ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE UN PORTAL PARA LA ADMINISTRACIÓN DE LA RAMA ESTUDIANTIL IEEE – ESPOL UTILIZANDO LA ARQUITECTURA J2EE"

TESIS DE GRADO

Previo a la obtención del Título de:

INGENIERO EN COMPUTACIÓN
ESPECIALIZACIÓN EN SISTEMAS DE INFORMACIÓN

Presentada por:

Luis Enrique Loor Maspons

GUAYAQUIL – ECUADOR

2008

AGRADECIMIENTO

Agradezco a Dios por toda su ayuda y aliento para terminar este tema de tesis a pesar de todas las adversidades, y todas las personas que de manera directa o indirecta ayudaron a la realización de este tema de tesis; de manera especial quiero dar las gracias a:

A la Ing. Carmen Vaca, Directora de Tesis por toda su ayuda, paciencia y colaboración en la realización de este proyecto.

A mi madre que supo guiarme por el buen camino durante toda mi vida.

A mis compañeros y amigos por todo su apoyo.

DEDICATORIA

A Dios que siempre ha sido ese pilar fundamental en mi vida.

A mi madre que siempre veló por mi bienestar y supo darme valores para ser una persona de bien a la sociedad.

A mis amigos que me apoyaron al desarrollo de esta tesis.

TRIBUNAL DE GRADUACIÓN

Ing. Holger Cevallos
SUBDECANO DE LA FACULTAD

Ing. Carmen Vaca
DIRECTORA DE TESIS

Ing. Carlos Monsalve
MIEMBRO DEL TRIBUNAL

Ing. Verónica Macías
MIEMBRO DEL TRIBUNAL

Escaneado con CamScanner

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL).

Luis Enrique Loor Maspons

Escaneado con CamScanner

RESUMEN

La Rama Estudiantil IEEE – ESPOL ofrece en la actualidad a todos sus miembros un sinnúmero de beneficios como las membresías a la IEEE, revistas de información tecnología de actualidad, eventos en varias áreas de especialización que permiten a sus participantes estar al corriente de los últimos acontecimientos tecnológicos y otros beneficios que ofrece el ser miembro de la IEEE.

La gestión de todas las actividades que realiza la rama en pro de sus miembros demanda una gran inversión de tiempo por parte de sus voluntarios, lo cual en muchas ocasiones dificulta ofrecer el mejor servicio posible a los miembros y público en general. Esta tesis propone un sistema web como una herramienta que permita administrar más eficientemente (desde el punto de vista del tiempo invertido) los procesos más críticos que tiene la rama, entre los cuales se encuentran la gestión de los eventos, registro de nuevos miembros, manejo económico, entre otros.

En el Capítulo 1 se exponen la introducción general, objetivos y justificaciones del presente proyecto de tesis.

En el Capítulo 2 se revisan los fundamentos teóricos de este proyecto de tesis, como la descripción y justificación del uso de la arquitectura J2EE, y la utilización de patrones de diseño implementados en el desarrollo del portal.

En el capítulo 3 se describe el análisis de los requerimientos funcionales y no funcionales de la aplicación; así como el listado de casos de uso y la definición de los roles del sistema.

En el capítulo 4 se detalla el diseño del sistema, es decir la arquitectura basada en patrones y el diseño de interfaces; así como el modelo lógico de la base de datos.

En el capítulo 5 se revisa la implementación del sistema, el uso de *Ajax* en la implementación de módulos, hardware, software, pruebas y seguridades de la aplicación.

ÍNDICE GENERAL

RESUMEN ..... **VI**

ÍNDICE GENERAL ..... **VIII**

LISTA DE ABREVIATURAS ..... **XII**

ÍNDICE DE FIGURAS ..... **XIII**

ÍNTRODUCCIÓN **1**

CAPITULO 1 ..... **3**

1. PLANTEAMIENTO DEL PROBLEMA **3**

1.1. Antecedentes 3

1.2. Objetivos 5

1.3. Justificación 6

CAPITULO 2 ..... **9**

2. FUNDAMENTOS TEORICOS **9**

2.1. Arquitectura y Patrones J2EE **9**

2.1.1. Descripción de la Arquitectura J2EE 9

2.1.2. Justificación del uso de la Arquitectura J2EE 12

2.1.3. Patrones de diseño utilizados en el desarrollo del portal 14

2.2. AJAX **15**

2.2.1. Descripción de la tecnología AJAX 15

2.2.2. Justificación del uso de AJAX 17

2.3. Ventajas de estándares utilizados **19**

2.3.1. CSS 20

2.3.2. XHTML 22

CAPITULO 3 24

3. ANÁLISIS DEL SISTEMA 24

3.1. Requerimientos funcionales 24

3.2. Requerimientos no funcionales 27

3.3. Diagrama de casos de uso 28

3.4. Definición de roles del sistema 33

CAPITULO 4 35

4. DISEÑO DEL SISTEMA 35

4.1. Modelos de diseño 35

4.1.1. Arquitectura basada en patrones de diseño 35

4.1.1.2. Data Value Object 37

4.1.1.3. Data Access Object 40

4.1.1.4. Custom Value Object 44

4.1.1.5. Composite View 49

4.1.1.6. Model View Controller 50

4.1.2. Diseño de Interfaces 54

4.2. Modelo lógico de la base de datos 62

CAPITULO 5 73

5. IMPLEMENTACIÓN Y PRUEBAS 73

5.1. Implementación 73

5.1.1. Menús dinámicos 73

5.1.2. Módulo de roles 74
5.1.3. Módulo de Anuncios 76
5.1.4. Módulo de Correspondencia 79
5.1.5. Módulo de Inventario 83
5.1.6. Módulo de Membresías 87
5.1.7. Módulo de Eventos 88
5.1.8. Módulo de Capítulos 93
5.1.9. Módulo de Sociedades 93
5.1.10. Módulo Financiero 94
5.1.11. Módulo de Usuarios 97
5.2. Uso de AJAX en las interfaces del sistema 99
5.2.1. Tablas Dinámicas 99
5.2.2. Búsquedas Dinámicas 103
5.2.3. Formularios Dinámicos 107
5.2.4. Dobles Combos 110
5.3. Plataforma 114
5.3.1. Hardware 114
5.3.2. Software 116
5.4. Seguridad 123
5.4.1. Autorización 123
5.4.2. Roles 126
5.4.3. Filtros 129
5.5. Configuraciones 133

5.6. Pruebas 134

5.6.1. Plan de Pruebas..... 134

5.6.2. Resultado de las Pruebas 139

CONCLUSIONES Y RECOMENDACIONES

ANEXOS

BIBLIOGRAFÍA

LISTA DE ABREVIATURAS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascade StyleSheet
DAO	Data Access Object
DOM	Document Object Model
DTD	Document Type Definition
DVO	Data Value Object
HTTP	HyperText Transfer Protocol
JSP	Java Server Page
J2EE	Java to Enterprise Edition
MVC	Model View Controller
RDBMS	Relational Database Management System
W3C	World Wide Web Consortium
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

ÍNDICE DE FIGURAS

Figura 1.1 Logo de RIEEEWEB 5

Figura 3.1 Diagrama de Casos de uso de usuario Administrador General ... 29

Figura 3.2 Diagrama de Casos de usuario Administrador De Membresías y Correspondencia..... 30

Figura 3.3 Diagrama de Casos de uso de usuario Miembro IEEE..... 30

Figura 3.4 Diagrama de Casos de uso de usuario Normal 31

Figura 3.5 Diagrama de Casos de uso de usuario Administrador Financiero…..... 31

Figura 4.1 UML de la tabla persona en la base de datos..... 39

Figura 4.2 UML de la clase PersonaDVO en la aplicación. 40

Figura 4.3 Diagrama de implementación del patrón DAO..... 44

Figura 4.4 UML de las tablas correspondencia, correspondencia – miembro y sociedadesieee 47

Figura 4.5 UML de la clase Correspondencia CVO 47

Figura 4.6 Modelo Vista Controlador 53

Figura 4.7 Diseño modular de páginas a través de sub-vistas 59

Figura 4.8 Diseño de la interfaz del sistema RIEEEWEB 61

Figura 4.9 Diagrama de datos que representa el módulo de Anuncios 65

Figura 4.10 Diagrama de datos que representa el módulo de Sociedades y Capítulos..... 65

Figura 4.11 Diagrama de datos que representa el módulo de Inventario 66

Figura 4.12 Diagrama de datos que representa el módulo de Correspondencia..... 67

Figura 4.13 Diagrama de datos que representa el módulo de Eventos 68

Figura 4.14 Diagrama de datos que representa el módulo Financiero 69

Figura 4.15 Diagrama de datos que representa el módulo de Membresía ... 70

Figura 4.16 Diagrama de datos que representa el módulo de Roles..... 71

Figura 4.17 Diagrama de datos que representa el módulo de Usuarios..... 72

Figura 5.1 Menú Dinámico del sistema..... 73

Figura 5.2 Creación de un nuevo rol en el sistema..... 76

Figura 5.3 Publicación de Anuncios..... 78

Figura 5.4 Visualización completa de un anuncio..... 78

Figura 5.5 Visualización de Correspondencia Personal..... 82

Figura 5.6 Entrega de correspondencia..... 83

Figura 5.7 Búsqueda de ítems para registro de movimientos en inventario . 84

Figura 5.8 Reporte de Inventario 85

Figura 5.9 Consulta de Inventario en Jasper Reports..... 86

Figura 5.10 Ingreso de las conferencias con AJAX 91

Figura 5.11 Reporte de Ingreso por eventos 92

Figura 5.12 Formulario de registro de transacción..... 95

Figura 5.13 Ejemplo de un reporte de utilidad 97

Figura 5.14 Creación de tablas dinámicas..... 101

Figura 5.15 XML que contiene los datos de conferencias ingresadas..... 102

Figura 5.16 Código JavaScript que utiliza DOM para modificar una página y añadir una tabla dinámica 103

Figura 5.17 Ejemplo de búsquedas con AJAX 104

Figura 5.18 XML que contiene los datos de la búsqueda de un usuario..... 105

Figura 5.19 Ejemplo de Auto completar en la aplicación 106

Figura 5.20 XML que contiene los datos de un conferencista 107

Figura 5.21 Creación de Formularios dinámicos..... 109

Figura 5.22 XML que contiene los datos de un participante 110

Figura 5.23 XML con los datos de sub-áreas de un evento 112

Figura 5.24 Código JavaScript que manipula un comboBox mediante los métodos del DOM 113

