANDAR UML 47

2.5.3 JAVA ENTERPRISE EDITION 5.0 (JEE 5) 56

2.5.4 TECNOLOGÍA SERVLET JAVA 60

2.5.5 JAVA SERVER FACES 63

2.5.6 SEAM 67

2.5.7 SERVIDOR DE APLICACIONES – JBOSS 71

2.5.8 ENTERPRISE ARCHITECT 71

2.5.9 CONSTRUCCIÓN DE PROTOTIPOS DE SOFTWARE 72

3. METODOLOGÍA DE DESARROLLO 75

3.1 CICLO DE VIDA DEL PROYECTO 75

3.1.1 ANÁLISIS DE REQUERIMIENTOS 75

3.1.2 DISEÑO 76

3.1.3 IMPLEMENTACIÓN DE LA APLICACIÓN 77

3.1.4 PRUEBAS DEL SOFTWARE 77

3.1.5 AJUSTES 78

3.2 METODOLOGÍA DE DESARROLLO DEL PROYECTO 78

3.2.1 MODELO DE CONSTRUCCIÓN DE PROTOTIPOS 78

3.2.2 ESTRUCTURA DEL MODELO DE CONSTRUCCIÓN DE PROTOTIPOS 79

3.2.3 PROCEDIMIENTO PARA LA METODOLOGÍA PLANTEADA 80

3.3 PLAN DE TRABAJO 81

3.3.1 FASE PREVIA: DE ACOPLAMIENTO 82

3.3.2 PRIMERA FASE: RECOLECCIÓN Y ANÁLISIS DE REQUERIMIENTOS 82

3.3.3 SEGUNDA FASE: DISEÑO DEL PROTOTIPO 83

3.3.4 TERCERA FASE: DESARROLLO DEL PROTOTIPO 83

3.3.5 CUARTA FASE: REFINAMIENTO DEL PROTOTIPO 84

3.3.6 QUINTA FASE: IMPLANTACIÓN Y ENTREGA DEL PRODUCTO FINAL 84

PARTE II. DESARROLLO DEL SISTEMA 85

4. APLICACIÓN DE LA METODOLOGÍA 85

4.1 LEVANTAMIENTO DE REQUERIMIENTOS 85

4.1.1 DESCRIPCIÓN GENERAL 85

4.1.2 FUNCIONES DEL SISTEMA 85

4.2 ESTÁNDARES DE LA DIVISIÓN DE SERVICIOS DE INFORMACIÓN. 88

4.2.1 ASPECTOS GENERALES 88

4.2.2 DOCUMENTACIÓN DE LOS DIAGRAMAS DE DISEÑO 91

4.2.3 SINTAXIS DE NOMBRES EN JAVA 92

4.2.4 DOCUMENTACIÓN 96

4.2.5 CAPA DE PRESENTACIÓN 96

4.3 ROLES Y USUARIOS 101

4.4 DIAGRAMAS UML 102

4.4.1 DIAGRAMA: CASOS DE USO 102

4.4.2 DIAGRAMAS DE SECUENCIA 108

4.4.3 DIAGRAMA DE CLASES 110

4.4.4 DIAGRAMA DE DATOS 115

4.5 PROTOTIPO INICIAL 115

4.5.1 GENERALIDADES DEL PROTOTIPO INICIAL 115

4.5.2 PRIMER PROTOTIPO : PROTOTIPO NO FUNCIONAL 116

4.6 PROTOTIPO FINAL 119

4.6.1 ESQUEMA DE SEGURIDAD DE LA UIS, IMPLEMENTADO EN SIMIN 122

4.6.2 ORGANIZACIÓN DE DIRECTORIOS 127

4.6.3 DOCUMENTACIÓN DE PROGRAMAS FUENTE. 130

5. CONCLUSIONES 139

6. RECOMENDACIONES 141

7. BIBLIOGRAFÍA 142

LISTA DE TABLAS

Tabla 1 - Guía alimentaria para la población colombiana (ICBF).....39

Tabla 2 - Grupos etáreos Población Beneficiarios.....39

Tabla 3 - Fórmula Sintética.....41

Tabla 4 - Actores del sistema SIMIN.....103

Tabla 5 - Detalle del Caso de Uso: Gestionar Minutas.....107

Tabla 6 - Casos de Uso para Minutas.....107

Tabla 7 - Atributos Clase MenuPrograma.....112

Tabla 8 - Atributos Clase Elemento Adicional.....112

Tabla 9 - Atributos Clase TotalElementoMenu.....113

Tabla 10 - Atributos Clase Menu.....114

Tabla 11 - Atributos Clase ServicioTipoServicio.....114

LISTA DE FIGURAS

Figura 1 - Estructura Organizacional de la UIS 30
Figura 2 - Estructura Organizacional de Vicerrectoría Académica de la UIS 33
Figura 3 - Estructura Organizacional DBU 36
Figura 4 - Recommended Dietary Allowances (RDA) 38
Figura 5 - Elementos de un Sistema de Información 43
Figura 6 - Representación de la Nota (UML) 48
Figura 7 - Representación de la Dependencia (UML) 49
Figura 8- Representación de un Actor en UML 50
Figura 9 - Representación de un Caso de Uso en UML 50
Figura 10 - Representación de una relación entre Casos de Uso en UML 51
Figura 11 - Representación de un Diagrama de Secuencia (UML) 52
Figura 12 - Representación de un Diagrama de Actividades (UML) 53
Figura 13 - Representación Diagrama de Clases (UML) 54
Figura 14 - Representación Diagrama Entidad - Relación 55
Figura 15 - Modelo de Construcción de Prototipos 79
Figura 16 - Estructura del Modelo de Construcción de Prototipos 80
Figura 17 - Plantilla Principal División de Servicios de Información 97
Figura 18 - Plantilla de Contenido División de Servicios de Información 97
Figura 19 - Plantilla de Contenido con Formulario DSI 98
Figura 20 - Modelo de Casos de Uso SIMIN 102

Figura 21 - Actores del Modelo de Casos de Uso SIMIN.....103
Figura 22 - Estructura del Modelo de Casos de Uso SIMIN 104
Figura 23 - Diagrama Casos de Uso: Minutas 105
Figura 24 - Diagrama Secuencia: Crear Nuevo Servicio 109
Figura 25 - Diagrama Secuencia: Establecer cantidades elemento.....110
Figura 26 - Diagrama de Clases, Modulo Minutas.....111
Figura 27 - Nueva Receta (Prototipo Inicial) 118
Figura 28 - Nuevo Menu (Prototipo Inicial) 119
Figura 29 - Programar Minuta (Prototipo Final).....121
Figura 30 - Consulta de Pedidos (Prototipo Final) 122

GLOSARIO DE TERMINOS

AJAX (asynchronous Javascript y Xml) : Es una técnica de desarrollo para aplicaciones web interactivas que ejecutan del lado cliente, por medio de navegador del usuario y conserva una comunicación asíncrona con el servidor web, aumentando la rapidez de la aplicación web.

BALANCEO NUTRICIONAL : Es una operación que se utiliza para encontrar el punto justo para cerciorarse si la alimentación brindada a su cuerpo, está acorde con las necesidades energéticas y nutricionales, tomando una muestra de una población nutricionalmente normal.

BIENESTAR UNIVERSITARIO : Es una dependencia integradora, cuya misión es participar en la formación integral de la comunidad universitaria con énfasis en la construcción de una cultura de auto cuidado y del bien común, para que cada persona de la comunidad desde su singularidad y autonomía asuma con calidad su proyecto de vida.

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

COMEDORES : Es el servicio de alimentación a estudiantes de la universidad, que tengan bajos recursos y que cumplan algunos requisitos establecidos por Bienestar Universitario.

JAVA : Es un lenguaje de programación, que se caracteriza por tener una arquitectura que permite generar programas independientes de la plataforma, es decir, que pueden ejecutarse en cualquier sistema operativo o hardware.

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

MINUTA : Lista de recetas que componen una dieta nutricional, es la base de todas las funciones en una unidad de servicio de alimentación.

MÓDULO : Se utiliza como sinónimo de subsistema.

OBJETO : Se define como la unidad que en tiempo de ejecución realiza las tareas de un programa. También a un nivel más básico se define como la instancia de una clase.

RECETA : Es una lista de ingredientes y una serie de instrucciones para realizar un plato de cocina particular.

SIMIN : Sistema de Información para el manejo de Minutas de la División de Bienestar Universitario de la Universidad Industrial de Santander.

SISTEMA DE INFORMACIÓN : Aplicación para computador. Construida por la base de datos, los programas de aplicación, los procedimientos manuales y automatizados e incluye los sistemas computacionales que realizan procesamiento.

RESUMEN

TITULO:

SISTEMA DE INFORMACIÓN ORIENTADO A LA WEB COMO SOPORTE PARA EL PROCESO DEL MANEJO DE LAS MINUTAS OFRECIDO POR LA SECCIÓN DE CAFETERIA Y COMEDORES DE LA DIVISIÓN DE BIENESTAR UNIVERSITARIO.*

AUTORES:

FRANCY ZULAY CARRILLLO VEGA
ANGELICA MARIA AMAYA GIRALDO**

PALABRAS CLAVES:

Minutas, Bienestar Universitario, nutrición, software, menú, comedores, Sistema de Información Web, JavaEE5, JSF, RichFaces, Hibernate, Seam.

DESCRIPCIÓN:

El presente trabajo de grado fue desarrollado en conjunto con la División de servicios de Información, unidad encargada de administrar y desarrollar el software que soporta el funcionamiento de la Universidad Industrial de Santander (UIS) y la División de Bienestar Universitario.

Se presenta la solución práctica y completa para mejorar una situación específica que se encuentra latente en las necesidades que tiene la sección de Cafetería y Comedores de la división de Bienestar Universitario de la Universidad Industrial de Santander, con un software de calidad y con la garantía de todo trabajo realizado por la División de Servicios de Información.

El proyecto plantea el diseño y desarrollo de una herramienta software orientada a la web de nivel empresarial, de acuerdo a los parámetros de calidad y siguiendo con los lineamientos nutricionales establecidos por el balanceo nutricional, permitiendo a Bienestar Universitario crecer en cobertura, en cantidad y calidad de servicios de alimentación ofrecidos por la Sección de Cafetería y Comedores. El desarrollo de este trabajo de grado utilizó tecnologías de punta y se tomaron como base los estándares definidos por la División de Servicios de Información UIS, los cuales contemplan las herramientas de desarrollo, la organización de directorios, la diagramación UML y su documentación, la organización y documentación del código fuente.

---

*Proyecto de grado, en la modalidad de de Investigación

**Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas. Director: Ingeniero Enrique Torres López. Codirectora: Consuelo Serrano Vega

SUMMARY

TITLE:
WEB-ORIENTED INFORMATION SYSTEM AS SUPPORT FOR MANAGEMENT PROCESS OF MINUTES OFFERED BY THE CAFETERIA AND DINING SECTION OF THE DIVISION OF STUDENT WELFARE. *

AUTHORS:

FRANCY ZULAY CARRILLLO VEGA
ANGELICA MARIA AMAYA GIRALDO **

KEYWORDS:

Minutas, welfare, nutrition, software, menus, dining, Web Information System, JavaEE5, JSF, RichFaces, Hibernate, Seam.

DESCRIPTION:

The present degree project was developed in joint with the Division of Information Services, a branch in charge of managing and developing the software that supports the functioning of the UIS, and the Division of student welfare of the Universidad Industrial de Santander (UIS).

We are presenting a practical and comprehensive solution to improve a specific situation that is latent in the needs of cafeteria and dining section of the Division of student welfare of the Universidad Industrial de Santander. This is software with quality assurance performed by the Division of Information Services.

The project proposes the design and development of a software tool for the web aimed at enterprise level, according to the quality parameters and following the nutritional guidelines established by the nutritional balancing and thus allowing the Division of student welfare grows in coverage, quality and quantity of food services offered by Cafeteria and Dining Section. The development of this degree project has done using edge technologies and standards defined by the Information Services Division of UIS as baseline, including development tools, folders organization, UML documentation & diagramming, the documentation and organization of source code.

---

*Degree Project in the modality Research.
**Physique-Mechanics Sciences Department, Computer Science Engineering, Engineer Enrique Torres Lopez, Consuelo Serrano Vega

INTRODUCCION

La División de Bienestar universitario ha crecido en servicios, buscando cubrir de manera eficaz las necesidades tanto nutricionales como económicas de toda la comunidad universitaria. Con el propósito de impulsar esta meta la División de Servicios de información se da a la tarea de apoyarla, brindando una solución informática de alta calidad, facilitando un proceso de toma de decisiones y de modernización de la institución.

La División de Bienestar universitario hoy por hoy cuenta con un sistema de información que no se ajusta a las condiciones actuales, lo que hace que limite al usuario final en la toma de decisiones forzándolo a recurrir a métodos más tradicionales que son tediosos a la hora utilizar.

En este proyecto se propone el desarrollo de un sistema de información orientado a la web como soporte para el proceso del manejo de las minutas cumpliendo con los estándares de desarrollo de la División de servicios de Información de la Universidad Industrial de Santander, y teniendo en cuenta los nuevos requerimientos resultado de los nuevos servicios ofrecidos por la sección de Cafetería y Comedores de la División de Bienestar Universitario.

PARTE I FUNDAMENTOS

1 DESCRIPCIÓN DEL PROYECTO

1.1 Definición Del Problema

La finalidad básica de la alimentación es conseguir una buena nutrición para conservar la integridad física y asegurar el correcto ejercicio de las funciones que hacen posible que la vida continúe.

Se exige elegir bien los alimentos para gozar de una buena salud y sentirse bien, comer variedad de alimentos y bebidas con los nutrientes que el cuerpo necesita, provenientes de los distintos grupos alimenticios del modelo nutricional adoptado, modelos como la Pirámide nutricional o la Esfera Alimentaria o el Tren Alimentario, para mantenerse dentro de las necesidades calóricas de cada uno, y reforzar el modelo nutricional con una tabla de equivalencias para obtener una visión más clara de las porciones que se deben consumir de cada grupo de alimentos, teniendo en cuenta que aunque el peso o tamaño de los alimentos sea el mismo, sus porciones no son equivalentes, los volúmenes son radicalmente diferentes en comparación con su aportación energética.

La nutrición puede ser mala no sólo por un exceso de alimentos sino también por su carencia o por que existe un desequilibrio en lo consumido. En todos los casos se produce trastornos que afectan a la salud física y mental, obesidad, desnutrición o enfermedades debidas a la falta o exceso de algún nutriente en concreto.

Los órganos y funciones que participan más directamente en la nutrición, el aparato digestivo con sus funciones características y en especial al metabolismo, son los que sufren con más frecuencia las consecuencias de una incorrecta alimentación.

Es fundamental cerciorarse que la alimentación que brinde la División de Bienestar Universitario, esté acorde con las necesidades energéticas y nutricionales de los estudiantes y trabajadores de la Universidad.

La Sección Comedores y Cafetería de Bienestar Universitario, tiene como misión, contribuir al mejoramiento de las condiciones nutricionales de la comunidad Universitaria a través de acciones educativas y servicios con calidad que ofrece a toda la comunidad universitaria[1], por lo tanto es la responsable de la planeación de la MINUTA, para los diferentes servicios que ofrece: comedores, cafetería, combo saludable y eventos especiales, y otros servicios que pueda considerar en el futuro, servicios que tienen sus propios objetivos y características.

La base de todas las funciones en una unidad de servicio de alimentación es la MINUTA, consiste en una lista de recetas que componen la comida.
Las minutas ofrecidas por la División de Bienestar Universitario, deben cumplir con las características de la alimentación saludable y fomentar hábitos alimentarios correctos en la comunidad universitaria.

Los alimentos ofrecidos para los diferentes servicios, deben tener una gran variedad de texturas, consistencia, color, sabor, forma, humedad, y métodos de preparación, esto se logra con un buen número de recetas estandarizadas, y generando combinaciones adecuadas.

[1] Portafolio de Programas y Servicios 2010, Bienestar Universitario, Universidad Industrial de Santander.

Las minutas debe agradar a los usuarios, cubrir las necesidades nutricionales de los usuarios, mantenerse dentro de los límites presupuestales, contribuir a crear la imagen del servicio y establecer las bases para la realización de otras actividades como la compra, estándares de producción, etc.

Las minutas pueden activarse o inactivarse de acuerdo a muchos factores, dependiendo de características específicas y pueden ser modificados debido a diversas circunstancias, tales como escasez de alimentos, cosecha de alimentos, daños en los equipos de producción de determinada minuta, y otros factores.

1.1 Obejtivos

1.1.1 *Objetivo General*

Analizar, Diseñar, implementar y poner en funcionamiento un Sistema de información orientado a la web como soporte para el proceso del manejo de las minutas ofrecido por la Sección de Cafetería y Comedores de la División de Bienestar universitario de la Universidad Industrial de Santander.

1.1.2 *Objetivos Específicos*

El sistema a desarrollar permitirá:

- Realizar el balanceo de los menús cumpliendo con las recomendaciones nutricionales para cada servicio que ofrece la Sección de Comedores y Cafetería: comedores, cafetería, combo saludable y para los eventos especiales que se requiera, con las características definidas por la División de Bienestar Universitario, para la población que utiliza los diferentes servicios.

- Facilitar el proceso de elaboración de las minutas diarias, para comedores, cafetería y combo saludable generando los reportes de:
    - ✓ Recetas a utilizar.
    - ✓ Alimentos a pedir.
    - ✓ Alimentos que van a pique.
    - ✓ Alimentos que no van a pique.
    - ✓ Costos de los insumos, por minuta y por pedido.

- Registrar las minutas ofrecidas por la Sección de Comedores y Cafetería con los debidos controles para garantizar la calidad de la información.

- Seguir las políticas de seguridad establecidas para los Sistemas de Información de la Universidad Industrial de Santander, estableciendo:

    - ✓ Los roles para los usuarios del sistema, con el fin de definir los permisos de acceso al mismo.
    - ✓ Los alcances y funciones que pueden desempeñar cada usuario dentro del Sistema, para mantener su integridad y seguridad.

- Ofrecer al administrador la facilidad de conocer de los beneficiarios, el nivel de aceptabilidad de las minutas, con ayuda de una escala de calificación establecida por la Sección de Comedores y Cafetería de Bienestar Universitario.

- Proporcionar información estadística sobre:

    - ✓ Las personas presupuestadas con respecto a las que utilizan el servicio.
    - ✓ El grado de satisfacción de los clientes de acuerdo a las calificaciones hechas dentro del sistema.

- Integrar el Sistema de Información, con los demás Sistemas de Información desarrollados por la División de Servicios de Información para la sección de Comedores y Cafetería, y con el Sistema de Información Financiero de la Universidad Industrial de Santander, con el fin de garantizar que se proporcione información actualizada para la División de Bienestar Universitario.

