SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LA SOLICITUD Y ASIGNACIÓN DE CITAS A LOS SERVICIOS INTEGRALES DE SALUD OFRECIDOS POR LA DIVISIÓN DE BIENESTAR UNIVERSITARIO

HÉCTOR FERNELY RÍOS MORALES

FERNANDO ANDRÉS ZAMBRANO VILLAR

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2012

SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LA SOLICITUD Y ASIGNACIÓN DE CITAS A LOS SERVICIOS INTEGRALES DE SALUD OFRECIDOS POR LA DIVISIÓN DE BIENESTAR UNIVERSITARIO.

AUTORES

HÉCTOR FERNELY RÍOS MORALES

FERNANDO ANDRÉS ZAMBRANO VILLAR

Trabajo de grado para optar el título de Ingeniero de Sistemas

DIRECTOR

ING. JACKSSON SONNY GONZÁLEZ

CODIRECTOR

ING. ENRIQUE TORRES LÓPEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2012

AGRADECIMIENTOS

A Dios por estar siempre presente y permitirme culminar con éxito esta etapa de mi vida.

A mi familia por su apoyo incondicional, su paciencia y su entrega.

A todos mis compañeros y amigos que estuvieron conmigo a lo largo de este proceso, especialmente, Fernando Zambrano quien contó conmigo para la realización de este proyecto; y Viviana Jaimes, Carlos Prieto, Francy Carrillo y Fidel Jiménez por su valiosa colaboración.

A la División de Servicios de Información, y Bienestar Universitario quienes confiaron en nosotros la responsabilidad de sacar adelante este proyecto.

Héctor Fernely Ríos Morales

AGRADECIMIENTOS

A mis padres por ser mi apoyo incondicional, a Gabriela García por ser la voz que me motiva a conseguir mis metas, a Viviana Jaimes, Carlos Prieto, Francy Carrillo, Fidel Jiménez y Danny Vergel por su ayuda para que esto pudiera llegar a su término, a los ingenieros Enrique Torres y Jacksson González por darnos la oportunidad de realizar este proyecto y a mi compañero Héctor Ríos por su apoyo.

FERNANDO ANDRÉS ZAMBRANO VILLAR

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO

- **AJAX:** (JavaScript Asíncrono y XML) Técnica de desarrollo web que facilita la creación de aplicaciones interactivas; éstas se ejecutan desde el navegador del usuario, manteniendo una comunicación asíncrona con el servidor que permite realizar cambios más fácilmente, mejorando la interacción del usuario con la página.

- **API:** (Application Programming Interface; Interfaz de Programación de Aplicaciones). Grupo de rutinas que provee un sistema operativo, una aplicación o una biblioteca, que definen cómo invocar desde un programa un servicio que éstos prestan. En otras palabras, una API representa un interfaz de comunicación entre componentes software.[1]

- **BASE DE DATOS:** Se define como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

- **COMPILACIÓN:** Proceso de traducción de un código fuente a lenguaje máquina para que pueda ser ejecutado por el computador.

- **CONTRASEÑA:** (Clave o Password) Conjunto de caracteres que pertenecen a un solo usuario y permiten su acceso a un determinado recurso.

- **DBA:** (Data Base Administrator). Administrador de la Base de Datos, es quien se encarga del control general del Sistema de Base de Datos.

- **EJB:** Son componentes del contexto de servidor que cubren la necesidad de intermediar entre la capa web y diversos sistemas empresariales. Son

[1] http://www.alegsa.com.ar/Dic/api.php

una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE 5.0).[2]

- **FRAMEWORK:** Se refiere a "ambiente de trabajo, y ejecución". En general los framework son soluciones completas que contemplan herramientas de apoyo a la construcción y motores de ejecución. Por ejemplo; dentro de Java en el ámbito especifico de aplicaciones Web tenemos los framework: Struts, "Java Server Faces", o Spring.[3]

- **HTML:** (HyperText Markup Language). Lenguaje de Marcado de Hipertexto, es el lenguaje de programación predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, además de complementar el texto con objetos tales como imágenes.[4]

- **INFORMIX:** Es un manejador de base de datos (DBMS) adquirido por IBM a la compañía INFORMIX SOFTWARE en el año 2001, sirve de enlace entre el usuario, las aplicaciones, y la base de datos del sistema.

- **INTERFAZ:** En software, parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos. Esta parte de un programa está constituida por un conjunto de comandos y métodos que permiten estas intercomunicaciones.[5]

- **INTERNET:** Es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando

[2] http://www.proactiva-calidad.com/java/ejb/introduccion.html

[3] http://daitesrc.wikispaces.com/Frameworks+para+Java

[4] http://es.wikipedia.org/wiki/HTML

[5] http://www.alegsa.com.ar/Dic/interfaz.php

que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.[6]

- **INTRANET:** Es una red de ordenadores privada basada en los estándares de Internet. Las Intranets utilizan tecnologías de Internet para enlazar los recursos informativos de una organización, desde documentos de texto a documentos multimedia, desde bases de datos legales a sistemas de gestión de documentos.[7]

- **JAVA:** Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.[8]

- **JAVA EE:** (Java Platform Enterprise Edition). Anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.[9]

- **JBOSS DEVELOPER STUDIO (JBDS):** Es un entorno de desarrollo comercial creado y actualmente desarrollado por JBoss (una división de Red Hat) y Exadel, Inc.[10]

- **JBOSS HIBERNATE:** Hibernate es un potente servicio de mapeo objeto/relacional de alto desempeño para consulta y persistencia. Le

[6] http://es.wikipedia.org/wiki/Internet

[7] http://www.masadelante.com/faqs/intranet

[8] http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

[9] http://es.wikipedia.org/wiki/Java_EE

[10] http://es.zettapedia.com/jboss-developer-studio.htm

permite desarrollar clases persistentes siguiendo el lenguaje orientado a objetos, incluyendo la asociación, herencia, polimorfismo, composición y colecciones.[11]

- **JBOSS SEAM**: Es un framework que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. Ha sido diseñado intentado simplificar al máximo el desarrollo de aplicaciones, basando el diseño en Plain Old Java Objects (POJOs) con anotaciones. Estos componentes se usan desde la capa de persistencia hasta la de presentación, poniendo todas las capas en comunicación directa.[12]

- **JAVA DEVELOPMENT KIT (JDK):** Es un grupo de herramientas para el desarrollo de software provisto por Sun Microsystems Inc. Incluye las herramientas necesarias para escribir, testear, y depurar aplicaciones y applets de Java.[13]

- **JAVA PERSISTENCE API (JPA):** Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).[14]

- **JAVA PERSISTENCE QUERY LANGUAGE (JPQL):** Es un lenguaje de consulta orientado a objetos, definida como parte de la especificación Java Persistence API. JPQL se utiliza para hacer consultas a las entidades almacenados en una base de datos relacional. Está fuertemente inspirado en SQL y sus preguntas se asemejan a las consultas SQL en la sintaxis, pero operan contra los objetos entidad JPA y no directamente con las tablas de base de datos.[15]

---

[11] http://www.latam.redhat.com/products/jboss/frameworks/hibernate/

[12] http://emanuelpeg.blogspot.com/2011_02_01_archive.html

[13] http://www.pergaminovirtual.com.ar/definicion/JDK.html

[14] http://es.wikipedia.org/wiki/JPA

[15] http://www.aprendiendojava.com.ar/index.php?topic=59.0

- **MÁQUINA VIRTUAL JAVA (JVM):** Es una maquina virtual de programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.[16]

- **MAPEO OBJETO-RELACIONAL:** Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).[17]

- **MÉTODO:** Es la operación que define el comportamiento de un objeto.

- **PERSISTENCIA:** Es la acción de preservar la información de un objeto de forma permanente para poder acceder a ella cada vez que sea necesario.

[16] http://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java
[17] http://es.wikipedia.org/wiki/Mapeo_objeto-relacional

RESUMEN

TÍTULO: SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LA SOLICITUD Y ASIGNACIÓN DE CITAS A LOS SERVICIOS INTEGRALES DE SALUD OFRECIDOS POR LA DIVISIÓN DE BIENESTAR UNIVERSITAR. *

AUTORES: HECTOR FERNELY RIOS MORALES, FERNANDO ANDRES ZAMBRANO VILLAR.**

PALABRAS CLAVE: JAVA, Bienestar Universitario, Software, Sistema de Información, Citas.

DESCRIPCIÓN:

Dado el interés por parte de la Universidad Industrial de Santander en la actualización de la plataforma existente para el proceso de control de citas, y teniendo en cuenta las ventajas que brinda el uso de las nuevas tecnologías de la comunicación y la información (TIC,) es necesario el desarrollo de un nuevo sistema de información para dar cumplimiento al proceso de solicitud y asignación de citas con el fin de acceder a los servicios integrales de salud ofrecidos por Bienestar Universitario a la comunidad estudiantil.

Para el desarrollo de este proyecto es necesario analizar, diseñar, implementar y poner en funcionamiento el nuevo sistema de información de citas orientado a la Web basados en Java 5, siguiendo con los estándares establecidos por la DSI (División de Servicios de Información), satisfaciendo las necesidades presentes y futuras previstas por Bienestar Universitario, cumpliendo de esta forma con los objetivos propuestos en el proyecto.

Una vez puesto en funcionamiento el software desarrollado, será una gran herramienta que facilitará y optimizará los procesos administrativos presentes en todas las funciones de la sección de servicios integrales de salud, haciéndolos más autónomos, algunos de los procesos se automatizarán, y otros tendrán un seguimiento especial, dando la oportunidad al administrador del sistema tener una visión global del comportamiento de las reglas del negocio, lo que permitirá a Bienestar Universitario brindar un mayor y mejor servicio a los beneficiarios en sus diferentes áreas.

---

* Proyecto de grado en la modalidad de Investigación

**Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Ing. Jacksson Sonny González Bayona. Codirector: Ing. Enrique Torres López.

ABSTRACT

TITLE: INFORMATION SYSTEM FOR THE SUPPORT OF THE APPLICATION AND ALLOCATION OF APPOINTMENTS TO HEALTH INTEGRATED SERVICES OFFERED BY THE DIVISION OF UNIVERSITY WELFARE.*

AUTHORS: HECTOR FERNELY RIOS MORALES, FERNANDO ANDRES ZAMBRANO VILLAR.**

KEYWORDS: JAVA, University Welfare, software, Information System, Appointment.

DESCRIPTION:

Given the interest and the need from the Industrial University of Santander in updating the existing platform to control the appointments process, and taking into account the advantages offered by the use of new communication and information technologies (ICT) is necessary the development of a new information system to comply with the application process and appointment scheduling to access to the health integrated services offered by University Welfare for the student community.

For the development of this project was necessary to analyze, design, implement and operate the new information system to control the appointments Web-oriented based on Java 5, following the standards set by the DSI (Information Services Division), satisfying the present and future needs provided by the University welfare, thus complying with the proposed objectives of the project.

Once started up the software developed will be a great tool to facilitate and optimize administrative processes present in every section features of the health integrated services, some of the processes will be automated, and others will have a special track, giving the opportunity for the system administrator to get an overview of the conduct of business rules, which will enable the University Welfare provide better service to beneficiaries in their different areas.

* Degree Project in the modality of research

**Faculty of Physical-Mechanical Engineering. Systems Engineering. Director: Ing. Jacksson Sonny González Bayona. Codirector: Ing. Enrique Torres López.

INTRODUCCION

Con el auge de las tecnologías de la información y de la comunicación TIC, el país ha tenido que ir actualizando su plataforma tecnológica para así llegar a ser competitivos en un mundo globalizado. Por eso la División de Bienestar Universitario, en busca de cumplir los objetivos institucionales de mejorar los servicios ofrecidos a la comunidad estudiantil, uno de dichos servicios es la solicitud y asignación de citas médico asistenciales.

A causa del aumento de los beneficiarios, Bienestar Universitario junto con la División de Servicios de Información, han tenido la necesidad de implementar un nuevo sistema, el cual ofrezca mayor cobertura, facilidad de uso, mejor soporte y reducción de tiempos de los usuarios.

Este proyecto pretende diseñar, implementar e implantar un sistema de información para la solicitud y asignación de citas medico asistenciales, el cual permitirá a los profesionales de la salud poder realizar las agendas, solicitar citas disponibles a los usuarios y llevar un control adecuado del uso del servicio. Para ello se utilizaron los estándares definidos por la División de servicios de Información en el desarrollo de software y utilizando herramientas como Java EE 5 e Informix.

1. PRESENTACION DEL PROYECTO

1.1 Especificaciones

Título

SISTEMA DE INFORMACIÓN PARA EL SOPORTE DE LA SOLICITUD Y ASIGNACIÓN DE CITAS A LOS SERVICIOS INTEGRALES DE SALUD OFRECIDOS POR LA DIVISIÓN DE BIENESTAR UNIVERSITARIO.

Director del proyecto:

Nombre: Ing. Jacksson Sonny González

Institución: Universidad Industrial de Santander

Cargo: Profesional adscrito a la división de Servicios de Información.

Codirector del proyecto:

Nombre: Ing. Enrique Torres López

Institución: Universidad Industrial de Santander

Cargo: Profesional adscrito a la división de Servicios de Información.

Autores:

Nombre: Héctor Fernely Ríos Morales

Código: 2060548

Carrera: Ingeniería de sistemas e informática

Nombre: Fernando Andrés Zambrano Villar

Código: 2061458

Carrera: Ingeniería de sistemas e informática

Entidades interesadas en el proyecto:

División de Servicios De Información de la Universidad Industrial de Santander

División de Bienestar Universitario de la Universidad Industrial de Santander

1.2 Objetivos Del Proyecto

1.2.1 Objetivo general

Desarrollar el sistema de información para la solicitud, asignación y control de citas médicas para los servicios integrales de salud de Bienestar Universitario.

1.2.2 Objetivos específicos

Implementar el software para el Sistema de Información contemplando los componentes que permitan alcanzar los siguientes objetivos:

- Administrar las agendas de los profesionales de la salud de Bienestar Universitario.
- Generar las citas ofrecidas teniendo en cuenta los recursos de planta física disponible y la agenda de los profesionales de la salud de Bienestar Universitario.
- Asignar las citas médicas programadas de acuerdo a los criterios establecidos, que son: a) solicitud directa de los usuarios a través de la Web, b) citas de control, c) citas de urgencias.
- Controlar el manejo de citas y de la asistencia a las consultas de los usuarios solicitantes.

- Asignar automáticamente las citas médicas y odontológicas a los estudiantes nuevos, para su examen de ingreso a la universidad.
- Generar las sanciones a los estudiantes que incumplan las citas según reglas establecidas en Bienestar Universitario.
- Elaborar las estadísticas que sirvan como soporte administrativo para la toma de decisiones en lo relacionado con el proceso de programación, asignación y control de citas médico asistencial..
- Generar automáticamente recordatorios vía e-mail tanto a pacientes como profesionales, de la programación de citas solicitadas, de citas de control programadas y de agendas.

1.3 Justificación y definición del problema

Dado el tamaño del sistema de información existente y la alta calidad académica con la que cuenta la universidad, se decidió dar la oportunidad para que estudiantes de Ingeniería de Sistemas e Informática que deseen realizar su proyecto de grado y colaborar a la DSI en dicha actualización puedan hacerlo.

Con el cambio en las organizaciones a través de los años, es necesario realizar una actualización periódica del software y en algunos casos un cambio en la plataforma utilizada. En el caso en particular la DSI, decidió migrar el sistema de información existente en la Universidad Industrial De Santander a una plataforma más robusta y de mayor cobertura como lo es JAVA EE 5.

Este nuevo sistema orientado a Web, permite a los usuarios utilizarlo en cualquier equipo con conexión a Internet, disminuyendo las filas que se generan diariamente en Bienestar Universitario al momento de solicitar una cita médica, además contará con un módulo estadístico el cual permitirá llevar un control de las citas solicitadas, asignadas, no asignadas, inasistidas y canceladas, para así poder brindar un mejor servicio a toda la comunidad estudiantil. También se podrá tener un control a las diferentes sanciones

aplicadas a los estudiantes, originadas por el mal uso del servicio que ofrece Bienestar Universitario.

1.4 Impacto

El sistema reducirá el tiempo y las congestiones presentadas al momento de requerir los servicios integrales de salud ofrecidos por Bienestar Universitario, además de automatizar la adjudicación de la primera cita a los estudiantes nuevos y el proceso de sanciones a los estudiantes infractores.

La realización de este proyecto servirá para aumentar la calidad y cobertura del servicio prestado puesto que permitirá a los estudiantes acceder al sistema vía Web, con la facilidad de tener a su alcance las alternativas de solicitud de servicios de salud ofrecidos por Bienestar Universitario.

1.5 Viabilidad

La viabilidad del proyecto es total, puesto que se cuenta con los recursos humanos y técnicos necesarios para garantizar el éxito de su desarrollo. El sistema se desarrolla bajo la supervisión de los directores del proyecto, los cuales cuentan con una amplia experiencia en diseño y desarrollo de este tipo de sistemas.

Además las herramientas de desarrollo utilizadas son de distribución libre o con licencias adquiridas por la División de Servicios de Información, también se cuenta con las instalaciones, equipos y soportes necesarios para su cumplimiento.

2. MARCO TEORICO

2.1 División de Bienestar Universitario

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

2.1.1 Sección de Servicios Integrales De Salud

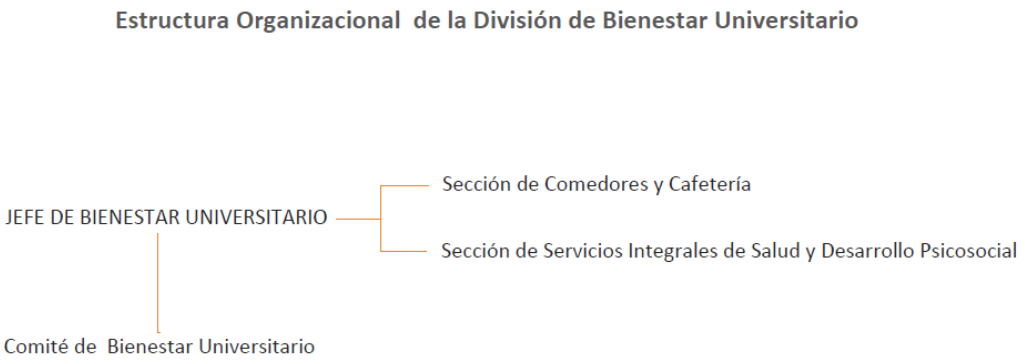


Figura 1. Estructura Organizacional de la División de Bienestar Universitario

La DBU está dividida en tres secciones, y una de ellas, sobre la cual se trabajará en este proyecto es la sección de Servicios Integrales de Salud y Desarrollo Psicosocial, la cual se encarga de velar por los siguientes objetivos del Bienestar Universitario:

- Ofrecer y mantener servicios y programas que promuevan la formación integral y el mejoramiento de la calidad de vida de la comunidad estudiantil.