Figura 5.25 Dobles Combos con Ajax..... 114

Figura 5.26 Servidor Web Apache Tomcat 117

Figura 5.27 EMS como interfaz gráfica para MySQL 119

Figura 5.28 Entorno de desarrollo Eclipse con plugin MyEclipse..... 121

Figura 5.29 Entorno gráfico para la creación de los reportes 123

Figura 5.30 Página de ingreso del sistema 125

Figura 5.31 Mensaje de Recurso no autorizado 126

Figura 5.32 Configuración de un filtro en el archivo web.xml..... 130

Figura 5.33 Funcionamiento un filtro..... 133

Figura 5.34 Impresión de las credenciales antes del cambio 143

Figura 5.35 Impresión de las credenciales después del cambio..... 144

Figura 5.36 Esquema anterior de reportes de los miembros de la rama. .. 146

Figura 5.37 Esquema actual de reportes de los miembros de la rama 147

Figura 5.38 Entrega de correspondencia antes del cambio 149

Figura 5.39 Entrega de correspondencia después del cambio 150

Figura B.1 Imagen que muestra la validación de una página XHTML en modo estricto 179

Figura C.1 Referencia a JSTL dentro de una página JSP 180

Figura C.2 Invocación de los objetos a través de JSTL 181

Figura D.1 Variable que recoge el parámetro “accion” 182

Figura D.2 Implementación del MVC en un servlet a través de la sentencia condicional if. 182

Figura E.1 Configuración de un servlet dentro del archivo web.xml 183

Figura E.2 Configuración de filtro dentro del archivo web.xml. 183

Figura E.3 Configuración de páginas de error personalizadas dentro del archivo web.xml 184

Figura E.4 Configuración de tiempo de expiración de sesión dentro del archivo web.xml 184

INTRODUCCÍON

Desde el año 2002 la ESPOL y la Facultad de Ingeniería en Eléctrica y computación cuentan con la colaboración de un grupo de voluntariado cuya meta es tratar de difundir entre los estudiantes de la universidad la integración de profesionales y estudiantes de ingeniería con el fin de promover la investigación tecnológica, a través de la difusión de eventos que aportan temas de actualidad.

Este grupo de voluntariado llamado Rama estudiantil IEEE - ESPOL, intenta difundir a las personas los beneficios de ser miembro de la IEEE que es un organismo internacional para ingenieros electrónicos y de computación. La IEEE consta de varias ramas estudiantiles alrededor del mundo a través de las cuales difunde su misión.

La gestión que deben realizar las ramas se basa en voluntariado lo que provoca que muchas veces ésta se vuelva compleja por la falta del número de voluntarios necesarios.

La meta principal de esta tesis es la de ofrecer una solución tecnológica que ayude a la gestión más eficiente de los procesos principales de la Rama

Estudiantil IEEE- ESPOL, lo que permitirá ofrecer mejor atención a sus miembros y voluntarios.

CAPITULO 1

1. PLANTEAMIENTO DEL PROBLEMA

1.1. Antecedentes

El IEEE, (por sus siglas en inglés Institute of Electrical and Electronics Engineers) es el Instituto de Ingenieros Eléctricos y Electrónicos más grande del mundo, y tiene su sede en los Estados Unidos. Está formado por más de 365,000 miembros, incluyendo 68,000 estudiantes, en 150 países. Posee 311 secciones en 10 regiones geográficas alrededor del mundo. Tiene creados 1,450 Capítulos Técnicos, más de 1,300 Ramas Estudiantiles en 80 países, 39 Sociedades Técnicas y 128 Transactions, Journals y revistas. Además 300 Conferencias en todo el mundo cada año y 900 estándares IEEE activos y más de 400 en desarrollo [1].

El IEEE brinda la oportunidad a Universidades e Institutos Académicos de nivel superior a ser parte de toda la organización a través de las Ramas Estudiantiles. Una Rama Estudiantil es un grupo de estudiantes miembros y voluntarios del Instituto, que representan a determinada universidad. Dicho grupo tiene la oportunidad, a través de

los beneficios que brinda el IEEE, de fomentar la investigación y el liderazgo en los estudiantes de la Universidad.

Actualmente nuestra universidad cuenta con una Rama Estudiantil conformada por un comité ejecutivo para la coordinación general de las actividades de la rama y un comité de membresía que tiene como misión: administrar la renovación y registros de nuevos estudiantes y profesionales al IEEE, administrar la correspondencia de cada miembro, difundir los beneficios del IEEE y sus Sociedades Técnicas y resolver cualquier inquietud que tenga cualquier miembro.

Hoy en día la Rama Estudiantil IEEE – ESPOL no cuenta con un sistema que le permita automatizar los procesos de administración de la correspondencia, membresías, inventario, recursos económicos y los eventos que realiza durante el año. Esto suele constituir un problema para los voluntarios pues representa una mayor inversión de tiempo en la organización de eventos y administración de documentación relacionada a las actividades de la rama.

Este proyecto de tesis propone la implementación de un sistema Web que daría solución a varios de los problemas en los procesos que lleva a cabo a la Rama Estudiantil IEEE – ESPOL y permitiría ofrecer un

mejor servicio a sus miembros y voluntarios. Con la implementación de este sistema, la Rama IEEE - ESPOL sería la primera en Latinoamérica en poseer un sistema para la automatización de sus procesos. Así nace RIEEEWEB (Rama IEEE Web) como una solución a las necesidades de la Rama Estudiantil.

La figura 1.1 muestra el logo del sistema RIEEEWEB.

Figura 1.1 Logo de RIEEEWEB

1.2. Objetivos

El objetivo general planteado en este proyecto consiste en implementar un portal para la administración de la Rama IEEE – ESPOL utilizando la arquitectura J2EE y aplicando patrones de diseño para obtener una aplicación con una arquitectura robusta y fácil de mantener.

Los objetivos específicos propuestos para el desarrollo de este proyecto se centran alrededor de la creación de una herramienta que permita:

- Automatizar el proceso de manejo de la correspondencia de sus miembros, es decir, revistas, sobres, cds y toda la información que llega del Instituto.
- Administrar la membresía de cada miembro, sus datos personales y su tipo de registro al IEEE.
- Administrar toda la planificación anual de la Rama, sus capítulos y grupos de afinidad. Esto es poder administrar de manera dinámica la creación de eventos.
- Llevar un control del inventario de todos los recursos que se adquieren.
- Llevar un control de ingresos y egresos y mostrar un estado de resultados para tener una idea global del estado financiero de la Rama.
- Administrar los anuncios y publicaciones.
- Administrar la suscripción de los miembros para acceder a la información de correspondencia.

1.3. Justificación

La Rama Estudiantil IEEE – ESPOL necesita automatizar sus procesos internos como el registro de nuevos miembros, envío de correspondencia, registro de ingreso y egresos, ingreso de los activos de la rama, entre otros.

Existen muchas razones por las cuales se ha vuelto imprescindible que la Rama cuente con una herramienta para la administración de sus actividades. Los procesos manuales pueden generar muchos errores, pérdida de información y administración poco eficiente de los recursos. De hecho en la actualidad, se pueden observar los siguientes inconvenientes:

- La realización del registro de nuevos miembros se realiza tomando los datos personales en hojas de papel lo cual significa gastos en papel para el proceso de registro. Además este proceso se realiza siempre que un estudiante va a registrarse o a renovar su membresía lo que significa que siempre hay que volver a tomar los mismos datos personales ya que no se encuentran registrados en ningún lado.
- La correspondencia que envía la IEEE a sus miembros debe ser recogida por los miembros en las oficinas de la rama. Sin embargo, mucha de esta correspondencia nunca es retirada ya que nunca se sabe a ciencia cierta cuando la correspondencia ha llegado por lo que los miembros no la retiran.
- Los recursos de la rama no están inventariados por lo que la rama puede ser fácilmente presa de robo de sus activos, especialmente de los útiles de oficina que utilizan

- La rama no cuenta con una manera eficiente de llevar un control e ingresos y egresos de toda la rama así como de cada uno de sus capítulos, por lo que es difícil conocer con cuántos recursos cuenta cada uno de ellos.
- La gestión de los eventos que promociona la rama es compleja debido a los problemas con el registro de los estudiantes, el cobro de los valores y el no tener un repositorio común para los datos de los participantes.

CAPITULO 2

2. FUNDAMENTOS TEORICOS

2.1. Arquitectura y Patrones J2EE

2.1.1. Descripción de la Arquitectura J2EE

J2EE (*Java to Enterprise Edition*) es una plataforma creada por la Sun Microsystems en 1997 que permite generar contenido Web de manera dinámica y basada en herramientas de código abierto [2].

La arquitectura de J2EE define un modelo de programación que se encamina a la creación de aplicaciones basadas en n-capas. Típicamente se identifican cinco capas diferentes para una aplicación Web:

- **Capa de presentación**: Representa el conjunto de componentes que contiene la información que se presentará en la interfaz de usuario. Cuando se utiliza la arquitectura J2EE esta capa está compuesta básicamente de páginas JSP, cuya finalidad consiste en

mostrar los datos que han sido generados por un controlador. Este controlador pertenece a la capa de lógica de negocio que se describe a continuación.

- **Capa de lógica de negocio**: Contiene todos los componentes en los que se implementará la lógica de la aplicación. Esta capa recibe los datos y requerimientos provenientes de la capa de presentación y, de acuerdo a las reglas del negocio, genera los datos, que serán enviados como resultado de la petición hecha por el cliente. En la arquitectura J2EE la capa lógica de negocio se implementa utilizando clases de Java llamadas servlets. Los servlets se describen más adelante en esta sección.
- **Capa de integración**: Aquí se encuentran componentes que nos permiten hacer más transparente el acceso a la capa de sistemas de información, es decir a los datos utilizados por nuestra aplicación. En esta capa es donde se implementará la lógica de objetos de acceso a datos utilizando el patrón de diseño DAO (Data Access Objects) explicado más adelante en este capítulo.
- **Capa de sistemas de información**: Esta capa engloba los sistemas de información utilizados por la empresa

como por ejemplo: bases de datos relacionales, bases de datos orientadas a objetos, repositorios de mails o de documentos, etc.

Para ejecutar una aplicación Web basada en la arquitectura J2EE se requiere que la misma resida en un servidor de Aplicaciones o un contenedor que le permitirá enlazarse al WWW (World Wide Web).

Los servidores de aplicaciones son muy utilizados para aplicaciones grandes que requieren estar distribuidas en varios servidores para poder atender la demanda de requerimientos que llegan a la aplicación.