1.2 Justificación

Últimamente se ha notado la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través de la web, enfocada al usuario final. Se trata de aplicaciones que generen colaboración de servicios que reemplacen las aplicaciones de escritorio.

Esta etapa ha definido nuevos proyectos en Internet y está orientada a brindar mejores soluciones para el usuario final. La plataforma Web se está implantando en las organizaciones, los nuevos desarrollos se están realizando enfocados a la Web y la evolución de la informática va en esa dirección.

El sistema con que cuenta la División de Bienestar Universitario para manejar la información de minutas, es un software limitado, no orientado a la web, complejo para su mantenimiento, desarrollado en modo carácter y desactualizado según los nuevos requerimientos de la división de Bienestar Universitario. No se encuentra integrado con los demás sistemas de información de la universidad, en especial con los sistemas de información financiera, y de servicio de comedores y cafetería de Bienestar Universitario, lo cual es importante para poder contar con información actualizada y para la toma de decisiones en cuanto a los servicios que se pueden ofrecer.

La herramienta usada actualmente no cumple con los actuales estándares de desarrollo de la División de Servicios de Información, que tiene como parte de su plan de trabajo, la integración de los sistemas de información de la Universidad, del cual se encuentra actualmente excluido el sistema de minutas por su obsolescencia, poca cobertura, por su rigidez para ser adaptado a los cambios permanentes que se dan y por las limitantes que tiene para su administración y uso.

Con la introducción del Internet y de la Web, se han abierto infinidad de posibilidades en cuanto al acceso y uso de información desde cualquier parte del mundo.

Con los avances en tecnología cada vez se demandan aplicaciones más rápidas, ligeras y robustas que permitan ser utilizadas, sin distinción de lugar u horario, pero llevando la tendencia de integración de servicios y el fácil acceso de la información, que deben ser tenidas en cuenta, para el desarrollo de este sistema de información, con el fin de solucionar las limitantes que tiene el sistema actual.

1.3 Impacto

El proyecto contribuye a la gran integración y actualización de los Sistemas de Información de la universidad; administrados por la División de Servicios de Información de la UIS, y utiliza la implementación de nuevas metodologías, como lo es el desarrollo basado en componentes.

El sistema de información desarrollado agiliza las actividades asociadas a la asignación y control de la parte nutricional de todos los servicios que ofrece la División de Bienestar Universitario, mejora la productividad logrando disminuir el

tiempo de las operaciones y favorece la calidad en los servicios de comedores, cafetería, combos saludables y futuros servicios que deseen ofrecer.

Sin duda, la elaboración del proyecto ayudará a mejorar las condiciones de trabajo de la Sección de Comedores y Cafetería de la universidad y fortalecerá la buena calidad de los servicios brindados a sus beneficiarios para que ellos estén más conformes, esto se logra mediante la facilitación de la elaboración de informes ejecutivos para la Dirección de la Universidad.

SIMIN[2], es una herramienta que cumple con las características de calidad interna y externa del software en el proceso de desarrollo, funcionalidad, confiabilidad, usabilidad, eficiencia, capacidad de mantenimiento y portabilidad, además de la calidad en el uso del software[3], brindando con esto al usuario final un mejor entendimiento, haciendo al software, más atractivo y a los usuarios, más satisfechos.

1.4 Viabilidad

El proyecto se desarrolló en la nueva plataforma JAVA 5 EE, con herramientas de libre distribución como lo son Jboss Developer Studio 3.0 G.A., el servidor de aplicaciones Jboss-seam, productos patrocinados por la compañía Red Hat Software Inc, la cual tiene contrato de soporte con la Universidad Industrial de Santander para brindar una mayor facilidad de uso, éstas herramientas están en continua evolución en calidad y seguridad, debido a la interacción de miles de personas que trabajan en ellas a través de internet y con las que se puede hacer un constante intercambio de conocimientos mediante foros y grupos de discusión para asesoramientos y capacitaciones a desarrolladores.

[2] Sistema de Información para el manejo de Minutas en Bienestar Universitario, UIS.
[3] Norma ISO/IEC 9126 : 2001.

Las herramientas que no son de libre distribución fueron adquiridas y licenciadas por la Universidad Industrial de Santander para su uso.

Se dispuso de las instalaciones adecuadas, equipos necesarios, red e internet, servidor y soporte tecnológico apto para el desarrollo y para la implementación de SIMIN, además el proyecto contó con una permanente supervisión por parte del director del proyecto y de un equipo de ingenieros de la División de Servicios de Información, expertos en las herramientas utilizadas.

2. MARCO TEORICO

2.1 Estructura De La Universidad Industrial De Santander

La Universidad Industrial de Santander, UIS, es una institución oficial, del orden departamental, académica de educación superior, encaminada a la formación del hombre alrededor de los saberes, con cinco facultades en las que se relacionan los campos del conocimiento, armonizando las actividades de docencia, investigación y extensión.[4]

La Universidad cuenta con una Estructura administrativa conformada por Divisiones y dependencias destinadas al buen funcionamiento y cumplimiento de su misión.

2.1.1 *Estructura Organizacional de la UIS*[5]

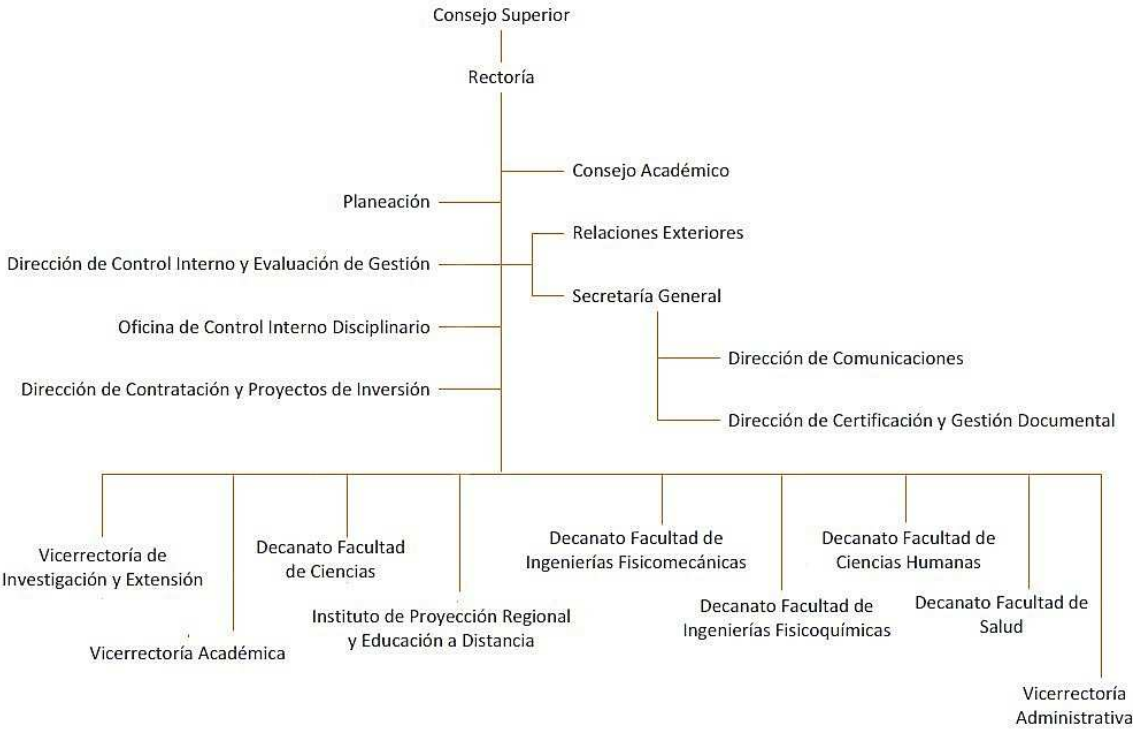
Como institución académica de educación superior enmarca su estructura organizacional en torno a los saberes en cinco facultades: Ingenierías Físico-Mecánicas, Ingenierías Físico-Químicas, Ciencias, Salud y Humanidades se conjugan los campos del conocimiento en los que la Universidad adelanta las actividades de docencia, investigación y extensión.

Las Facultades son unidades académicas y administrativas que agrupan campos y disciplinas afines del conocimiento, profesores, personal administrativo, bienes y recursos, con el objeto de orientar, planificar, fomentar, coordinar, integrar y evaluar actividades de las Escuelas y Departamentos a su cargo, de conformidad con las políticas y criterios emanados del Consejo Superior (máximo órgano de dirección y gobierno de la Universidad) y del Consejo Académico (máxima

[4] Tomado de http://www.uis.edu.co/ en 2011-01-08
[5] Estructura Organizacional de la UIS – http://www.uis.edu.co/ en 2011-01-08

autoridad académica). Cada Facultad está dirigida por el Decano y el Consejo de Facultad y tiene para la orientación, fomento y coordinación de las actividades de investigación y de extensión, un Director de Investigaciones dependiente del Decano.

Figura 1 - Estructura Organizacional de la UIS



Fuente: http://www.uis.edu.co/ en 2011-01-08

Las Escuelas son unidades académicas y administrativas que agrupan uno o varios campos afines del conocimiento y desarrollan programas académicos de pregrado o postgrado, de investigación y de extensión. Cada Escuela tiene un Director quien está asesorado por el Consejo de Escuela y a su cargo se encuentra el personal docente y administrativo adscrito a ésta. Solamente la Escuela de Medicina tiene subdirector, por la cantidad de programas académicos de especialización que maneja.

De la Facultad de Ingenierías Fisicomecánicas dependen las Escuelas de Ingeniería Eléctrica, Electrónica y Telecomunicaciones; Ingeniería Mecánica; Estudios Industriales y Empresariales; Ingeniería Civil; Ingeniería de Sistemas y Diseño Industrial

La Facultad de Ingenierías Fisicoquímicas está conformada por las Escuelas de Ingeniería Química, Ingeniería Metalúrgica, Ingeniería de Petróleos y Geología.
Hacen parte de la Facultad de Ciencias, las Escuelas de: Física, Química, Matemáticas y Biología.

De la Facultad de Salud, las Escuelas de: Medicina, Enfermería, Bacteriología y Laboratorio Clínico, Fisioterapia y Nutrición.

Conforman la Facultad de Ciencias Humanas, las Escuelas de: Trabajo Social, Idiomas, Educación, Artes, Derecho y Ciencia Política, Historia, Filosofía y Economía y Administración.

Los Departamentos son unidades académicas y administrativas dependientes de una Facultad o Escuela, que prestan servicios a una o varias Escuelas y desarrollan programas de investigación y extensión, de conformidad con las políticas y directrices de la Universidad.

Así, el Departamento de Deportes pertenece a la Facultad de Ciencias Humanas, y de la Escuela de Medicina dependen los Departamentos de Ciencias Básicas, Cirugía, Ginecobstetricia, Medicina Interna, Patología, Pediatría, Salud Mental y Salud Pública.

La Dirección General de Regionalización es la encargada de planificar, fomentar, dirigir, coordinar, evaluar y propender por la calidad académica de los programas de regionalización de la Universidad Industrial de Santander. Además, este

organismo se constituye en la instancia correspondiente para la toma de decisiones directamente relacionadas con las sedes, para permitir una mayor agilidad y participación activa de las personas directamente relacionadas con su actividad.

Por su parte, del Instituto de Estudios a Distancia (INSED) unidad académica y administrativa adscrita a la Vicerrectoría Académica, dependen los programas de educación a distancia de la Universidad. El INSED, ofrece además apoyo técnico y logístico para la utilización de metodologías convencionales en las distintas Escuelas. Al frente del Instituto se halla un Director General y los Coordinadores de los Programas Académicos que ofrece.

2.2 Bienestar Universitario[6]

La División de Servicios de Información y la División de Bienestar Universitario y de Servicios de Información, hacen parte del grupo de unidades que dependen de la Vicerrectoría Administrativa de la Universidad Industrial de Santander.

La División de Bienestar Universitario (DBU) es la dependencia administrativa que brinda apoyo directo a la actividad académica, contribuyendo activamente en la formación integral de los estudiantes a través del desarrollo de programas educativos y el ofrecimiento de servicios asistenciales que propenden por el mejoramiento de la calidad de vida de los estudiantes.

[6] Tomado de http://www.uis.edu.co/ en 2011-01-08

Figura 2 - Estructura Organizacional de Vicerrectoría Académica de la UIS

Fuente: http://www.uis.edu.co/ en 2011-01-08

La División de Bienestar Universitario tiene las siguientes funciones[7]:

- Dirigir, orientar, coordinar y ejecutar servicios para el desarrollo integral de los miembros de la comunidad universitaria en un proceso de mejoramiento continuo de la calidad de vida de quienes la conforman.

- Promover la creación de grupos de estudio, artísticos, culturales, deportivos y recreativos.

- Propiciar oportunidades para compartir experiencias.

[7] Funciones aprobadas por Acuerdo Superior Nº 057 de 1994, Universidad Industrial de Santander.

2.2.1 *Misión de Bienestar Universitario*

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.

2.2.2 *Orientación Filosófica*

- Desarrollo integral del individuo.
- Énfasis en lo preventivo y promocional.
- Búsqueda de equilibrio entre una concepción paternalista y la indiferencia social.
- Preponderancia del proceso continuo sobre el asistencialismo aislado.
- Responsabilidad con lo académico y formativo en la prestación de servicios.

2.2.3 *Objetivos de Bienestar Universitario*

- Propiciar un ambiente de humanización.
- Promover el desarrollo de potencialidades.
- Proponer y desarrollar investigaciones.
- Diseñar y ejecutar programas y servicios con orientación preventiva.
- Promover y apoyar propuestas socio-culturales, ecológicas y académicas.
- Favorecer la calidad de vida de la comunidad universitaria.

2.2.4 *Políticas de Bienestar Universitario*

- Propiciar un clima institucional amable.

- Crear espacios propicios para el conocimiento y fortalecimiento del ser social.
- Aumentar la eficiencia y solidez financiera en el área tradicional de los servicios de Bienestar Universitario.
- Articular el trabajo con las unidades académicas y administrativas.
- Establecer relaciones con entidades similares a nivel nacional e internacional.
- Crear e impulsar estrategias para estimular la participación de la comunidad UIS.
- Propiciar la creación de espacios que permitan obtener la opinión y crítica del estudiantado.
- Propiciar la cooperación interinstitucional.

2.2.5 *Estructura Organizacional de la DBU*[8]

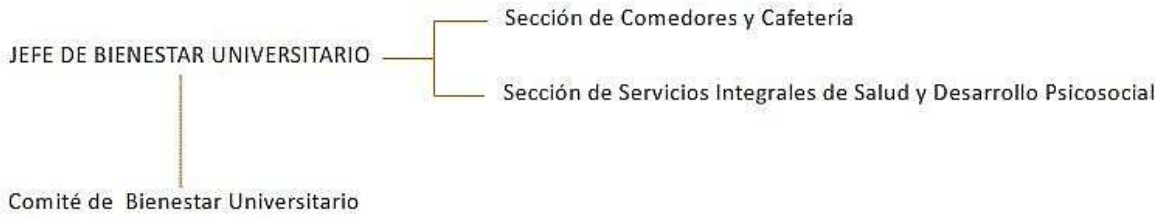
La División de Bienestar Universitario, cuenta con dos secciones para la el cumplimiento de sus funciones, estas son:

- Sección de Comedores y Cafetería, que tiene como misión apoyar a los estudiantes en su desarrollo de sus potencialidades en el área académica, con el suministro de servicio de alimentación que cumpla con los parámetros nutricionales y de calidad estipulados.

- Sección de Servicios Integrales de Salud y Desarrollo Psicosocial, que tiene como misión apoyar a los estudiantes en su actividad académica, mediante el suministro de servicios de salud y de prevención, así mismo apoyar a todos aquellos estudiantes de bajos recursos socioeconómicos.

[8] Tomado de http://www.uis.edu.co/ en 2011-01-08

Figura 3 - Estructura Organizacional DBU



Fuente: http://www.uis.edu.co/ en 2011-01-08

2.3 Sección Comedores Y Cafetería

La sección con la que se puede participar en el proceso de formación integral del estudiante mediante acciones que permitan el logro de hábitos nutricionales adecuados.

2.3.1 *Misión de la Sección de Comedores y Cafetería DBU*[9]

Contribuir al mejoramiento de las condiciones nutricionales de la comunidad universitaria a través de acciones educativas y servicios con calidad.

En la actualidad la sección ofrece los siguientes servicios:

- Cafetería
- Almuerzos Corrientes
- Eventos Especiales
- Comedores
- Combos Saludables

[9] Tomado de http://www.uis.edu.co/ en 2011-01-08

2.4 Servicio De Nutrición

2.4.1 *Planeación*

La planeación de la minuta tiene diferentes objetivos específicos y características nutricionales, de acuerdo con el tipo de servicio de alimentación.

Al momento de planear se debe tener en cuenta que la minuta deba:

- Agradar a los usuarios
- Llenar las necesidades nutricionales de los usuarios.
- Mantenerse dentro de los límites presupuestales definidos.
- Establecer las bases para la organización del servicio.
- Contribuir a crear la imagen del servicio.

2.4.2 *Balanceo Nutricional*

Es indispensable que la Sección de Comedores y Cafetería de BU, garantice que la alimentación que se brinde a la comunidad Universitaria, esté acorde con las necesidades energéticas y nutricionales de los usuarios. El método utilizado por Sección Comedores y Cafetería Bienestar Universitario, para ofrecer los diferentes servicios, es el método de balanceo de dietas normales.