- Prestar servicios de salud en el primer nivel de complejidad para favorecer las condiciones de salud y contribuir a la formación y desarrollo integral de los estudiantes.

- Fomentar en la comunidad estudiantil la promoción de la salud, la prevención de enfermedades, el autocuidado y la adopción de estilos de vida saludables que propendan por una mejor calidad de vida y una nueva cultura de salud.

Actualmente, esta sección ofrece dentro del campus universitario los siguientes servicios:

- Consultas asistenciales: Consultas asistenciales de atención en salud en las áreas de: Medicina General, Odontología General, Fisioterapia, Nutrición, Psicología, Trabajo Social y Psicopedagogía.

- Consultas Especializadas: Consultas en Oftalmología y Optometría por profesionales adscritos, prestadas en consultorios particulares; consulta especializada en Ginecología y Psiquiatría por médicos pertenecientes a la Escuela de Medicina, en consultorios del servicio de Salud de Bienestar Universitario; consulta especializada en Homeopatía, Sexología y Medicina Familiar con profesionales especializados de Bienestar Universitario.

- Atención de Enfermería: Se presta servicios de enfermería en lo relacionado con inyectología, curaciones, pequeña cirugía, lavado de oídos, lavado de ojos, toma de tensión arterial, suministro de medicamentos y atención de urgencias menores.

- Programas Educativos Preventivos: Son programas de atención especial al servicio de la comunidad estudiantil creados con el propósito de motivar y promover en los estudiantes un mejor autocontrol sobre su salud integral, se tienen, entre otros, los siguientes servicios: Mantenimiento de la salud, espalda sana, salud oral, salud visual, prevención de cáncer de cérvix, mama, testículos, control de la fecundidad.

2.2 División de Servicios de Información

2.2.1 Misión

La División de Servicios de Información de la Universidad Industrial de Santander tiene como fin la administración y el desarrollo de la tecnología de la información en los ámbitos académico y administrativo, definiendo las políticas necesarias para la gestión de la infraestructura de servicios informáticos institucionales, garantizando el adecuado uso de los recursos e impulsando la Innovación tecnológica de la Universidad.

2.2.2 Visión

La División de Servicios de Información de la Universidad Industrial de Santander se proyecta como una organización orientada a utilizar la tecnología de la información como vehículo para el hacer y el saber institucional promoviendo la participación de la comunidad universitaria en la generación y uso de soluciones informáticas de la más alta calidad técnica que faciliten el proceso de modernización institucional.

2.3 Diseño Orientado a Objetos con UML

2.3.1 UML

El lenguaje Unificado de Modelado (UML por sus siglas en inglés) es un lenguaje usado en el diseño de aplicaciones software que ayuda a especificar, visualizar, y documentar modelos de sistemas, incluyendo su estructura y diseño, de manera que satisfaga todos los requerimientos extraídos.

El modelado, es el diseño de las aplicaciones software antes de empezar con la codificación, es una parte fundamental en los grandes proyectos, y también es muy útil en el buen desarrollo de medianos y pequeños sistemas.

Para la realización óptima de estos modelos se trabaja mediante conjuntos de diagramas que permiten trabajar a altos niveles de abstracción.

La División de Servicios de Información ha establecido dentro de sus estándares de desarrollo de software la elaboración de los diagramas de casos de uso, de clases y de secuencia.

2.3.2 Diagramas de Casos de Uso

El modelo de casos de uso especifica la funcionalidad que el sistema ha de ofrecer desde la perspectiva de los usuarios y lo que el sistema ha de realizar para satisfacer las peticiones de estos usuarios. El diagrama de casos de uso visualiza el comportamiento de un sistema, de un subsistema, o de una clase, tal como se muestra a un usuario.

Este modelo utiliza tres elementos básicos:

Actor: Los actores del sistema son un tipo o categoría de usuarios distinguibles, que señalan cuando uno de ellos realiza una tarea con el sistema. Un actor define los diferentes papeles que puede desempeñar un usuario, así, un conjunto de actores dentro de un sistema es lo que constituye toda acción externa al sistema que se está desarrollando.

El símbolo UML que identifica a los actores de un sistema en un diagrama de casos de uso es el siguiente:

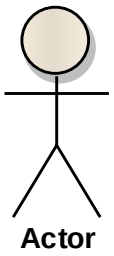


Figura 2. Actor

Casos de Uso: Representa la secuencia de transacciones que se realizan en un diálogo con el sistema, y que se encuentran relacionadas por su comportamiento. La colección de todos los casos de uso relacionados con un

sistema especifica todas las maneras en que se puede utilizar el sistema. Los casos de uso se identifican por un óvalo que contiene la acción a realizar.

Figura 3. Caso de Uso

La especificación de los casos de uso incluye:

- **Secuencia básica:** Es la descripción secuencial de cada caso de uso al ser activado directamente por un actor.
- **Secuencia alternativa:** Es una variante de la secuencia básica, describen subflujos presentados por posibles errores en la ejecución de los casos de uso.

Relaciones: Son los conectores usados para identificar la comunicación existente entre los actores y los casos de uso.

En UML se identifican varios tipos de relaciones:

- **Relación de Generalización Entre Actores:** Se usa para dar orden a los distintos actores, su descripción indica si ciertas características del actor son compartidas con otros.
- **Relación de Generalización Entre Casos de Uso:** Señala que un caso de uso comparte y añade propiedades a un caso de uso general.
- **Relación de Extensión:** Se utiliza entre casos de uso para definir cuando un caso de uso es extendido por otro, especificando cómo la descripción de un caso de uso puede ampliar la descripción de otro.
- **Relación de Inclusión:** Indica que un caso de uso incorpora a su comportamiento natural las características del comportamiento de otro caso de uso.

- **Relación de Asociación:** Señala la comunicación existente entre un actor y un caso de uso.

Los símbolos UML para las relaciones son:

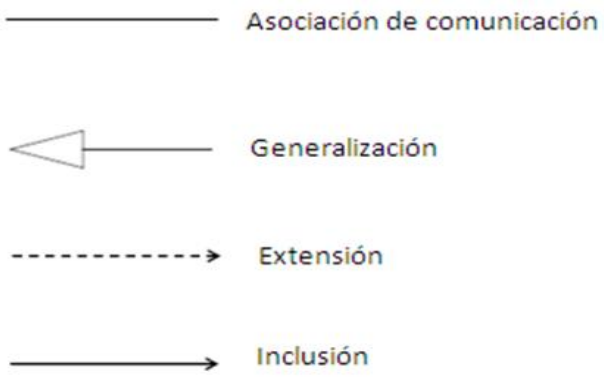


Figura 4. Relaciones Casos de Uso

2.3.3 Diagrama de Clases

El diagrama de clases modela la vista estática del sistema, ya que no describe el comportamiento del sistema en función del tiempo. Un diagrama de clases recoge tanto los conceptos del dominio de la aplicación como aquellos que forman parte de la implementación de la aplicación. [18]

Los elementos principales son:

a) **Clases:** Una clase es una descripción de un conjunto de objetos con las mismas propiedades, el mismo comportamiento, las mismas relaciones con otros objetos y la misma semántica. Una clase puede representar un concepto del mundo real o un concepto de implementación del sistema modelado. Las clases se representan de la siguiente manera:

[18] DE AMESCUA SECO, Antonio; CUADRADO GALLEGO, Juan José. Análisis Y Diseño Estructurado Y Orientado A Objetos De Sistemas Informáticos. Mc Graw Hill, 2003. p. 25 – 27.

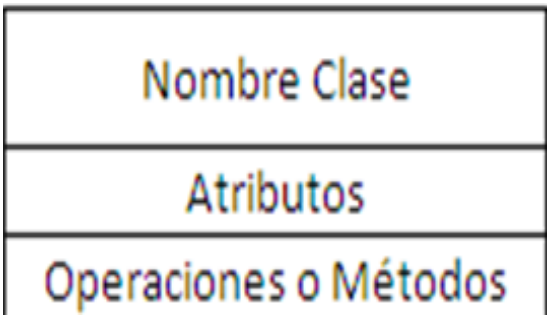


Figura 5. Clase

Tanto los atributos cómo las operaciones pueden ser:

- Públicos: Todo elemento puede ver a la clase con sus atributos y métodos correspondientes. Se representa con el signo +.
- Protegidos: Sólo se puede ver el atributo u operación perteneciente a su propia clase o una hija de ésta. Se representa con el signo #.
- Privados: Los atributos y operaciones indicadas sólo pueden ser vistos por elementos de esa misma clase. se identifica con el signo -.

b) **Relaciones:** Se definen como una abstracción de un conjunto de interrelaciones semánticas puntuales que se dan sistemáticamente entre diferentes tipos de objetos; las relaciones que se presentan entre las clases pueden ser de asociación, agregación o herencia, ésta última se puede encontrar mediante generalización o especialización.

Las clases pueden ser interrelacionadas por medio de las siguientes relaciones:

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

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

depende del tiempo de vida de otro objeto que lo incluye, se dice que es una *Relación de Composición.*

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

La herencia puede darse de dos formas:

- **Generalizada:**
  Ocurre cuando la Súper clase encapsula las propiedades y comportamientos de una o más subclases. En este tipo de relación las subclases no pueden heredar.
- **Especializada:**
  Se da cuando las subclases heredan las propiedades y comportamientos de una clase mayor dándole a esta última un mayor nivel de detalle.

Los símbolos UML para las relaciones ente clases son los siguientes:

Figura 6. Relaciones En Clases

2.3.4 Diagrama de secuencia.

Los diagramas de secuencia son un tipo de diagramas de interacción. Se utilizan especialmente cuando se trata de sistemas en tiempo real, pueden ser excesivamente grandes si intervienen muchas clases.

Una clase en el diagrama de secuencia se representa con un rectángulo en cuyo interior aparece el nombre de la clase. Todas las clases involucradas en el diagrama de secuencia que se esté contrayendo se colocan una al lado de la otra. Debajo de cada clase se coloca una línea vertical. Entre las clases se pueden enviar mensajes, si se está en fase de análisis, o llamadas a métodos, si se está en fase de diseño.[19]

Los símbolos UML para un diagrama de secuencia son:

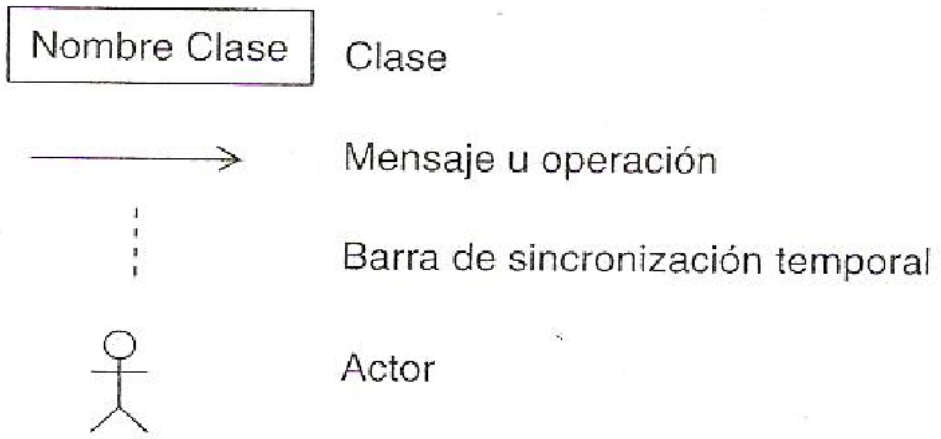


Figura 7. Símbolos Diagrama de Secuencia

2.4 Tecnologías de Desarrollo

2.4.1 AXURE

Es Un constructor de interfaz gráfica del tipo del tipo *WYSIWYG*(“lo que ves es lo que obtienes”, por sus siglas en inglés).

Axure RP es un software interactivo que brinda el poder de entregar prototipos de forma más fácil y rápida que otras herramientas, puesto que genera interfaces de usuario sin necesidad de programación, y permite enviar un

[19] DE AMESCUA SECO, Antonio; CUADRADO GALLEGO, Juan José. Análisis Y Diseño Estructurado Y Orientado A Objetos De Sistemas Informáticos. Mc Graw Hill, 2003. p. 64.

enlace con el resultado final a los clientes o usuarios, para la respectiva revisión del prototipo creado.

2.4.2 ENTERPRISE ARCHITECT

Es una herramienta desarrollada por *Sparx System* que ofrece la capacidad de realizar el modelado de un proyecto de ingeniería de software, y apoyar su desarrollo durante todo su ciclo de vida. Sus repositorios de alta escalabilidad, su habilidad de reunir dominios complejos en una visión unificada, su capacidad para crear sistemas precisos y complejos, y su soporte para más de 10 lenguajes de programación hacen que Enterprise Architect sea una de las herramientas de diseño de software de mayor éxito.

2.4.3 JAVA EE5

Los desarrolladores actualmente reconocen la necesidad de presentar aplicaciones distribuidas, transaccionales, y portables, que exploten la seguridad, velocidad y fiabilidad de las tecnologías del lado del servidor. El diseño de las aplicaciones en las empresas debe estar orientado a producir más con menos recursos. A través de Java Enterprise Edition, el desarrollo de estas aplicaciones se concibe de forma fácil y rápida, cumpliendo con las demandas de los desarrolladores.

JAVA EE5 es una versión liberada de la plataforma de programación JAVA EE, que tiene como objetivo equipar al desarrollador con un potente conjunto de APIs, reduciendo la complejidad de las aplicaciones.

Una novedad en JAVA EE5 es la presencia de la JAVA Persistence API, la cual provee un mapeo objeto/relacional para el manejo de datos relacionales en *Enterprise beans,* componentes Web y aplicaciones cliente. Ésta API también puede ser usada en un entorno externo a JEE tal como aplicaciones JSE.[20]

[20] An Introduction to the Java EE Platform <http://download.oracle.com/javaee/5/firstcup/doc/> [Consultado: 5 de junio de 2011]

2.4.3.1 Arquitectura Java. La arquitectura Java EE puede dividirse en contenedores, éstos son las interfaces entre componente y funcionalidad de bajo nivel proporcionadas por Java EE para brindar soporte a ese componente. Sin embargo, el servidor Java EE permite a los diferentes tipos de componentes trabajar juntos para garantizar la funcionalidad de la aplicación.

La siguiente figura muestra las relaciones de los elementos que conforman la arquitectura de la plataforma Java EE:

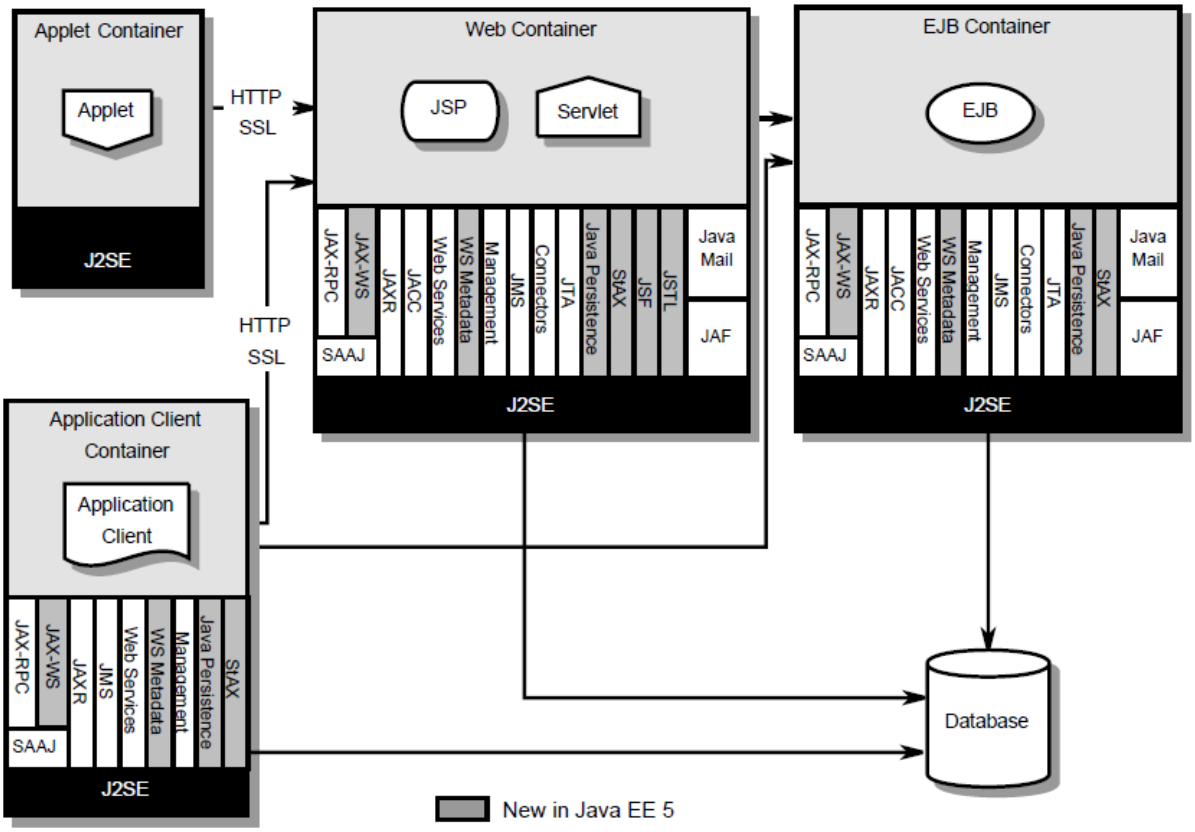


Figura 8. Arquitectura Java EE

Cada uno de los rectángulos separados son entornos de ejecución en la plataforma Java EE, que proporcionan los servicios requeridos para cada uno de los componentes de la aplicación identificados en cada cuadro; por ejemplo, el contenedor de aplicación cliente proporciona Java Message Service (JMS) para clientes de aplicaciones.