Un servidor de aplicaciones típicamente está formado por un contenedor de EJB (Enterprise Java Beans), un contenedor Web y un sistema de mensajería. Los EJBs son objetos distribuidos que pueden ser accesados remotamente y que permiten implementar operaciones de lógica del negocio y de persistencia de datos. Entre los servidores de aplicación más conocidos tenemos el JBoss, Jerónimo, IBM WebSphere Application Server, Sun Java System Application Server entre otros.

Un contenedor Web es el que recibe las peticiones HTTP y puede ejecutar las clases de Java para gestionar y formar las repuestas a dichas peticiones. En un contenedor Web que soporta J2EE las aplicaciones pueden tomar la forma de servlets o JSP, ambas tienen como objetivo generar páginas Web en el servidor en base a la lógica del negocio. Entre los contenedores Web más conocidos en el mercado tenemos: Apache Tomcat, Jetty, etc. Una aplicación que no requiere utilizar plataformas distribuidas, no necesitará tampoco de un servidor de aplicaciones. En ese caso es suficiente utilizar un contenedor Web. Los contenedores habitualmente consumen menor cantidad de recursos en la máquina en donde se encuentran instalados.

2.1.2. Justificación del uso de la Arquitectura J2EE

J2EE plantea una arquitectura robusta para el desarrollo de aplicaciones Web con características que permiten obtener soluciones fáciles de mantener y de buen rendimiento. Entre las principales ventajas de J2EE tenemos:

- J2EE se basa en Java por lo que hace uso de herramientas de software libre lo cual permite crear aplicaciones Web con ningún o muy bajo costo.

- J2EE al ser basado en Java permite aprovechar las ventajas de los conceptos relacionados a orientación a objetos y de todas las librerías que dispone el kit de desarrollo de Java.
- J2EE es soportado por múltiples sistemas operativos ya que al utilizar el lenguaje de programación Java, las aplicaciones resultantes pueden ser ejecutadas en cualquier sistema operativo que incluya una Java Virtual Machine (JVM). Java Virtual Machine es un entorno de ejecución para clases de Java que interpreta los bytecodes generados por un compilador y los traduce a código nativo de la plataforma en la cual se ejecuta.
- J2EE permite crear aplicaciones de gran calidad y rendimiento al igual que su principal competidor: .NET.
- Gracias a lo extendido que está el uso de la plataforma J2EE, existe una amplia comunidad de desarrolladores que constantemente publican propuestas de arquitectura de software para construir soluciones más confiables. Un ejemplo de estas propuestas se reflejan en los patrones de diseño que se han diseñado para ser aplicados en aplicaciones Web basadas en J2EE.

2.1.3. Patrones de diseño utilizados en el desarrollo del portal

Como se menciona en [3], los desarrolladores a menudo confunden aprender tecnología con aprender a diseñar con tecnología. Los patrones de diseño atacan precisamente esta falencia: proponen mini-arquitecturas de software que pueden ser seguidas por el programador al momento de diseñar e implementar sus aplicaciones.

Existe una amplia gama de patrones de diseño propuestos para ser utilizados en el desarrollo de aplicaciones Web. El autor de este trabajo, después de hacer un análisis de todos ellos determinó el conjunto de patrones a ser utilizados para el desarrollo del portal de la rama. Para el análisis se consideró la facilidad de implementación de los patrones, la claridad al momento de implementar el código y cuáles eran los patrones que mejor se ajustaban a los requerimientos de la aplicación a desarrollar.

Considerando lo expuesto en el párrafo anterior, el conjunto de patrones de diseño seleccionados para el portal fueron los siguientes:

- Data Value Object (DVO)

- Custom Value Object (CVO)
- Data Access Object (DAO)
- Composite View
- Model View Controller (MVC)

En el capítulo 4 se presentan los detalles de implementación de los patrones utilizados, junto con una breve descripción de cada uno de ellos.

2.2. AJAX

2.2.1. Descripción de la tecnología AJAX

AJAX (***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*) es una técnica para la creación de aplicaciones Web interactivas. Esta técnica propone el envío de requerimientos HTTP desde el cliente hacia el servidor de forma asíncrona. El servidor responde a estos requerimientos (de forma asíncrona) y la respuesta recibida se procesa en la página web utilizando métodos y propiedades del estándar DOM [4].

El DOM es un estándar de la W3C que propone una serie de interfaces y tipos de datos para manipular los elementos de una página web. Estas interfaces pueden ser implementadas en

cualquier lenguaje de programación pero actualmente la implementación utilizada en AJAX es la de JavaScript.

Utilizando este método de comunicación asíncrona con el servidor, se pueden realizar cambios sobre una misma página sin necesidad de hacer un refrescamiento total de la página sino solamente de los elementos que deseamos cambiar. De esta manera se logra una aplicación con una mayor interactividad y mayor rendimiento.

El término AJAX es relativamente nuevo puesto que fue introducido en el año 2005. Sin embargo, las técnicas para realizar peticiones asíncronas para cargar contenidos en una página datan desde los tiempos del Iframe (introducido en Internet Explorer 3 en 1996). El Iframe tiene un atributo llamado src a través del cual se puede cargar cualquier URL. Utilizando JavaScript se pueden manipular los elementos de la página que contiene al IFrame y lograrse efectos parecidos a los de AJAX.
AJAX en sí no es una tecnología sino un término que engloba a varias tecnologías que se utilizan de manera conjunta.
Las tecnologías utilizadas en el modelo AJAX son las siguientes:

- XHTML (o HTML) y hojas de estilo en cascada (CSS): Estas sirven para presentar la información.
- Document Object Model (DOM): Mediante métodos implementados en un lenguaje de scripting como JavaScript (más conocido), los métodos definidos en el estándar DOM permiten manipular a través de programación todos los elementos de una página web.
- XML: Es el formato comúnmente usado para la transferencia de datos desde el servidor, aunque se puede usar cualquier formato de representación de información como HTML o texto plano. Para la transferencia de datos con el servidor, AJAX usa el objeto XMLHttpRequest, que es el que permite realizar el intercambio de datos de manera asíncrona con el servidor.

2.2.2. Justificación del uso de AJAX

Como fue descrito en el punto anterior AJAX permite el intercambio de datos de manera asíncrona con un servidor, lo cual nos permite crear aplicaciones Web interactivas para el usuario.

Entre las principales razones por las cuales las aplicaciones Web están haciendo cada vez más uso del modelo propuesto por AJAX tenemos:

- AJAX está basado en tecnologías conocidas como Javascript, HTML, CSS, XML y el objeto XMLHttpRequest que aunque no es parte del estándar si es soportado por la gran mayoría de los navegadores como Internet Explorer, Mozilla, Firefox, Opera, etc.
- Permite crear aplicaciones en las que se mejora la usabilidad y se evita que el usuario deba recargar la página por cada requerimiento sin considerar si el requerimiento requiere cambiar toda la página o sólo una parte de la misma. Con AJAX se realizan peticiones asíncronas que nos permiten hacer actualizaciones a nuestras páginas solamente donde es necesario sin necesidad de recargar toda la página.
- El uso de peticiones asíncronas hace que la aplicación sea más eficiente en cuanto a rendimiento puesto que se reduce el tiempo de respuesta de las peticiones y reduce el ancho de banda requerido puesto que se envían mucho menos datos que cuando se recarga toda la página.

- AJAX puede ser usado por cualquier aplicación Web independientemente de qué tipo de tecnología de servidor y lenguaje de programación se use.

2.3. Ventajas de estándares utilizados

En la actualidad existen estándares que son aplicables al desarrollo web y cuyo objetivo es que se construyan sitios web que cada vez sean accesibles a más personas y que se puedan visualizar en cualquier dispositivo con acceso a internet.

Estos estándares son publicados por la W3C (World Wide Web Consortium) en conjunto con otras organizaciones internacionales que buscan llevar al Web a su máximo potencial (W3C) [6].

Entre las ventajas que ofrece el usar los estándares web tenemos:

- Estos estándares son desarrollados por expertos.
- Son neutrales.
- Son creados en base a consenso.
- Son ampliamente difundidos por la industria del Web.
- Son de gran aceptación.
- Continua revisión para posteriores mejoras.

A continuación se describen los estándares utilizados: CSS y XHTML.

2.3.1. CSS

CSS (Cascading Style Sheets, u Hojas de Estilo en Cascada) es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación del contenido [7].

Entre las principales ventajas de usar CSS en una Aplicación Web tenemos:

- Separación del contenido y la presentación: Esto permite que la aplicación Web sea más flexible ya que si se quieren realizar cambios a la apariencia sólo se tiene que modificar la hoja de estilo sin tener que modificar el código HTML de las páginas.
- Unificación en el diseño de las páginas: Al usar hojas de estilos se obtiene homogeneidad en las páginas de la aplicación con sólo enlazar nuestras páginas con dichas hojas de estilo. Si no utilizáramos CSS la única manera de lograr esta homogeneidad consistiría en copiar el formato que se encuentra en una página a todas las demás. En una aplicación con decenas de páginas esto

llevaría a un trabajo extremadamente largo al momento de cambiar el diseño de las páginas. Al utilizar correctamente CSS, los cambios se pueden realizar en un solo archivo (la hoja de estilo) sin importar si la aplicación Web está formada por 10 o 1000 páginas.

- Optimización en los tiempos de carga de los archivos: Se reducen los tiempos de carga de nuestra aplicación. Si bien es cierto que para la primera vez que se cargue la aplicación, el navegador traerá los dos archivos tanto el de contenido como el de formato (HTML y el CSS), para posteriores visitas nuestra hoja de estilos se recuperará de la caché del browser y podrá ser utilizada en la página Web sin necesidad de ser descargada desde el servidor nuevamente.
- Permite que el código fuente de nuestra página sea menos extenso y contenga solamente información referente a los datos y no a la presentación de los mismos.
- Con el uso de CSS se pueden tener diferentes hojas de estilo según el tipo de dispositivo que se utilice como cliente de la aplicación o diferentes estilos según los gustos de los usuarios.

2.3.2. XHTML

XHTML (eXtensible Hypertext Markup Language) es una versión más estricta y limpia de HTML pensado para sustituir a HTML como lenguaje estándar para las páginas Web [8].

Entre las ventajas más importantes de usar XHTML en vez de HTML tradicional tenemos:

- XHTML es en sí mismo XML por lo que sigue las reglas del mismo que son: bien formado y válido. Que esté bien formado significa que debe contener un solo elemento raíz y que todas las etiquetas que se abren deben tener una etiqueta de cierre. Que sea válido en cambio significa que debe seguir las reglas especificadas en un DTD (Document Type Definition) o un schema que son los que definen las reglas de cómo debe estar estructurado el documento XML a la hora de ser validado.
- Las aplicaciones Web de la actualidad pueden ser accedidas desde cualquier dispositivo. XHTML surge como el lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el

que se accede a ella. Además los dispositivos pequeños tienen menor capacidad de procesamiento y por lo tanto no pueden emplear muchos recursos para poder afrontar la complejidad de la sintaxis del HTML.