Figura 4 - Recommended Dietary Allowances (RDA)

Recommended Daily Allowances (RDA), 1989

Life-Stage Group	Energy (kcal)	Protein (g)	Vitamin A (mcg RE)	Vitamin E (mg)	Vitamin K (mcg)	Vitamin C (mg)	Iron (mg)	Zinc (mg)	Iodine (mcg)	Selenium (mcg)
INFANTS										
0-6 months	650	13	375	3	5	30	6	5	40	10
7-12 months	850	14	375	4	10	35	10	5	50	15
CHILDREN										
1-3 years	1300	16	400	6	15	40	10	10	70	20
4-6 years	1800	24	500	7	20	45	10	10	90	20
7-10 years	2000	28	700	7	30	45	10	10	120	30
MALES										
11-14 years	2500	45	1000	10	45	50	12	15	150	40
15-18 years	3000	59	1000	10	65	60	12	15	150	50
19-24 years	2900	58	1000	10	70	60	10	15	150	70
25-50 years	2900	63	1000	10	80	60	10	15	150	70
>50 years	2300	63	1000	10	80	60	10	15	150	70
FEMALES										
11-14 years	2200	46	800	8	45	50	15	12	150	45
15-18 years	2200	44	800	8	55	60	15	12	150	50
19-24 years	2200	46	800	8	60	60	15	12	150	55
25-50 years	2200	50	800	8	65	60	15	12	150	55
>50 years	1900	50	800	8	65	60	10	12	150	55
PREGNANCY										
	+300	60	800	10	65	70	30	15	175	65
LACTATION										
1st 6 months	+500	65	1300	12	65	95	15	19	200	75
2nd 6 months	+500	62	1200	11	65	90	15	16	200	75

Fuente: http://www.sparknotes.com/health/assessment/dietaryassessment/section6.rhtml en 2011-01-08

Para que una minuta esté balanceada dietéticamente debe suministrar un número de calorías determinado por factores como: edad, sexo y tipo de actividad de las personas usuarias; en el caso de la Universidad Industrial de Santander, fue definido con base en la recomendación diaria de consumo de calorías y nutrientes de Instituto Colombiano de Bienestar Familiar (ICBF) y de RECOMMENDED DIETARY ALLUWANCES(RDA) de la academia nacional de ciencias de Estados Unidos, sacando una media entre los valores correspondientes al rango de edades entre 19 y 24 años, tanto de mujeres, como hombres, obtenemos un valor calórico de referencia con un margen de error de más o menos 100 calorías; el cual comprende el valor calórico de referencia obtenido para la población universitaria.

Tabla 1 - Guía alimentaria para la población colombiana (ICBF)

GRUPO	RANGO EDAD	REQUERIMIENTOS
HOMBRES	18-24 años	3000 Kcal
MUJERES	18-24 años	2250 Kcal

Se calculan los porcentajes requeridos para cada grupo etáreo:

Tabla 2 - Grupos etáreos Población Beneficiarios

GRUPO	PORCENTAJE
HOMBRES	60%
MUJERES	40%

Cantidad Referencia:

Hombres: 60% de 3000 Kcal. =	1800	Kcal
Mujeres: 40% de 2250 Kcal=	900	Kcal
Cantidad de Referencia	2700	Kcal

Cantidad de referencia que es usada para distribuirla entre los macronutrientes necesarios en las minutas.

Como fue previamente indicado, cada servicio o programa que ofrece la DBU, como lo son: comedores, cafetería, combos, etc, ofrece un número de tipos de servicio como lo son: desayuno, almuerzo, cena, etc, cada uno con unas características nutricionales especiales y que cumplen con el requisito de Cantidad de referencia de calorías para un día.

El sistema permite crear tantos servicios como el Administrador de SIMIN lo desee, y crear aquellas especificaciones nutricionales con las que debe cumplir el servicio.

El Sistema, SIMIN, ofrece la facilidad de elaborar la Minuta Patrón propia para cada servicio con el objetivo de hacer una correcta distribución de las propiedades nutricionales requeridas.

2.4.3 *Cálculos de la Dieta*

La distribución del valor de referencia de calorías necesarias para una correcta dieta se hace dentro de los tres macro nutrientes, Carbohidratos, Proteínas y Grasas.

Para la realización del balanceo para una población normal, los porcentajes y valores de los nutrientes se han definido de la siguiente manera:

Tabla 3 - Fórmula Sintética

NUTRIENTES PRINCIPALES	RANGO PERMISIBLE	PORCENTAJE	CALORIAS/GRAMO DE NUTRIENTE
CARBOHIDRATOS	55-60	58	4
PROTEINAS	12-15	16	4
GRASAS	25-30	26	9

2.5 Esquema De Desarrollo

2.5.1 *Aplicaciones Orientadas a la WEB*

En los primeros días de la Web, los sitios Web consistían de páginas estáticas, permitiendo una interacción limitada con el usuario. Al comienzo de los años 90, estas limitaciones fueron superadas cuando los servidores Web fueron reemplazados para permitir comunicaciones a través del desarrollo de fragmentos de código que eran ejecutados del lado del servidor. A partir de entonces las aplicaciones dejaron de ser estáticas y solamente editadas por aquellos “gurúes” del HTML y se permitieron a usuarios normales interactuar con las aplicaciones por primera vez.

Este fue un paso fundamental para llegar a la Web que hoy en día conocemos. Sin la interacción no existiría el comercio electrónico, el Web-mail, Internet-banking, blogs, comunidades online.

2.5.1.1 La WEB como Sistema de Información[10]

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

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

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

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

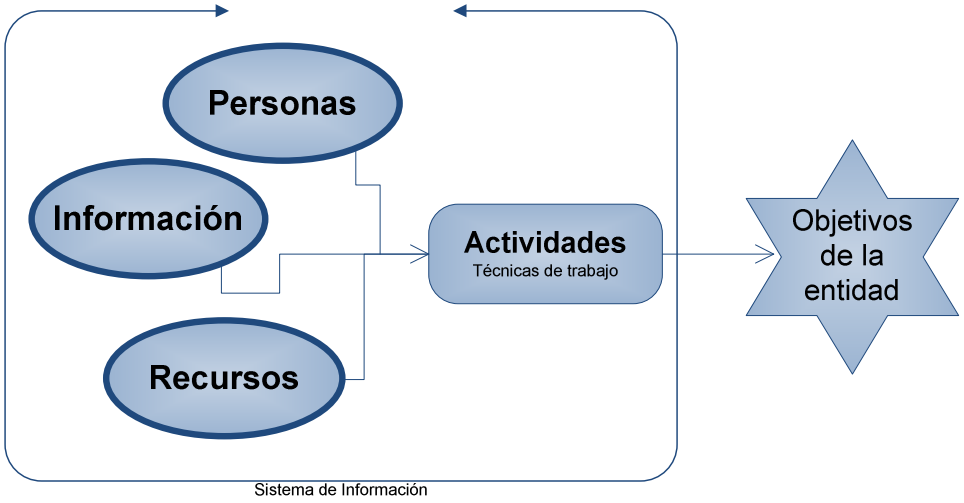
Todos estos elementos interactúan entre sí para procesar los datos (incluyendo procesos manuales y automáticos) dando lugar a información más elaborada y

[10] Rodríguez Perojo Keilyn, Ronda León Rodrigo. El Web como Sistema de Información. http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm en 2010-12-12

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

Normalmente el término es usado de manera errónea como sinónimo de sistema de información informático, en parte porque en la mayoría de los casos los recursos materiales de un sistema de información están constituidos casi en su totalidad por sistemas informáticos, pero siendo estrictos, un sistema de información no tiene por qué disponer de dichos recursos (aunque en la práctica esto no suela ocurrir). Se podría decir entonces que los sistemas de información informáticos son una subclase o un subconjunto de los sistemas de información en general.

Figura 5 - Elementos de un Sistema de Información



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

2.5.1.2 Tecnologías Web.[11]

Inicialmente, era difícil la construcción de aplicaciones sofisticadas. La primera generación de aplicaciones Web era primitiva, en general basada en formularios con información y aplicaciones de búsqueda. Incluso estas aplicaciones básicas requerían de un alto seniority para su construcción.

A través del tiempo, el conocimiento necesario para construir aplicaciones ha sido reducido. Hoy en día, es relativamente sencillo construir aplicaciones sofisticadas utilizando las modernas plataformas y lenguajes, como pueden ser PHP, .NET o Java.

Primera generación – CGI:
Common Gateway Interface (CGI) fue la tecnología reinante desde aproximadamente 1993 hasta fines de los '90 cuando los lenguajes de scripting comenzaron a ganar importancia.

CGI trabaja encapsulando la información provista por el usuario en variables de ambiente. Estas luego son accedidas por scripts o programas desarrollados comúnmente en Perl o C. Estos programas procesan la información provista por los usuarios, y luego envían código HTML con la información procesada a la salida estándar, que a su vez es capturada por el servidor Web y pasada al usuario.
Scripting:

La falta de manejos de sesiones y control de autorización por parte de CGI impidió el desarrollo de aplicaciones Web comerciales con esa tecnología.
Los desarrolladores Web comenzaron entonces a utilizar lenguajes de script, como JavaScript o PHP para resolver esos problemas. Básicamente los lenguajes

[11] López Gustavo, Blanco Ignacio. Tesis en Ingeniería Informática.
http://materias.fi.uba.ar/7500/blanco-tesisingenieriainformatica.pdf en 2010-10-09

de script son ejecutados en el servidor Web y como son no compilados son desarrollados e implementados más fácilmente.

Los lenguajes de script tienen algunas desventajas:

- La mayoría de los lenguajes no son tipados y no promueven buenas prácticas de programación.
- Son más lentos en comparación con los lenguajes compilados (a veces hasta 100 veces más lentos).
- Es difícil (no imposible) escribir aplicaciones de múltiples capas porque en general las capas de presentación, aplicación y datos residen en la misma máquina, limitando de esta forma la escalabilidad y seguridad.
- La mayoría no soporta nativamente métodos remotos o llamadas a Web services, lo que hace difícil la comunicación entre servidores de aplicación y con Web services externos.

De cualquier manera a pesar de las desventajas aplicaciones grandes y frecuentemente accedidas han sido desarrolladas utilizando lenguajes de script, como eGroupWare (egroupware.org), que está escrita en PHP. Además muchas aplicaciones de Internet banking han sido desarrolladas en ASP.

Los lenguajes de script incluyen, ASP, Perl, Cold Fusion y PHP. De cualquier manera, muchos de esos podrían ser considerados como lenguajes interpretados híbridos, en particular las últimas versiones de PHP y Cold Fusion.

2.5.1.3 Plataformas de Desarrollo WEB

Una vez que los lenguajes de script alcanzaron los límites de performance y escalabilidad, los proveedores más grandes evolucionaron hacia la plataforma de Sun J2EE y cuya evolución se encuentra en JEE6 y la de Microsoft .NET.

J2EE:

- Utiliza el lenguaje Java para producir aplicaciones Web.
- Permite la creación de grandes aplicaciones distribuidas.
- Provee un buen control de sesión y manejo de autorización.
- Permite la creación de aplicaciones de múltiples capas.

Una de las desventajas de J2EE es que posee una curva de aprendizaje importante, lo que provoca una difícil inserción de diseñadores Web y programadores en sus primeros pasos.

Microsoft actualizó su tecnología ASP a ASP.NET que imita a J2EE en muchas maneras.

- Simplifica la creación de aplicaciones pequeñas a programadores que se están iniciando y a diseñadores gráficos.
- Permite la creación de grandes aplicaciones distribuidas.
- Provee un buen control de sesión y manejo de autorización.
- Permite a los programadores la utilización de su lenguaje de programación favorito, el que es compilado a código nativo.

La elección entre J2EE y .NET es dependiente de la plataforma. Las aplicaciones J2EE teóricamente pueden ser ejecutadas en la mayoría de las plataformas, desde Linux a AIX, MacOS X o Windows.

2.5.1.4 WEB 2.0

El concepto original de la Web (en este contexto, llamada Web 1.0) eran páginas estáticas HTML que no eran actualizadas frecuentemente. El éxito de las punto com dependía de webs más dinámicas (a veces llamadas Web 1.5) donde los

CMS servían páginas HTML dinámicas creadas al vuelo desde una actualizada base de datos. En ambos sentidos, el conseguir hits (visitas) y la estética visual eran considerados como unos factores muy importantes.

Los propulsores de la aproximación a la Web 2.0 creen que el uso de la Web está orientado a la interacción y redes sociales, que pueden servir contenido que explota los efectos de las redes con o sin crear webs interactivas y visuales. Es decir, los sitios Web 2.0 actúan más como puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales.

2.5.2 *Diseño Con el Estandar UML*[12]

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

2.5.2.1 Introducción a UML

UML se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.
Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

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

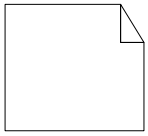
Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE (Object-oriented software engineering). UML ha puesto fin a las llamadas “guerras de métodos”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

2.5.2.2 Elementos comunes a todos los diagramas

Nota

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

Figura 6 - Representación de la Nota (UML)



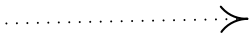
Una nota se representa como un rectángulo con una esquina doblada con texto en su interior.

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

Dependencias

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

Figura 7 - Representación de la Dependencia (UML)



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

2.5.2.3 Diagramas de Casos de Uso

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

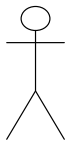
Elementos del diagrama de casos de uso

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

Actores

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

Figura 8- Representación de un Actor en UML

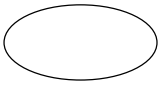


Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).

Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica.

Figura 9 - Representación de un Caso de Uso en UML



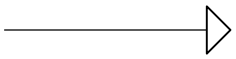
Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre Casos de Uso

Entre dos casos de uso puede haber las siguientes relaciones:

- ✓ Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.
- ✓ Uso: Cuando un caso de uso utiliza a otro.

Figura 10 - Representación de una relación entre Casos de Uso en UML



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

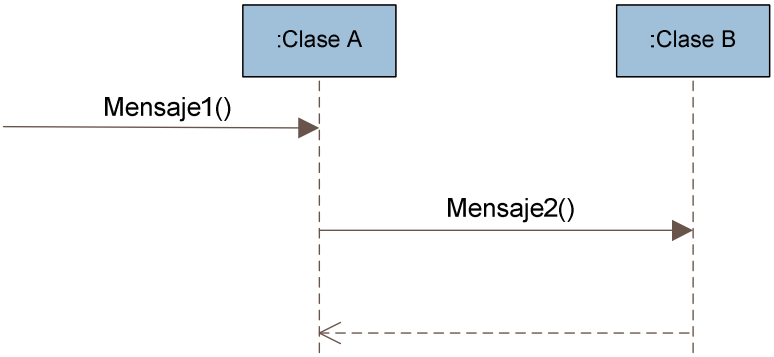
En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

2.5.2.4 Diagramas de Secuencia

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

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

Figura 11 - Representación de un Diagrama de Secuencia (UML)



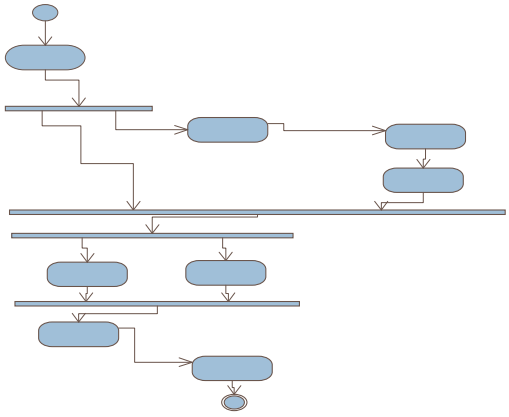
Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

2.5.2.5 Diagrama de Actividades

La idea es generar una especie de diagrama Pert, en el que se puede ver el flujo de actividades que tienen lugar a lo largo del tiempo, así como las tareas concurrentes que pueden realizarse a la vez. El diagrama de actividades sirve para representar el sistema desde otra perspectiva, y de este modo complementa a los anteriores diagramas vistos.

Gráficamente un diagrama de actividades será un conjunto de arcos y nodos.
Desde un punto de vista conceptual, el diagrama de actividades muestra cómo fluye el control de unas clases a otras con la finalidad de culminar con un flujo de control total que se corresponde con la consecución de un proceso más complejo.

Figura 12 - Representación de un Diagrama de Actividades (UML)



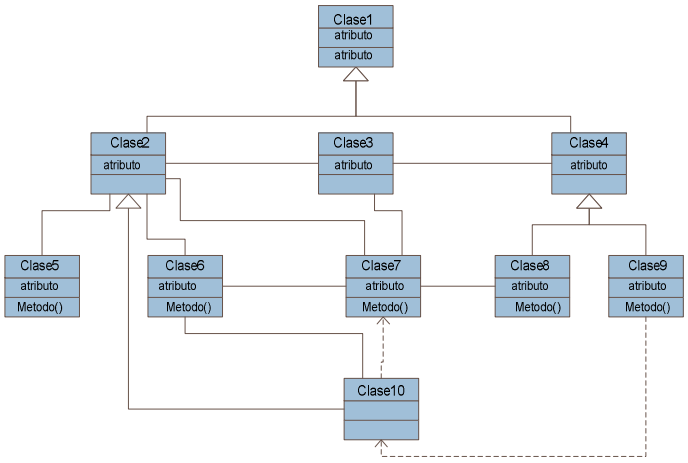
Por este motivo, en un diagrama de actividades aparecerán acciones y actividades correspondientes a distintas clases. Colaborando todas ellas para conseguir un mismo fin.

2.5.2.6 Diagrama de Clases

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

Representación de: - Requerimientos en entidades y actuaciones. - La arquitectura conceptual de un dominio - Soluciones de diseño en una arquitectura - Componentes de software orientados a objetos

Figura 13 - Representación Diagrama de Clases (UML)



- Propiedades también llamados atributos o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.
- Operaciones comúnmente llamados métodos, son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc.
- Interfaz es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo.
- Herencia se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una

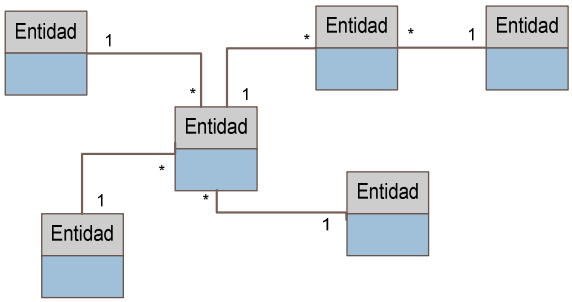
persona puede especializarse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además cada uno tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

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

2.5.2.7 Diagrama Entidad-Relación[13]

El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Figura 14 - Representación Diagrama Entidad - Relación



Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos

[13] Fuente: http://www3.uji.es/~mmarques/f47/apun/node83.html en 2011-01-08

compuestos y las jerarquías de generalización, en lo que se ha denominado *modelo entidad-relación extendido*.

El Modelo Entidad-Relación.

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

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

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

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

2.5.3 *Java Enterprise Edition 5.0 (JEE 5)*

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.

2.5.3.1 Seguridad

La plataforma Java EE dispone de reglas estándar para controlar el acceso. Estas son definidas por el desarrollador y se implementan al desplegar la aplicación en el servidor.

Java EE pone a disposición de los desarrolladores mecanismos de acceso predefinidos para que ellos no tengan que implementarlos en sus aplicaciones. Una sola aplicación trabaja en diversos ambientes de desarrollo sin necesidad de modificar el código fuente.

2.5.3.2 Componentes Java EE

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

La plataforma Java EE define los siguientes componentes:
Aplicaciones cliente y Applets, estos componentes se ejecutan en el cliente.
Los componentes Web:Servlets, JavaServer Faces, y tecnología JavaServerPagesTM (JSPTM);se ejecutan en el servidor.

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

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

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.

2.5.3.3 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.

2.5.3.4 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.