Las flechas en la figura indican el acceso a otras partes de la plataforma Java EE. Cómo se observa, el contenedor de aplicación cliente proporciona a los clientes acceso directo a la base de datos de Java EE a través de la API de Java para la conectividad.

2.4.3.2 Características del Lenguaje Java. Algunas de las características principales que nos ofrece Java son:

- **Portabilidad:** Las aplicaciones o applets creados en Java son indiferentes a la arquitectura sobre la cual se está trabajando, los programas se ejecutan de igual forma en cualquier plataforma gracias a la máquina virtual de Java.
- **Seguridad:** Aunque el desarrollo para múltiples plataformas es, sin duda, una razón importante para utilizar java, la seguridad del lenguaje es lo que ha recibido la mayor atención durante los últimos años. El lenguaje java fue diseñado para ejecutarse en la red, el medio de los hackers de todo el mundo, por lo que se desarrolló pensando en la seguridad.
- **Rendimiento:** Se considera a Java como un lenguaje de alto rendimiento por su velocidad al momento de la ejecución de los programas, su facilidad de cumplir varias funciones al mismo tiempo gracias al uso de multihilos, y su capacidad de ser compilado e interpretado en tiempo real.
- **Orientado a Objetos:** Java fue diseñado como un lenguaje orientado a objetos desde el principio, agrupando los objetos en estructuras encapsuladas que contendrían tanto sus datos como sus métodos (o funciones) facilitando así su manipulación.
- **Integración:** Su capacidad de integrar servicios y componentes hace del desarrollo de aplicaciones en Java un desarrollo sencillo, ágil, robusto, y ampliamente escalable.

2.4.3.3 Modelo de Aplicación Java EE. Este modelo inicia con el lenguaje Java y la máquina virtual de Java. La portabilidad, seguridad, y productividad que proveen estas herramientas son la base del modelo de aplicación de la

plataforma empresarial. Java EE fue diseñado para soportar aplicaciones que implementaran servicios de la empresa para todos aquellos usuarios que interactúen con ella.

El modelo de aplicación Java EE define una arquitectura para la implementación como las aplicaciones de varios niveles, ofreciendo escalabilidad, accesibilidad, y facilidad de gestión que necesitan las aplicaciones de nivel empresarial. Este modelo divide el trabajo necesario para implementar un servicio de varios niveles en dos partes: La lógica y la presentación responsabilidad del desarrollador, y los servicios proporcionados por la plataforma Java EE.

2.4.3.4 Componentes Java EE Un componente. Java EE es una unidad autónoma de software que se ensambla en una aplicación Java EE con sus clases y archivos y que se comunica con otros componentes.

En el estándar JavaEE se especifican los siguientes componentes:

- Aplicaciones cliente y applets, componentes que se ejecutan en el cliente.
- Java Servlets, JavaServer Faces y JavaServer Pages (JSP), componentes web ejecutados del lado del servidor.
- Enterprise Java Beans (EJB) (enterprise beans), componentes empresariales que se ejecutan en el servidor.

Los componentes Java EE son desarrollados en el lenguaje de programación Java compilados de la misma forma que cualquier otro programa. La diferencia radica que los componentes Java EE son ensamblados en una aplicación Java EE, verificando su funcionamiento cumpliendo con la especificación Java EE, y se implementan en la producción, para ser ejecutados y gestionados por el servidor Java EE.

2.4.4 SEAM

SEAM es una poderosa plataforma de desarrollo que facilita la construcción de aplicaciones Web en Java. SEAM permite, a través de sofisticadas

herramientas, integrar en una pila unificada de soluciones diversas tecnologías tales como Asynchronus JavaScript and XML (AJAX), JavaServer Faces (JSF), Java Persistence (JPA), Enterprise Java Beans (EJB 3.0) y Business Process Managment (BPM).

SEAM fue diseñado con el propósito de eliminar la complejidad en la arquitectura y en los niveles API de desarrollo. Esto proporciona al desarrollador la capacidad de ensamblar cómodamente complejas aplicaciones Web mediante simples clases anotadas de java, un conjunto rico en componentes de interfaz de usuario, y algo de XML.[21]

2.4.5 Servidor de Aplicaciones JBOSS (JBOSSAS)

Los desarrollos en Java se realizan sobre el stack de JBoss: JBoss AS (Servidor de aplicaciones)

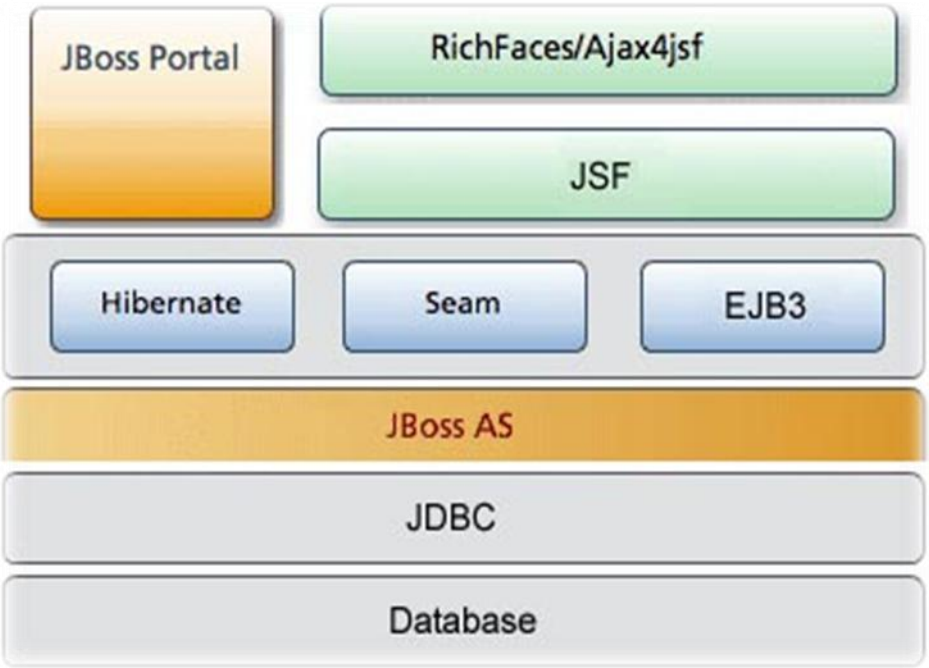


Figura 9: Stack de JBoss[22]

[21] El Framework SEAM <http://seamframework.org/> [Consultado: 5 de junio de 2011]

[22] BDF Tech. Informática y sistemas freelance Group. <http://www.bdftech.com/bdftech-portal/images/development/jboss-stack.jpg > [Consultado: 6 de junio]

JBOSSAS es un servidor de aplicaciones java creado por la comunidad de desarrolladores de código abierto; es una plataforma para la producción de aplicaciones empresariales, Web y portables con el fin de abarcar todo el repertorio ofrecido en JAVA EE 5, así como los servicios adicionales incluyendo: clustering, caching y persistencia.[23]

2.4.6 Aplicaciones Web

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

- **Orientada a la presentación:** Una aplicación web orientada a la presentación genera páginas web interactivas que contienen varios tipos de lenguajes de marcas (HTML, XML, etc) y el contenido dinámico en respuesta a las solicitudes.
- **Orientada a servicios:** Una aplicación web orientada a servicios implementa el punto final de un servicio web. Las aplicaciones orientadas a la presentación son a menudo clientes de las aplicaciones web orientadas a servicios.

Desde la introducción de Java Servlets y la tecnología JSP, Java y otras tecnologías de aplicaciones web interactivas se han desarrollado. La figura 10 ilustra estas tecnologías y sus relaciones.

Cada tecnología agrega un nivel de abstracción que hace más rápido el prototipado y desarrollo de aplicaciones web, logrando al final aplicaciones más robustas, escalables y fáciles de mantener.

[23] Servidor de Aplicaciones JBOSS <http://www.jboss.org/jbossas/> [Consultado: 5 de junio de 2011]

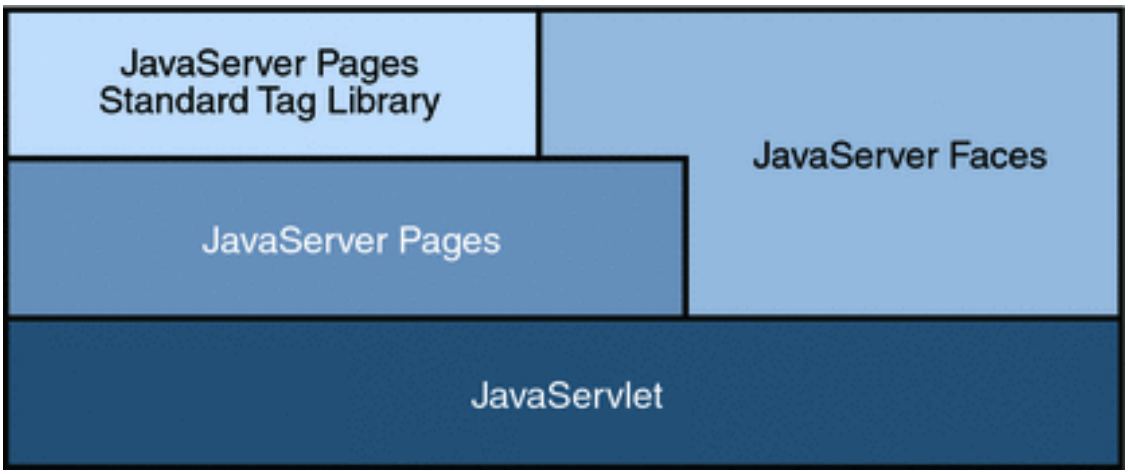


Figura 10. Tecnologías Java para Aplicaciones Web

Los componentes Web son soportados por los servicios de una plataforma de ejecución llamada *contenedor web*. Un contenedor web proporciona servicios tales como solicitud de envío, gestión de la seguridad, concurrencia, y el ciclo de vida. También da a los componentes acceso a las API de web, tales como nombres, transacciones y correo electrónico.

2.4.6.1 Ciclo de Vida de una Aplicación Web. Una aplicación web consta de componentes web, archivos de recursos estáticos como imágenes, las clases de ayuda y las librerías. El contenedor web proporciona muchos servicios de apoyo que fortalecen las capacidades de los componentes web. Sin embargo, debido a que una aplicación web debe tener en cuenta los servicios de aplicación, el proceso de creación y funcionamiento de una aplicación web es diferente al proceso de clases en Java.

El proceso para la creación, implementación y ejecución de una aplicación web se puede resumir de la siguiente manera:

1. Desarrollo del código del componente web.
2. Desarrollo del descriptor de despliegue de la aplicación web.
3. Compilación de los componentes de la aplicación web y las clases de ayuda que referencian los componentes.
4. Empaquetado de la aplicación en una unidad desplegable (opcional).

5. Implementación de la aplicación en un contenedor web.
6. Acceso a una URL que referencia la aplicación web.

2.4.6.2 Módulos Web. En la arquitectura Java EE, los componentes estáticos y archivos de contenido, se llaman **recursos de la web.** Un **módulo** es la unidad más pequeña de despliegue y uso de los recursos web. Un módulo Java EE Web corresponde a una **aplicación** tal como se define en la especificación Java Servlet.

Junto con los componentes web y recursos de Internet, un módulo web puede contener otros archivos:

- Del lado del servidor clases de utilidad (beans de bases de datos, carritos de compra, etc.). A menudo, estas clases, se ajustan a la arquitectura de componentes JavaBeans.

- Del lado del cliente (applets y clases de utilidad).

Un módulo web tiene una estructura específica. El directorio de nivel superior de un módulo web es la raíz del documento de la solicitud. La raíz del documento es donde serán guardadas las páginas JSP, las clases y archivos del lado del cliente, y recursos web estáticos, tales como imágenes.

La raíz del documento contiene un subdirectorio llamado WEB-INF, en el que se encuentran los siguientes archivos y directorios:

- web.xml: La aplicación web descriptor de despliegue
- Archivos descriptores de librerías de etiquetas
- clases: un directorio que contiene las clases del lado del servidor: servlets, clases de servicios públicos, y los componentes JavaBeans
- tags: Un directorio que contiene los archivos de etiquetas, que son implementaciones de bibliotecas de etiquetas
- lib: un directorio que contiene los archivos JAR de bibliotecas importadas por las clases del lado del servidor.

La estructura de un módulo web que se puede implementar en el servidor de aplicaciones se muestra a continuación.

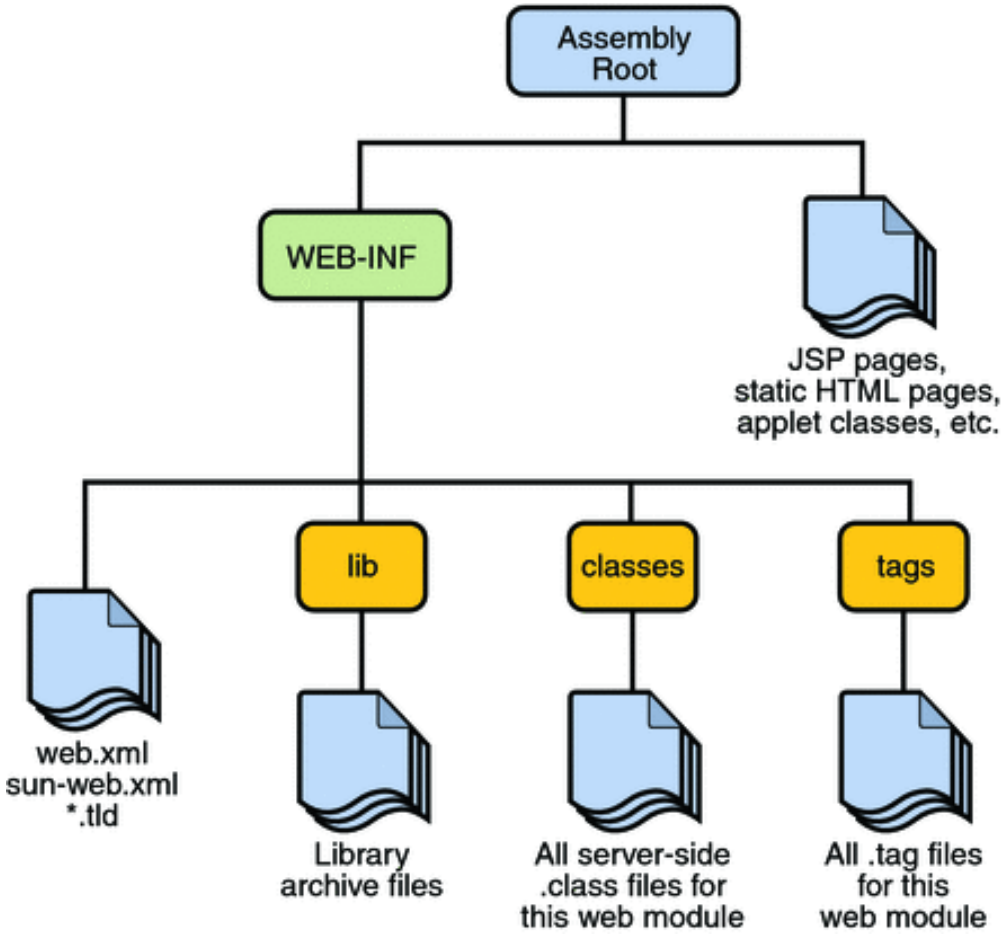


Figura 11. Estructura Módulo Web

Un módulo web puede ser desplegado como una estructura de ficheros sin empaquetar o puede ser empaquetado en un fichero JAR, dado que el contenido y uso de éstos ficheros difiere de otros ficheros JAR, a este empaquetamiento se le asigna la extensión .war.

Para implementar un archivo WAR en el servidor de aplicación, el archivo también debe contener un descriptor de despliegue en tiempo de ejecución. El descriptor de despliegue en tiempo de ejecución es un archivo XML que contiene información tal como la raíz del contexto de la aplicación web y la

asignación de los nombres de los recursos móviles de una aplicación a los recursos del servidor de aplicaciones.

2.4.7 Tecnología Java Servlet

Tan pronto como la web comenzó a ser usada para la prestación de servicios, los proveedores de servicios reconocieron la necesidad de implementar contenido dinámico. A partir de esta necesidad se realizaron varios intentos para ofrecer esta dinámica, pero presentaban deficiencias, como la dependencia de la plataforma o la falta de escalabilidad. Estas limitaciones son superadas con la tecnología Java Servlets, la cual ofrece contenido dinámico al usuario.

Un servlet es una clase Java utilizada para ampliar las capacidades de los servidores que almacenan aplicaciones accesibles a través de un modelo de programación basado en petición-respuesta. Aunque los servlets pueden responder cualquier tipo de solicitud, se usan más para extend de las aplicaciones en servidores web. Para esas aplicaciones, la tecnología Java Servlets HTTP define clases específicas de servlets.

2.4.8 Java Server Faces (JSF)

La tecnología Java Server Faces consiste en un framework de interfaces de usuario del lado del servidor para aplicaciones web basadas en Java.

Esta tecnología se compone principalmente por:

- Una API que representa componentes de interfaz de usuario y gestiona su estado; controla eventos, realiza validaciones del lado del servidor y conversión de datos; define la navegación entre páginas; soporta internacionalización y accesibilidad; y proporciona extensibilidad para todas sus características.
- Dos librerías de etiquetas personalizadas JavaServer Pages(JSP) para expresar componentes de interfaz de usuario dentro de una página JSP y para conectar los componentes con los objetos del lado del servidor.