Como hemos visto existe la W3C que conjuntamente con otras organizaciones han desarrollado los estándares web que buscan que el web evolucione para que sea universal.

En el siguiente capítulo se describe el análisis del sistema, profundizaremos en lo que concierne a requisitos del sistema, casos de uso, modelo de base de datos y demás modelos de diseño.

CAPITULO 3

3. ANÁLISIS DEL SISTEMA

3.1. Requerimientos funcionales

Los requerimientos funcionales definen cómo se debe comportar la aplicación internamente, es decir cómo se procesa y manipula los datos que ingresan al sistema. De manera general los casos de uso del sistema definen los requerimientos funcionales del mismo.

A continuación se exponen los requerimientos funcionales del sistema.

- Permitir la publicación de anuncios con una fecha de expiración de los mismos, que pueden ser vistos en la página principal del portal Web por todos los visitantes. Se permitirá también la modificación de los datos de dichos anuncios.
- Permitir ingresar toda la correspondencia que envía la IEEE a sus miembros para que ésta pueda ser vista en la sección de correspondencia por todos los que se encuentran registrados en el sistema y enviar un correo electrónico a cada uno de los miembros para informar el arribo de nueva correspondencia.

- Permitir registrar la entrega de la correspondencia a los miembros de manera que exista constancia que ésta ha sido retirada.
- Permitir el registro de miembros que se suscriban al IEEE y de las membresías de cada uno, así como la fecha en que cada una de las membresías expiran. Así mismo se debe asignar un tipo de usuario con el cual el nuevo miembro podrá ingresar al sistema dependiendo de si ejerce algún tipo de voluntariado dentro de la rama.
- Generar reportes de los miembros de la rama cuyas membresías se encuentran activas o inactivas, así como un reporte de los miembros que pertenecen a cada una de las sociedades IEEE.
- Permitir registro del inventario físico de la rama y de todos los movimientos que se registren sobre los mismos para tener un mejor control de los activos de la rama, así como la modificación de los datos de los ítems del inventario y de sus movimientos.
- Permitir generar reportes de todos los ítems que constan en el inventario de la rama, así como reportes de los movimientos del inventario permitiendo filtrar los reportes según el tipo de ítem de inventario y el tipo del movimiento.

- Permitir el registro de ingresos y egresos de la rama y de cada uno de los capítulos para llevar un control de los ingresos y gastos. Así mismo la generación de comprobantes de cada transacción que se realice.
- Permitir la generación de reportes de utilidades de la rama o de cada uno de los capítulos técnicos detallando los ingresos y egresos.
- Permitir la creación de eventos y sus conferencias, así también el ingreso de los conferencistas de las mismas. Así también se debe permitir la publicación de estos nuevos eventos en la página principal del portal Web para que los visitantes puedan verlos y tengan la opción de registrarse en los mismos.
- Permitir realizar reportes de los asistentes a un evento y un reporte de los ingresos que se generaron por el evento.
- Creación de nuevos roles en el sistema en base a las opciones que ya se encuentran disponibles.
- Permitir a los miembros registrados el ingreso al sistema generar un menú con las opciones que pertenecen al rol que tiene asignado.
- Conocer a ciencia cierta que miembros están aptos para las votaciones que realiza la rama para elegir sus dignidades.

3.2. Requerimientos no funcionales

Los requerimientos no funcionales describen criterios del sistema que sirven para evaluar su operación. Los requerimientos funcionales más comunes son disponibilidad, eficiencia, rendimiento, robustez, precio, etc.

A continuación se exponen los requisitos no funcionales:

- El sistema garantizará que los recursos del sistema sólo estén disponibles para usuarios con acceso permitido y que cada usuario sólo tiene acceso a la información que su rol en el sistema le permite acceder.
- El sistema será desarrollado de manera que pueda evolucionar e incorporar nuevos requerimientos y funcionalidades, tratando de afectar lo menos posible al código ya implementado.
- El sistema deberá tener una interfaz de usuario amigable de manera que facilite su uso de manera que la capacitación de los usuarios no supere el tiempo de 1 hora por cada uno de los módulos que componen la aplicación.
- El sistema deberá ser orientado al Web por lo que se usará estándares de diseño web.
- El sistema garantizará eficiencia en el registro de los miembros de la Rama.

- Garantizar un buen rendimiento de las consultas para que no se afecte el desempeño de la base de datos. En condiciones normales de operación del servidor, las consultas de datos que corresponden a información a mostrar en el portal no deben tomar más de 10 segundos; el tiempo de generación de reportes no debe ser mayor a 45 segundos cuando el volumen de datos sea grande.

3.3. Diagrama de casos de uso

Los casos de uso son un conjunto de acciones que describen las funcionalidades con las que debe cumplir un sistema proporcionando un listado de escenarios posibles entre los casos y usuarios.

Las siguientes figuras representan los diagramas de casos de uso del sistema. Los casos de uso están separados de acuerdo a los roles que tienen los usuarios en el sistema.

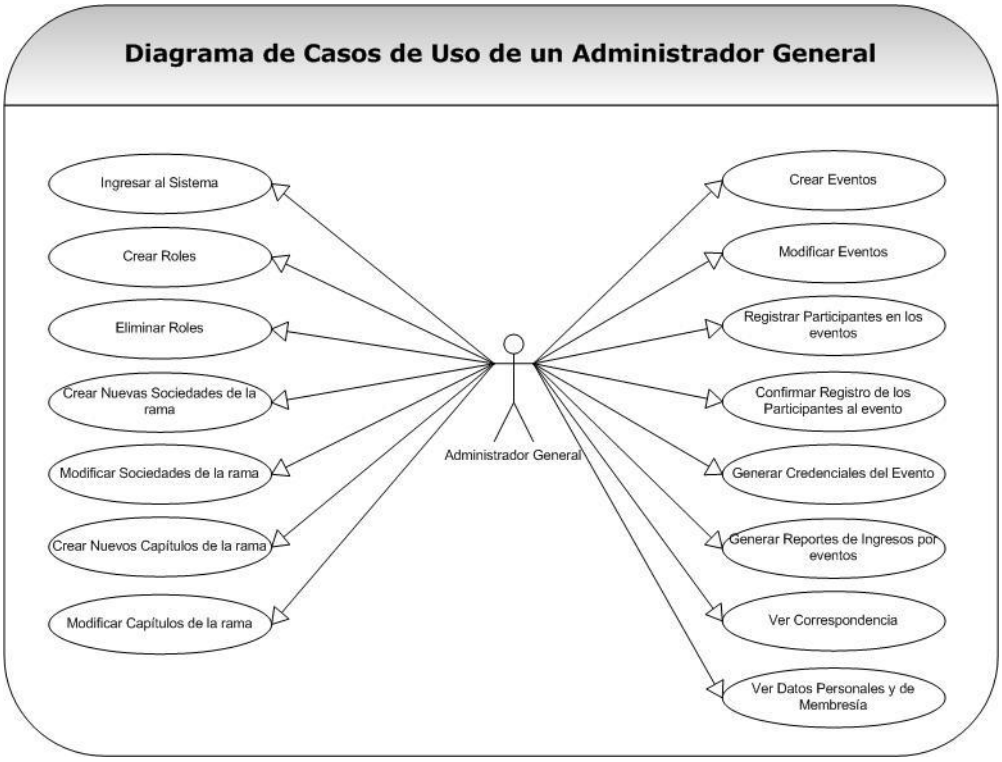


Figura 3.1 Diagrama de Casos de un usuario Administrador General.

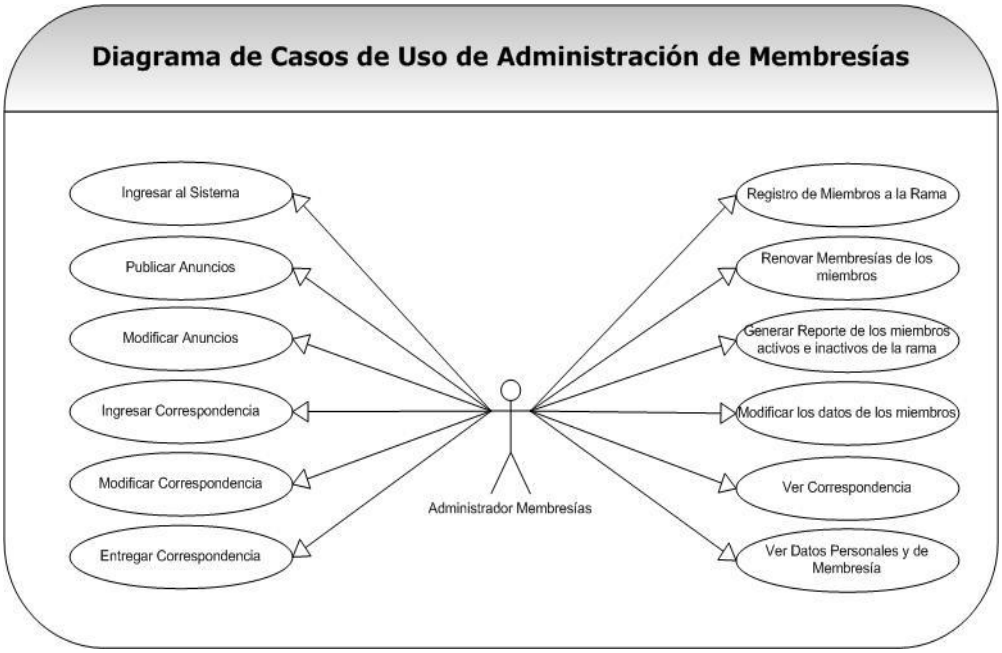


Figura 3.2 Diagrama de Casos de un usuario Administrador de Membresías y Correspondencia.

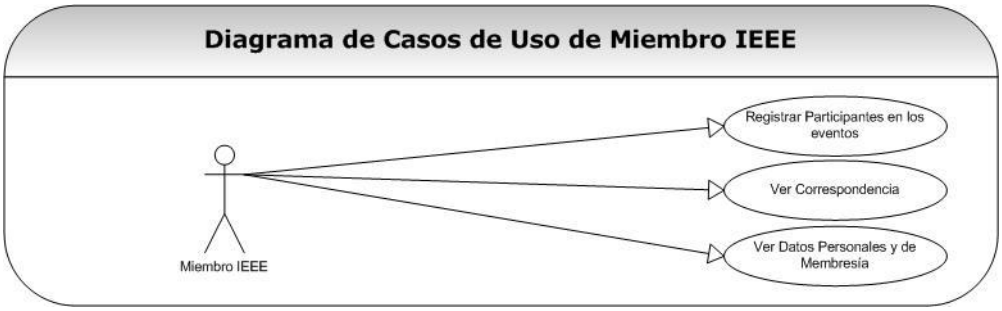


Figura 3.3 Diagrama de Casos de un usuario Miembro IEEE.