2.5.3.5 Ensamble 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.

2.5.3.6 Empaquetado de aplicaciones

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

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

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

2.5.4 *Tecnología Servlet Java*

2.5.4.1 Servlet

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

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

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.

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.

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.

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.

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.

2.5.5 *Java Server Faces*

2.5.5.1 Características principales

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

Los principales componentes de la tecnología JavaServer Faces son:

- Una API y una implementación de referencia para:
    - Representar componentes de interfaz de usuario (UI-*User Interface*) y manejar su estado.
    - Manejar eventos, validar en el lado del servidor y convertir datos.
    - Definir la navegación entre páginas.
    - Soportar internacionalización y accesibilidad.
    - Proporcionar extensibilidad para todas estas características.

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

Este modelo de programación bien definido junto con la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y

mantenimiento de aplicaciones web con UIs en el lado servidor. Con un mínimo esfuerzo, es posible:

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

- Mapear componentes UI a una página de datos en el lado servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.

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

2.5.5.2 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.

2.5.5.3 Definición JavaServer Faces

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

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

• Páginas, (principalmente páginas JSP).
• Clases de utilidad del lado del servidor, como beanspara acceder a las bases de datos.

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

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

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

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

• **Autores de páginas**, que utilizan un lenguaje de marcas, como HTML, para construir páginas para aplicaciones web. Cuando se utiliza la tecnología

JavaServer Faces, los autores de páginas casi siempre usarán exclusivamente la librería de etiquetas.

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

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

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

2.5.6 *Seam*

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

- Un tipo de cosas

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

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

- Integrar JSF con EJB 3.0

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

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

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

- Integrar AJAX

Seam soporta mejor las soluciones de código abierto basado en AJAX JSF: JBossRichFaces e ICEfaces. Estas soluciones le permiten agregar la capacidad de AJAX para la interfaz de usuario sin necesidad de escribir código JavaScript.
Por otra parte, seam proporciona una capa incorporada del Javascript que le permite llamar a los componentes JavaScript de forma asincrónica del lado cliente sin la necesidad de una capa de acción intermedia.

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

- Procesos de negocio como el primer constructor de clase

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

- Biyección

La biyección es dinámica y bidireccional, podríamos pensar en esto como un mecanismo de alias para variables contextuales (nombres en alguno de los contextos enlazados al hilo de ejecución actual) a atributos del componente.
La biyección permite auto ensamblaje de componentes con estado por el contenedor, esto incluso permite a un componente asegurar y fácilmente manipular el valor de una variable contextual, simplemente asignándola a un atributo del componente

- Preferir anotaciones a XML

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

- La prueba de integración es fácil

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

- Las especificaciones técnicas no son perfectas

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

- Hay más de una aplicación web que sirve páginas HTML

Un framework de aplicaciones web realmente completo debe abordar problemas como la persistencia, la concurrencia, a sincronía, administración del estado, seguridad, correo electrónico, mensajería, pdf, generación de gráficos, servicios web, caché, entre otros.
Seam integra JPA e Hibernate3 para persistencia, EJB TimerService y Quartz para ligeras a sincronías, jBPM para flujo de trabajo, JBoss rules para reglas del negocio, Meldware Mail para correo electrónico, HibernateSearch y Lucene para búsqueda de texto, JMS para mensajes y JBoss Caché para la página de almacenamiento en caché.

2.5.7 *Servidor De Aplicaciones – JBoss*

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

Es reconocido por soportar los estándares más recientes. De hecho, es el primer servidor de aplicaciones en alcanzar la certificación J2EE 1.4 cuando salió su versión 4.0 (actualmente va en la versión 5, liberada a finales de 2008). Pero JBoss no sólo marca la pauta en la adopción de estándares con su servidor de aplicaciones, sino en la imposición de los mismos. Recientemente se le eligió para hacer parte del Java Community Process (JCP). Además en los últimos años ha estado a la cabeza del desarrollo de Java Enterprise llegando a establecerse en todas las especificaciones de requerimientos de Java (Java Specification Requests, JSRs).

2.5.8 *Enterprise Architect*

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

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

2.5.9 *Construcción de Prototipos de Software*

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

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

Ventajas:

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

En sistemas grandes y complejos una forma de resolver la dificultad de evaluación es utilizar un enfoque evolutivo para el desarrollo de sistemas. Esto significa proporcionar al usuario un sistema incompleto y después modificarlo y aumentarlo en el momento en que los requerimientos del usuario sean claros.
El enfoque de construcción de prototipos desechables es para ayudar a refinar y clasificar la especificación del sistema. El prototipo se escribe, evalúa y modifica.

La evaluación del prototipo informa del desarrollo de la especificación detallada del sistema que se incluye en el documento de requerimientos de este. Una vez que se ha redactado la especificación, el prototipo ya no es útil y se desecha.

2.5.9.1 Prototipo No funcional[14]

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

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

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

2.5.9.2 Axure RP Pro 5.6

Axure RP es una herramienta completa para la creación rápida de WireFrames, prototipos y especificaciones para aplicaciones y sitios Web. La presentación, pantallas interactivas, ayudan a comunicar el diseño claramente, y colabora en la retroalimentación con los clientes, identificando claramente los requisitos que por una u otra manera se pasaron por alto en las primeras entrevistas.
Con Axure RP se pudo crear un prototipo interactivo basado en el navegador, sin necesitar programación de código.

Axure muestra de manera gráfica el esquema de navegación del Sistema, brindando widgets y utilidades con las que el público en general está familiarizado, además de aportar librerías de elementos creados para rapidez en la elaboración del prototipo y facilitar la introducción del cliente al sistema.

Las características principales de Axure RP son [15]:

- Generar prototipos interactivos HTML con el clic de un botón, sin una sola línea de código.

- Muestra una amplia gama de interacciones en su diseño como base de unión, expansión y contracción de contenido.

- Comparte sus prototipos HTML con otros compañeros y clientes para ser observados en los navegadores más populares sin necesidad de instalar para observar el prototipo.

Es una herramienta distribuida por Axure Software Solutions, Inc, empresa privada dedicada al desarrollo de software, la versión piloto de Axure RP que es su producto líder, fue en enero de 2003.

[15] Tomado de: http://www.axure.com/ en 2010-12-12

3. METODOLOGÍA DE DESARROLLO

A continuación se hace una descripción de las diferentes actividades que se llevaron a cabo durante el transcurso del proyecto, buscando aplicar la metodología que se estableció en el desarrollo del nuevo Sistema de información orientado a la web, como soporte para el proceso del manejo de las minutas ofrecido por la sección de cafetería y comedores de la División de Bienestar Universitario de la Universidad Industrial de Santander.

3.1 Ciclo De Vida Del Proyecto

3.1.1 *Análisis de Requerimientos*

El proceso de análisis de requerimientos se especificó junto con los funcionarios de la Sección de Comedores y Cafetería de la DBU y bajo la supervisión del director del proyecto.

Para el análisis de requerimientos se fijaron y se realizaron reuniones de manera periódica, junto con los funcionarios de la Sección de Comedores y Cafetería de la DBU, ingenieros de la DSI y en algunas ocasiones directivos de la División de Bienestar Universitario.

El análisis de requerimientos permitió conseguir una visión amplia de las necesidades de la Sección de Comedores y Cafetería, conocer sus procesos y las posibles mejoras que se podrían implementar con la nueva herramienta, y finalmente se recibió toda la información y requerimientos para traducirlos posteriormente en datos, arquitectura y diseño.

Durante unas reuniones posteriores, se fijaron los casos de uso, el diagrama de clases y diagrama de secuencia, todos ellos evaluados y aprobados mediante el prototipo no funcional, por los funcionarios de la División de Bienestar Universitario en reuniones programadas conjuntamente.

3.1.2 *Diseño*

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

En esta etapa de diseño se realizó una traducción de los requisitos a una representación del software donde se pudiera evaluar su calidad previamente al inicio de la codificación.

El diseño se efectuó, mediante modelos UML (Lenguaje de Modelado Unificado) que incluyeron los diagramas que han sido seleccionados dentro de los estándares de desarrollo de software utilizados en la División de Servicios de Información, que son:

- Diagramas de Clases
- Diagramas de Casos de Uso
- Diagramas de Secuencia

Y como consecuencia del Diagrama de Clases se elaborò el Diagrama Entidad-Relación, para el inicio de la programación. Se utilizó la herramienta de modelaje Enterprise Architect licenciada por la Universidad Industrial de Santander a través de División de Servicios de Información.

3.1.3 *Implementación de la Aplicación*

El diseño se debe traducir en una forma legible por la máquina. El paso de generación de código se llevó a cabo en esta tarea.
En esta etapa, se desarrolló el software que se ha diseñado, teniendo en cuenta los parámetros establecidos por la División de Servicios de Información, en cuanto a los estándares técnicos y de calidad que caracterizan las aplicaciones que son generadas para el servicio de la Universidad, teniendo como base el Lenguaje de programación JAVA 5, frameworks como: seam, java server faces (JSF), Enterprise Java Beans (EJB 3.0) e Informix como motor de base de datos.

3.1.4 *Pruebas del software*

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

Las pruebas se integraron dentro de las diferentes fases del ciclo de desarrollo del software, fueron de carácter permanente a lo largo del desarrollo del proyecto por parte del equipo de trabajo.

Una vez generado el código, comienzan las pruebas, proceso utilizado para identificar posibles fallos de implementación, calidad, o usabilidad de un programa.

Las pruebas se centran en los procesos lógicos internos del software, asegurando que todas las sentencias sean probadas y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

3.1.5 *Ajustes*

Los ajustes, son el proceso de mejora del sistema desarrollado e implementado, tomando como base las sugerencias y observaciones que se plantearon en el proceso anterior, las pruebas, aquí también se incluyen las correcciones a los posibles fallos detectados, mejoras en la interfaz y el diseño de la misma.
Los ajustes están incluidos dentro del proceso iterativo de refinamiento que se planteó para al sistema, para adaptarse plenamente a las necesidades del cliente Sección de Comedores y Cafetería de la División de Bienestar Universitario.

Los ajustes se realizaron de manera permanente a lo largo del desarrollo del proyecto, a la par con el proceso de las pruebas, en la medida en que se detectaron errores o inconsistencias en lo desarrollado.

3.2 Metodología De Desarrollo Del Proyecto

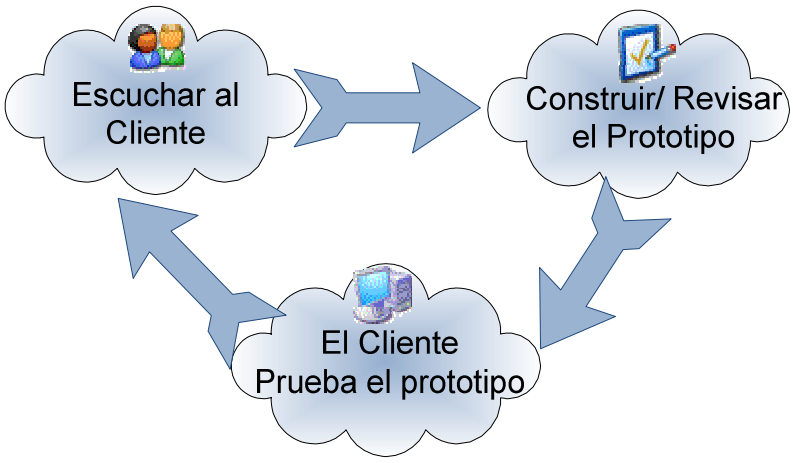
Teniendo como base las actividades anteriormente descritas, se utilizó la metodología de desarrollo Modelo de Construcción por Prototipos, para la implementación de éste proyecto.

3.2.1 *Modelo de Construcción de Prototipos*

Se consideró esta metodología, para alcanzar una mejor definición del conjunto de objetivos generales para el software solicitado por la Sección de Comedores y Cafetería de Bienestar Universitario, y para definir aquellos 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.

Figura 15 - Modelo de Construcción de Prototipos



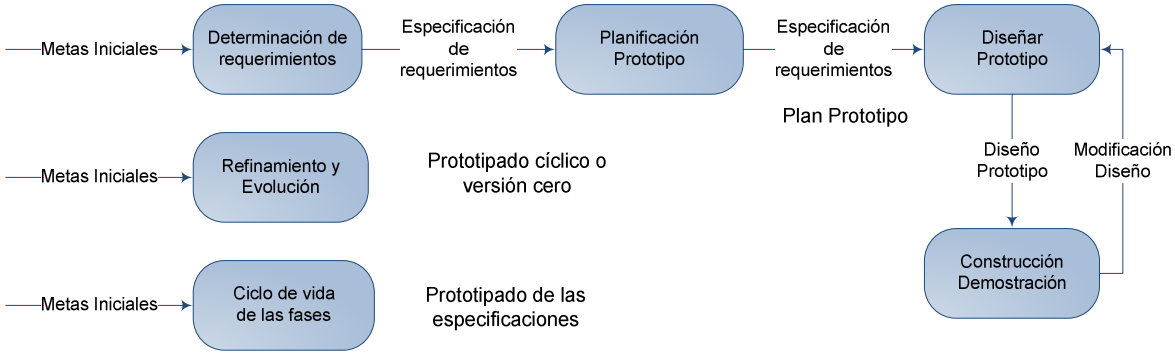
3.2.2 *Estructura del Modelo de Construcción de Prototipos*

Esta metodología resultó viable para el desarrollo del proyecto y se escogió debido a las siguientes razones:

- En la creación de los prototipos iniciales se debe trabajar con unas ideas aproximadas de lo que desea la Sección de Comedores y Cafetería de 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.

Figura 16 - Estructura del Modelo de Construcción de Prototipos



- 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.

3.2.3 *Procedimiento Para La Metodología Planteada*

- La construcción de prototipos comienza con la recolección de los requisitos, con la Jefe de la sección de Comedores y Cafetería de la División de Bienestar Universitario.

- El grupo de desarrollo y el usuario se reúnen y definen los objetivos globales del software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesaria una mayor definición.

- Se produce el diseño del prototipo que se enfoca sobre la representación de los aspectos del software visibles al usuario (métodos de entrada, esquemas de navegación y formatos de salida). En ésta etapa el usuario adquiere mayor claridad de las funcionalidades del sistema y da las sugerencias necesarias para la construcción de un prototipo más avanzado.

- Se procede con la construcción del prototipo que se ha propuesto.

- El prototipo es evaluado por el usuario y se utiliza para refinar los requisitos del software a desarrollar.

- Se produce un proceso iterativo en el que el prototipo es “afinado” (Refinamiento del prototipo) para que satisfaga las necesidades del usuario, al mismo tiempo que facilita al que lo desarrolla una mejor comprensión de lo que hay que hacer y poder entregar el producto final requerido.

3.3 Plan de Trabajo

El plan de trabajo del proyecto se precisó por FASES y con éstas pudo ser medido el nivel de avance alcanzado por el proyecto, dichas fases se encuentran descritas a continuación:

3.3.1 *Fase Previa: de Acoplamiento*

- Esta fase consiste en la realización de reuniones previas de usuarios con el grupo desarrollador, para dar a conocer los alcances del proyecto y definir los objetivos.

- Se realizará el planteamiento de prioridades sobre las cuales se va a trabajar y establecer la metodología y reglas de trabajo a utilizar dentro de la División de Servicios de Información.

3.3.2 *Primera Fase: Recolección Y Análisis de Requerimientos*

- Se procederá con reuniones periódicas con los funcionarios de la Sección de Comedores y Cafetería de la División de Bienestar Universitario de la Universidad Industrial de Santander, con el objetivo de establecer las necesidades que se quieren satisfacer con el desarrollo del proyecto.

- En esta Fase se debe establecer todo lo que el sistema debe realizar, y todas las restricciones sobre la funcionalidad que se deben tener en cuenta.

- Se especificarán el conjunto completo de resultados a ser obtenidos al utilizar el sistema.

- La importancia de esta fase radica en que se debe mostrar a los desarrolladores y a los usuarios qué se necesita del sistema, en un lenguaje que todos comprendan, por lo tanto es primordial la excelente comunicación entre usuarios y desarrollador.

- Al terminar esta fase se debe tener un modelo completo que represente el sistema total en algún nivel de abstracción mental que puede ser especificada

a nivel de detalle en un documento denominado “Especificación de Requerimientos”.

3.3.3 *Segunda Fase: Diseño del Prototipo*

- Teniendo la especificación de los requerimientos en un lenguaje natural a nivel de detalle, se procede entonces a abstraer esa información y traducirlo en lenguaje de modelado unificado (UML).

- Dentro de ésta fase se definirá la interfaz del usuario y la interfaz de comunicación entre los programas.

- En esta etapa se deberá evaluar el grado de aceptación y satisfacción del usuario frente a los modelos de entrada y salida de datos y la interfaz de usuario que se ha propuesto.

- Se podrá volver a llegar a esta fase durante el proceso de refinamiento del prototipo en caso de ser necesarias mejoras en los diseños planteados en el primer prototipo.

3.3.4 *Tercera Fase: Desarrollo del Prototipo*

- Durante esta fase se realizará la programación de los requerimientos previamente modelados.

- Se llegará a esta fase cada vez que sea necesario una nueva iteración o un nuevo proceso de refinamiento del sistema, en donde se deberá implementar el nuevo prototipo.

3.3.5 *Cuarta Fase: Refinamiento del Prototipo*

- Dentro de esta fase se llevará a cabo la corrección de las fallas que surgieron tras la realización de las pruebas funcionales de las secciones de automatización de los módulos o de la realización de la evaluación por parte del usuario en cuanto al prototipo y su aceptación o de nuevas sugerencias para un mejoramiento o maduración del prototipo.

- Se realizarán los ajustes y refinación del funcionamiento de los servicios, regresando a las fases de diseño y desarrollo hasta obtener el producto final.

- En todo caso, se iniciará una nueva iteración partiendo de este punto, cuando el usuario considere que se deben hacer mejoras, o atender sugerencias para la maduración del prototipo.

3.3.6 *Quinta Fase: Implantación y Entrega del Producto Final*

- Implantación del Sistema de información orientado a la web como soporte para el proceso del manejo de las minutas ofrecido por la Sección de Cafetería y Comedores de la División de Bienestar Universitario, integrado con el Sistema Financiero de la Universidad Industrial de Santander.

- Capacitación de los Usuarios en el funcionamiento del sistema que se está entregando.

- Socialización del sistema en la comunidad universitaria.

PARTE II. DESARROLLO DEL SISTEMA

4. APLICACIÓN DE LA METODOLOGÍA

4.1 Levantamiento De Requerimientos

Se presentan los requerimientos, generados después de las primeras reuniones con el cliente, en donde se pudo abstraer la idea principal de lo que se quiere alcanzar durante el desarrollo del proyecto, que sirvió como base para las primeras etapas del análisis y diseño de SIMIN.