Un modelo de programación bien definido junto con las librerías de etiquetas permite aliviar la carga de desarrollo y mantenimiento de aplicaciones web. Con un mínimo esfuerzo, se logra:

- Situar los componentes en una página mediante la adición de etiquetas de componentes.
- Ligar los eventos de componentes generados al código de la aplicación del lado del servidor
- Mapear componentes UI a una página de datos en el lado del servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.
- Guardar y restaurar el estado de la interfaz de usuario más allá de la vida de las solicitudes realizadas por el servidor.

Como se observa a continuación, la interfaz de usuario creada con la tecnología JavaServer Faces se ejecuta del lado del servidor para ser mostrada al cliente:

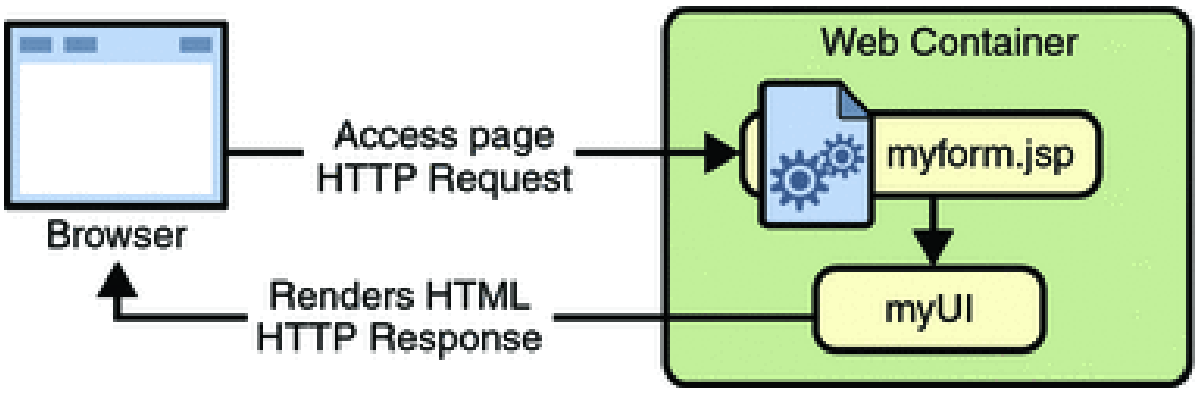


Figura 12. Interfaz De Usuario Ejecutada Del Lado Del Servidor

2.4.8.1 Beneficios de la tecnología JavaServer Faces: Una de las mayores ventajas de la tecnología JSF es que ofrece una clara separación entre el comportamiento y la presentación. Las aplicaciones web construidas con tecnología JSP no logran completamente esta separación. Una aplicación JSP no puede mapear peticiones HTTP para la gestión de eventos específicos de

cada componente, ni administrar los elementos de la interfaz de usuario como objetos con estado en el servidor, algo que sí es posible gracias a JSF.

La separación de la lógica de la presentación permite también a cada miembro del equipo de desarrollo de la aplicación web centrarse en su parte del proceso de desarrollo, y a su vez proporciona un modelo de programación sencillo para unir las piezas creadas. Por ejemplo, un miembro del equipo que no tenga conocimientos de programación puede acceder a los objetos del lado del servidor desde una página web a través de etiquetas de componentes sin necesidad de usar ningún script.

Otra ventaja importante de la tecnología JSF está en que aprovecha los conceptos conocidos de componentes de interfaz de usuario y la capa web de primer nivel sin limitarse a una tecnología de secuencias de comandos o lenguajes de marcado.

Pero lo más importante, la tecnología JSF proporciona una arquitectura sólida para la gestión de estado de un componente, el procesamiento de datos de los componentes, la validación de la entrada del usuario, y el control de eventos.

2.4.8.2 Aplicación JSF. Básicamente, una aplicación JSF es como cualquier otra aplicación Web de Java. Una aplicación JSF contiene los siguientes elementos:

- Un conjunto de páginas JSP (aunque no se limitan a la utilización de las páginas JSP como su tecnología de presentación).
- Un conjunto de *BackingBeans,* componentes JavaBeans que definen las propiedades y funciones de los componentes de interfaz de usuario en una página.
- Una configuración de la aplicación de los recursos de archivos, definiendo las reglas de navegación de páginas, configurando los beans y otros objetos personalizados.
- Un descriptor de despliegue (un archivo web.xml).

- Un conjunto de objetos personalizados creados por el desarrollador de la aplicación. Estos objetos pueden incluir componentes personalizados, validadores, convertidores, o los oyentes de eventos.
- Un conjunto de etiquetas personalizadas para representar los objetos personalizados en la página

Una aplicación Java Server Faces, que incluye las páginas JSP también utiliza las librerías de etiquetas estándar definidos por la tecnología Java Server Faces para la representación de los componentes de interfaz de usuario y otros objetos en la página.

2.4.8.3 Arquitectura Modelo Vista Controlador En JSF: La arquitectura Modelo Vista Controlador (MVC) es una arquitectura software en el que la lógica de presentación se encuentra separada de la lógica del negocio, permitiendo el control, desarrollo, pruebas y mantenimiento de cada capa de la aplicación por separado.

En la tecnología JSF esta arquitectura se encuentra dividida de la siguiente forma:

a) **Modelo:** En las aplicaciones software se presentan a los usuarios ciertos datos procedentes de una realidad sobre la que se pretende actuar; el modelo, pues, es el objeto que representa y trabaja directamente con los datos del programa, respondiendo a los eventos generados por los componentes JSF.
   En JSF se puede distinguir dentro del modelo, principalmente, los *Managed Beans(objetos de respaldo);* los objetos Java, responsables de la lógica de la aplicación; y los métodos de acción.

b) **Vista:** La vista es el objeto que maneja la representación visual de los datos gestionados por el modelo. Genera una representación del modelo al usuario que interacciona con el modelo a través de una referencia propia a éste.

En JSF la vista es el conjunto de ficheros JSP con las librerías de JSF; ficheros xhtml; y otros lenguajes de declaración de páginas. Describen la jerarquía de componentes JSF que conforman cada una de las páginas de la interfaz de usuario de la aplicación, y vinculan los componentes con los *Managed Beans* y sus componentes.

c) **Controlador:** Es el objeto que proporciona significado a las órdenes del usuario al actuar sobre los datos representados. El controlador entra en acción al presentarse un evento sobre el modelo o sobre la vista.
El controlador en JSF se compone del Faces Servlet y los métodos de acción de los Managed Beans. Todas las peticiones HTTP del usuario pasan por el Faces Servlet, se examinan las peticiones recibidas, se renderiza la interfaz del cliente, y se invocan los manejadores de eventos y las acciones del modelo a través de los métodos de los Managed Beans.

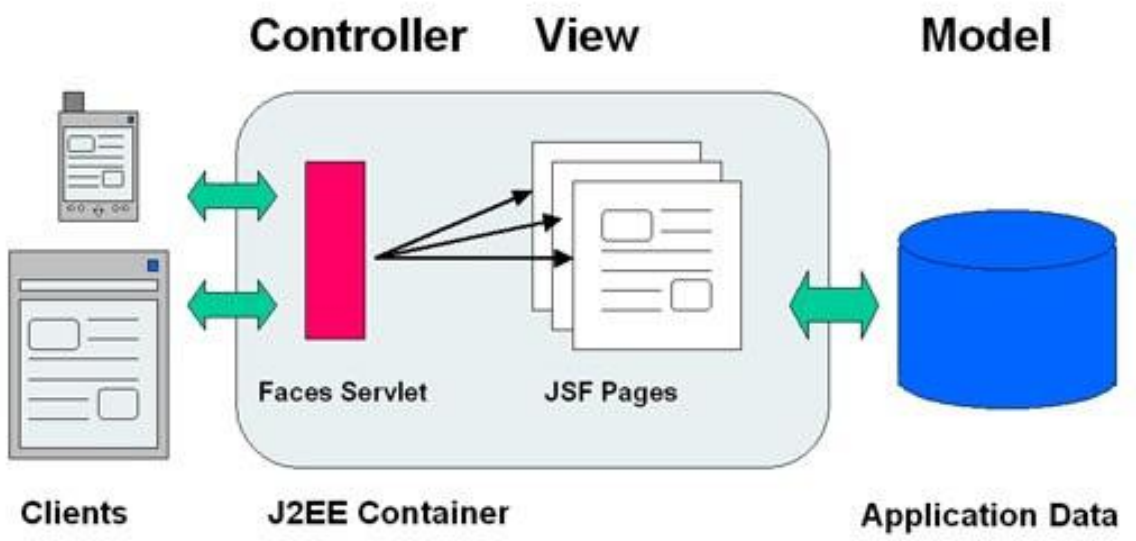


Figura 13. Arquitectura Modelo Vista Controlador En JSF

Este modelo de arquitectura en JSF brinda, entre otras, las siguientes ventajas:

- Existe una clara separación entre los componentes de un programa, lo cual permite su implementación por separado.

- Se encuentra una API muy bien definida, cualquiera que use la API, podrá reemplazar fácilmente el modelo, la vista, o el controlador.
- La conexión entre el modelo y sus vistas es dinámica y se produce en el tiempo de ejecución.

2.4.9 Enterprise Java Beans (EJB 3.0)

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

El modelo de EJB simplifica el desarrollo eliminando la necesidad de una interfaz 'Home' y descriptores de despliegue al reemplazarlos por anotaciones, facilitando la implementación de una nueva manera gracias a la API de persistencia JPA.

Dentro de la arquitectura EJB podemos contemplar diferentes tipos de *beans.*

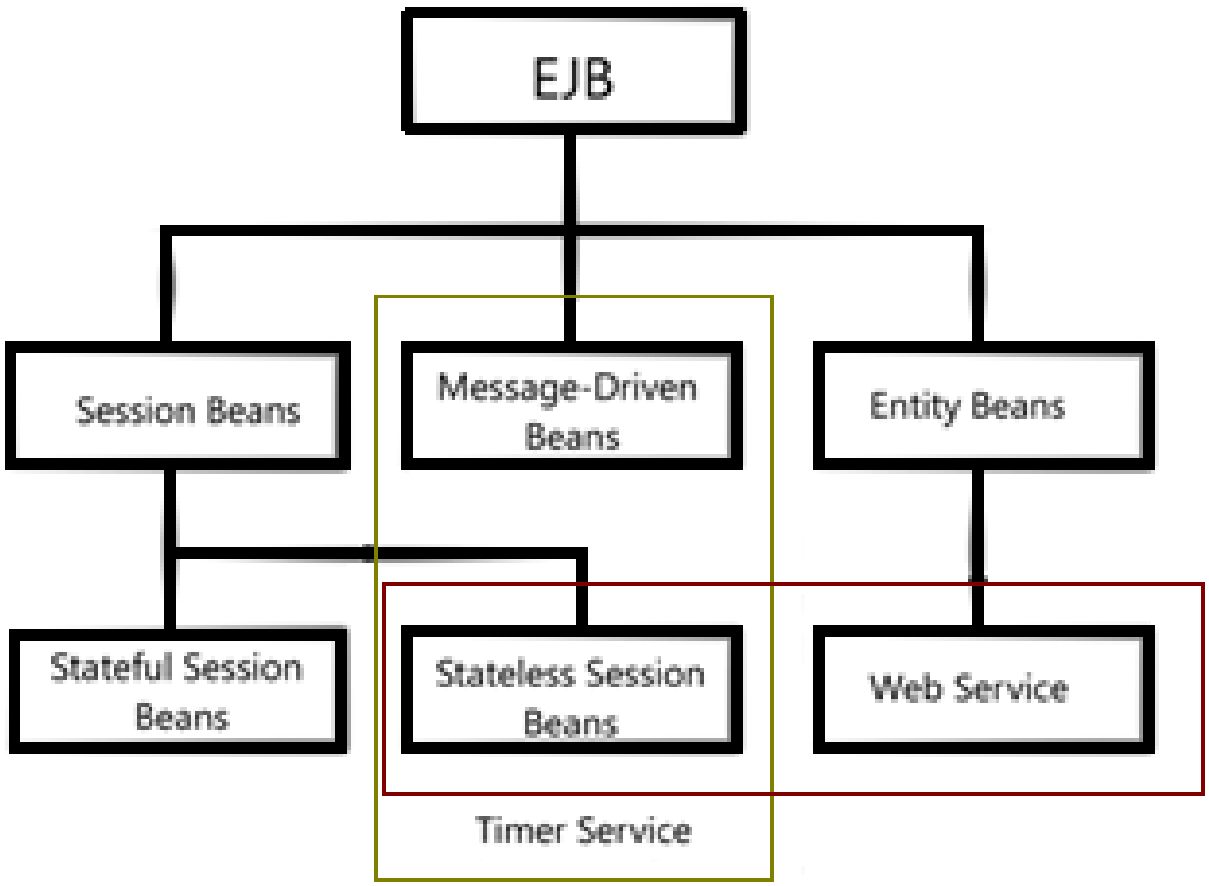


Figura 14. Arquitectura EJB 3.0

Se diferencia entre varios tipos de beans, representados en la figura anterior.

- Los **Session Beans** representan la lógica de la aplicación; y se dividen entre **Stateful Session Beans** (beans con estado que guardan información desde el servidor) y **Stateless Session Beans** (beans sin estado que no necesitan administración desde el servidor).
- Los Messages **Driven Beans**, contienen también la lógica, pero son controlados mediante mensajes.
- Los **Entity Beans** representan los datos de una tabla de base de datos.
- El Timer Service y los Web Services no hacen propiamente parte de la arquitectura EJB ya que son elementos implementados por los Message-Driven Beans, o Stateless Session Beans.

2.4.10 Java Persistence API (JPA)

La Java Persistence API fue desarrollada para la plataforma Java EE y está incluida en el estándar EJB 3.0.

JPA es un modelo de persistencia basado en POJOs que facilita el mapeo de bases de datos relacionales en Java, manteniendo así las ventajas de la orientación a objetos al interactuar con una base datos. Este modelo utiliza anotaciones bean en las entidades, lo cual permite que no se requiera usar archivos descriptores XML.

Una de sus mayores ventajas, es que permite realizar cambios al diseño de la base de datos sin tener que reescribir completamente las aplicaciones mediante la introducción de JPQL.

2.4.11 Java Persistence Query Language (JPQL)

JPQL es un lenguaje de consultas independiente de la plataforma, orientado a objetos, definidos como parte de la especificación Java Persistence API. Permite al desarrollador realizar consultas en las entidades almacenadas en bases de datos relacionales, operando contra objetos entidad JPA y no actuando directamente sobre las tablas de base de datos. Si la base de datos sufriese cambios físicos, sólo habría que adaptar las anotaciones que hacen referencia a ella en el proyecto JPA sin afectar las consultas realizadas por JPQL.

2.4.12 IReport

Cuando se habla de IReport un informe. Si vamos un poco más allá podemos visualizar la inteligencia de negocio, es decir, un conjunto de herramientas enfocadas a la administración y generación de conocimientos mediante el análisis de datos. Aplicando ese concepto, podemos usar la inteligencia de negocio para crear informes, reportes que ayuden a los usuarios finales a tomar decisiones.

2.4.12.1 IReport VS. JasperReports. Son conceptos que la gente puede llegar a confundir fácilmente, hay que tener en cuenta que **IReport** es el entorno de desarrollo donde se elabora la plantilla, informe o reporte, el cual usa lenguaje Groovy o Java, y las extensiones de los informes creados son **.jrxml**(fuente), **. jasper**(compilado); mientras que **JasperReports** hace referencia a las librerías, código, compiladores que permiten el enlace entre una aplicación y el reporte elaborado en IReport.

3. METODOLOGIA DE DESARROLLO

3.1 Ciclo de vida del proyecto

A continuación se procederá a explicar el procedimiento utilizado para el desarrollo del nuevo sistema de información para el soporte de la solicitud y asignación de citas a los servicios integrales de salud ofrecidos por la división de bienestar universitario.

3.1.1 Análisis de Requerimientos.

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel del sistema a desarrollar, y el paso que precede al diseño estructural y funcional del sistema.

En este proceso se especifica la función y comportamiento que tendrá el sistema de información presentado en este proyecto, se identificarán las solicitudes e inquietudes de las entidades interesadas, y se establecerán los estándares, principios y objetivos que se deberán cumplir.

Para esta etapa primero se organizó una reunión con los miembros directamente responsables del proyecto (directores y autores), para discutir la propuesta a presentar a las entidades interesadas (División de Bienestar Universitario), se habló de la problemática actual y se dieron ideas de lo que se quería logar. Llegado a un acuerdo, se prosiguió a presentar la propuesta a través de una serie de entrevistas y reuniones particulares con todos los funcionarios de Bienestar Universitario, quienes aportaron sus puntos de vista acerca del funcionamiento del sistema y los posibles cambios que se podrían realizar en éste, definiendo así las características primarias del sistema a desarrollar contemplando las observaciones presentadas.

Al finalizar las entrevistas se concretó otra reunión con los responsables del proyecto en la que quedaron definidos los objetivos del sistema después de analizar la viabilidad de las exigencias del cliente. Una vez acordados los

objetivos se elaboró un plan de trabajo para el completo desarrollo del sistema de información.

El análisis de requerimientos es un proceso que estará presente durante todo el ciclo de vida del proyecto puesto que en el camino se darán nuevas ideas y se desecharán otras.

3.1.2 Diseño.

El diseño es una parte fundamental en el desarrollo de todo proyecto de ingeniería de software, con un buen diseño, el desarrollo de cualquier sistema es más fácil y su resultado será de muy buena calidad.

La fase de diseño ha de iniciarse una vez se tenga una idea clara de lo que se desea obtener del sistema, esto permite que el proceso entre en constante realimentación cada vez que presenten ideas nuevas o complementarias. El proceso de diseño consta realmente de muchos pasos centrados en cuatro aspectos: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo).

El diseño, en su parte de modelado, se realizó siguiendo los estándares de la DSI, donde se contempla la elaboración de diagramas UML para representar las características del sistema, dichos diagramas fueron creados apoyados en la herramienta Enterprise Architect licenciada por la Universidad Industrial de Santander a través de la División de Servicios de Información.

En esta fase, se desarrolló también un prototipo no funcional que fue presentado a las entidades interesadas con el fin de refinar el diseño del software basado en las nuevas observaciones presentadas.