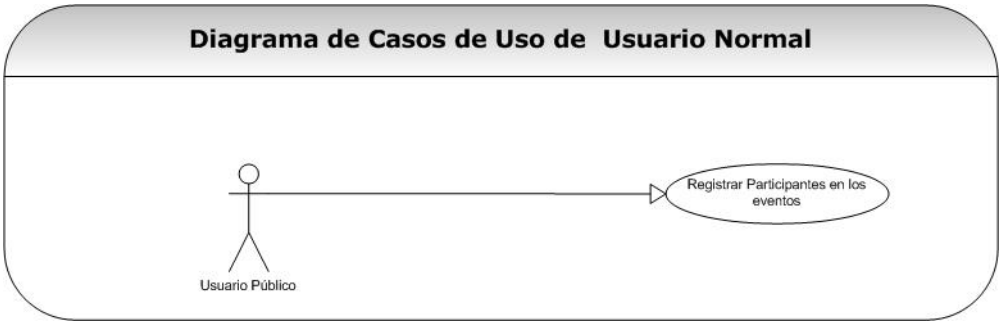


Figura 3.4 Diagrama de Casos de un usuario normal.

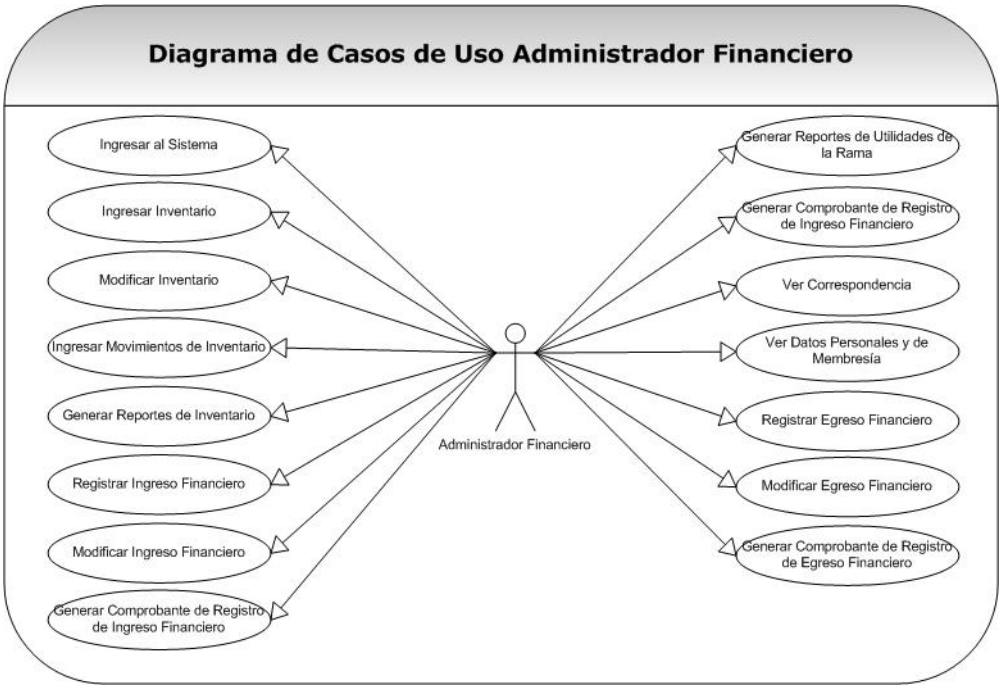


Figura 3.5 Diagrama de Casos de un usuario Administrador Financiero.

Listado de Casos de Uso

1. Ingresar al Sistema
2. Publicar Anuncios

3. Modificar Anuncios
4. Ingresar Correspondencia
5. Modificar Correspondencia
6. Entregar Correspondencia
7. Ver correspondencia Personal
8. Crear Roles
9. Eliminar Roles
10. Ingresar Inventario
11. Modificar Inventario
12. Ingresar Movimientos de Inventario
13. Generar Reporte de Ítems de Inventario
14. Registrar Ingreso Financiero
15. Modificar Ingreso Financiero
16. Generar Comprobante de Registro de Ingreso
17. Registrar Egreso Financiero
18. Modificar Egreso Financiero
19. Generar Comprobante de Registro de Egreso
20. Generar Reporte de Utilidades de la Rama
21. Generar Reporte de Utilidades por cada Capítulo.
22. Registrar nuevos Miembros a la Rama IEEE – ESPOL
23. Renovar las membresías de los miembros.
24. Generar Reporte de Miembros activos e inactivos.

25. Generar Reporte de todos los miembros activos por cada capítulo de la rama.
26. Modificar los datos de los miembros.
27. Crear Eventos
28. Modificar Eventos
29. Registrar participantes en los eventos creados.
30. Confirmar registro de los participantes.
31. Generar credenciales de los eventos.
32. Generar Reporte de ingresos por cada evento.
33. Crear nuevas sociedades para el registro de los miembros.
34. Modificar sociedades.
35. Crear nuevos capítulos de la Rama.
36. Modificar los capítulos de la Rama.

La descripción de los casos de uso se encuentra en el anexo A, ubicado al final de este documento.

3.4. Definición de roles del sistema

En la definición de roles en el sistema fue necesario realizar una comparación entre las funcionalidades ofrecidas por el sistema y la organización interna de los miembros voluntarios de la rama IEEE – ESPOL.

Internamente, la rama designa a sus miembros voluntarios diferentes tareas según sean las necesidades de la rama y la disponibilidad de tiempo de los voluntarios. Entre las diferentes actividades que cumplen los voluntarios están el registro de nuevos miembros, coordinación de eventos, entrega de correspondencia, etc. Al hacer un análisis de las tareas de los voluntarios y las tareas que debe cumplir el sistema se llegó a definir los siguientes roles:

- Administrador Principal o General
- Administrador Financiero
- Administrador de Membresía y Correspondencia.

Al definir varios roles que cumplen diferentes tareas, se pueden tener varios miembros voluntarios en un mismo rol con lo que existe disponibilidad de miembros que puedan cumplir las tareas asignadas en el sistema en caso de que otro voluntario esté ausente.

La definición más amplia de los roles y de las tareas asignadas a cada rol, se describen más adelante en el capítulo 5.

CAPITULO 4

4. DISEÑO DEL SISTEMA

4.1. Modelos de diseño

4.1.1. Arquitectura basada en patrones de diseño

Los patrones de diseño aparecen por los años 70 a través de Christopher Alexander quien escribe acerca de patrones aplicados a la arquitectura, más tarde este conocimiento es aplicado al campo del desarrollo de software orientado a objetos. En 1987 Ward Cunningham y Kent Beck mientras diseñaban interfaces de usuario para Smalltalk (Lenguaje de programación) decidieron utilizar las ideas de Alexander de tal manera que puedan ser utilizadas como guía para los programadores. El resultado de esto fue el libro “Using Pattern Languajes for Object – Oriented Programs”. A partir de esto surgieron varias publicaciones de libros que hablaban de patrones de diseño para diferentes lenguajes de programación [9].

Pero es 1998 cuando Mark Grand publica “Patterns in Java” que fue el primer libro de patrones de diseño para Java. En el 2001 el mismo autor publica “Java Enterprise Design Patterns” donde se añadieron 41 nuevos patrones que contribuyeron a construir aplicaciones más robustas para J2EE.

Sin duda los patrones de diseño son un gran aporte al desarrollo de aplicaciones y existen una variedad de patrones de diseño que no sólo se aplican a la arquitectura J2EE, sino que también se aplican para otras arquitecturas diseñadas usando.NET o PHP. Sin embargo, estas otras tienen una menor difusión.

Los patrones de diseño J2EE son soluciones bien documentadas de problemas comunes que surgen en el diseño de aplicaciones J2EE [10].

En todo patrón J2EE se define lo siguiente:

- Contexto: Circunstancias bajo las cuales el patrón existe.
- Problema: Describe el problema que trata se trata de resolver por medio del patrón.

- Motivación: Lista de las razones y motivaciones que afectan al problema y a la solución.
- Solución: Describe la solución brevemente y los elementos de la solución en detalle.
- Estrategias: Describen diferentes formas de cómo un patrón puede ser implementado.
- Consecuencias: Las ventajas y desventajas cuando se aplica el patrón.

Existen una gran variedad de patrones de diseño para J2EE, sin embargo a continuación se describen aquellos patrones que fueron utilizados durante la implementación.

4.1.1.2. Data Value Object

El patrón de diseño Data Value Object (DVO) propone definir clases de Java de tal manera que cada uno de los atributos corresponda a los campos que componen las tablas de la base de datos [11].

Para implementar este patrón de diseño hay que modelar los datos residentes en la base de datos en clases simples de Java que reflejen el modelo físico existente con métodos getters y setters para poder

acceder a los atributos de las mismas. En una clase de Java, los métodos getters se utilizan para recuperar información sobre los atributos y los métodos setters para modificar el valor de los mismos. Todos los cambios realizados a la base implican un cambio en los objetos implementados usando este patrón.

La utilización de este patrón de diseño facilita el mantenimiento de la aplicación ya que los cambios que resultan de modificar las clases que representan los *data value objects* son mínimos obteniendo una mayor eficiencia en la escritura del código. Además se promueve la reutilización de código y se evita tener que reescribir partes del programa cuando se necesite hacer las mismas operaciones en diferentes módulos de la aplicación.

Si no se utilizara este patrón, el desarrollador tendría que declarar el mismo número de variables según los campos en una tabla de la base de datos, esto lo debería hacer las veces que se quiera obtener los

datos de dicha tabla. Esto resulta en un código repetitivo que se encuentra disperso en varias partes de la implementación lo cual puede dificultar el mantenimiento de la aplicación si se requieren muchos cambios.

La figura 4.1 muestra el diagrama UML de la tabla persona en la base de datos y a continuación se muestra la figura 4.2 donde se visualiza el diagrama UML de la clase PersonaDVO en la aplicación. Al analizar ambas figuras se puede apreciar la relación entre los campos de la tabla y los atributos de la clase *como se había explicado en el párrafo anterior.*

Figura 4.1 UML de la tabla persona en la base de datos.