4.1.1 *Descripción General*

Se permitió identificar la naturaleza y funciones del proceso, establecer el conjunto de requerimientos, y obtener un mayor conocimiento de la Sección de Comedores y Cafetería de la DBU.

Luego de una revisión del sistema actual de minutas, en el que solo se puede usar para el servicio de comedores, se decidió comenzar desde cero, por estar enfocado de manera diferente los objetivos del antiguo sistema, al sistema SIMIN.
Se estudiaron los procesos con ayuda de los funcionarios de la DBU y se establecieron las funciones principales del sistema.

4.1.2 *Funciones del Sistema*

Elementos de Bienestar:
Es el módulo de registro de un elemento en el almacén de Bienestar, trayéndolo del Sistema Financiero, relacionándolo a datos propios de SIMIN, como grupo

alimenticio, valores nutricionales de las propiedades nutricionales que han sido asociados al elemento.
En el módulo se puede crear Elementos, modificarlos y Eliminarlos.

Recetas:
Es el módulo de administrar las recetas que maneja el sistema, asociando los elementos que se utilizan para realizarlas, el procedimiento que se debe seguir para armar el plato especifico, relacionarla con un tipo de receta, y especificar los valores nutricionales de la misma.
En el módulo se puede crear Recetas, modificarlas e Inactivarlas.

Menús:
Es el módulo que se encarga de administrar los Menús que maneja SIMIN, en él se especifica las recetas para armar el menú y se le asigna una descripción para hacer más fácil su selección a la hora de hacer la programación de las minutas, resumidamente se totalizan los valores nutricionales del menú. Se le asocia, también una imagen del menú para hacer su promoción en la página.
En el módulo se puede crear Menús, modificarlos e Inactivarlos.

Servicios:
Es el módulo de registro de un nuevo servicio ofrecido por Bienestar Universitario para toda la comunidad universitaria. Los servicios pueden ser activados o inactivados.
A cada servicio se le asocia un fondo presupuestal y los tipos de servicio a ofrecer, el número de calorías permitido para el tipo de servicio y las observaciones pertinentes.

Programar Minuta:

Para la programación de una minuta, primero se selecciona la fecha, el servicio y el tipo de servicio a programar, se le asigna el número de beneficiarios, en el caso del servicio Comedores, se trae el número de beneficiarios desde el Sistema de Comedores, para sugerir el número de beneficiarios de acuerdo con los adjudicados para dicho servicio.

Se asigna el menú para este día y los adicionales que desea proporcionar para acompañar el menú, si así lo requiere. Quedan visibles los totales nutricionales para mantener el equilibrio para el servicio y el tipo de servicio que se requiere programar.

Se ofrece la programación de minutas que ya han sido utilizadas en semanas anteriores.

Almacenes:

Es el inventario de los registros de elementos que se encuentran en el almacén bienestar, es útil para conocer la disposición que hay de elementos, es por eso que para cada elemento se requiere especificar su costo unitario, su existencia en el almacén y su existencia en el almacén intermedio para saber que tanto hay disponible.

Entrada de Sobrantes:

En este módulo se hace el ingreso de aquellos elementos que salieron del almacén Bienestar pero que no fueron usados en la minuta para la que fueron solicitados, se necesita que el almacenista haga el ingreso al almacén intermedio, que es el que corresponde a los productos sobrantes, con el fin de hacer un ahorro en el consumo y evitar el desperdicio de elementos que pueden ser perfectamente aprovechados en nuevas minutas.

Estadísticos:

Es el módulo en el que se muestran los estadísticos que proporciona el sistema, consultas de elementos utilizados en determinado rango de fecha, resultado de la encuesta de satisfacción y estadístico de la cantidad de Beneficiarios que realmente reciben los servicios ofrecidos.

Tablas de Soporte:

Son todas aquellas tablas importantes para el sistema y que se necesitan llenar previamente para que los módulos funcionen correctamente.

Dentro de las Tablas soporte se encuentran:

- Grupos Alimenticios
- Alimentos Equivalentes
- Unidades de Medida
- Conversiones
- Propiedades Nutricionales
- Grados de Satisfacción

4.2 Estándares De La División De Servicios De Información.[16]

4.2.1 *Aspectos Generales*

Interfaz de desarrollo:

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

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

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

Servidor de aplicaciones:

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

JAVA:

La versión del compilador de JAVA debe ser la 1.5, la cual debe ser instalada en el directorio C:\java1.5.

SEAM:

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

Espacio de trabajo:

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

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

[Sistema]Entidades

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

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

[Sistema]JPA

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

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

Servidor de versiones:

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

subclipse-site-1.4.7

ajdt_1.6.1a_for_eclipse_3.4

org.tmatesoft.svn_1.2.1.eclipse

Plantillas, estilos, imágenes y formateador:

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

Patrones a utilizar:

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

Capa de presentación: Modelo Vista Controlador, el cual es implementado por Java Server Faces (JSF).

Capa de lógica de negocio: Session Façade

Capa de persistencia: Entity Access Object (EAO)

Rich Faces:

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

Código HTML:

Está prohibido el uso de etiquetas HTML en las páginas.

Anotaciones en la entidad:

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

Llaves compuestas:

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

HashCode e Equals en EJB de entidad:

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

JPA externos:

Para utilizar los JPA externos (academico.jar, recursos-humanos.jar, etc), éstos deben copiarse en la carpeta raíz. En el caso de Windows C:\, para Linux \).
En cada uno de los archivos persistence.xml de su aplicación, se debe utilizar <jar-file>file:/jpa.jar</jar-file>.
Por ejemplo:

```
<jar-file>file:/academico.jar</jar-file>
<jar-file>file:/recursos-humanos.jar</jar-file>
<jar-file>file:/general-UIS.jar</jar-file>
```

Servicios:

Para crear un servicio se debe tener en cuenta las siguientes reglas:
La interfaz debe contener la anotación Remote
La implementación de la interfaz debe contener la anotación RemoteBinding con su respectivo nombre jndi (igual al utilizado por la anotación Name).
Por ejemplo: `@RemoteBinding(jndiBinding = "ConsultarGenerales")`

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

```
@EJB(mappedName = "ConsultarGenerales")
```

4.2.2 *Documentación de los Diagramas de Diseño*

Casos de Uso:

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

Identificación de Actores:
Se identifican con el rol que desempeñan en el sistema.

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

Casos de Uso:
Se deben identificar con una acción. Los verbos que se pueden utilizar se encuentran al final del documento.
La información mínima requerida por cada caso de uso es la siguiente:
Descripción completa
Precondiciones y postcondiciones
Descripción del escenario básico y alternos
Diagrama de clases
Si el caso de uso es complejo se debe incluir:
Diagrama de secuencia y/o diagrama de actividades

4.2.3 *Sintaxis de Nombres en Java*

Reglas de sintaxis generales:

- En esta sección se especifican las reglas de sintaxis generales para todos los identificadores (nombres de variables, clases, métodos, etc.)
- Todos los nombres de los identificadores deben estar en español.
- Siempre se deben utilizar nombres que sean claros, concretos y libres de ambigüedades. Usando palabras completas evitando acrónimos y abreviaturas.

- Los nombres deben estar definidos sin espacios en blanco, sin guiones (_ , -), ni comillas (“,’), sin operadores (+, - , / , *), sin tildes, utilizar la n en vez de la ñ y sin caracteres especiales.
- No se debe utilizar la mayúscula para diferenciar entre identificadores distintos. Ejemplo: contador y Contador.
- No se deben diferenciar dos identificadores solo con numerales en cualquier posición. Ejemplo: contador1, contador2, 1contador, 2contador.
- Las siguientes partículas están prohibidas en la declaración de los nombres de identificadores: artículos (el, la, los, unos, unas, un), determinantes demostrativos (este, ese, aquel, aquellos), cardinales (uno, dos, etc.), pronombres de cualquier tipo (yo, tu, el, me, te, se, este, ese, mi, tu, su, etc.).
- Se deben utilizar máximo 5 palabras por nombre, las 3 primeras palabras van completas, a partir de la cuarta palabra se quitan las vocales a la palabra exceptuando la última vocal y la primera si la palabra empieza por vocal. No se utiliza ningún separador entre las palabras, se separa cada palabra utilizando su primera letra en mayúscula.

  Ejemplo:

  `hacerMantenimientoConsultaUsros,`

  `hacerMantenimientoAsignaturasCntxto.`

Clases:

- Los nombres de las clases deben iniciar siempre en mayúscula, deben ser simples y descriptivos.
- Los nombres de las clases de Entidad deben ser sustantivos en singular.
- Para las clases de Entidad siempre se debe definir la anotación @Table que indica la tabla de la base de datos relacionada para su persistencia. Además se debe utilizar el parámetro name de la anotación @Entity para identificar el EJB (@Entity(name = “xxx”)). El nombre de la clase puede

ser distinto al nombre de la tabla y debe seguir el estándar de identificadores mencionado en el numeral anterior.

- Los nombres de los EJB utilizados por el patrón Session Façade está conformado por un verbo autorizado. Además debe incluir al final las letras “EJB”. Ejemplo: RegistrarMatriculaEstudianteEJB.
- La interfaz asociada al EJB debe llevar el mismo nombre del EJB sin el sufijo “EJB”. Ejemplo: RegistrarMatriculaEstudiante.
- Para los EJB de entidad, cuando la clase representa una relación entre dos entidades, el nombre se forma uniendo los nombres de las entidades involucradas.
- Los nombres de las clases que representan excepciones terminaran siempre con el sufijo “EXCEPCION”.
- El nombre de la clase no contendrá detalles sobre la implementación interna de la misma. Por ejemplo ArrayEstudiantes no es nombre válido.

Métodos:

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

Paquetes:

El nombre de los paquetes debe iniciar con minúscula la primera palabra, las siguientes palabras inician en mayúscula, sin separadores.

La estructura de los paquetes es la siguiente:

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

Ejemplo:

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

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

co.edu.uis.sistema.entidades

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

co.edu.uis.sistema.servicios

Variables:

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

una instancia, para diferenciarlas se debe adicionar una palabra que identifique el rol que desempeña. Ejemplo: Estudiante estudiantePregrado, Estudiante estudiantePostgrado.

- La variable del EntityManager se debe llamar em.
- La variable de tipo FacesMessages se debe llamar facesMessages.

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

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

4.2.4 *Documentación*

La documentación relacionada con el diseño del sistema reside en la base de datos de *Enterpirse Architect*.
La documentación del código fuente se debe hacer en cada una de las clases, siguiendo el estándar de *JAVADOC* y documentando la definición de la clase, descripción de los métodos get y set y descripción de los parámetros de entrada y salida de cada uno de los métodos que componen la clase.

4.2.5 *Capa de Presentación*

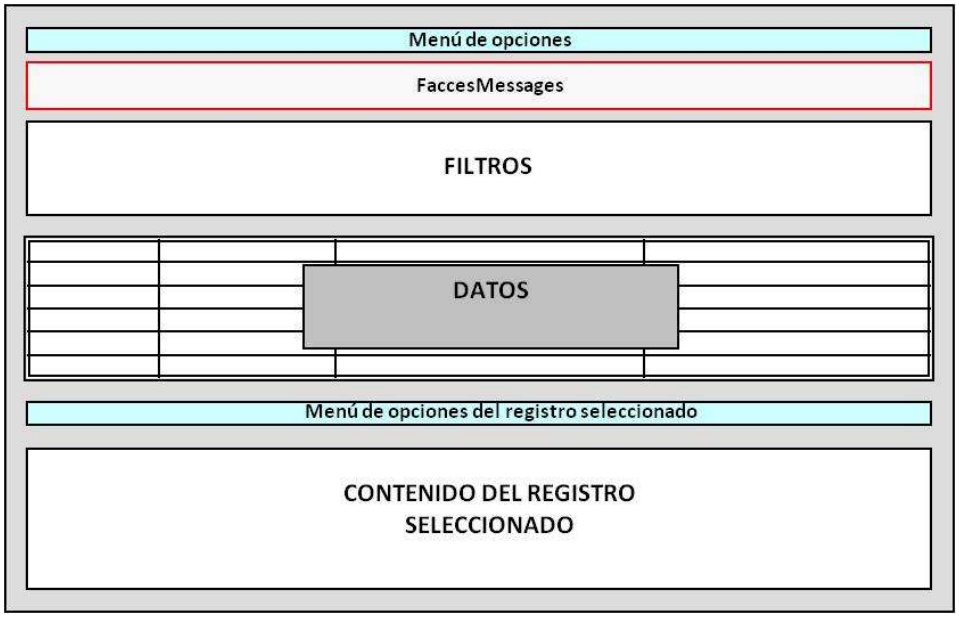
Plantilla principal: La plantilla principal de una página es la siguiente:

Figura 17 - Plantilla Principal División de Servicios de Información

Fuente: Estándares de la División de Servicios de Información de la UIS 2010

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

Figura 18 - Plantilla de Contenido División de Servicios de Información



Fuente: Estándares de la División de Servicios de Información de la UIS 2010

Para las páginas que muestran formularios:

Figura 19 - Plantilla de Contenido con Formulario DSI

Fuente: Estándares de la División de Servicios de Información de la UIS 2010

Paginación:

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

Por ejemplo, se quiere consultar los estudiantes de la sede Bucaramanga. De acuerdo al análisis realizado, no hay un consumo alto de memoria, por lo que se decide retornar 100 estudiantes máximos. Esto indica que cuando se implemente el método de consulta en JPQL, el parámetro setMaxResults debe tener el valor de 100.

Texto:

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

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

La sintaxis para utilizar las plantillas son:

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

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

Para la plantilla mostrar.xhtml:

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

Edición:

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

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

     <ui:define name="label">
          #{FormatoWeb.mostrarMensaje('primerNombre', true)}
     </ui:define>
     <h:inputText id="txt[nombreCajaDeTexto]" value="[valor]"
          required="true">
          <a:support event="onblur"
     reRender="dcr[identificador]"/>
     </h:inputText>
</s:decorate>
```

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

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

Tablas estáticas:

Se debe usar el control panelGrid de Java Server Faces.

Tablas dinámicas:

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

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

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

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

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

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

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

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

4.3 Roles Y Usuarios

La determinación de los roles o niveles de usuario es importante para la seguridad y auditoria del sistema. Consiste en restringir el acceso a la información del sistema, las tablas, los módulos de acuerdo a su rol de usuario.
Los niveles de usuario se determinan de acuerdo a lo que denominamos ROL, los roles de los diferentes usuarios que tienen acceso al sistema SIMIN son los siguientes:

Administrador:	Encargado de la administración total del sistema.
Invitado:	Es para el público en general, usuarios foráneos del sistema, tienen acceso a hacer consultas del menú y a opinar en los servicios prestados.

Jefe de Unidad: Son aquellos Jefes de las diferentes unidades de la universidad, que tienen algunos privilegios dentro del sistema como Reservar.

Almacenista: Encargado de mantener actualizado el almacén de Bienestar Universitario.

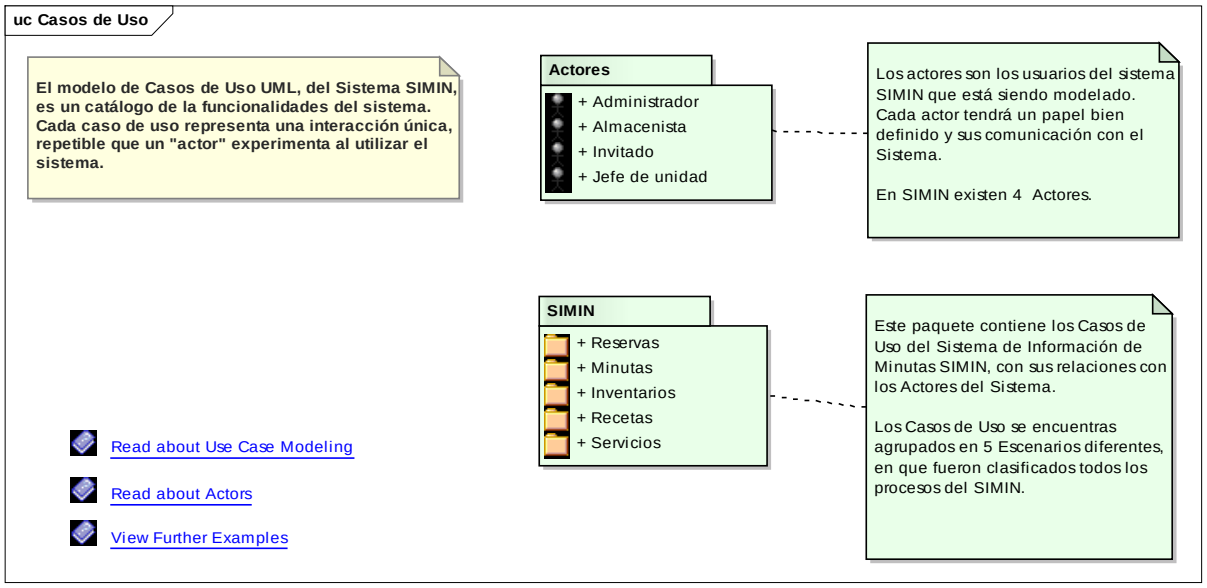
4.4 Diagramas Uml

En el proceso de análisis y diseño del sistema de información se construyeron los diagramas UML, durante el transcurso del proyecto, el diseño se fue adaptando a medida que se generaban cambios luego de presentar el prototipo inicial.

4.4.1 *Diagrama: Casos de Uso*

Los esquemas básicos de cada uno de los diagramas que se generaron, producto del diseño, se presentaran a continuación la estructura del Modelo de Casos de Uso y un ejemplo.

Figura 20 - Modelo de Casos de Uso SIMIN



4.4.1.1 Identificación de Actores

Figura 21 - Actores del Modelo de Casos de Uso SIMIN

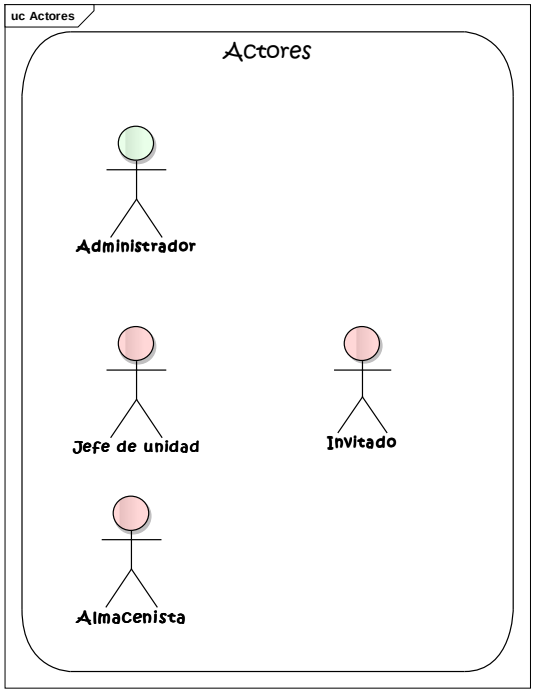


Tabla 4 - Actores del sistema SIMIN

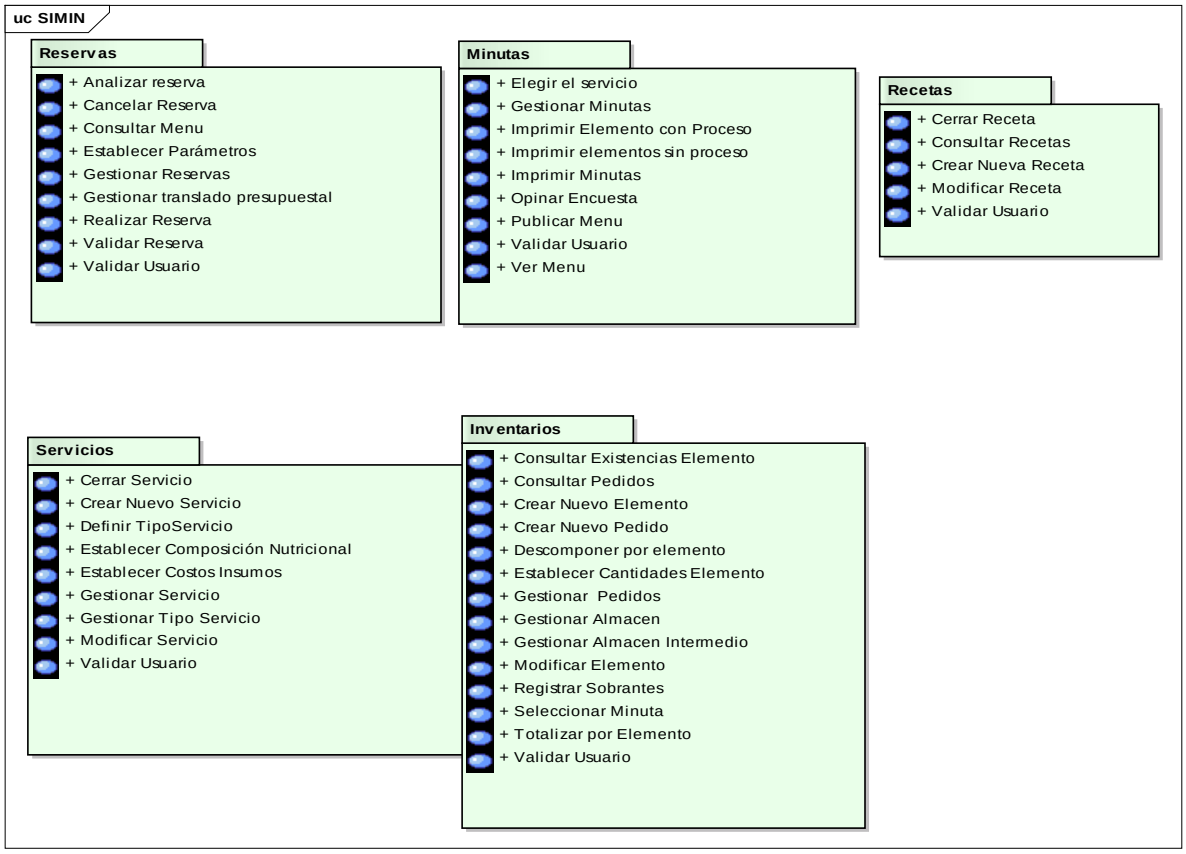
Actor	Descripción
Administrador	El Administrador, es el es el Encargado o el Jefe de Sección de Comedores y Cafería, quien administra todos los procesos que realiza el SIMIN
Invitado	El invitado, es el usuario del sistema que sin necesidad de estar registrado puede acceder algunas secciones que ofrece el sistema SIMIN.
Jefe de unidad	El Jefe de Unidad, es aquel usuario del sistema que se encuentra registrado como Jefe de Unidad, debe tener un login y un password para acceder al módulo de

	reservas del SIMIN.
Almacenista	El Almacenista, es aquel usuario del sistema que se encuentra registrado como Almacenista, debe tener un login y un password es el encargado de mantener los inventarios actualizados.

4.4.1.2 Casos de uso SIMIN

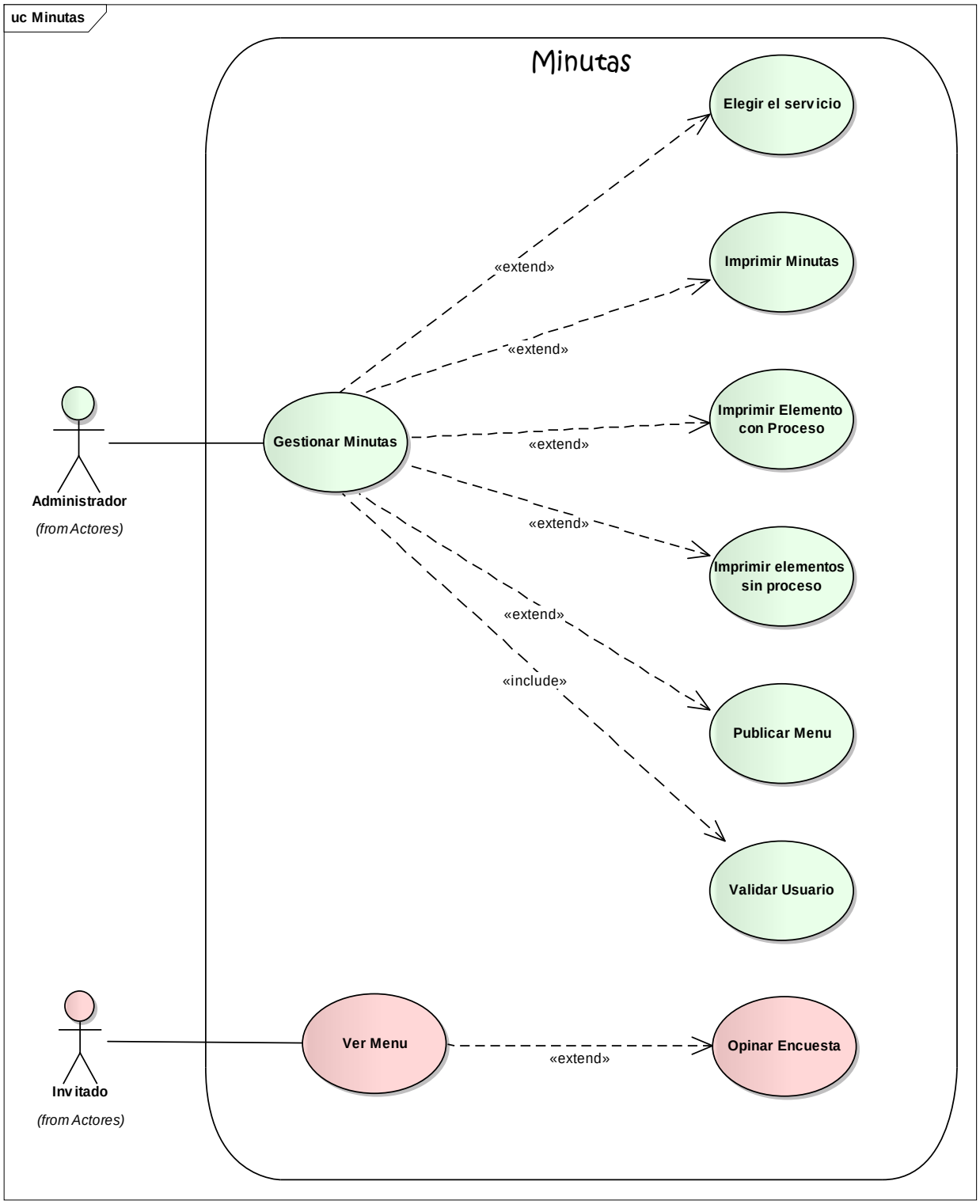
Para el Sistema de Información SIMIN, se integró toda la información recopilada, analizada, en cinco (5) diagramas de Casos de Uso.

Figura 22 - Estructura del Modelo de Casos de Uso SIMIN



CASO DE USO	GESTIONAR MINUTAS

Figura 23 - Diagrama Casos de Uso: Minutas



Objetivo:	Administrar las minutas.	
Actores:	Administrador	
Pre condiciones	Validarse como Administrador	
Post Condiciones	Actualizar datos y Generar Informes	
Pasos	Acción del Actor	Respuesta del Sistema
	Ingresar al Sistema	
		Muestra Pantalla inicial las minutas creadas
	Registrar nueva Minuta	
		El actor hace el registro de registro de una nueva minuta.
	Asignar Menú	
		El actor asigna al servicio y tipo de Servicio de una fecha especifica un menú.
	Validación	
		El actor prueba los menús para verificar que cumplan con los requerimientos específicos del servicio.
	Asignar Adicionales	
		El Actor asigna elementos adicionales a la minuta a programar
	Totalizar	
		El actor decide totalizar las minutas de un rango de fecha, para hacer el pedido de elementos al almacén.
Variaciones:	El usuario puede escoger las opciones para administrar, sea ver, ingresar, modificar o eliminar una minuta, y puede o no asignar Elementos Adicionales.	
Extensiones:	El sistema accede a la base de datos para visualizar los datos que corresponden a la Programación de Minutas.	

Tabla 5 - Detalle del Caso de Uso: Gestionar Minutas

Tabla 6 - Casos de Uso para Minutas

CASO DE USO	DESCRIPCIÓN
Elegir el servicio	Elegir el Servicio, Como hay varios servicios previamente establecidos y relacionados con diversos Tipos de Servicio, es necesario elegir el servicio para el cual va asignada la minuta a programar.
Gestionar Minutas	Gestionar Minutas, para hacer toda la programación y las operaciones sobre las minutas diarias para cada tipo de servicio asociado a los diferentes servicios prestados por la Sección de Comedores y Cafetería de Bienestar Universitario.
Imprimir Elemento con Proceso	Imprimir Elemento con Proceso, Luego de ser programadas las minutas, se imprimen los elementos que requieren PIQUE y que han sido solicitados a almacén.
Imprimir Minutas	Imprimir Minutas, Luego de ser programadas las minutas, se imprimen las minutas diarias o semanales.
Imprimir elementos sin proceso	Imprimir Elemento sin Proceso, una vez programadas las minutas, se imprimen los elementos que NO requieren PIQUE y que han sido solicitados a almacén.

Opinar Encuesta	Opinar Encuesta, acceder a una encuesta para dar su opinión sobre el servicio disfrutado. El SIMIN podrá deshabilitar las minutas que tenga bajo grado de satisfacción, de acuerdo a la opinión de los usuarios.
Publicar Menú	Publicar Menú, en el SIMIN serán publicados todos los menús previamente programados para los diversos Tipos de Servicio que ofrece la Sección de Comedores y Cafetería de la División de Bienestar Universitario, para que todos puedan acceder a dicha información.
Validar Usuario	Validar Usuario, es el proceso de verificación para los Usuarios del SIMIN.
Ver Menú	Ver Menú, en el SIMIN serán publicados todos los menús programados para los diversos Tipos de Servicio que ofrece la Sección de Comedores y Cafetería de la División de Bienestar Universitario, para acceder a toda ésta información.

4.4.2 *Diagramas de Secuencia*

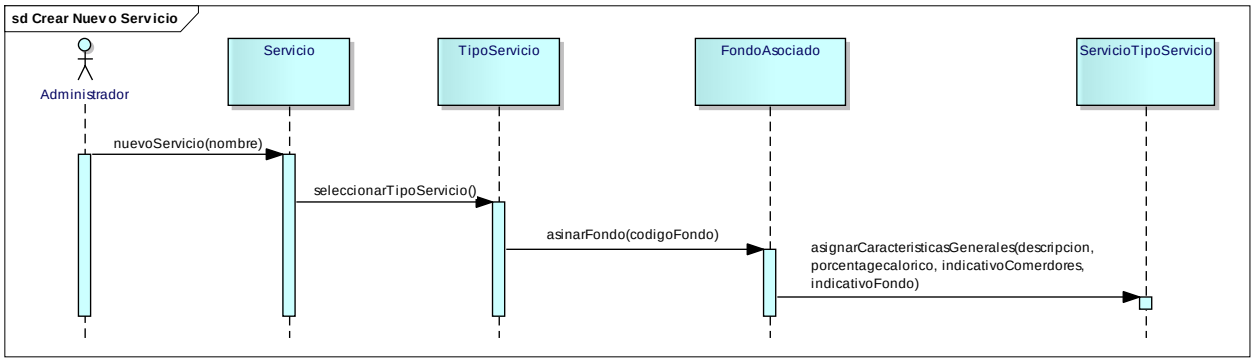
Los diagramas de secuencia amplían la visión general del funcionamiento e interacción del sistema con el usuario final, visión que previamente se conformó a partir de la recolección de requerimientos del sistema.

Se escogieron los casos de uso más complejos para representarlos en los diagramas de secuencia, los cuales se presentan a continuación:

Crear Nuevo Servicio:

Crear nuevo servicio ofrecido por la División de Bienestar Universitario, incluye la relación de los tipos de servicio al servicio.

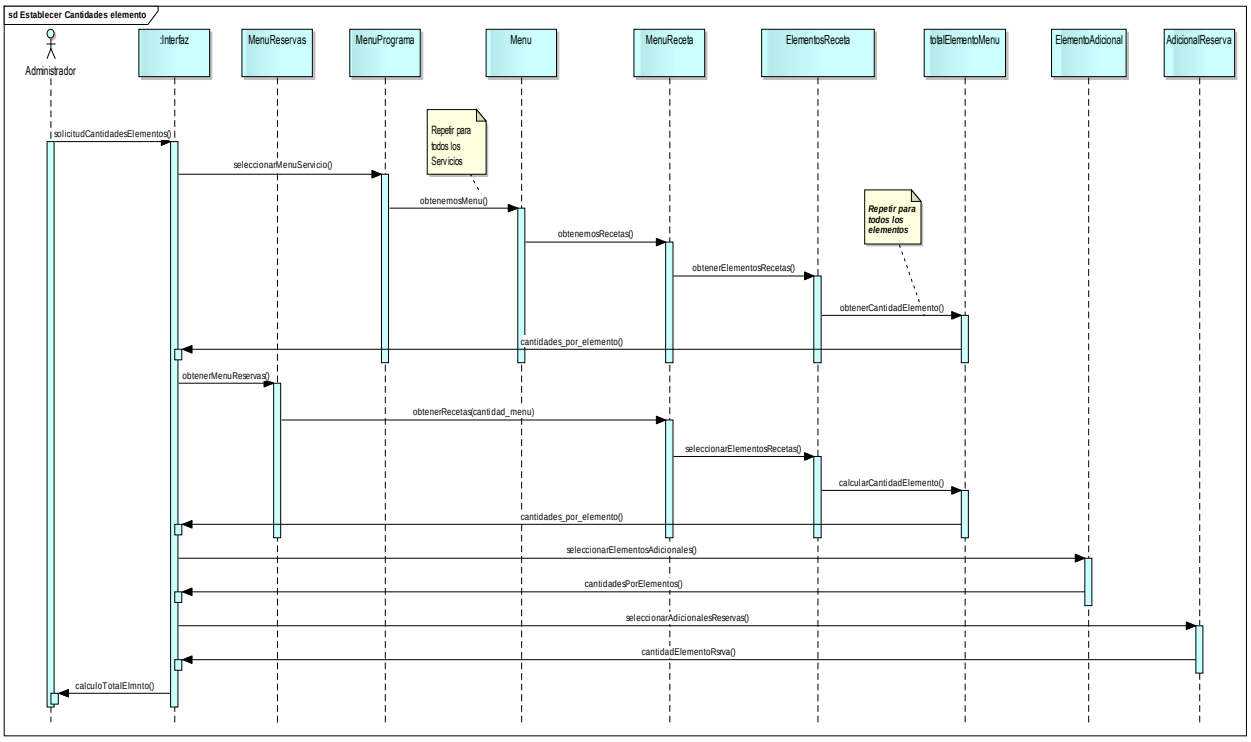
Figura 24 - Diagrama Secuencia: Crear Nuevo Servicio



Establecer Cantidades elemento:

Establece las cantidades totales por elemento que se pedirán a almacén luego de la programación de las minutas, incluyendo adicionales del menú y de las reservas.

Figura 25 - Diagrama Secuencia: Establecer cantidades elemento



4.4.3 *Diagrama de Clases*

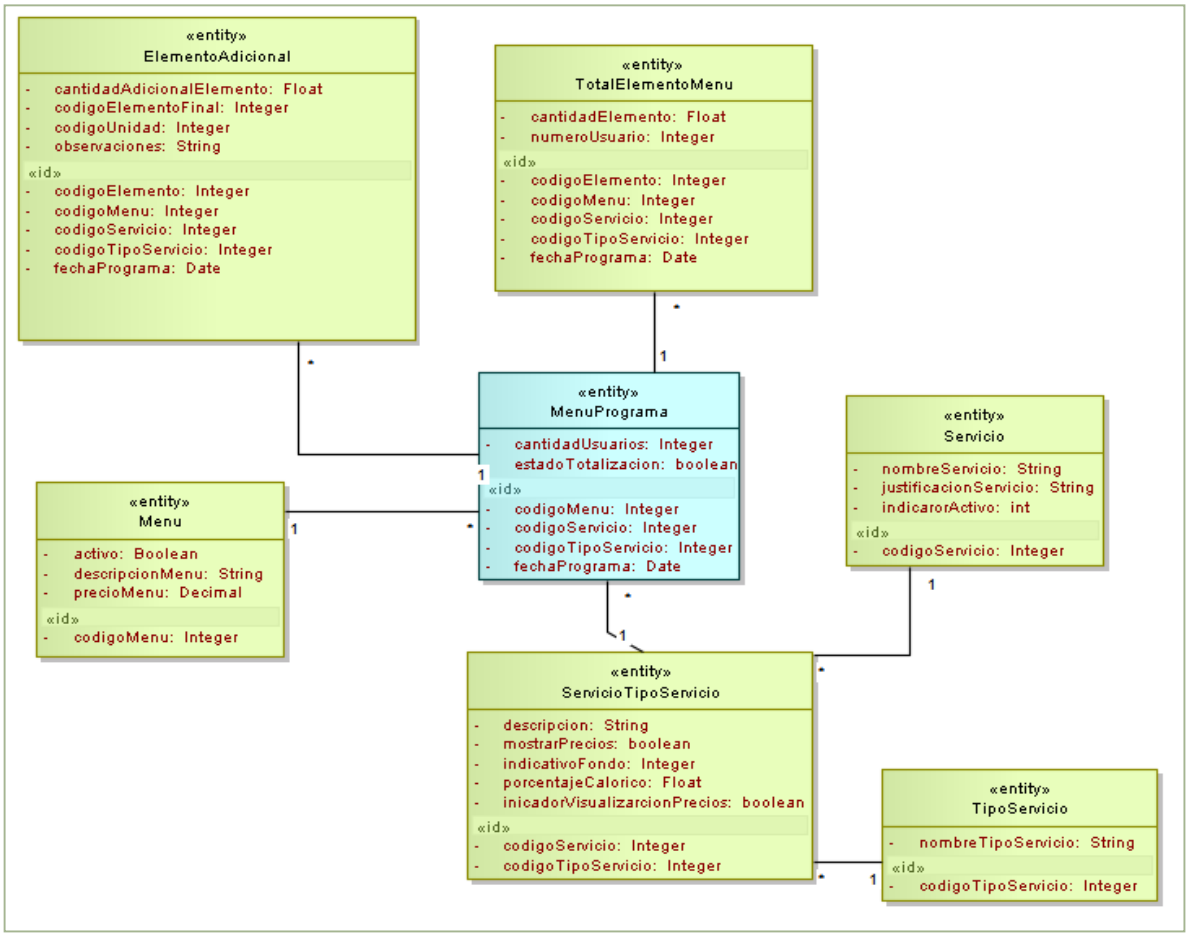
El modelo de Clases es la descripción más clara de la estructura estática del sistema SIMIN.

Durante el análisis, el modelo de clases fue conceptual e intentó representar en clases todos los conceptos de información que debía manejar el sistema de minutas. En la etapa de diseño este modelo de clases conceptual tomó mayor forma y las clases representadas se alimentaron con atributos incluidos y necesarios para SIMIN, así mismo, la inclusión de los métodos de las clases, y la decisión de navegación en las asociaciones entre clases.

El modelo de clases ha sido modificado durante todo el proyecto, con el fin de satisfacer al cliente y de hacer de SIMIN un software de calidad.

Para ilustrar la documentación dada del diagrama de clases se ha tomado como ejemplo el correspondiente modulo de Gestionar Minutas.

Figura 26 - Diagrama de Clases, Modulo Minutas



Clase MenuPrograma: Esta entidad contiene la información acerca de la programación de las minutas, de acuerdo al dia, servicio, tipo de servicio.

Tabla 7 - Atributos Clase MenuPrograma

Atributo	Tipo	Descripcion
fechaPrograma	Date	Es la fecha para la cual se va a programar la minuta.
codigoServicio	Integer	Es el código del servicio que se desea programar.
codigoTipoServicio	Integer	Es el código del Tipo de Servicio que se desea programar.
codigoMenu	Integer	Es el código del Menu a programar para la minuta.
cantidadUsuarios	Integer	Es la cantidad de Usuarios Beneficiarios para los cuales se va a preparar la minuta.
estadoTotalizacion	Boolean	Es el estado en que se encuentra la minuta, si ya fue totalizada para poder realizar el pedido a almacén Financiero, o no.

Clase ElementoAdicional: Esta entidad contiene la información acerca de los elementos que se le adicionan a una minuta programada.

Tabla 8 - Atributos Clase Elemento Adicional

Atributo	Tipo	Descripcion
codigoElementor	Integer	Es el código del elemento que se va a adicionar a la minuta.
codigoServicio	Integer	Es el código del servicio que se desea programar.

codigoTipoServicio	Integer	Es el código del Tipo de Servicio que se desea programar.
codigoMenu	Integer	Es el código del Menú a programar para la minuta.
fechaPrograma	Date	Es la fecha para la cual se va a programar la minuta.
cantidadAdicionalElemento	Float	Es la cantidad del elemento que se va a adicionar a la minuta.
codigoElementoFinal	Integer	Es el código del elemento.
codigoUnidad	Integer	Es el código de la unidad de medida del elemento.
observaciones	String	Son las observaciones que se le hacen a la asignación de elementos adicionales a la minuta.

Clase TotalElementoMenu: Esta entidad contiene la información acerca de los elementos utilizados en las minutas programadas junto con sus adicionales.

Tabla 9 - Atributos Clase TotalElementoMenu

Atributo	Tipo	Descripcion
codigoElementor	Integer	Es el código del elemento
codigoServicio	Integer	Es el código del servicio que se desea programar.
codigoTipoServicio	Integer	Es el código del Tipo de Servicio que se desea programar.
codigoMenu	Integer	Es el código del Menu a

		programar para la minuta.
fechaPrograma	Date	Es la fecha para la cual se va a programar la minuta.
cantidadElemento	Float	Es la cantidad del elemento usado para la minuta.
numeroUsuarios	Integer	Es la cantidad de Usuarios Beneficiarios para los cuales se va a preparar la minuta.

Clase Menu: Esta entidad contiene la información acerca de los Menús.

Tabla 10 - Atributos Clase Menu

Atributo	Tipo	Descripcion
codigoMenu	Integer	Es el código del menú.
descripcionMenu	String	Es el nombre que se le da al menú..
precioMenu	Decimal	Es el precio del Menú.
activo	Boolean	Es el estado de activación del Menú.

Clase ServicioTipoServicio: Esta entidad contiene la información acerca de los servicios y tipos de servicios.

Tabla 11 - Atributos Clase ServicioTipoServicio

Atributo	Tipo	Descripcion
codigoServicio	Integer	Es el código del servicio que se desea programar.
codigoTipoServicio	Integer	Es el código del Tipo de

		Servicio que se desea programar.
descripcion	String	Es la justificación del servicio y tipo de servicio.
mostrarPrecio	Boolean	Es el estado para saber si se deben o no mostrar precio.
indicativoFondo	Integer	Es el código del fondo presupuestal del servicio.
porcentajeCalorico	Float	Es el porcentaje calórico del servicio tipo servicio.
indicadorVisualizacionPrecios	Boolean	Es el estado para indicar si se pueden ver o no los precios.

4.4.4 *Diagrama de Datos*

El diagrama de datos es el resultado del Diagrama de Clases diseñado, y adecuado para la programación del sistema.

4.5 Prototipo Inicial

4.5.1 *Generalidades del prototipo inicial*

El prototipo es la forma más efectiva para los diseñadores y las partes interesadas para plasmar todas las ideas sobre el sistema de manera previa al desarrollo.
Se tomó atenta nota de cada uno de los requerimientos para el sistema, se realizó un correcto análisis de ellos y se comenzó a construir un prototipo no funcional

que proporcionara una facilidad de exponer al cliente final, la manera en que se podría brindar una solución a los requerimientos analizados previamente.

El proceso de construcción del prototipo comenzó una vez se estabilizó el diseño, y se trabajó de manera conjunta con el Director del proyecto Ingeniero Enrique Torres, con el Ingeniero Jacksson Sonny Gonzalez, profesional DSI, la Nutricionista Consuelo Serrano Vega y las autoras del proyecto, durante reuniones periódicas en donde se fue perfeccionando el diseño del sistema, haciendo las debidas modificaciones en el diseño y se aterrizó la idea del sistema.

Para la elaboración del prototipo no funcional se necesitó la herramienta Axure RP, el cual creó la sensación de estar navegando en el producto final debido a los efectos en la interfaz. Tomó tiempo lograr la satisfacción del cliente final, y la optimización del diseño del sistema y garantizar la eficiencia del desarrollo y la programación del sistema de información SIMIN.

Para el caso de SIMIN, el prototipo inicial resultó tan claro que el trabajo fue efectivo y evitó cambios drásticos que podrían haber solicitado los usuarios al final del proyecto.

4.5.2 *Primer Prototipo : Prototipo No Funcional*

Para el prototipo no funcional se crearon los siguientes módulos para el sistema SIMIN:

Gestionar Minutas En el que se podrían administrar los elementos, las recetas y los menús a manejar dentro del sistema, con las funciones de asignar, modificar, eliminar y asignar.

Administrar Servicios En el que se podrían administrar los servicios y asociarlos a los tipos de servicio, con las acciones de modificar e inactivar los servicios.

Administrar Minutas En este módulo se pueden administrar las minutas, haciendo las programaciones de las minutas y un historial de minutas.

Administrar Reservas En el que se pueden hace el cambio de estado de las solicitudes de las reservas recibidas

Almacenes Módulo encargado de administrar el almacén intermedio y de hacer la los pedidos y consulta de pedidos de las minutas programadas.

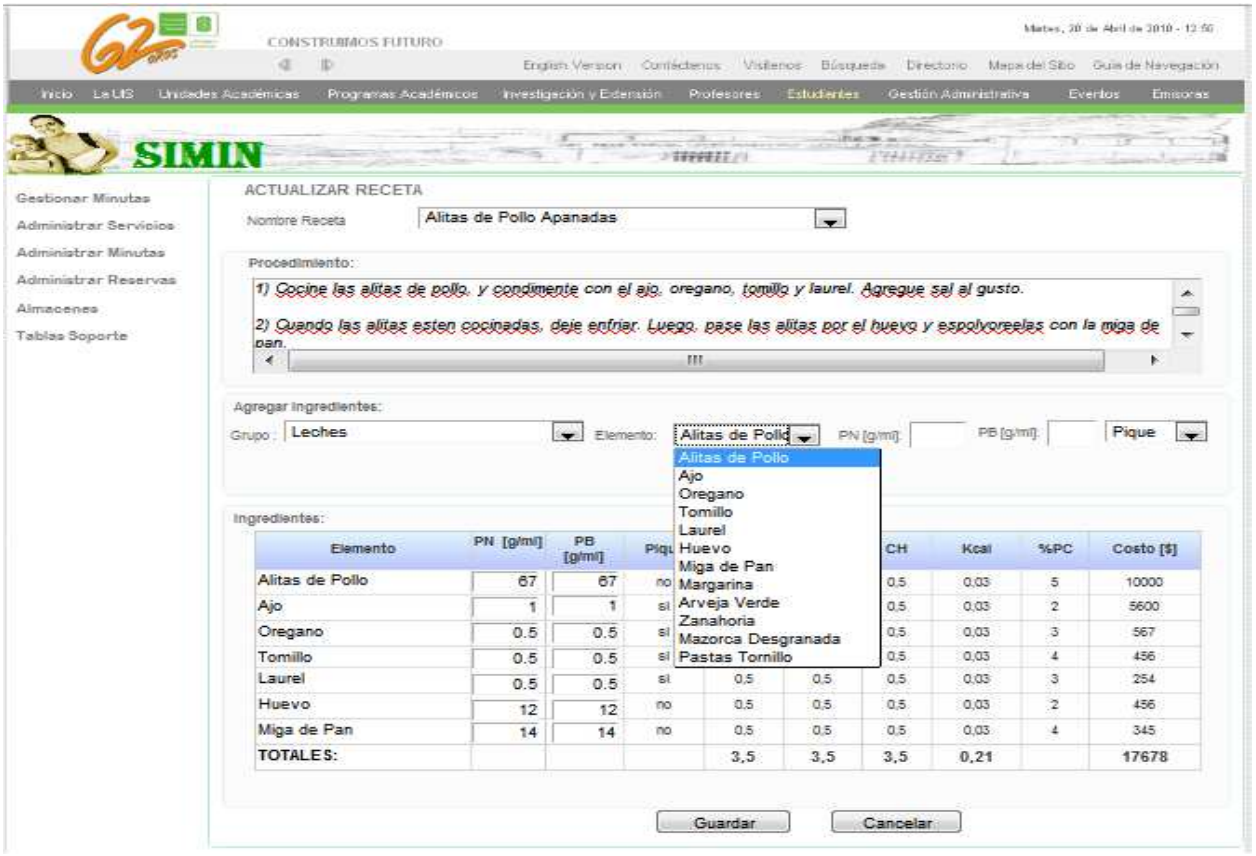
Tablas Soporte Éste módulo está encargado de hacer el llenado de las tablas necesarias para el funcionamiento del sistema, grupos alimenticios, equivalentes, conversiones, unidades, propiedades nutricionales, equivalentes y tipos de servicio.

Su opinión cuenta Esta es la encuesta de satisfacción para los usuarios de los Servicios que ofrece la Sección de Comedores y Cafetería, de Bienestar Universitario UIS.

Ver Minuta Semanal Aquí, todos los usuarios, pueden acceder a la información de las minutas semanales de acuerdo al dia, servicio y tipo de servicio.

Para describir de manera detalla se escogió un par de ejemplos dentro del módulo Gestionar Minutas, se pensó en la Actualización de Recetas, en esta parte se hace el ingreso de los elementos que se necesitan para la elaboración de una Receta y su procedimiento.

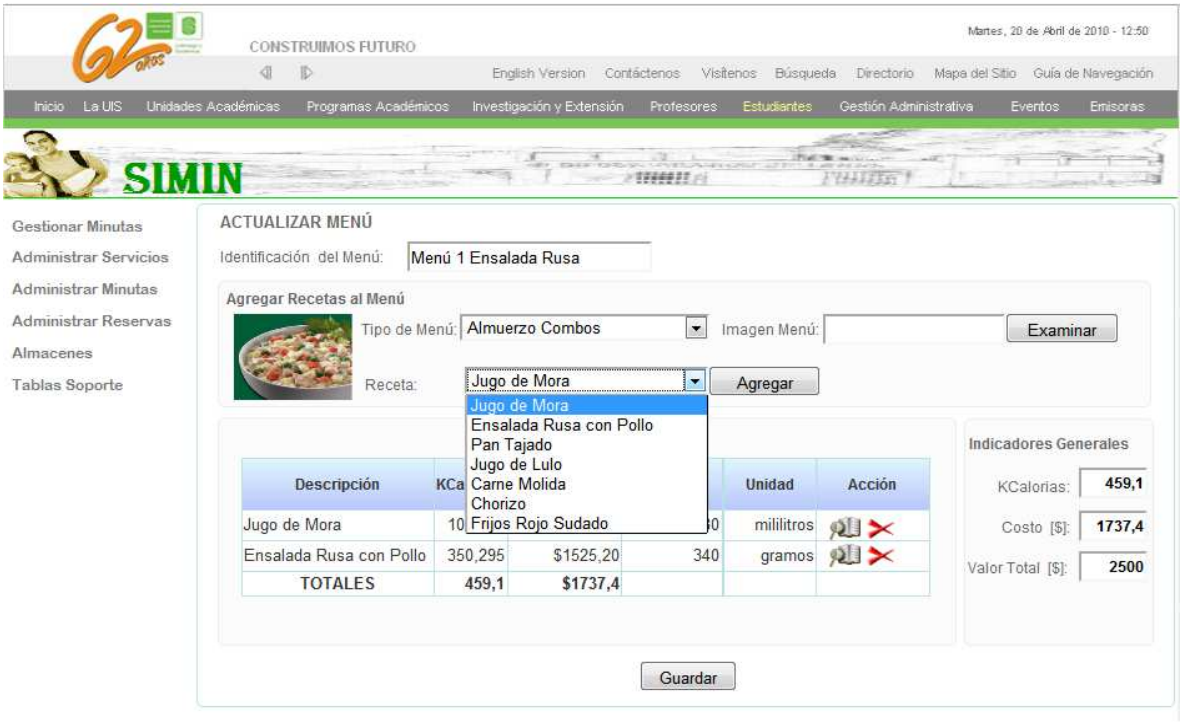