3.1.3 Implementación.

Esta fase del desarrollo requiere de mucho estudio y cuidado por parte de los autores del proyecto, ya que de algunas decisiones depende que el resultado final satisfaga a las entidades interesadas.

Es en esta etapa donde se empieza a ver reflejado el funcionamiento del sistema, y se debe ser muy cuidadoso con los parámetros establecidos, y con los estándares técnicos y de calidad requeridos.

Las herramientas y tecnologías a utilizar para este desarrollo son principalmente: Lenguaje de programación Java 5; Framework SEAM; Java Server Faces (JSF); Enterprise Java Beans(EJB 3.0); e Informix cómo motor de base de datos.

3.1.4 Pruebas.

La fase de pruebas del software está siempre presente en el desarrollo del sistema, las pruebas se pueden ver como los procesos que permiten verificar la calidad de un producto software.

Desde el momento en que se empieza a generar código, es fundamental iniciar también con el proceso de identificación de fallos de sintaxis, implementación o usabilidad. Las pruebas son esenciales para asegurarse de probar que las sentencias sean correctas, y garantizar que la entrada definida produzca los resultados esperados.

Durante la codificación del proyecto se realizaron pruebas unitarias, Gracias al servidor de aplicaciones JBoss y al acceso a un servidor de versiones, se estuvo monitoreando y probando contantemente el correcto funcionamiento del software durante la fase de desarrollo.

A parte de las pruebas realizadas por el equipo de desarrollo, se dio a conocer el resultado final de la implementación a los clientes finales para que interactuaran con la aplicación y realizaran algunas observaciones que indicaran su grado de satisfacción, que podrían significar posibles ajustes finales.

3.1.5 Ajustes

A lo largo del desarrollo del sistema siempre estuvieron presentes los ajustes, éstos hacen referencia a una constante realimentación en los requerimientos,

en el diseño, en la implementación del software y en la realización de pruebas, a medida que se encontraban errores en alguno de los procesos.

Los ajustes son necesarios para que el sistema sea lo más robusto posible, y se adapte fielmente a los requerimientos del cliente.

3.2 Metodología de Desarrollo

Partiendo de las actividades realizadas durante el ciclo de vida del proyecto, y teniendo en cuenta que cada una requiere de una constante realimentación de requerimientos y ajustes, se implementó como metodología de desarrollo, el modelo de prototipos ó prototipado evolutivo.

La metodología de prototipado consiste en elaborar un modelo software para la evaluación posterior del cliente, con el que posteriormente se llegue a un acuerdo conjunto sobre lo que se desea.

Antes de obtener una visión de lo que serían las páginas en la interfaz de usuario final del software, se tuvo un enfoque de prototipado evolutivo en las fases de análisis y diseño, es decir, se elaboraron y dieron a conocer varios modelos software de cómo los desarrolladores veían el sistema. Estos modelos se discutieron con los directores del proyecto y con los clientes finales hasta conseguir que la propuesta lograra cumplir las expectativas esperadas.

4. DESARROLLO DEL SISTEMA

4.1 Levantamiento de Requerimientos

Por lo general todo desarrollo de software empieza teniendo una idea clara y sólida de las características del sistema que se plantea. Esta idea se forma a través de la interacción desarrollador – director – cliente, en un proceso que se conoce cómo levantamiento de requerimientos.

En el levantamiento de requerimientos deben quedar bien definidos, entre otros, los siguientes aspectos del sistema:

4.1.1 Requerimientos Funcionales

El sistema debe brindar soporte a los usuarios de los servicios integrales de salud ofrecidos por Bienestar Universitario en el proceso de citas médico asistenciales. Para lograrlo se deben tener en cuenta una serie de requerimientos que garantizaran su correcto funcionamiento. Por ejemplo:

- El sistema debe permitir la creación, y edición de registros en tablas soporte profesional, especialidad, consultorio, actividades no asistenciales, indicadores, estado cita, y tipo cita, sin perder nunca integridad.
- El sistema debe contener un módulo para administrar la agenda programada por los profesionales.
- El sistema debe establecer parámetros para su buen funcionamiento, y que éstos pueden ser susceptibles a cambios.
- El sistema debe aceptar el ingreso de estudiantes independientemente del estado en el que se encuentre adscrito a la universidad.
- El sistema debe controlar si el estudiante realizó el pago de matrícula con o sin el monto adicional por salud.
- El sistema debe mostrar si el estudiante tiene deudas con Bienestar Universitario.
- El sistema debe llevar un control de las solicitudes y asignaciones de las citas médico-asistenciales programadas.

- El sistema debe permitir el control de citas prioritarias y citas no programadas.
- El sistema estará en la plataforma web de la Universidad.
- El sistema se realizará bajo los estándares de la DSI.

4.1.2 Usuarios Del Sistema

Se espera que los principales usuarios del sistema sean los estudiantes de la universidad, gracias al beneficio que implica para ellos acceder a este servicio a través de la web.

El sistema también cuenta con un módulo para la administración de Agendas, el cual es de uso exclusivo del profesional para que éste pueda programar de manera fácil y sencilla las actividades relacionadas con las funciones asignadas por el Servicio.

Los usuarios restantes son los asistentes del sistema, que van a controlar y monitorear el proceso de solicitud y asignación de citas; y los administradores que velarán por el buen funcionamiento del sistema.

Todos los usuarios del sistema cuentan con un nivel medio en el manejo de computador, y se les capacitará en el manejo del sistema dejando claro el rol de cado uno.

4.1.3 Restricciones Del Sistema

- No se tiene estipulado que las personas externas a la universidad puedan hacer uso de los servicios de Bienestar Universitario a través de este sistema.
- El profesional no podrá editar su agenda asistencial en días futuros mientras existan citas solicitadas en esos días.
- El estudiante no podrá tener asignadas más de dos citas médico-asistenciales y dos citas de PyP (Programas de Promoción y Prevención) por día.

- El estudiante tendrá permitido cancelar una cita hasta dos horas antes de la hora asignada.

4.2 Estándares de la División de Servicios de Información

4.2.1 Aspectos Generales

4.2.1.1 Interfaz de desarrollo. El IDE de desarrollo a utilizar es el Jboss Developer Studio, el cual debe ser instalado en la carpeta establecida.

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

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

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

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

4.2.1.4 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

4.2.1.5 Espacio de Trabajo (Workspace). El espacio de trabajo se debe crear como 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).

[Sistema]JPA

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

4.2.1.6 Plantillas, estilo, imágenes y formateador. Descargar del servidor de versiones de documentación: las carpetas Estilos, Plantillas e imágenes y copiarlas en la carpeta view de la aplicación.

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

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

4.2.1.8 Código HTML. Está prohibido el uso de etiquetas HTML en las páginas.

4.2.2 Entidades

4.2.2.1 Anotaciones en entidades. Las anotaciones en las entidades se deben colocar antes del método get correspondiente y no en la declaración del atributo. Algunas de las anotaciones requeridas para las entidades se mencionan a continuación:

- La entidad debe llevar las anotaciones: @Entity, @Table, @Name y debe ser una clase Serializable.
- En la anotación @Entity utilizar el parámetro name para identificar el EJB @Entity(name="xxx").
- Las anotaciones requeridas para un campo llave son: @Id, @Column.
- Las anotaciones requeridas para un campo fecha son: @Column, @Temporal, @NotNull (Si el campo es obligatorio).
- Las anotaciones requeridas para un campo alfanumérico son: @Column, @NotNull(Sie el campo es obligatorio) y @Length.
- Las anotaciones requeridas para un campo numérico son: @Column y @NotNull.

4.2.2.2 Llaves compuestas. Se debe utilizar la anotación *EmbeddedId*, la cual obliga a declarar un objeto del tipo de la llave dentro del EJB de entidad.

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

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

Por ejemplo:

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

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

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

4.2.2.5 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 antación 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")

4.2.3 Diagramas UML

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

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

- ✓ Descripción completa.

- ✓ Precondiciones y post-condiciones.
- ✓ Descripción del escenario básico y alternos.
- ✓ Diagrama de clases.
- ✓ Si el caso de uso es complejo se debe incluir el diagrama de secuencia y/o diagrama de actividades.

4.2.3.2 Diagrama de Clases. Para el diagrama de clases se debe tener en cuenta:

- Incluir en cada clase todos los atributos. Esto implica adicionar aquellos que son propios y generados por las relaciones (objetos o listas) que tenga la clase.
- Especificar la dirección (unidireccional o bidireccional) de la relación en el gráfico del diagrama de clases.
- Todo diagrama debe mostrar los atributos de cada una de las clases.
- No incluir los métodos set y get como operaciones de la clase.

4.2.4 Declaración y Control de Acceso

A continuación se presentan los aspectos más importantes que se deben tener en cuenta al momento de crear una clase o método y el control de acceso para los atributos de la misma.

4.2.4.1 Sintaxis Generales

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

4.2.4.2 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.contratación.ordenarPrestaciónServicio

- La estructura para el paquete donde van a estar las entidades comunes para todos los sistemas es el siguiente: co.edu.uis.sistema.entidades.

- La estructura para el paquete donde van a estar los servicios comunes para todos los sistemas es el siguiente: co.edu.uis.sistema.servicios.

4.2.4.3 Clases

- Los nombres de las clases deben iniciar siempre en mayúscula, deben ser simples y descriptivos.
- 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.
- El nombre de los EJB utilizados por el patrón EAO está conformado por un verbo autorizado. Además debe incluir al final las letras "EAOImp". Por ejemplo: RegistrarMAtriculaEstudianteEAOImp.
- La interfaz asociada al EAO debe llevar el mismo nombre del EJB sin el sufijo "Imp". Ejemplo: "RegistrarMatriculaEstudianteEAO.
- Para los EJB de entidad, cuando la clase representa una relación entre dos entidades, el nombre de forma uniendo los nombres de las entidades involucradas.
- El nombre de la clase no contendrá detalles sobre la implementación interna de la misma. Por ejemplo, ArrayEstudiantes no es nombre válido.

4.2.4.4 Atributos

- Para el nombre de los atributos utilizar palabras completas en singular (máximo tres palabras). El nombre debe 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, sólo 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.
- Los atributos de tipo booleano deben usar el prefijo is.
- Todos los atributos declarados a nivel de clase deben ser privados.

4.2.4.5 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 contenido real del método.
- Para los métodos que retornan objeto(s) se usa el prefijo ***get***. Ejemplo: **getProgramasAcademicos(parámetros [opcional])** si este método retorna un listado, en caso de retornar un objeto será: **getProgramaAcademico(parámetros[opcional]).**

- Se debe colocar el modificador de visibilidad private para los métodos que son invocados desde la misma clase.
- En la clase EJB se debe hacer uso de los métodos privados para ser invocados desde los métodos get.

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

4.2.4.7 Nombres de Archivos. Para los nombres de los archivos se sigue el mismo estándar descrito en las reglas de sintaxis generales.

4.2.4.8 Documentación. La documentación relacionado con el diseño del sistema reside en la base de datos de Enterprise 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

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

Figura 15. Plantilla Principal

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

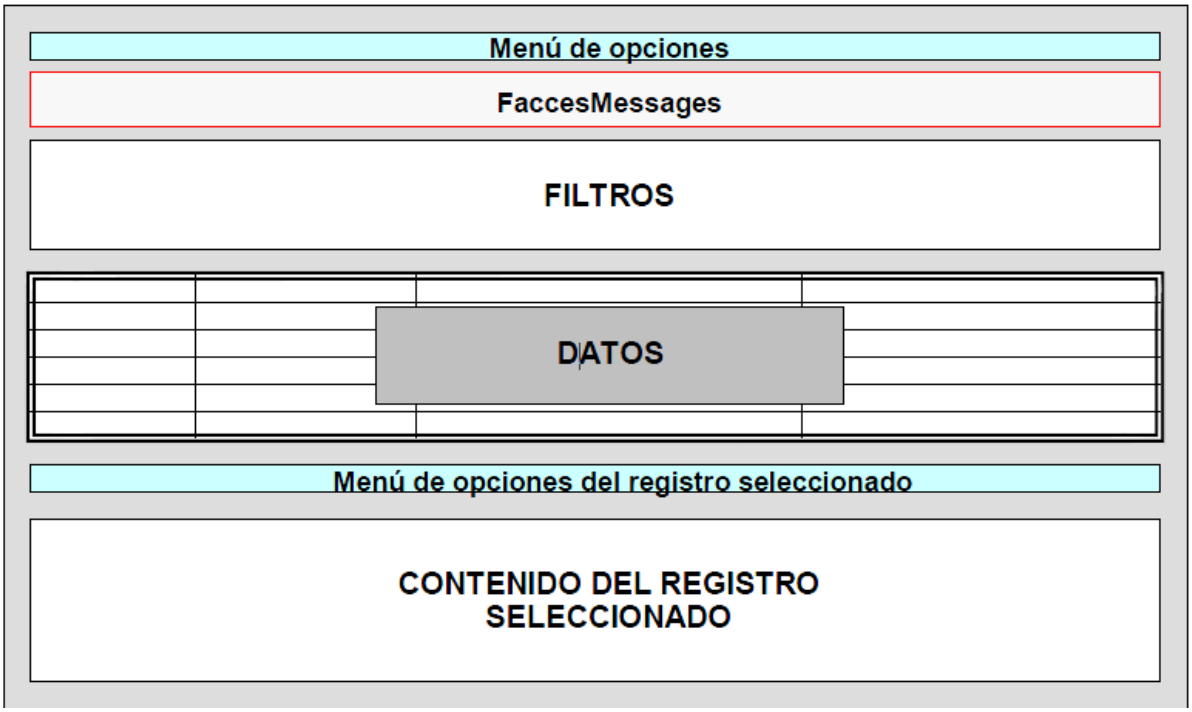


Figura 16. Plantilla de Contenido

Para las páginas que muestran formularios:

Figura 17. Plantilla Formularios

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

Por ejemplo, se quiere consultar los estudiantes de la sede Bucaramanga, trayendo la siguiente información: nombre del estudiante, documento código, programa académico, nivel de condicionalidad. 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.

Para su implementación en la capa de presentación se debe utilizar el control **datascroller**, de richfaces, con 20 registros por página. Por ejemplo:

```
<h:form>

  <a:outputPanel id="tblTabla">

  <rich:dataTable value="#{condicionalidades}"
var="v"
rendered="#{condicionalidades != null and
```

```
condicionalidades.rowCount > 0}"
width="100%" onRowMouseOver="this.style.backgroundColor='#E1E1E1'"
   onRowMouseOut="this.style.backgroundColor=
'#{a4jSkin.tableBackgroundColor}'"
   id="dtbCondicionalidades"
   rows="20">

   <f:facet name="header">
     <rich:columnGroup>
       <rich:column width="15%">
         <rich:spacer></rich:spacer>
       </rich:column>
       <rich:column
         width="85%">#{FormatoWeb.getMensaje('descripcion',true)}
       </rich:column>
     </rich:columnGroup>
   </f:facet>

   <rich:column width="15%">
     <div align="center">
       <h:outputText value="#{v.codigo}" />
     </div>
   </rich:column>

   <rich:column width="85%">
     <div align="left">
       <h:outputText value="#{v.descripcion}" />
     </div>
   </rich:column>
  <f:facet name="footer">
   <rich:datascroller
     id="dscCondicionalidades">
     <f:facet name="next">
       <h:outputText
         value="#{FormatoWeb.getMensaje('siguiente',true)}" />
     </f:facet>
     <f:facet name="previous">
       <h:outputText
         value="#{FormatoWeb.getMensaje('anterior',true)}" />
     </f:facet>
   </rich:datascroller>
  </f:facet>

  </rich:dataTable>

  </a:outputPanel>

</h:form>
```

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

- Etiqueta.xhtml: Texto o datos.
- etiquetaColumna.xhtml: El título de la columna de una tabla
- titulo.xhtml: Títulos
- 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.getMensaje(„primerNombre‟, true)}
  </ui:define>
</s:decorate>
```

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

Para la plantilla mostrar.xhtml:

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

4.2.5.5 Edición. Para capturar cualquier tipo de dato en una caja de texto se debe utilizar la plantilla edición.xthml de la siguiente manera:

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

  <ui:define name="label">
```

```
    #{FormatoWeb.getMensaje(„primerNombre“, true)}
  </ui:define>

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

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

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

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

4.2.5.6 Tablas Estáticas. Se debe usar el control panelGrid de Java Server Faces.

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