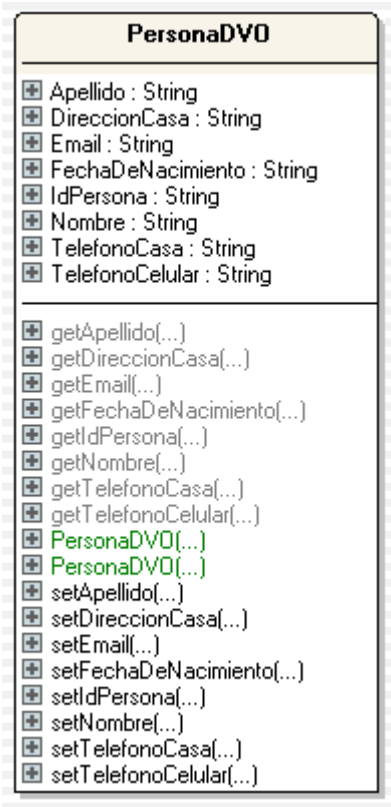


Figura 4.2 UML de la clase PersonaDVO en la aplicación.

4.1.1.3. Data Access Object

El patrón de diseño Data Access Object (DAO) propone separar y encapsular toda la lógica de acceso a los datos que maneja la aplicación, logrando de esta maneja desligar la lógica de negocio y presentación del mecanismo de manejo del repositorio de datos que se utilice para almacenar la información [12].

El mecanismo para el acceso al repositorio de datos puede ser manejado mediante JDBC o cualquier herramienta de persistencia de datos. En el mercado existen varias, entre las más conocidas están TopLink e Hibernate.

Una clase de Java que implementa el patrón de diseño DAO es la responsable del manejo de la conexión a la fuente de datos, de tal manera que éstos se puedan recuperar utilizando siempre la misma interface independientemente de cuál sea la fuente. Además de proveer el método para abrir la conexión a la base de datos, contendrá también métodos para acceder a los datos, ya sea para leerlos, actualizarlos o eliminarlos.

Las clases implementadas usando este patrón frecuentemente utilizarán clases DVO para leer o modificar datos.

En la aplicación web implementada para este trabajo de tesis se implementa un objeto DAO que encapsula

todos los métodos de acceso a datos a una base creada en MySql Server 5.0. Cada objeto DAO tiene todos los métodos para establecer una comunicación con la base de datos, ejecutan cierta acción y cierran la conexión con la base.

Las ventajas que ofrece el uso de este patrón son muchas. Cuando se utiliza el patrón de diseño DAO, la conexión a la fuente de datos se maneja de manera independiente. Dado que en la arquitectura utilizada se centraliza el manejo de la fuente de datos en las clases DAO, se podría decidir cambiar de plataforma para el almacenamiento de información sin afectar al resto de la aplicación.

El usar este patrón en la aplicación estableció un alto grado de similitud en todos los métodos para la lectura, actualización, inserción o eliminación de datos facilitando el trabajo del desarrollador que puede manejar un modelo muy parecido de manipulación de datos aunque las vistas sean completamente distintas.

Otra ventaja es que los Objetos DAO utilizan objetos DVO, lo cual nos permite definir métodos cuyos prototipos no requieren una lista indefinida de parámetros (uno por cada campo de la tabla de datos). En su defecto los prototipos de los métodos solamente requieren como parámetro un objeto DVO. Los métodos del objeto tendrán un prototipo que lucirá como sigue: `metodo(objDVO obj)` en lugar de `metodo(param1 p, param2 q, ……)`, que es lo que se tendría en una aplicación que no utiliza los patrones ya mencionados en esta sección.

Quizás la mayor fortaleza de este patrón, consiste en permitir migrar con facilidad de fuente de datos, en nuestro caso de motor de base de datos. Bastaría con cambiar el objeto que maneja la conexión con la base de datos y eso es suficiente para que toda la aplicación funcione de igual manera que con el motor de base de anterior. Por supuesto, para que este cambio no afecte la aplicación, es importante utilizar sentencias SQL que sean parte del estándar de tal manera que puedan ejecutarse en cualquier RDBMS.

La figura 4.3 que muestra un diagrama de la implementación de un patrón DAO. Como se observa en el diagrama la clase que implementa el patrón DAO es la que maneja la conexión con la base de datos y los objetos del negocio acceden a ella a través de esta clase.

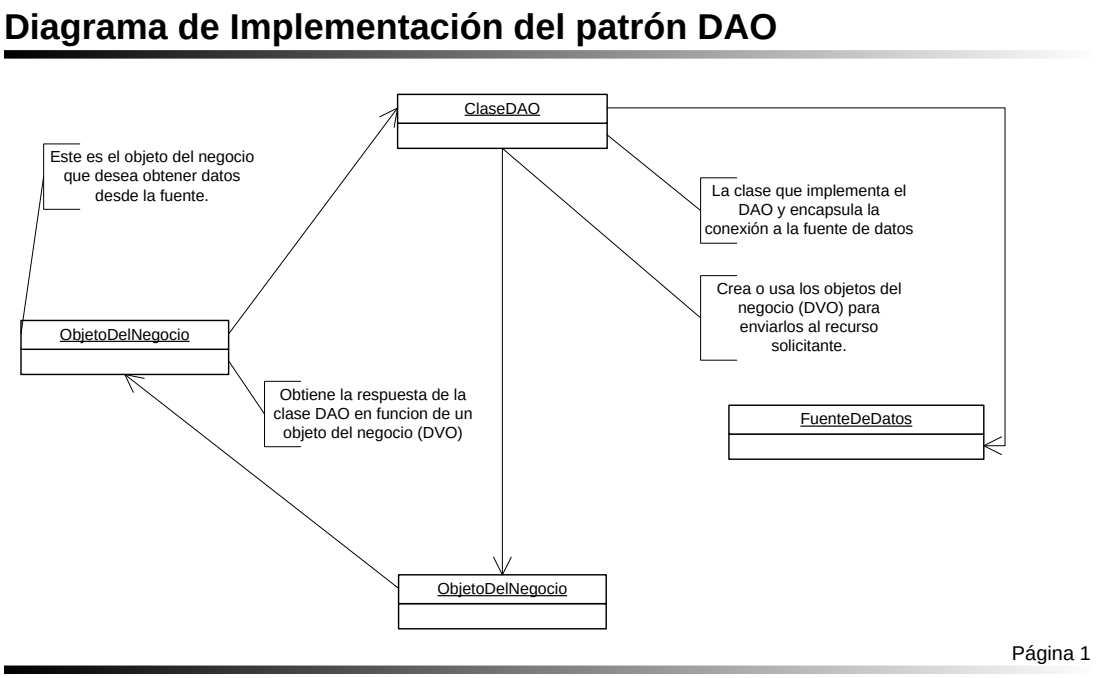


Figura 4.3 Diagrama de implementación del patrón DAO

4.1.1.4. Custom Value Object

El patrón CVO (Custom Value Object) es muy semejante al patrón DVO (Data Value Object) y su objetivo es encapsular los campos de las tablas de la

base de datos en clases de Java. La diferencia es que este patrón encapsula los datos no de una tabla sino datos de varias tablas. Esto se da porque no siempre necesitamos los datos de una sola tabla para formar un objeto sino que necesitamos datos de varias tablas [11].

La implementación de este patrón es muy similar a la del DVO, lo que hay que hacer de manera adicional es un análisis de cuáles atributos y cuáles tablas necesito para modelar un objeto CVO.

Un caso específico de este escenario en la aplicación son los datos de la correspondencia. Existen dos tablas que son *correspondenciaiiee* y *correspondencia-miembro*. La primera almacena los datos de la correspondencia ingresada y la segunda almacena la correspondencia para cada miembro, sin embargo la segunda tabla hace referencia a la primera a través de una clave foránea. Para mostrar la información de correspondencia de cada usuario se necesita hacer una combinación de atributos

individuales de las dos tablas; es ahí donde hemos usado el patrón CVO.

La ventaja que ofrece el uso de este patrón es que permite eliminar el tener que usar varios objetos de tipo DVO cuando se requieren atributos de varias tablas, ya que la mayoría de atributos de los DVO quedarían sin usar y esto produciría un uso ineficiente de los recursos de la aplicación.

La figura 4.4 muestra el diagrama UML de varias tablas de la base de datos y sus atributos. La figura 4.5 muestra el diagrama UML de la clase correspondenciaCVO. Como se puede apreciar, esta clase contiene como atributos algunos de los campos de las tablas mostradas en la figura 4.5.

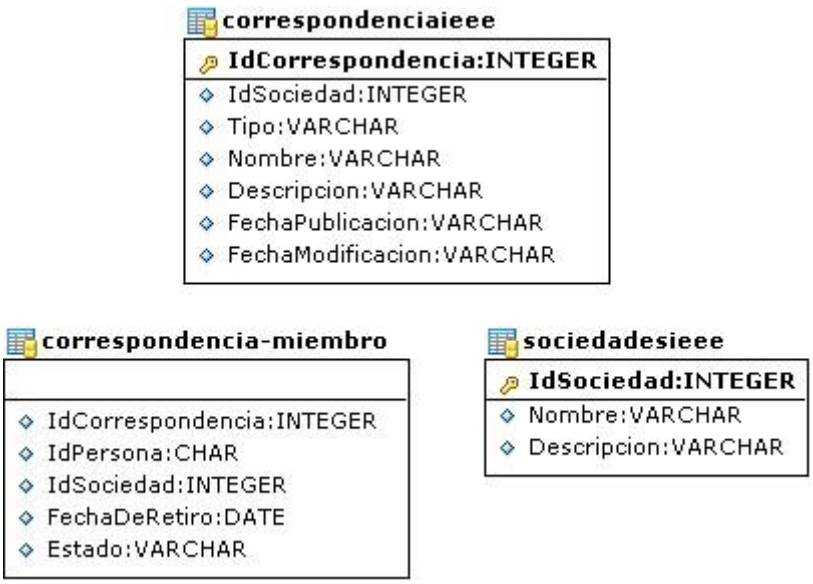


Figura 4.4 UML de las tablas correspondencia, correspondencia – miembro y sociedadesieee

Figura 4.5 UML de la clase Correspondencia CVO

En el caso mostrado el patrón CVO nos ayuda a crear una clase donde unimos los datos de las 3 tablas mostradas en la figura 4.5.