Figura 27 - Nueva Receta (Prototipo Inicial)



Como se puede observar en la figura 27, el sistema actualiza los valores de las calorías acumuladas para la receta de acuerdo a los elementos que se le van adicionando.

El segundo ejemplo contiene la creación de un nuevo menú dentro de SIMIN, el proceso comprende: asignar recetas al menú, darle un nombre, asociar una imagen, detallar la suma de calorías de acuerdo a las recetas asignadas al menú y establecer el precio de venta al público del menú.

Figura 28 - Nuevo Menu (Prototipo Inicial)



En la figura 28 se muestra la interfaz para modificar un Menú, con los datos que lleva el menú, la descripción, las recetas, la imagen y muestra los datos importantes de calorías, costo de insumos del menú y por último el valor total.

4.6 Prototipo Final

El prototipo final, es el prototipo resultado de las correcciones del prototipo inicial junto con una serie de cambios que aunque no fueron drásticos, son notorios en la interfaz e importantes en la programación de las minutas.

Las revisiones previas realizadas junto con los funcionarios de Bienestar Universitario, condujo a un resultado de prototipo funcional que cumple con todos

los objetivos estipulados en el plan de proyecto, incluyendo las políticas de seguridad instauradas por la DSI de la Universidad Industrial de Santander.

Para dar una idea más clara del trabajo realizado, y entregado como prototipo final, se escogieron dos ejemplos del prototipo final para detallarlos en el presente informe.

Administrar Minutas

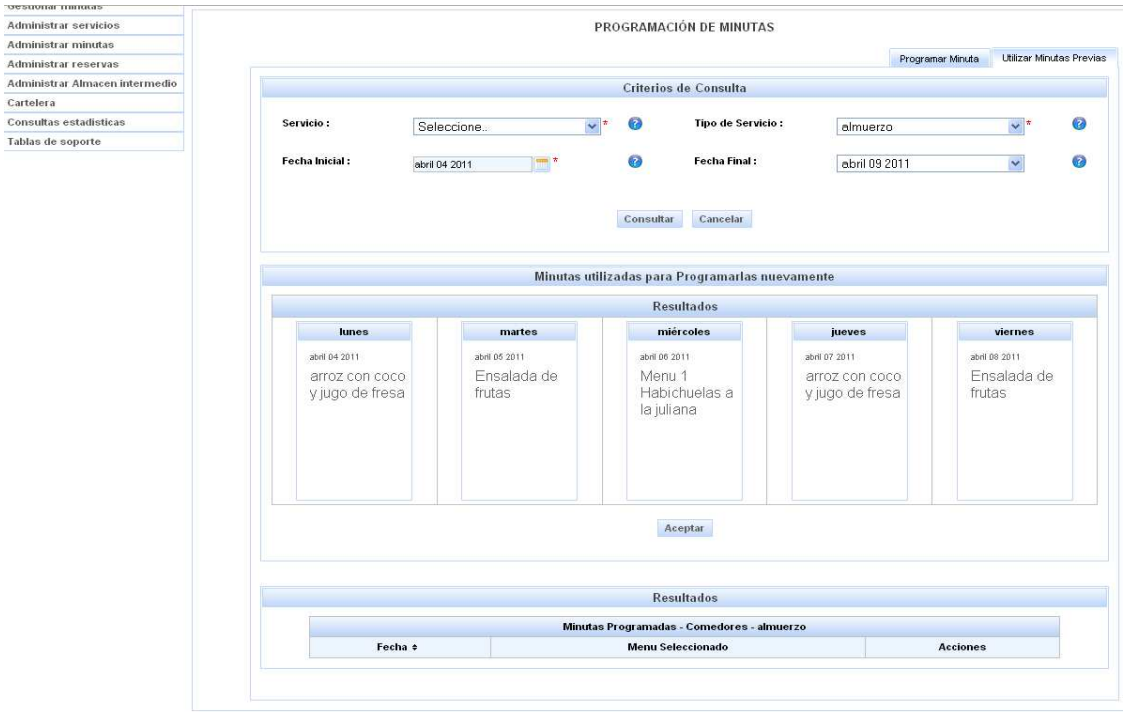
Teniendo previamente todo lo necesario para gestionar una minuta, registrados los elementos, construidas las recetas, conformados los menús, creados los servicios con sus tipos de servicio y sus respectivas características nutricionales, se procede a Programar la Minuta.

El primer ejemplo precisamente es el de Programar Minutas. Debido a los cambios que se generaron luego de la exposición del primer prototipo, se decidió ofrecer dentro del sistema dos maneras de realizar una programación de minutas.

Opciones para programar minutas:

- Programar la minuta de manera Diaria.
- Programa Minutas mediante la reutilización de minutas ofrecidas o programadas anteriormente.

Figura 29 - Programar Minuta (Prototipo Final)

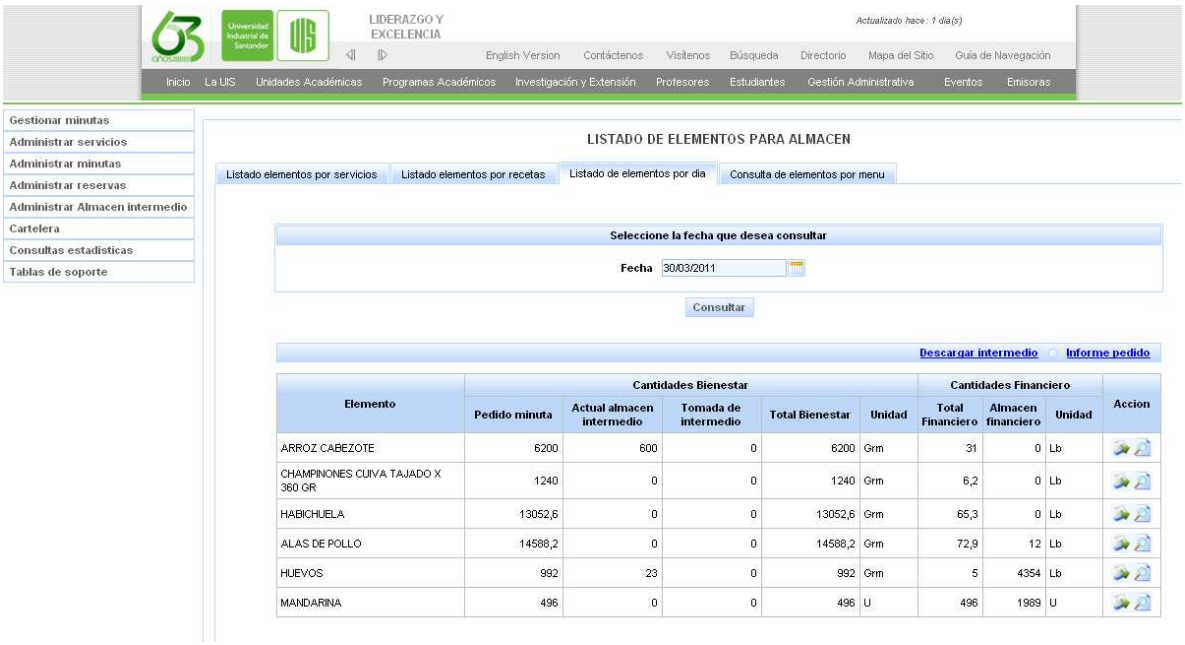


En la Figura 29, se muestra una minuta de una semana determinada, que se desea utilizar para otro periodo, en el mismo servicio.

El segundo ejemplo es la consulta generada de todos los elementos que se van a utilizar en todas las minutas programadas para un día específico.

Incluyendo en el informe los datos del consumo y de los saldos de almacén intermedio de Bienestar Universitario y almacén Financiero, permitiendo hacer las descargas al almacén intermedio según la necesidad y con la posibilidad de generar los informes en formato PDF.

Figura 30 - Consulta de Pedidos (Prototipo Final)



Elemento	Cantidades Bienestar					Cantidades Financiero			Accion
	Pedido minuta	Actual almacen intermedio	Tomada de intermedio	Total Bienestar	Unidad	Total Financiero	Almacen financiero	Unidad	
ARROZ CABEZOTE	6200	600	0	6200	Grm	31	0	Lb	
CHAMPINONES CUIVA TAJADO X 360 GR	1240	0	0	1240	Grm	6,2	0	Lb	
HABICHUELA	13052,6	0	0	13052,6	Grm	65,3	0	Lb	
ALAS DE POLLO	14588,2	0	0	14588,2	Grm	72,9	12	Lb	
HUEVOS	992	23	0	992	Grm	5	4354	Lb	
MANDARINA	496	0	0	496	U	496	1989	U	

En la figura 30 se puede observar la consulta de un día determinado y los elementos requeridos para dicho día.

4.6.1 *Esquema de Seguridad de la UIS, implementado en SIMIN*

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 de acuerdo a las actividades que realizan los usuarios. A cada uno de los roles definidos se le asocian los usuarios de acuerdo a las funciones que desempeñen.

4.6.1.1 Estructura de la Base de Datos soporte

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

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

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

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

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

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

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

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

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

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

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

Atributo–tabla: Establece los atributos sobre los cuales se debe controlar el acceso a una tabla, cuando a un rol se le concede permiso para hacer uso parcial de la información existente en una tabla. Contiene: Sistema de información, nombre de la tabla, nombre del atributo sobre el cual se controla el acceso a la tabla, descripción del atributo, fecha y hora de creación del atributo, fecha y hora de inicio de vigencia del atributo, fecha y hora de fin de vigencia del atributo.
Valor–atributo-proceso: Contiene los valores que deben tener los atributos definidos en cada tabla en la tabla atributo – tabla que permiten el acceso a la información asociada a estos valores. Específica: Sistema de información, nombre de la tabla, nombre del atributo, valor del atributo, descripción, fecha y hora de creación del valor del atributo, fecha y hora de inicio de vigencia del valor del atributo, fecha y hora de fin de vigencia del valor del atributo.

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

4.6.1.2 Entorno de Navegación

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

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

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

4.6.1.3 Entorno de Control de Datos

Para los roles definidos en cada uno de los sistemas de información se especifican las tablas a las cuales puede accesar, el tipo de transacción que puede realizar sobre estas tablas (inclusión, modificación o eliminación de registros), si tiene acceso total o parcial a la información que contiene la tabla.

Para el acceso a la información de la tabla de manera parcial, se debe establecer el atributo o atributos seleccionados, los valores que estos atributos deben tener para autorizar el acceso solicitado.

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

4.6.1.4 Auditoria

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

El historial de las transacciones de cada tabla contiene información de los registros incluidos en la tabla, de los registros modificados y de los registros eliminados. Adicionalmente, en cada transacción se especifica: Fecha de la

transacción, hora de la transacción, tipo de transacción (I/U/D), tipo y número de documento de identidad del usuario que realizó la transacción, login, rol asociado, dirección IP y MAC del equipo desde el cual llevó a cabo la transacción.

4.6.2 *Organización de Directorios*

Consideraciones iniciales

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

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

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

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

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

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

Carpetas

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

- bootstrap
- classes
- dist

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

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

Carpeta src

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

Carpeta main

Contiene las clases de apoyo al proceso. No incluye a las entidades, pues estas estarán disponibles en otro lugar que se describe más adelante en este documento.

Carpeta hot

Contiene los casos de uso reflejados en los EJBs y sus interfaces.

Carpeta resources

Contiene los archivos de configuración de la aplicación.

Carpeta view

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

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

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

Nombres de archivos
Se sigue el mismo estándar dado para el nombre de variables.

Organización del código fuente dentro de la estructura de directorios.

Paquetes
La composición del nombre de los paquetes se sigue teniendo en cuenta la siguiente sintaxis:
co.edu.uis.[sistema].[aplicación].[módulo].[caso de uso]
Ejemplo: co.edu.uis.minutas.administracion.servicio

En el caso de los proyectos principales (los que empaquetan entidades y componentes para cada sistema) los paquetes serán los siguientes:
Administración (src/action): co.edu.uis.[Sistema].[Aplicación].administracion
Clases generales (src/action): co.edu.uis.[Sistema].[Aplicación].general

Paginas (carpeta view)

Dentro de esta carpeta se tendrán la siguiente estructura:

- Plantillas : Plantillas del sitio

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

4.6.3 *Documentación de programas fuente[17].*

Nombre de archivos, variables, constantes, atributos, métodos y parámetros. Los nombres dentro del código fuente siguen los parámetros establecidos en el estándar general de nombres, con las siguientes particularidades.

- Los nombre de los parámetros de los métodos empiezan con guión de piso, el resto del nombre sigue el estándar para nombres de variables. Una vez dentro del código fuente, se deben asignar a una nueva variable con el mismo nombre pero sin el guión de piso.

- Los nombres de constantes o variables finally, se escriben en mayúscula sostenida separando las palabras por guiones de piso.

Comentarios Java:

Los programas en Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación están delimitados por /*.. */ cuando se trata de varias líneas y //

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

cuando se trata de una línea. Los comentarios de documentación (conocidos como "comentarios JavaDoc") son específicos de Java y están delimitados por : /** ... */. Los comentarios de documentación se pueden extraer a ficheros HTML usando la herramienta javadoc.

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

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

Orden dentro de los archivos de código fuente:

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

- Sentencias package.
- Sentencias import.
- Comentario de documentación de la clase/interfaz.
- Declaraciones de clase e interfaz.
- Comentario de la implementación de la clase/interfaz si fuera necesario.
- Declaración de constantes (solo se declaran dentro de interfaces).
- Declaración de atributos. (la declaración se debe hacer uno por línea y al frente debe ir un comentario de implementación de ser necesario.)

- Declaración de variables. (Según los estándares no se deben declarar variables públicas, para ello se definen los métodos gets() y sets()). Las variables aquí declaradas serán privadas y se utilizarán como indicadores de estado de la clase. Las variables se deben declarar en el siguiente orden:
  - Variables estáticas: organizadas por ámbito o accesibilidad de la siguiente manera: públicas, protegidas, de paquete (sin modificador) privadas.
  - Variables de instancia: organizadas de la misma manera que las estáticas.
  - Al igual que los atributos se debe declarar una por línea para facilitar los comentarios de implementación frente a ellas en caso de necesitarse.
  - Comentario de documentación sobre cada uno de los constructores (no aplica a entidades).
  - Declaración de constructores. (Siempre se declarará el constructor sin parámetros, se requiera o no una acción dentro de este. No aplica a entidades).
  - Comentario de documentación sobre cada uno de los métodos que lo requieran.
  - Declaración de métodos del la lógica del negocio: estos deben ir agrupados por funcionalidad en lugar de por ámbito, siendo el objetivo de esta organización el hacer el código de mas fácil lectura y compresión.
  - Declaración de métodos gets() , sets() e is().
  - Sobre escritura del método equals().
  - Sobre escritura del método hashCode().
  - Sobre escritura del método compare().
  - Sobre escritura del método compareTo().

Especificaciones sobre el formato del código fuente:

Tabulación:
La unidad de tabulación será de 2 espacios.

Longitud de línea:

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

Ruptura de líneas:

- Cuando una expresión no cabe en una única línea, se debe romper de acuerdo a estos principios generales:
  - Romper después de una coma.
  - Romper antes de un operador.
  - Preferir las rupturas de alto nivel a las de bajo nivel. (por ejemplo no romper por operador dentro de paréntesis.
  - Alinear la nueva línea con el principio de la expresión al mismo nivel que la línea anterior y dar cuatro tabulaciones.

Declaraciones y sentencias

Sentencias package e imports:

Deben seguir las siguientes reglas de formato.

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

Variables locales:

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

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

Clases, interfaces y métodos:

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

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

Sentencias simples:

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

Sentencias compuestas:

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

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

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

Líneas en blanco

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

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

Espacios en blanco:

Los espacios en blanco deberían usarse en las siguientes circunstancias:

- Una palabra reservada seguida por un paréntesis Por ejemplo:

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

- En las listas de argumentos, debe haber un espacio después de cada coma.
- Todos los operadores binarios, excepto el operador punto (.) deben estar separados de sus operandos por espacios. Los operadores unarios (incremento ++, decremento --, negativo -) nunca deben estar separados de sus operandos. Por ejemplo:

- Las expresiones de una sentencia for deben estar separadas por espacios en blanco. Por ejemplo:
  for (expr1; expr2; expr3);
- Las conversiones de tipo (cast) deberían estar seguidas de un espacio en blanco. Por ejemplo:
  variableEntera = (int) variableCadena;

4.6.3.1 Documentación básica del programa Fuente de un EJB de Sesión

En la labor desarrollo la documentación juega un papel muy importante para el mantenimiento de software. Esto es aun mas cierto en proyectos grandes, en los que pueden estar involucrados demasiados desarrolladores, todos con culturas de desarrollo diferentes. Aunque todos estos desarrollos cuentan con estándares muy definidos, siempre puede ser necesaria la aclaración sobre ciertos fragmentos o secciones de código.
Históricamente todos los lenguajes de programación han contado con facilidades para la inserción de comentarios dentro del código fuente. Estos comentarios no son compilados puesto que no tienen utilidad funcional pero cuando otro desarrollador ve el código fuente, son necesarios para entender partes que no son claras a simple vista. Java heredó de C la forma de hacer tales comentarios:

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

Sin embargo estas herramientas no son necesarias para documentar proyectos largos, como es el caso de las aplicaciones empresariales. Para esto Java ofrece

Javadoc, que es un generador de documentación a partir de código Java. El código que Javadoc utiliza es aquel que encuentre entre:

/** documentación */

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

Ejemplo de documentación de método