```
onRowMouseOver="this.style.backgroundColor='#E1E1E1'"
onRowMouseOut="this.style.backgroundColor='#{a4jSkin.tableBackgroundColor}'
```

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

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

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

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

4.2.6 Esquema de Seguridad

El esquema de seguridad de este sistema está definido por la División de Servicios de información de la misma forma que para los diferentes sistemas de información que apoyan la gestión de la Universidad Industrial de Santander, y que están a cargo de la DSI.

El esquema está basado en la estructura **Roles - Usuarios.**

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

4.2.6.1 Estructura de la Base de Datos soporte. La base de datos que soporta el esquema de seguridad contempla las siguientes tablas:

- **Sistema:** Contiene información de los sistemas de información de la universidad. Para cada sistema se especifica: 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 rol, descripción del rol, fecha y hora de creación, fecha y hora 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 encontramos: 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 sistema-rol, fecha y hora de inicio de vigencia del rol en el sistema, fecha y hora de cierre de vigencia del rol en el sistema.

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

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

- **Opción-Menú-Rol:** Contempla las opciones definidas para cada una de las opciones del menú establecido 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 en el menú, fecha y hora de inicio de vigencia de la opción, fecha y hora de cierre de vigencia 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 infromación, los roles que tienen permisos para incluir registros, para modificar registros o para eliminar registros en ella. Contiene: Nombre del sistema de información, nombre de la tabla, clase de permiso(inclusión, modificación, eliminación de registros), fecha y hora de creación del permiso, fecha y

hoya 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 ésta. Considera: Nombre del sistema de información, 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 cierre 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: Nombre del sistema, 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 cierre 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. Especifica: Nombre del sistema, 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 cierre 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.

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

Para cada rol se define el menú de inicio, el cual 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 la base de datos del esquema de seguridad: Rol, Usuario, Sistema, Sistema-Rol, Usuario-Rol, Menú-Rol, Opción-Menú-Rol, descritas anteriormente.

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

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

Este entorno está soportado por las siguientes tablas de la 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, ya descritas.

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

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

4.3 Roles y Usuarios

En este sistema se contemplaron los siguientes roles:

4.3.1 Administrador

Sección Servicios de Salud de la División de Bienestar Universitario. Esta persona tendrá una visión global del sistema, y será la encargada de evaluar y vigilar el cumplimiento de los procesos realizados.

4.3.2 Profesional

Adscrito a la Sección Servicios de Salud de la División de Bienestar Universitario. Los usuarios identificados como Profesionales tendrán una visión particular del sistema, y son los encargados del proceso fundamental de creación de agendas asistenciales.

4.3.3 Asistente

Los asistentes o secretarias de la sección de Servicios de Salud que vayan a tener acceso al sistema, tendrán cómo función principal brindar apoyo a través de consultas al sistema, también podrá hacer parte del proceso de asignación de citas médicas.

4.3.4 Estudiante

El rol estudiante contiene a toda persona que se encuentre ligada a un proceso formal de formación académica dentro de la Universidad Industrial de Santander. Es el principal beneficiario del proceso de solicitud y asignación de citas médicas, y será parte esencial para evaluar la calidad y resultados del sistema.

4.4 Diagramas De Diseño

A continuación se muestran fragmentos de los diagramas UML, con su respectiva documentación, usados en la fase de diseño del sistema.

4.4.1 Diagrama de Casos de Uso

El siguiente fragmento muestra las principales acciones del administrador del sistema.

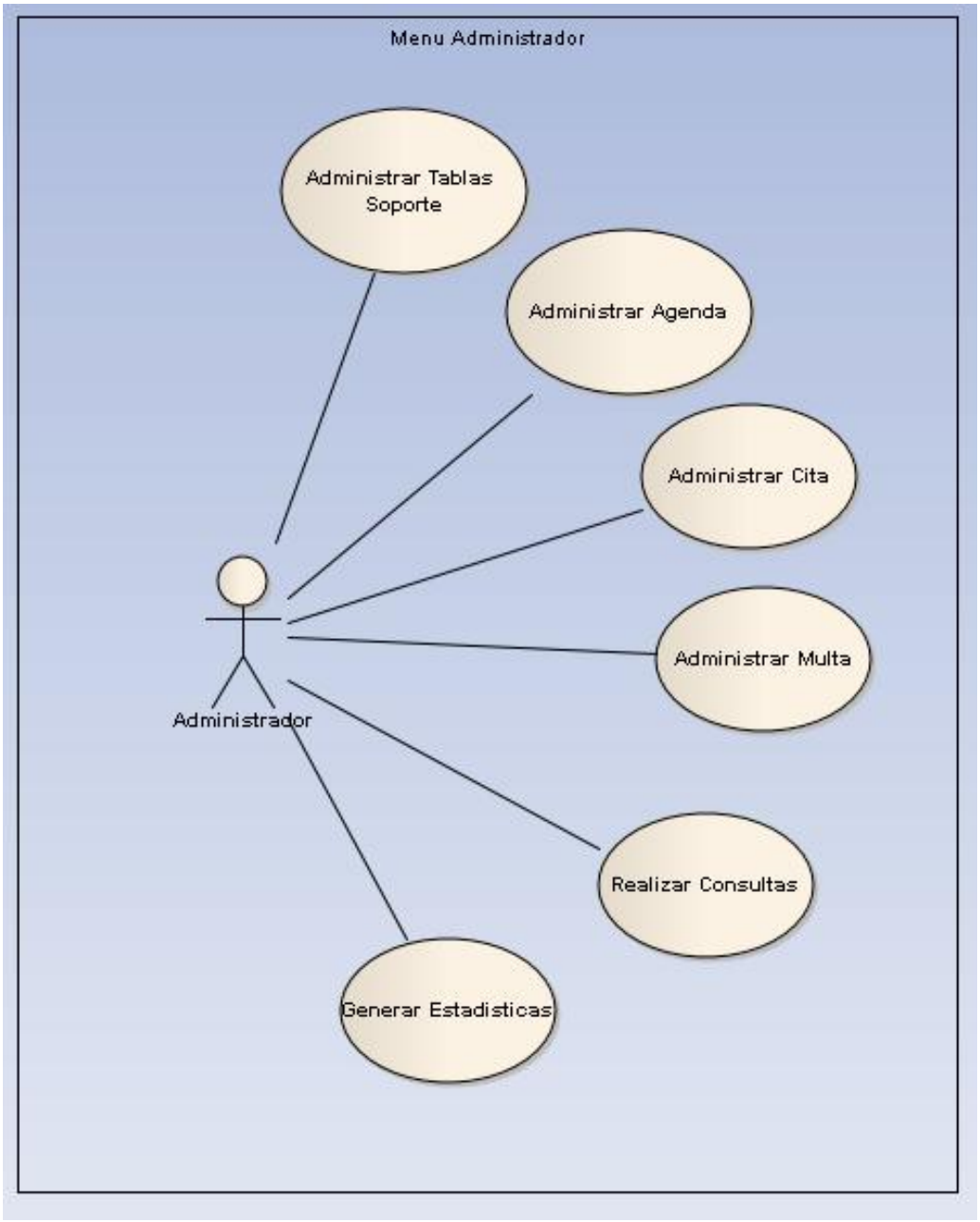


Figura 18. Fragmento Diagrama Caso De Uso Del Sistema

Las siguientes tablas corresponden a la documentación de los casos de uso presentados en la figura anterior (ver Figura 18.)

CASOS DE USO ADMINISTRAR TABLAS SOPORTE		
Caso de Uso	Administrar Tablas Soporte	
Actor(es)	Administrador	
Propósito	Ingresar y editar los registros de los atributos de cada una de las tablas base del sistema de información para su óptimo funcionamiento.	
Descripción	El administrador del sistema tiene acceso a este módulo donde podrá crear, consultar, editar y eliminar registros pertenecientes a las tablas soporte Especialidad, Profesional, Actividad No Asistencial, Consultorio, Tipo Cita, Estado Cita, Parámetros e Indicadores	
Precondición(es)	El administrador del sistema debe estar previamente identificado para tener acceso.	
Flujo Principal	Acciones de Actor(es)	Respuestas del Sistema
	1 Una vez el actor se encuentre autenticado en el sistema, éste escoge en el menú la tabla soporte sobre la cual desea actuar. 3 El actor encontrará en la	2 El sistema renderiza la página correspondiente al menú seleccionado por el administrador.

	página un formulario de creación de un nuevo registro, un panel de consulta, y el listado de los registros existentes en la tabla.	4 Dependiendo de la acción ejecutada por el administrador el sistema actualizará la base de datos y mostrará el mensaje de validación correspondiente a la acción.
Pos condición(es)	Actualización de datos.	

Tabla 1. Administrar Tablas Soporte

CASOS DE USO ADMINISTRAR AGENDAS	
Caso de Uso	Administrar Agendas
Actor(es)	Administrador, Profesional
Propósito	Organizar la agenda semanal de los profesionales según las actividades que planea hacer en su horario de trabajo.
Descripción	Los profesionales adscritos a Bienestar Universitario a través de esta opción, podrán crear, organizar, y visualizar sus actividades a realizar durante la semana seleccionada.

Precondición(es)	1. El usuario debe estar identificado y tener los permisos de acceso necesarios. 2. Las actividades a realizar deben existir en la base de datos.	
Flujo Principal	Acciones de Actor(es)	Respuestas del Sistema
	4. El actor ya identificado, ingresa al menú de agendas y selecciona una fecha específica. 3. El actor al visualizar el estado de su agenda semanal seleccionará un rango de hora y podrá realizar la creación, o eliminación de la actividad.	2. El sistema muestra el estado de la agenda para la semana perteneciente a la fecha seleccionada y valida las acciones que se podrán realizar. 4. El sistema realiza las validaciones correspondientes para la acción seleccionada y renderiza la agenda actualizada.
Pos condición(es)	Actualización de datos de la agenda personal.	

Tabla 2. Administrar Agendas

CASOS DE USO ADMINISTRAR CITAS		
Caso de Uso	Administrar Citas	
Actor(es)	Administrador, Profesional, Asistente, Estudiante	
Propósito	Solicitar, asignar y controlar las citas programadas.	
Descripción	Los actores del sistema pueden administrar las citas programadas ofrecidas por la división de Bienestar Universitario. Dependiendo de su rol tendrán ciertas restricciones para no perder el control del sistema.	
Precondición(es)	1. Los actores deben estar previamente identificados por su rol para poder limitar sus acciones. 2. La agenda asistencial debe existir.	
Flujo Principal	Acciones de Actor(es)	Respuestas del Sistema
	1 Una vez el actor este autenticado en el sistema ingresara al modulo de solicitud de citas. 3 El Actor ingresa los datos necesarios para la solicitud de la cita deseada.	2 El sistema muestra al usuario una interfaz con el formulario a diligenciar para la solicitud de la cita. 4 Dependiendo de la información ingresada por el actor el sistema valida que la

		cita se encuentre disponible y procede a asignarla, luego renderiza la lista de citas solicitadas por el estudiante.
Sub flujo	1 Si el sistema al validar la información ingresada por el profesional encuentra que la cita ya fue solicitada, muestra al Actor un mensaje. 2 Si el Estudiante al ingresar la información deseada no encuentra citas disponibles, podrá realizar un intento de solicitud para que el sistema lleve un control de los usuarios que no pudieron ser atendidos.	
Pos condición(es)	Actualización de datos de la agenda del profesional.	

Tabla 3. Administrar Citas

CASOS DE USO ADMINISTRAR MULTAS	
Caso de Uso	Administrar Multas
Actor(es)	Sistema, Asistente.
Propósito	Asignar, controlar, y levantar sanciones o suspensiones del servicio de salud según algunos parámetros establecidos por la División de Bienestar Universitario.
Descripción	Al finalizar la jornada el Sistema valida la lista de citas del día y asigna multas y/o sanciones a los estudiantes que hayan inasistido, en el caso de las Sanciones el Asistente

	podrá levantarlas desde el sistema.	
Precondición(es)	El estado de la cita debe estar Actualizado.	
Flujo Principal	Acciones de Actor(es)	Respuestas del Sistema
		1 El sistema al finalizar la jornada consulta la lista de citas del día y asigna las multas y/o sanciones a los estudiantes que hayan inasistido las multas.
Pos condición(es)	Actualización de datos de las tablas de multas y suspensiones,	

Tabla 4. Administrar Multas

CASOS DE USO REALIZAR CONSULTAS	
Caso de Uso	Realizar Consultar
Actor(es)	Administrador, Asistente.
Propósito	Hacer consultas sobre las especialidades y las citas de un profesional o de un estudiante dado.
Descripción	Se podrá consultar los detalles de las especialidades citas de un estudiante o un profesional.
Precondición(es)	No existen precondiciones ni restricciones puesto que es sólo un módulo informativo.

Flujo Principal	Acciones de Actor(es)	Respuestas del Sistema
	1. El actor accede al menú consultar y escoge la opción de la que desee obtener detalles.	2. Muestra al usuario una interfaz que permite ingresar criterios de búsqueda necesarios para realizar la consulta.
	3. El actor procede a llenar los campos de búsqueda según lo que quiere consultar.	4. El sistema mostrará una serie de opciones que correspondan a los criterios buscados.
	5. El usuario escogerá dentro de esas opciones la que crea más se acerca a su consulta y podrá ver más detalles.	6. El sistema mostrará un modal panel con la información detallada al ejecutar la acción.
Pos condición(es)	No ocurren cambios en la base de datos.	

Tabla 5. Realizar Consultas

CASOS DE USO GENERAR ESTADISTICAS		
Caso de Uso	Generar Estadísticas	
Actor(es)	Administrador	
Propósito	Generar estadísticas de las solicitudes de las citas por Especialidad.	
Descripción	El administrador del sistema tiene acceso a este módulo, donde se generaran las estadísticas de las citas dadas , y los intentos de solicitud de los estudiantes teniendo en cuenta un rango de fechas	
Precondición(es)	El administrador del sistema debe estar previamente identificado para tener acceso.	
Flujo Principal	Acciones de Actor(es)	Respuestas del Sistema
	1 Una vez el actor se encuentre autenticado en el sistema, éste escogerá el tipo de estadística que desee generar, ya sea estadísticas de las solicitudes dadas o de los intentos de solicitud. 3 El actor seleccionará el rango de fechas para la cual quiere generar las estadísticas.	2 El sistema renderiza la página correspondiente al menú seleccionado por el administrador. 4 El sistema generará las estadísticas, y tabulará los

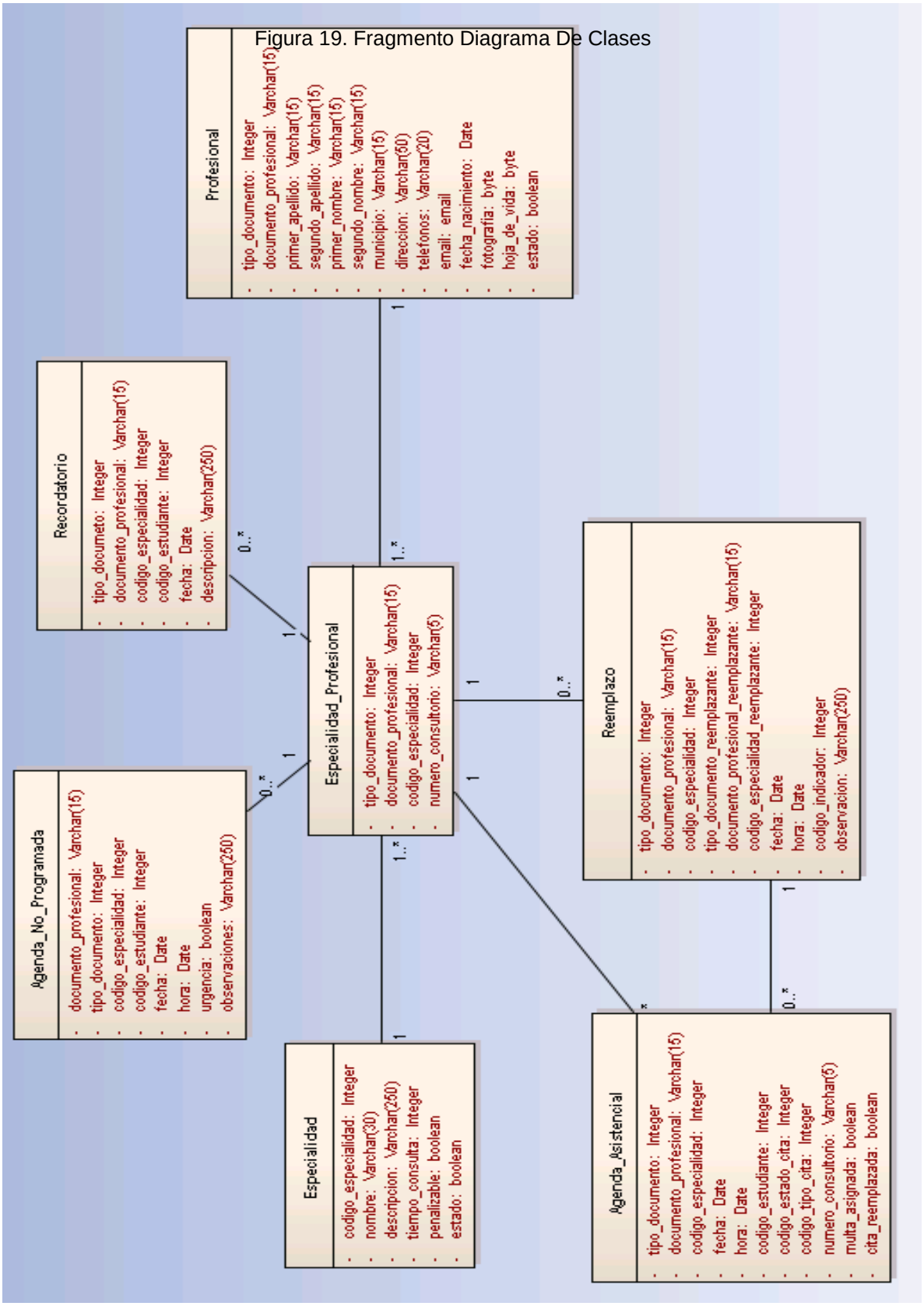
	5 Si el actor lo desea, generara la estadística en un documento PDF.	resultados.
Pos condición(es)		

Tabla 6. Generar Estadísticas

4.4.2 Diagrama de Clases

A continuación se muestra un fragmento tomado del diagrama de clases resultante en la fase del diseño del sistema, y se da una breve descripción de las tablas resultantes.

Figura 19. Fragmento Diagrama De Clases



- **Clase especialidadProfesional:** Contiene la lista de especialidades que pueden ser atendidas por un profesional en particular.