La clase CorrespondenciaCVO contiene como atributos de clase los siguientes:

De la tabla Correspondenciaieee

- descripcionCorrespondencia
- idCorrespondencia
- nombreCorrespodencia
- tipoCorrespondencia

De la tabla correspondencia-miembro

- fechaRetiro
- estadoCorrespondencia

De la tabla sociedadesieee

- nombreSociedad

Si no aplicáramos este patrón tendríamos que obtener los tres objetos desde la base y podríamos tener el mismo resultado. Sin embargo esto hubiese significado tres consultas contra la base de datos, lo

cual resultaría en un número mayor de operaciones a realizar en el servidor y un mayor uso del ancho de banda de la red, pues son más datos los que se transferirían.

4.1.1.5. Composite View

Composite View o Vistas Compuestas es un patrón que consiste en generar vistas totales a partir de sub-vistas atómicas. Cada vista más pequeña se puede incluir dinámicamente en una vista total [13].

En aplicaciones web actuales, el contenido puede ser generado dinámicamente desde diferentes fuentes de datos. Definir regiones cuyo contenido se presenta de manera dinámica, promueve el diseño modular de las páginas y la reutilización de código ya que las sub-vistas atómicas se pueden utilizar en diferentes secciones de la aplicación.

Una ventaja de usar este patrón es que se puede hacer un diseño preliminar de la aplicación sin necesidad de saber cuál será la información que se

va a colocar, así se puede hacer una distribución de las páginas.

4.1.1.6. Model View Controller

El patrón MVC (Modelo Vista Controlador) propone separar los datos que componen la aplicación, la interfaz del usuario y la lógica de control en tres componentes distintos: Modelo, Vista y Controlador. Estos componentes se definen a continuación:

Modelo: Es la representación de toda la información con la que funciona la aplicación.

Vista: Es la representación del modelo en forma visual para el usuario.

Controlador: Es el que responde a los eventos que envía el usuario para cambiar el modelo y probablemente la vista.

En un aplicación web J2EE que implementa este patrón, generalmente las **vistas** serán páginas HTML o páginas JSP, el **controlador** se implementará utilizando un servlet y el **modelo** se codificará utilizando clases de Java (en el caso de la aplicación

presentada en esta tesis serán clases que implementan el patrón DVO).

La vista es aquello que se muestra al usuario que puede ser código HTML viéndolo del lado del cliente, si lo vemos del lado del servidor es código JSP. Aquí en la vista es donde se visualizan los datos que el usuario ha solicitado al controlador.

El modelo son todas aquellas operaciones necesarias para preparar los datos para puedan ser enviados a la vista, esta solamente se encargará de hacerlos visibles.

La función del controlador consiste en recibir las acciones que el usuario ha seleccionado, y de acuerdo a esta selección invocar operaciones del modelo. Una vez que estas operaciones han sido completadas, el controlador se encargará de determinar la vista que debe ser enviada al cliente para que el usuario visualice el resultado del requerimiento enviado. En este proceso el

controlador creará los objetos necesarios para componer la vista que desplegará al usuario.

Este patrón es muy útil en la aplicación desarrollada debido a que toda la lógica de navegación e invocación de métodos del modelo se implementa en los servlets dejando las páginas JSP con ningún o escaso código.

Entre las ventajas que ofrece la implementación de MVC en J2EE tenemos que los servlets son clases pre – compiladas. Esto hace que los servlets se ejecuten más rápido que las páginas JSP que son compiladas en cada llamada al servidor. Si tenemos páginas JSP con código esto produce mayor retraso en las respuestas que da el servidor a las peticiones.

Los servlets al ser clases de Java proveen mayor facilidad para la corrección y detección de errores mientras que en las páginas JSP la detección de errores se vuelve una tarea muy complicada es un problema. A continuación se muestra gráficamente

las responsabilidades de cada uno de los objetos usados en este patrón.

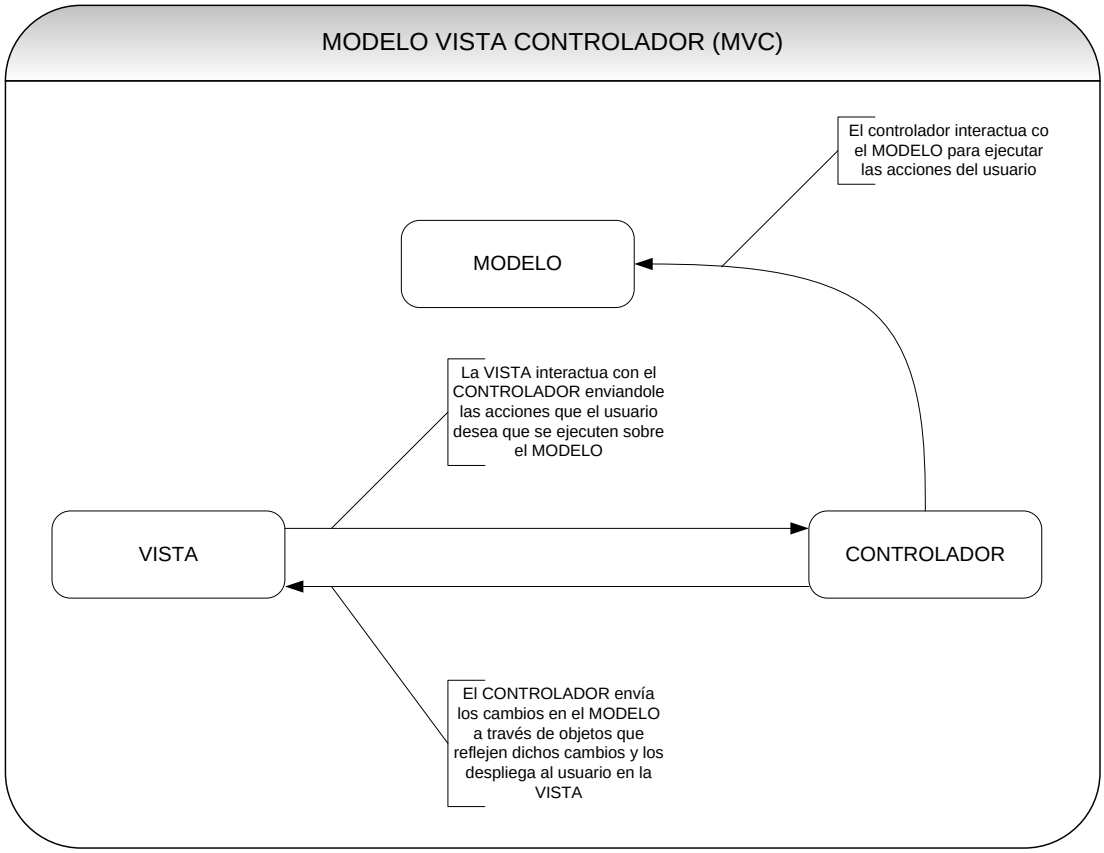


Figura 4.6 Modelo Vista Controlador

La figura 4.6 muestra un diagrama donde se muestran los componentes del modelo y la interacción entre ellos. Se puede observar como el controlador utiliza al modelo para obtener los datos necesarios para desplegar la vista al usuario.

4.1.2. Diseño de Interfaces

Para el diseño de las interfaces se tomaron en cuentas las consideraciones de la W3C XHTML 1.0 Strict para la construcción de páginas Web y se utilizaron los estándares sugeridos por dicha Institución para el desarrollo de interfaces web.

De acuerdo a la definición provista por la W3C, “XHTML es una familia de tipos de documentos y módulos basados en XML que reproducen y extienden de HTML 4”.

Las reglas que rigen el modo estricto de XHTML son las siguientes [14]:

- El elemento raíz del documento debe ser el elemento <html>
- El elemento raíz del documento debe contener una declaración xmlns (NameSpace).
- Debe existir la declaración del DOCTYPE previo a la declaración del elemento <html>.
- El documento XHTML debe estar bien formado. Esto quiere decir que todo elemento debe tener su correspondiente etiqueta de cierre.

- Todos los elementos y atributos deben estar escritos en letra minúscula.
- El valor de los atributos siempre debe ir entre comillas dobles.
- No puede ocurrir la minimización de atributos. Es decir cada atributo de un elemento debe contener su correspondiente valor.
- Los elementos vacíos deben estar cerrados.
- En XHTML los elementos script y style son declarados como elementos que requieren validación. Por lo tanto los caracteres ‘<’ o ‘>’ serán tomados en cuenta como inicio y fin de un elemento. Encerrar estos elementos dentro de una sección CDATA evitará que sean validados. Adicionalmente se puede usar scripts y hojas de estilo externas para evitar este problema.
- Los elementos deben contener un atributo llamado ID que sirve para su identificación y este atributo debe ser único por cada elemento.
- Existen prohibiciones con respecto a los elementos que deben considerarse a la hora de escribir un documento XHTML, estas son:
- El elemento <a> no puede contener otros elementos <a>

- El elemento <pre> no puede contener a los elementos <img>, <object>, <big>, <small>, <sub>.
- El elemento <label> no puede contener otros elementos <label>
- El elemento <form> no puede contener otros elementos <form>
- El elemento <button> no puede contener a los elementos <input>, <select>, <textarea>, <label>, <button>, <form>, <iframe>, <fieldset>

Se utilizaron archivos CSS para separar contenido de presentación. Así la aplicación será más fácil de mantener en cuanto a diseño se refiere.

En cuanto a la estructura del documento se revisó diferentes sitios web y ejemplos de cómo estructurar un documento HTML y la gran mayoría de los sitios visitados y los ejemplos se componían de varias vistas que completan una sola vista total que contiene a las demás.

Una aplicación Web típica puede estar compuesta de las siguientes vistas:

- Cabecera, generalmente contiene banners o logotipos.
- Barras laterales, son menús con acceso a las opciones de la aplicación.
- Cuerpo, es el centro de toda la vista, contiene la información principal que se debe presentar al usuario.
- Pie, donde se muestran información de copyright o de contacto.

El caso de la aplicación RIEEEWEB no es la excepción. Utilizando esta metáfora se han desarrollado las páginas que componen la aplicación.

Cada página está compuesta de 4 vistas:

- Cabecera: Donde se encuentra el banner con la imagen de la IEEE y el menú dinámico que se despliega para cada usuario según su rol en el sistema.
- Lateral: Donde se encuentra la sección de noticias. Además se muestra información de la sesión actual del usuario (usuario y rol)
- Cuerpo: Donde se muestra el contenido de principal de cada página.

- Pie: Donde se muestran ciertos enlaces importantes dentro y fuera de la aplicación.

A través de estas cuatro sub-vistas se construyen todas las páginas dentro de la aplicación. Utilizando este modelo de vistas compuestas se logra dar mayor flexibilidad al mantenimiento del diseño de la aplicación, ya que así, si se quieren cambiar el diseño de las páginas, simplemente bastará con cambiar el diseño de las subvistas y esto estará disponible para todas las páginas.