```
    /**
     * Método para habilitar los días que se pueden asignar como fecha final para la búsqueda de minutas semanales y para reprogramar
     * minutas previamente programadas.
     * @param aDia
     * @return diasSemana
     */
    private List<Date> getListaDiasPosibles(Date aDia){
        Date fechaDate = new Date();
        this.diasSemana = new ArrayList<Date>();
        Integer contador;
        Calendar fechaCalendar = new GregorianCalendar();
        fechaCalendar.setTime(aDia);
        this.diasSemana.add(aDia);
        for (contador = 1; contador<=(this.numeroDias-1); contador++)
        {
```

```
            fechaCalendar.add(fechaCalendar.DATE, 1);
            fechaDate = fechaCalendar.getTime();
            this.diasSemana.add(fechaDate);
        }
        return this.diasSemana;
    }
```

Antes de la firma del método va una descripción básica de la tarea que cumple el mismo. Pero en esta documentación se pueden especificar otras cosas. Por ejemplo, mediante @param aDia la documentación está haciendo referencia al parámetro aDia que llega al método y con @return diasSemana hace referencia al valor, en este caso la lista: diasSemana, que retorna el método.

5. CONCLUSIONES

1. El Sistema de Información de Minutas es una solución que llena las expectativas de la Sección de Cafetería y Comedores de la Universidad Industrial de Santander, para el manejo nutricional de cada uno de los servicios que ofrece, acorde con las necesidades de la comunidad universitaria.

2. El sistema de información desarrollado es una solución dinámica, flexible, que le permite a la División de Bienestar Universitarias crecer en cobertura, en cantidad y calidad de servicios, con los controles necesarios que garanticen el uso óptimo de los recursos que dispone.

3. SIMIN, se encuentra integrado con el Sistema de Información de Comedores, el Sistema de Recursos Humanos, el Sistema de Cafetería y el Sistema Financiero, garantizando de esta manera contar con información actualizada, que permita la correcta toma de decisiones.

4. El usar la metodología de desarrollo basada en prototipos es la adecuada para la construcción de sistemas, que permite un alto nivel de interacción con el usuario, garantizando de esta manera obtener los resultados esperados por el cliente.

5. La estructura de “rol-usuario” definido por la División de servicios de Información en su esquema de seguridad, permite asociar permisos a cada usuario de acuerdo a las funciones que realice en la Sección de Cafetería y Comedores.

6. El sistema le brinda a DBU una herramienta para evaluar la aceptación de los servicios que ofrece, control de los inventarios, seguimiento de los costos, con el fin de hacer los ajustes necesarios que garanticen una buena calidad del servicio.

7. El software se construyó utilizando tecnologías de punta, como JavaEE5, lo cual hizo necesario seguir un proceso de capacitación, el cual fue ofrecido por la DSI y requirió de una investigación permanente en los diferentes tópicos asociados con las herramientas utilizadas.

6. RECOMENDACIONES

- Llevar a cabo mejoras en el Sistema Financiero para facilitar una mejor integración con el Sistema de Minutas, en lo relacionado con el manejo de inventarios, realización de pedidos.

- Dotar a la DBU de la estructura tecnológica necesaria para aprovechar el potencial que ofrece esta solución, en cuanto a cobertura, seguimiento y control de los servicios ofrecidos, evaluaciones, entre otros.

- Socializar SIMIN, a todos los miembros de la comunidad universitaria para que puedan hacer uso de las facilidades que ofrece ésta solución para solicitar y obtener servicios que ofrece DBU.

- La cultura informática juega un papel importante en la utilidad de los productos de software y en muchos casos, las consecuencias de estas son difíciles de prever y van más allá de los límites del desarrollador. Es importante que la Sección de Cafetería y Comedores conozca el sistema con todo detalle para su uso correcto.

7. BIBLIOGRAFÍA

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

Estándares de desarrollo. División de Servicios de Información. Bucaramanga, UIS.

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

GROFF, James R. – WEINBERG, Paul N. APLIQUE SQL. OsBORNE-McGrawHill, 1991.

LÓPEZ GUSTAVO, Blanco Ignacio. Tesis en Ingeniería Informática

PIATTINI Velthuis, Mario G, Aplicaciones Informáticas de Gestión. Una perspectiva de Ingeniería del Software. Alfaomega Ra-Ma. México. 2004.

PRESSMAN, Roger. Ingeniería del Software, un enfoque práctico. Mc GraW Hill, 2005.

VASQUEZ de Plata, Gloria, Elieth Gomez Almeyda, Sistema de Alimentos Equivalentes, Ediciones UIS.

Enlaces sitios WEB:

- http://download.oracle.com/docs/cd/E17802_01/j2ee/javaee/javaserverfaces/2.0/docs/pdldocs/facelets/index.html

- http://livedemo.exadel.com/richfaces-demo/richfaces/comboBox.jsf?tab=usage&cid=170149

- http://www.sparxsystems.com.ar/