Atributo	Tipo de Dato	Descripción
tipo_documento	Integer	Indicativo del tipo de documento del profesional.
documento_profesional	Varchar(15)	Contiene el documento del profesional.
codigo_especialidad	Integer	Es el código correspondiente a la especialidad a asignar al profesional.
numero_consultorio	Varchar(5)	Contiene el número de consultorio sugerido por el profesional para la atención de esta especialidad.

Tabla 7. Clase especialidadProfesional

- **Clase especialidad:** Contiene información detallada de todas las especialidades ofrecidas dentro de la Sección de Servicios Integrales de Salud de Bienestar Universitario.

Atributo	Tipo de Dato	Descripción
codigo_especialidad	Integer	Número de serie que identifica una especialidad.
nombre	Varchar(30)	Es el nombre que corresponde a una única especialidad.
descripcion	Varchar(250)	Contiene detalles relevantes de cada especialidad.

tiempo_consulta	Integer	Indica la duración de la atención correspondiente a cada una de las especialidades.
penalizable	Boolean	Indica si la especialidad puede ser o no restringida mediante multas o sanciones.
estado	Boolean	Muestra el estado actual de dicha especialidad, si ésta está disponible o no para ofrecer su servicio.

Tabla 8. Clase especialidad

- **Clase recordatorios:** Esta clase crea un recordatorio a ser enviado a los usuarios del sistema a través de un correo electrónico.

Atributo	Tipo de Dato	Descripción
tipo_documento	Integer	Indicativo del tipo de documento del profesional
documento_profesional	Varchar(15)	Contiene el documento del profesional
codigo_especialidad	Integer	Es el código correspondiente a la especialidad del profesional.
codigo_estudiante	Integer	Es el código del estudiante al que se le generará un recordatorio.
fecha	Date	contiene la fecha en la cual se va a enviar el recordatorio
descripcion	Varchar(250)	Contiene una pequeña información del porqué se genera el recordatorio.

Tabla 9. Clase recordatorios.

- **Clase agendaNoProgramada:** Contiene el registro de los servicios integrales de salud que se presenten sin estar programadas en la agenda asistencial del profesional (Urgencias, citas prioritarias)

Atributo	Tipo de Dato	Descripción
tipo_documento	Integer	Indicativo del tipo de documento del profesional.
documento_profesional	Varchar(15)	Contiene el documento del profesional.
codigo_especialidad	Integer	Es el código correspondiente a la especialidad que se requiere.
codigo_estudiante	Integer	Es el código del estudiante que solicita la atención.
fecha	Date	Día en el que se presentó la cita no programada.
hora	Date(Hour To Minute)	Indica la hora en la que se dio la cita.
urgencia	Boolean	Indica si la cita no programada que se atendió fue consecuencia de una urgencia.
observaciones	Varchar(250)	Registra detalles de la eventualidad presentada.

Tabla 10. Clase agendaNoProgramada

- **Clase agendaAsistencial:** Es la agenda creada por cada profesional , donde programa en orden sus jornadas de atención de citas médicas.

Atributo	Tipo de Dato	Descripción
tipo_documento	Integer	Indicativo del tipo de documento del profesional.
documento_profesional	Varchar(15)	Contiene el documento del profesional.
codigo_especialidad	Integer	Es el código correspondiente a la especialidad que se requiere.
fecha	Date	Día programado para ofrecer las citas médicas.
hora	Date(Hour To Minute)	Indica la hora en la que puede darse una cita.
codigo_estudiante	Integer	Es el código del estudiante que solicita la atención.
Codigo_estado_cita	Integer	Indicativo del estado actual de la cita.
Codigo_tipo_cita	Integer	Indica la vía en la que fue solicitada la cita.
numero_consultorio	Varchar(5)	Contiene el número del consultorio en el cual está programado realizar las asistencias médicas.
multa_asignada	Boolean	Indica que la multa por inasistencia ya fue asignada.
cita_reemplazada	Boolean	Indica si una cita fue reemplazada

Tabla 11. Clase agendaAsistencial

- **Clase reemplazos:** Contiene la información del profesional que solicita un reemplazo, los detalles de la solicitud, y los datos del profesional que cubrirá la responsabilidad.

Atributo	Tipo de Dato	Descripción
tipo_documento_reemplazante	Integer	Indicativo del tipo de documento del profesional que realiza el reemplazo.
documento_profesional_reemplazante	Varchar(15)	Contiene el documento del profesional que realiza el reemplazo.
codigo_especialidad_reemplazante	Integer	Es el código correspondiente a la especialidad reemplazada.
tipo_documento	Integer	Indicativo del tipo de documento del profesional que solicita el reemplazo.
documento_profesional	Varchar(15)	Contiene el documento del profesional que solicita el reemplazo.
codigo_especialidad	Integer	Es el código correspondiente a la especialidad a reemplazar.
fecha	Date	Día en el que se realiza el reemplazo.
hora	Date(Hour To Minute)	Hora en la cual se efectuará el reemplazo del profesional.
codigo_indicador	Integer	Indica el posible motivo que generó la solicitud del

		reemplazo.
observacion	Varchar(250)	Breve descripción de la situación en la que se presenta el reemplazo.

Tabla 12. Clase reemplazos

- **Clase profesional:** En esta clase se encontrarán registrados todos los profesionales adscritos a la división de Bienestar Universitario en la sección de servicios integrales de salud.

Atributo	Tipo de Dato	Descripción
tipo_documento	Integer	Indicativo del tipo de documento del profesional.
documento_profesional	Varchar(15)	Contiene el documento del profesional.
Primer_apellido	Varchar(15)	Registra el primer apellido del profesional.
Segundo_apellido	Varchar(15)	Registra el segundo apellido del profesional.
Primer_nombre	Varchar(15)	Contiene el primer nombre del profesional.
Segundo_nombre	Varchar(15)	Contiene el segundo nombre del profesional.
Municipio	Varchar(15)	Contiene el municipio donde reside el profesional.
Direccion	Varchar(50)	Contiene la dirección de residencia

		del profesional.
Telefonos	Varchar(20)	Contiene los números telefónicos para contactar al profesional.
Email	email	Registra el email de contacto del profesional.
Fecha_nacimiento	Date	Registra la fecha de nacimiento del profesional.
Fotografia	byte	Contiene una fotografía del profesional.
Hoja_de_vida	byte	Contiene el enlace a la consulta de la hoja de vida del profesional.

Tabla 13. Clase profesional

4.4.3 Diagramas de Secuencia

Diagrama de secuencias que representa la creación de la agenda por parte del profesional.

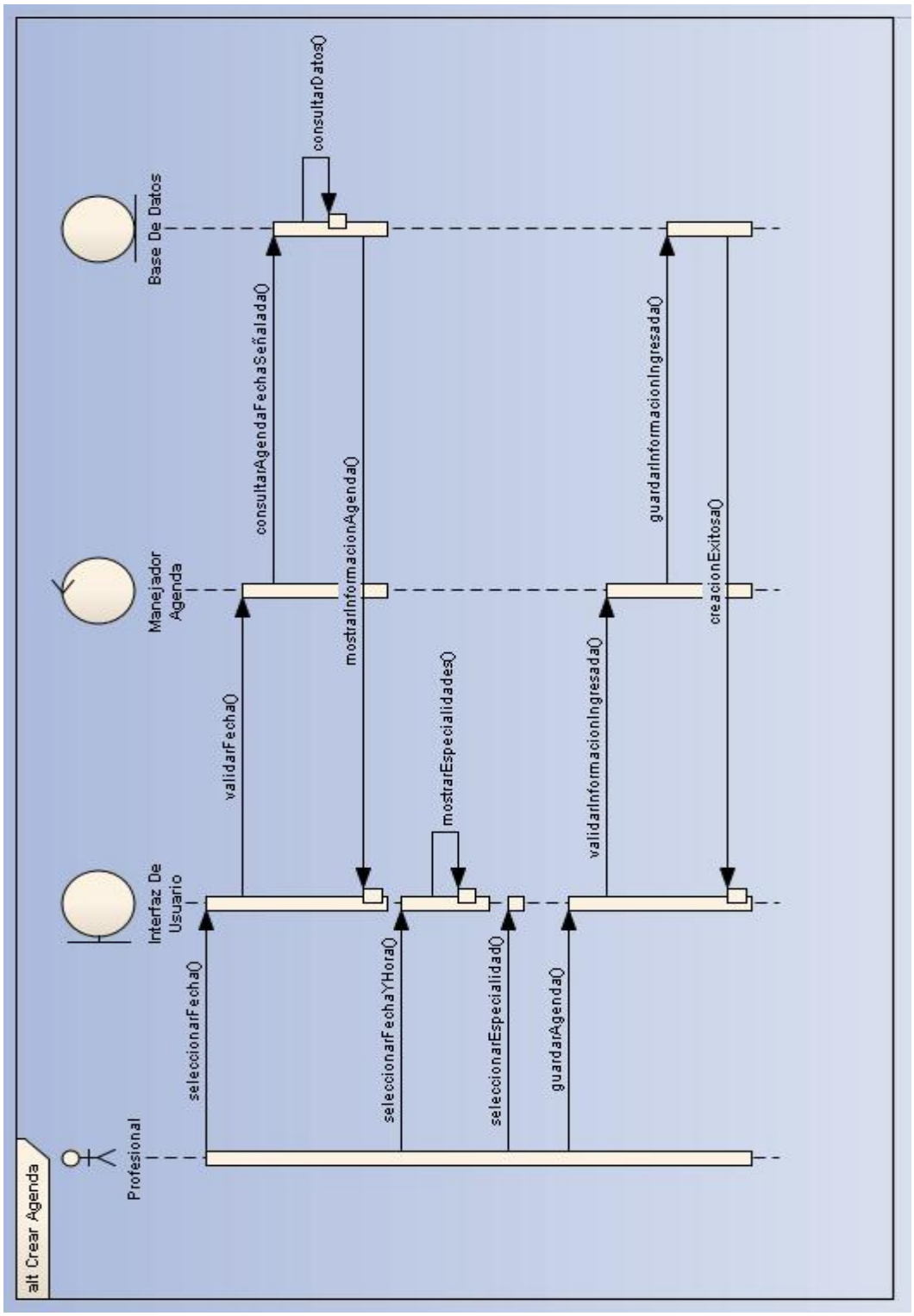


Figura 20. Diagrama De Secuencia Para La Creación De Agenda

Diagrama de secuencias que representa la solicitud de citas por parte de los estudiantes.

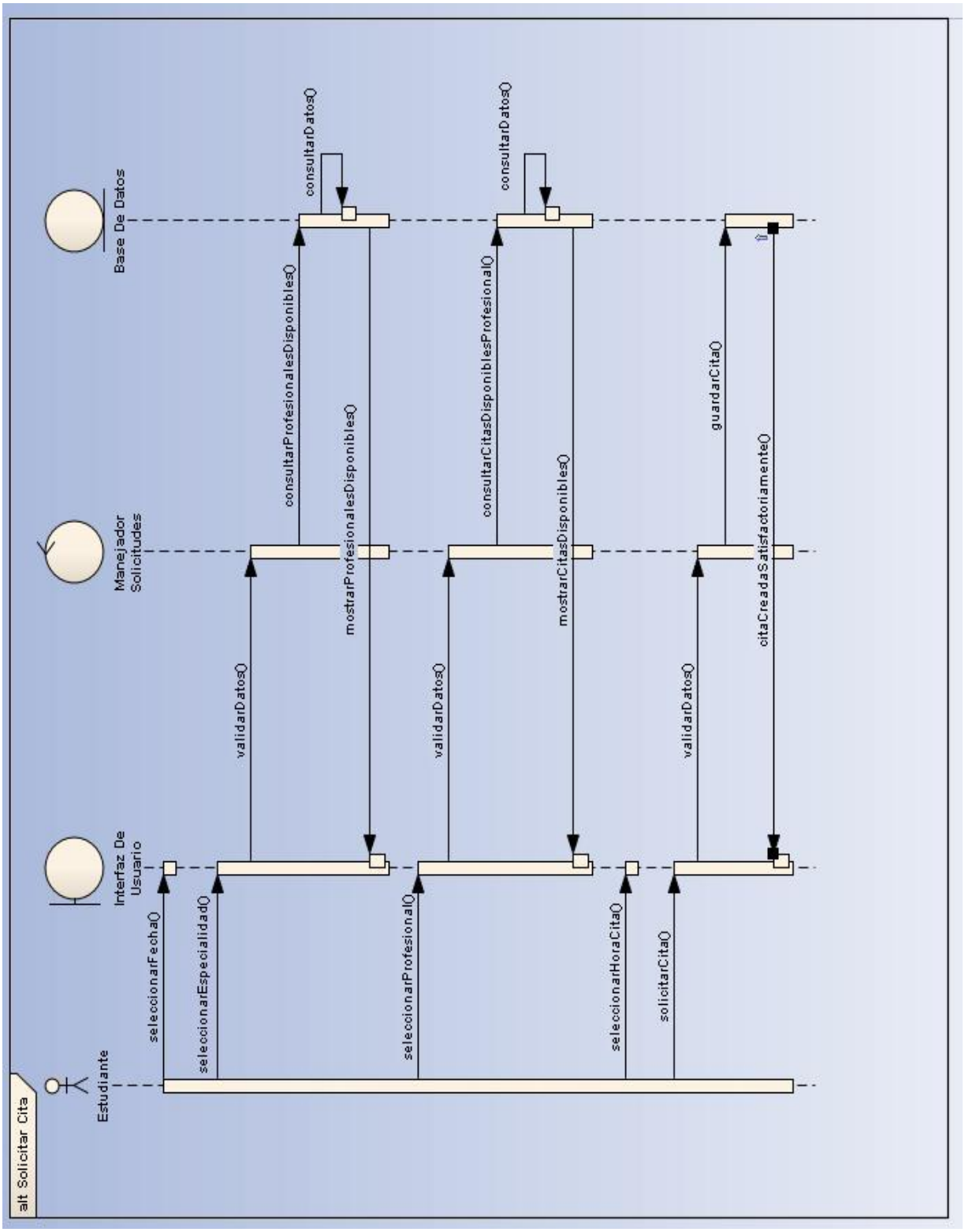


Figura 21. Diagrama de secuencia para la Solicitud de Citas

4.4.4 Prototipo No Funcional

A continuación se muestran dos ejemplos de cómo se construyó el prototipo no funcional:

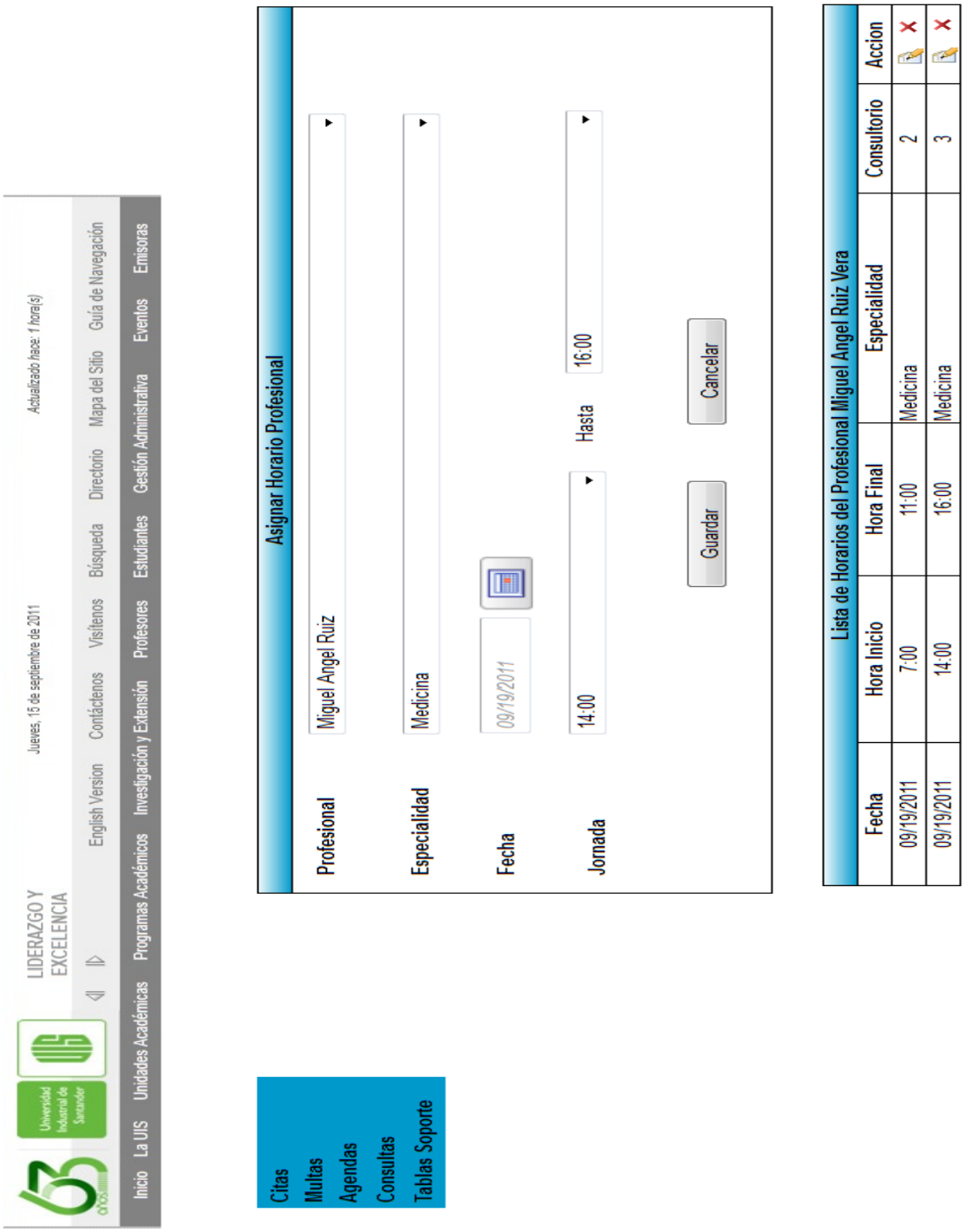


Figura 22. Interfaz Prototipo de Administración de la Agenda Profesional

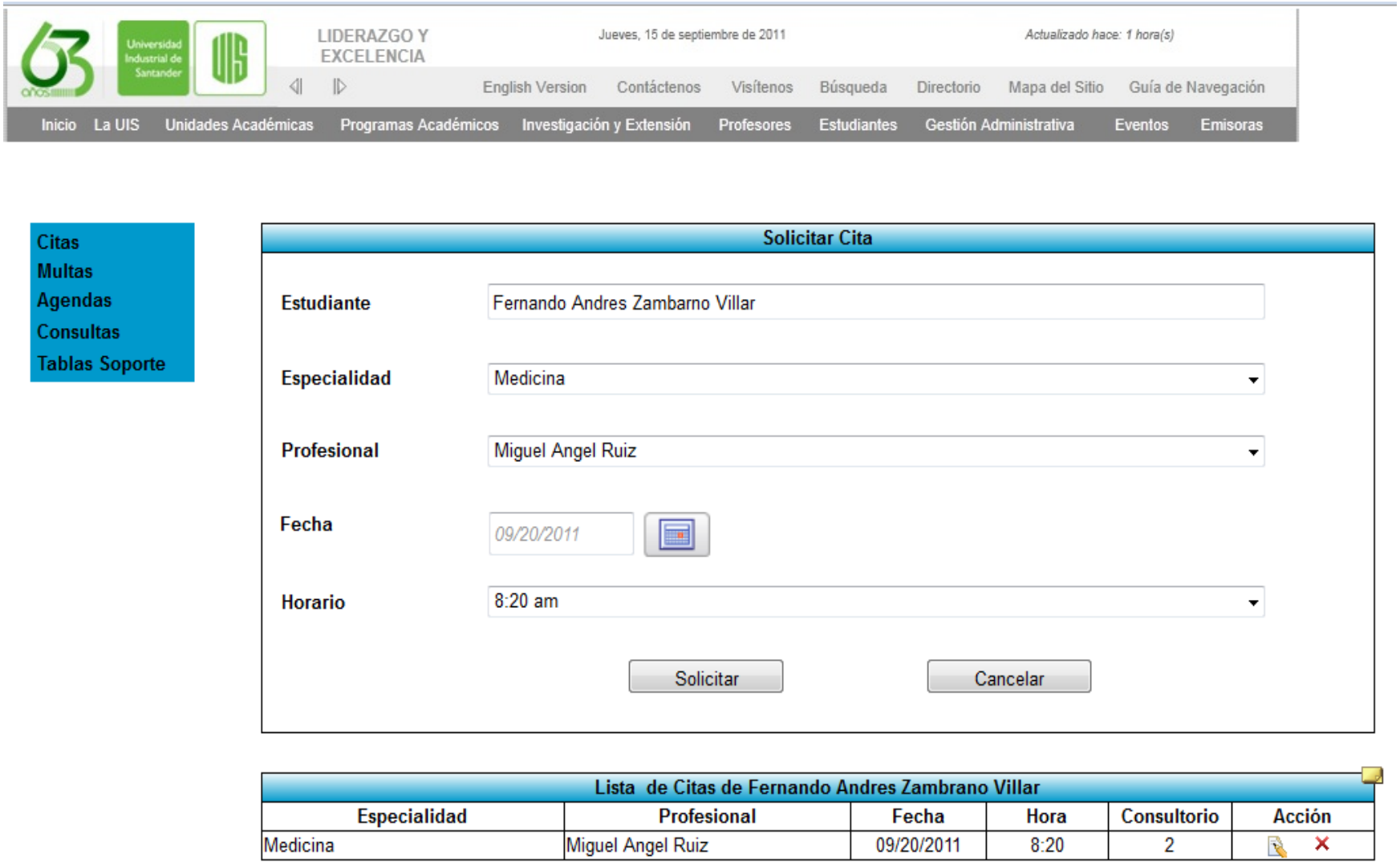


Figura 23. Interfaz Prototipo de Solicitud de Citas para Estudiantes

4.4.5 Prototipo Final

Las siguientes imágenes corresponden al resultado final de las interfaces del sistema:

Figura 24. Interfaz de Administración de la Agenda Profesional

Figura 25. Interfaz de Solicitud de Citas para Estudiantes.

4.4.6 Estructura del Sistema

El sistema de información para la Gestión de Solicitud y Asignación de Citas a los servicios integrales de salud está compuesto por módulos que brindan a los usuarios del sistema la capacidad de interactuar con el sistema de acuerdo a las necesidades que presenten y a los permisos asignados a cada rol.

La estructura y funcionalidad de cada módulo se muestra a continuación:

4.4.6.1 Administración. Este es el módulo encargado de alimentar y mantener las tablas que dan soporte al sistema. Aquí se encuentran los siguientes procesos:

- **Administrar Profesional.** Tabla soporte que permite el registro, edición y consulta de los datos pertenecientes a los profesionales adscritos a la sección de Servicios Integrales de Salud de Bienestar Universitario.

- **Administrar Especialidad.** Módulo usado para gestionar las especialidades médico-asistenciales ofrecidas por Bienestar Universitario a la comunidad Universitaria.

- **Administrar Consultorio.** Sección encargada del manejo de información relevante de los consultorios con los que se cuenta para llevar a cabo las consultas médico-asistenciales.

- **Administrar Actividad No Asistencial.** Permite al administrador del sistema crear diversas actividades que harán parte de la agenda de los profesionales.

- **Administrar Indicador.** Tabla soporte que contiene un listado de conceptos que afectan posibles campos de otras tablas. Ej: Conceptos de sanciones, indicadores de reemplazo.
- **Administrar Tipo Cita.** Sección en la que se encuentran los diferentes medios por los cuales una cita es solicitada. Ej: Web.

- **Administrar Estado Cita.** Módulo para la creación de los diferentes estados que puede presentar una cita. Ej: disponible, solicitada.

- **Administrar Parámetros.** Facilita al administrador Establecer diferentes parámetros para garantizar el correcto funcionamiento del sistema de acuerdo a las necesidades de los clientes. Ej: Tiempo de apertura para la solicitud, períodos de sanción.

4.4.6.2 Agendas. Este es el módulo desarrollado para la creación y organización de las agendas de cada uno de los profesionales adscritos a la sección de servicios integrales de salud de la división de Bienestar Universitario.

- **Agenda Profesional:** Interfaz de administración de las agendas profesionales. A través de un calendario compuesto por horarios semanales se permite al profesional organizar su tiempo y actividades de manera sencilla.

  A nivel lógico la Agenda Profesional se divide en una **Agenda Asistencial,** la cual se compone de las actividades médico-asistenciales que serán atendidas por el profesional; y una **Agenda No Asistencial,** compuesta por otras obligaciones adquiridas por el profesional con la división de Bienestar Universitario.

4.4.6.3 Consultas: Este módulo permite al usuario visualizar información correspondiente al estado actual de las citas ofrecidas, permitiendo llevar un control de las acciones del sistema.

- **Consultar Citas Profesional:** Desde aquí se podrá observar la agenda del profesional, para ver los detalles de las citas disponibles, solicitadas y asistidas.

- **Consultar Citas Estudiante:** Este módulo permite ver las características de las citas solicitadas o asignadas a un estudiante.

4.4.6.4 Citas. A través de este módulo se gestionará todo lo relacionado con las solicitudes a los servicios ofrecidos por el sistema.

- **Solicitud Cita:** Realiza el proceso de solicitud de una cita satisfaciendo la selección de criterios del usuario, siguiendo con los parámetros establecidos.

- **Solicitud Cita Especial:** Se encarga también del proceso de solicitud de una cita, sólo que se encuentran menos restricciones para llevarlo a cabo, pensado para cubrir ciertas eventualidades tales como: controles médicos, citas dobles, este tipo de solicitud sólo se lleva a cabo con la autorización del profesional.

- **Asignación Primera Cita:** Esta sección está orientada a agilizar el proceso de la asignación de citas médicas y odontológicas durante el examen de ingreso a los estudiantes de primer nivel.

4.4.6.5 Sanciones. Este módulo facilita el control de las sanciones que pueda tener un estudiante por deudas adquiridas con la sección de servicios de salud de Bienestar Universitario.

- **Gestionar Sanciones.** Espacio reservado para los usuarios con permisos del rol Asistente, desde aquí se levantan las sanciones a los estudiantes después de que hayan cumplido con los requisitos para saldar la deuda.

4.4.6.6 Estadísticas. Módulo que facilita al administrador tomar decisiones sobre el sistema tras analizar un conjunto de datos representativos.

- **Citas Solicitadas.** Desde aquí se crea un reporte detallado con datos relevantes a las citas solicitadas dentro de un rango de fechas requerido al usuario.

- **Intentos de Solicitud.** Facilita generar un reporte sobre los intentos de solicitud presentados en el sistema con la intención de analizar datos para monitorear el funcionamiento del sistema, y de ser necesario, redefinir parámetros.

Para poder generar los reportes estadísticos utilizando IReports se debe tener instalado el software Jaspersoft iReport Designer Professional y crear la plantilla deseada:

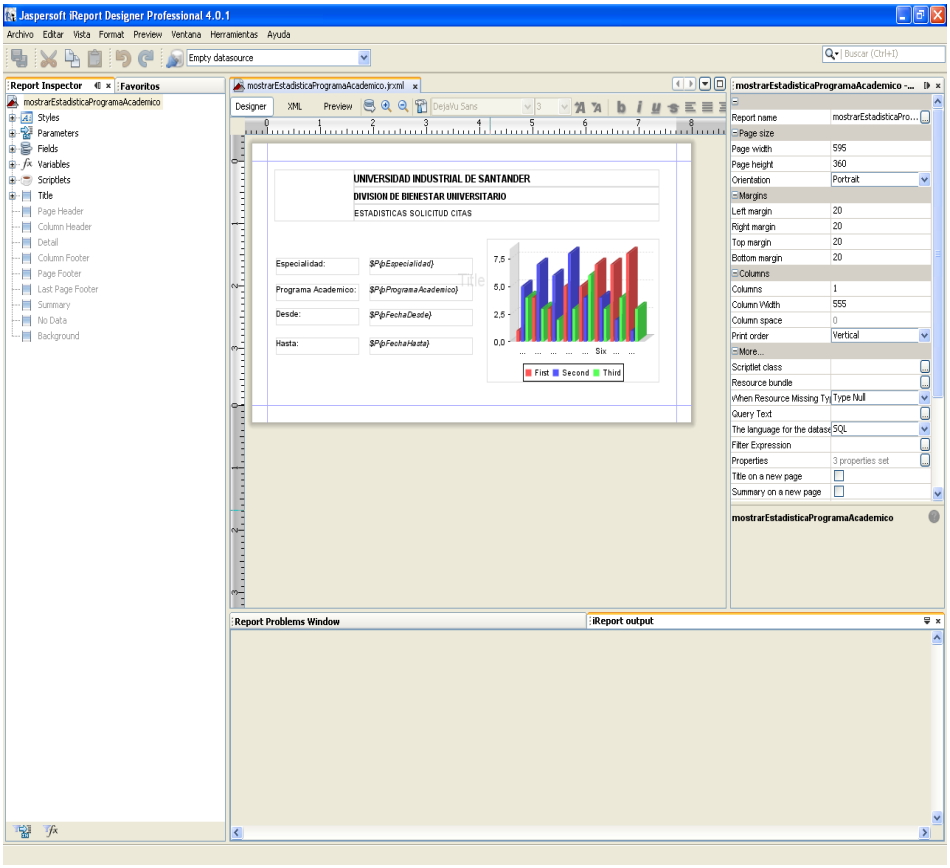


Figura 26. Creación de Plantillas en Jaspersoft iReport Designer Professional

Se debe agregar en el EJB las siguientes líneas de código:

```
public void asignarEstadisticaPrincipal(int aFila) {
    try {
      for (AgendaAsistencial a : this.agendasAsistenciales) {
```

```
        if
(a.getId().getEspecialidadProfesional().getId().getEspecialidad().getCodigo(
).equals(this.especialidad.getCodigo())) {

this.setEspecialidad(a.getId().getEspecialidadProfesional().getId().getEspe
cialidad());
          break;
        }
      }

      Map<String, Object> parametros = new HashMap<String, Object>();

      parametros.put("pRutaLogo", this.getPath("/imagenes/logoUIS.jpg"));
      parametros.put("pEspecialidad", this.especialidad.getNombre().trim());
      parametros.put("pProgramaAcademico",
this.valoresEstadisticasPrincipales.get(aFila).getProgramaAcademico().get
Nombre().trim());
      parametros.put("pFechaDesde", this.fechaDesde);
      parametros.put("pFechaHasta", this.fechaHasta);
      parametros.put("pCitasAsistidas",
this.valoresEstadisticasPrincipales.get(aFila).getCitasAsistidas());
      parametros.put("pCitasCanceladas",
this.valoresEstadisticasPrincipales.get(aFila).getCitasCanceladas());
      parametros.put("pCitasInasistidas",
this.valoresEstadisticasPrincipales.get(aFila).getCitasInasistidas());
      parametros.put("pCitasSolicitadas",
this.valoresEstadisticasPrincipales.get(aFila).getCitasSolicitadas());

      FacesContext context = FacesContext.getCurrentInstance();
      ExternalContext ext = context.getExternalContext();

      byte[] bytes =
JasperRunManager.runReportToPdf(this.getPath("/reportes/mostrarEstadis
ticaProgramaAcademico.jasper"), parametros, new JREmptyDataSource());
      HttpServletResponse response = (HttpServletResponse)
ext.getResponse();
      response.setContentType("application/pdf");
      response.setContentLength(bytes.length);
      ServletOutputStream servletOutputStream =
response.getOutputStream();

      servletOutputStream.write(bytes, 0, bytes.length);
      servletOutputStream.flush();
      servletOutputStream.close();
      FacesContext.getCurrentInstance().responseComplete();

    } catch (Exception e) {

      e.printStackTrace();
    } }
```

4.4.6.7 Procesos Automáticos. A parte de los módulos que se encuentran en la interfaz de usuario del sistema, su estructura y funcionalidad se complementa con los siguientes procesos automáticos, que se ejecutarán al final de cada día.

- **Recordatorios.** Al activarse el proceso automático se enviará un correo electrónico a los contactos que cumplan con los parámetros establecidos, éstos correos son útiles para recordar a los estudiantes que tienen una cita médica, o a alertar a los profesionales de la creación o ampliación de su agenda.

Para poder enviar correos desde el sistema es necesario modificar la siguiente línea de código del archivo components.xml ubicado en la carpeta WEB-INF del proyecto:

```
<mail:mail-session host="smtp.gmail.com" port="587"
username=xxxxx@gmail.com password="xxxxx" tls="true" />
```

Además en el EJB se debe agregar la siguientes líneas de código para el envió:

```
@In(create = true)
    private Renderer renderer;

this.renderer.render("/email/mailRecordatorioEstudiante.xhtml");
```

El siguiente es el diseño del correo electrónico “mailRecordatorioEstudiante.xhtml”

```
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:s="http://jboss.com/products/seam/taglib"
    xmlns:m="http://jboss.com/products/seam/mail"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
```

```
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:rich="http://richfaces.org/rich"
      xmlns:a="http://richfaces.org/a4j"
      xmlns:c="http://java.sun.com/jstl/core">

      <m:message>
          <m:from name="Bienestar Universitario"
              address="xxxxx@gmail.com" />
          <m:to
name="#{administrarSancionEJB.nombreEstudianteRecordatorio}">#{admi
nistrarSancionEJB.correoRecordatorio}</m:to>
          <m:subject>Sistema de Recordatorios Citas Medico
Asistenciales</m:subject>

          <m:body>


              <p>Cordial Saludo.<br></br>
              </p>


              <br></br>

              <p>Para BIENESTAR UNIVERSITARIO es importante que los
estudiantes
              asistan a los controles programados, por ese motivo le
recordamos que
              Usted tiene la siguiente cita de control programada:</p>

              <p>Fecha: #{administrarSancionEJB.fechaRecordatorio}</p>

              <p>Especialidad:
              #{administrarSancionEJB.especialidadRecordatorio}</p>

              <p>Profesional:
#{administrarSancionEJB.profesionalRecordatorio}</p>

              <p>Descripcion:
#{administrarSancionEJB.descripcionRecordatorio}</p>

              <p>Recuerde que puede solicitar esta cita vía WEB, llamando a
la línea xxxxxxx ext xxxx  o acercándose a
              la Ventanilla ubicada en el edificio de BIENESTAR
UNIVERSITARIO
              primer piso.</p>



              <br></br>
```

```
            <p>Cordialmente,</p>

            <br></br>

            <p>UNIVERSIDAD INDUSTRIAL DE
SANTANDER<br></br></p>
            <p>DIVISION BIENESTAR UNIVERSITARIO</p>


        </m:body>
    </m:message>
</ui:composition>
```

- **Sanciones.** Cada día se hará un barrido al sistema para asignar sanciones a los estudiantes que hayan incumplido con algunas de las normas establecidas en el proceso de solicitud y acceso a los servicios integrales de salud ofrecidos por la división de Bienestar Universitario.

CONCLUSIONES

- El nuevo sistema de información logra una reducción en los de los tiempos para la solicitud de citas médico asistenciales, y de promoción y prevención (PYP), ya que los estudiantes se ahorran las largas filas y conocen fácilmente los horarios disponibles.

- El sistema para la solicitud y asignación de citas ayuda a Bienestar Universitario a cumplir con los objetivos institucionales de mejorar la prestación y cobertura de sus servicios a la comunidad estudiantil.

- Gracias a la utilización de metodología de desarrollo basada en UML y prototipado evolutivo, se garantizó que el sistema cumpliera plenamente con las expectativas de las entidades interesadas, al incluirlas activamente durante todo el proceso.

- Con el módulo de Administración de Agendas se consigue la creación y modificación de las agendas de los profesionales de una forma dinámica y ordenada, logrando así la reducción de posibles errores.

- El sistema permite procesos de automatización, que reducen tiempo y esfuerzo a los funcionarios de Bienestar Universitario los cuales tenían que realizar estas tareas de forma manual.

- Gracias a los estándares establecidos por la División de Servicios de Información DSI, se logró desarrollar un software ordenado, eficiente y de mayor calidad.

RECOMENDACIONES

- Para el correcto funcionamiento del módulo de recordatorios es necesario mantener los correos electrónicos actualizados, tanto de los profesionales como de los estudiantes activos.

- Divulgar la información acerca del sistema a la comunidad en general para que esta pueda acceder a los servicios ofrecidos.

- Realizar la debida capacitación a los usuarios finales del sistema, sobre la funcionalidad de éste.

- Seguir permitiendo que la comunidad estudiantil participe en el desarrollo de este tipo de proyectos, puesto que ayuda a complementar los conocimientos adquiridos durante el ciclo de formación, y además ayuda a mejorar la infraestructura informática de la universidad.

BIBLIOGRAFÍA

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

[2] ROZANSKI, Uwe. Enterprise JavaBeans 3.0 con Eclipse Y Jboss. Alfaomega-Marcombo, 2009

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

[4] DAUM, Berthold. Profesional Eclipse 3 Para Desarrolladores Java. Anaya Multimedia-Anaya Interactiva.

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

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

[7] Estándares de desarrollo. División de Servicios de Información. Bucaramanga, UIS.

Enlaces sitios WEB

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

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

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

- http://docs.oracle.com/javaee/5/api/