A continuación se muestra la figura 4.7 donde puede observar como una página esta formada por varias sub-vistas

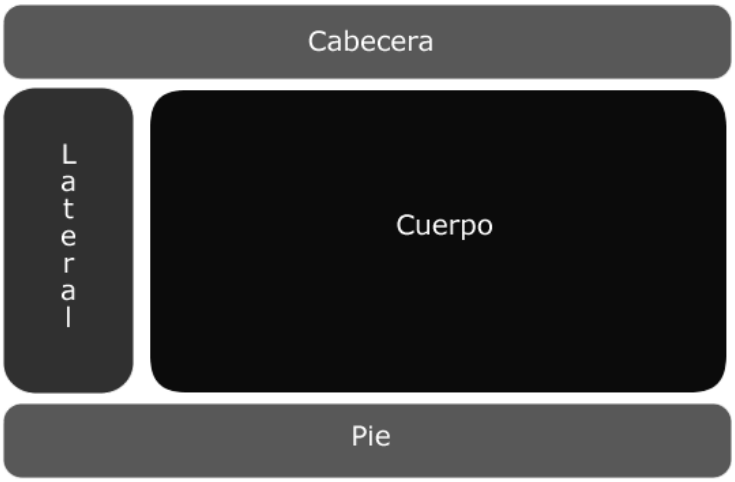


Figura 4.7 Diseño modular de páginas a través de sub-vistas

Otra ventaja de utilizar este enfoque para el desarrollo de la interfaz es el de la reutilización de código. A través de las sub-vistas se evita la inserción de código repetido en todas las páginas de la aplicación. Se tiene una sola vista y en cada página se cambia solamente la información que se quiere presentar. Si se decidiera cambiar el diseño de las páginas bastaría con cambiar las vistas y los cambios se reflejarían en toda la aplicación ahorrando así una gran cantidad de trabajo en el momento de hacer cambios radicales en el diseño del portal.

La única consideración que hay que tener con este modelo para el diseño de la interfaz es que las sub-vistas no son vistas completas y por lo tanto no pueden llevar etiquetas HTML que ya se encuentren en la vista principal. Ejemplo: una sub-vista no puede llevar la etiqueta <html></html> porque esto llevaría a que nuestra página no contenga XHTML válido.

La figura 4.8 muestra una página del sistema RIEEEWEB donde se aprecian las cuatro secciones mencionadas en un párrafo anterior.

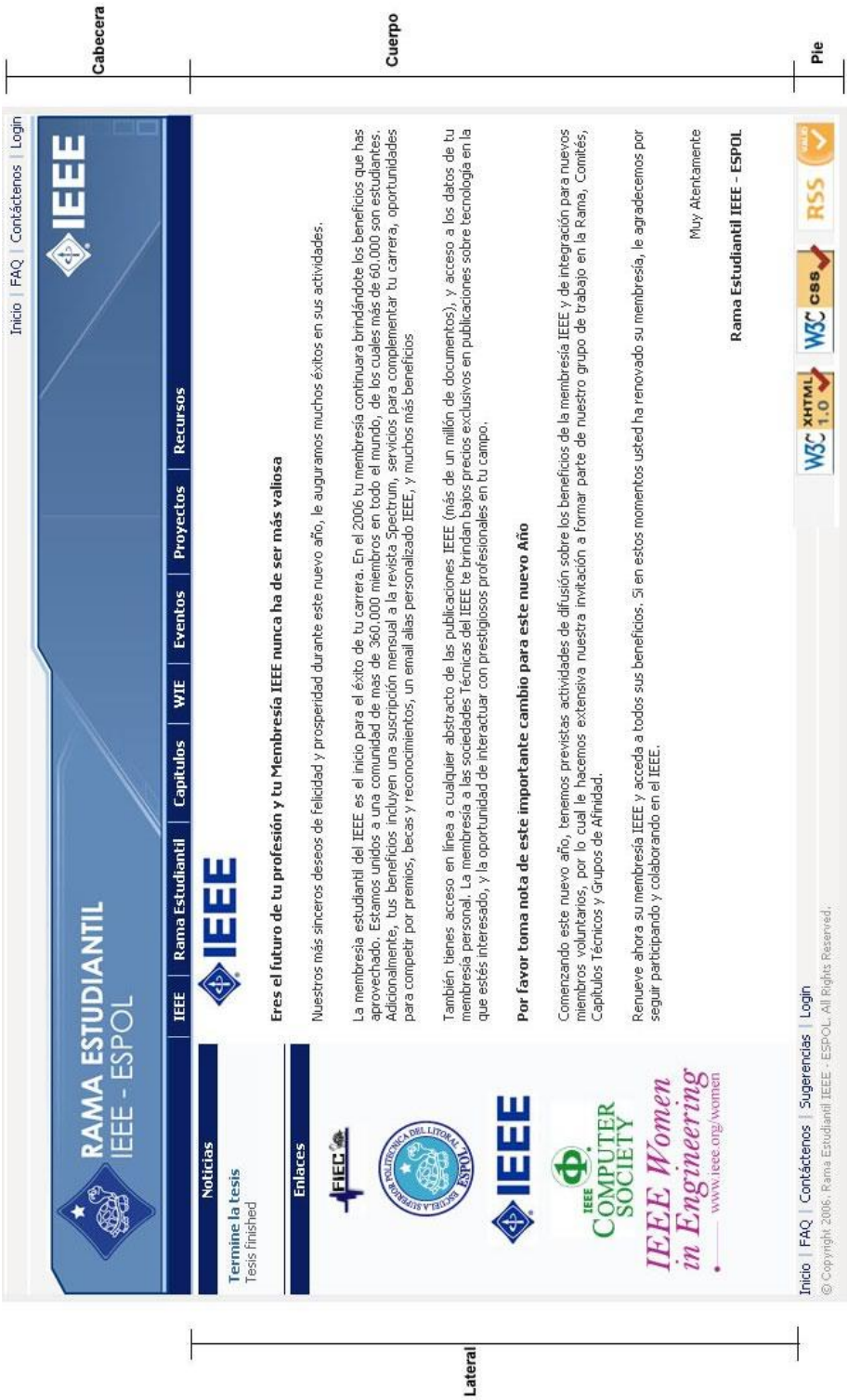


Figura 4.8 Diseño de la interfaz del sistema RIEEEWEB

4.2. Modelo lógico de la base de datos

A continuación mencionaremos algunas de las tablas más importantes del sistema, las cuales son vitales para el funcionamiento de la aplicación.

Persona: Donde se almacena todos los datos de participantes a eventos, miembros de la rama, conferencistas.

Para al funcionamiento del módulo de eventos son necesarias las siguientes tablas:

Eventoieee: Almacena todos los eventos.

Conferenciaieee: Almacena las conferencias de los eventos.

CurriculumConferencistaieee: Almacena los datos del conferencista. Está relacionada con la tabla persona en donde se almacenan los datos.

ParticipanteEventoieee: Almacena a los participantes a los eventos.

En el módulo de Membresías tenemos las siguientes tablas:

Miembroieee: Almacena los datos de la membresía IEEE.

Sociedades-miembro: Almacena las sociedades IEEE a las que se registra una persona.

Para la administración de la correspondencia encontramos las siguientes tablas:

CorrespondenciaIEEE: Almacena los datos de la correspondencia IEEE que ha llegado a la rama.

Correspondencia-miembro: Almacena la correspondencia que cada miembro IEEE tiene asignada.

Para el manejo de inventario de la rama tenemos las siguientes tablas:

Equipoieee: Almacena todos los ítems de inventario de la rama estudiantil.

MovimientoInventario: Registra todos los movimientos de los activos de la rama.

Para el manejo de los recursos financieros de la rama encontramos como principales tablas las siguientes:

ComprobanteIngreso: Comprobante de registro de un ingreso financiero.

ComprobanteEgreso: Comprobante de registro de un egreso financiero.

DetalleComprobanteIngreso: Almacena los detalles de cada comprobante de ingreso.

DetalleComprobanteEgreso: Almacena los detalles de cada comprobante de egreso.

El modelo de datos de la Rama Estudiantil IEEE – ESPOL consta de 40 tablas. En esta sección se describieron algunas de las tablas y su función dentro del sistema. A continuación se presentan en varios diagramas que representan la funcionalidades del sistema, todas las tablas del modelo de datos.

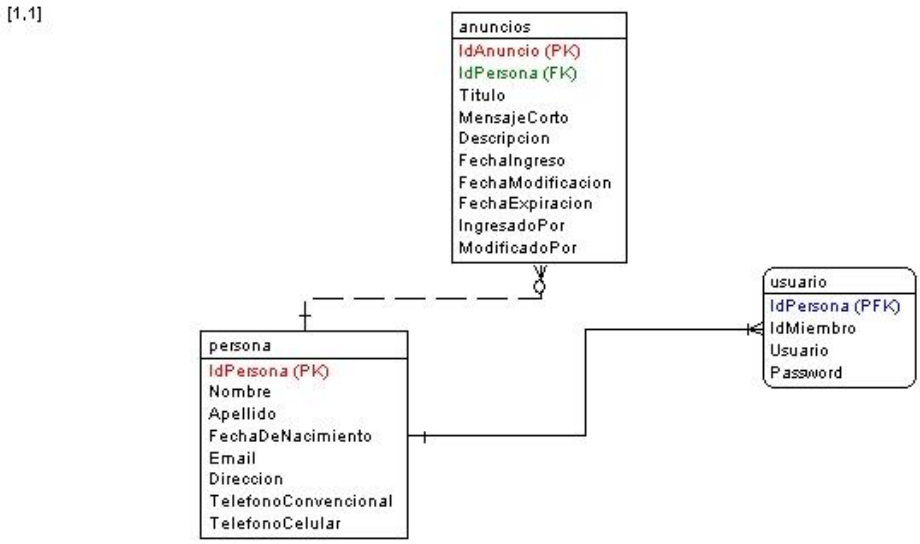


Figura 4.9 Diagrama de Datos que representa el Módulo de Anuncios.

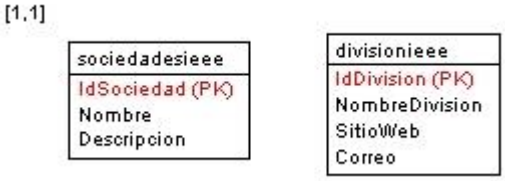


Figura 4.10 Diagrama de Datos que representa el Módulo de Sociedades y Capítulos.

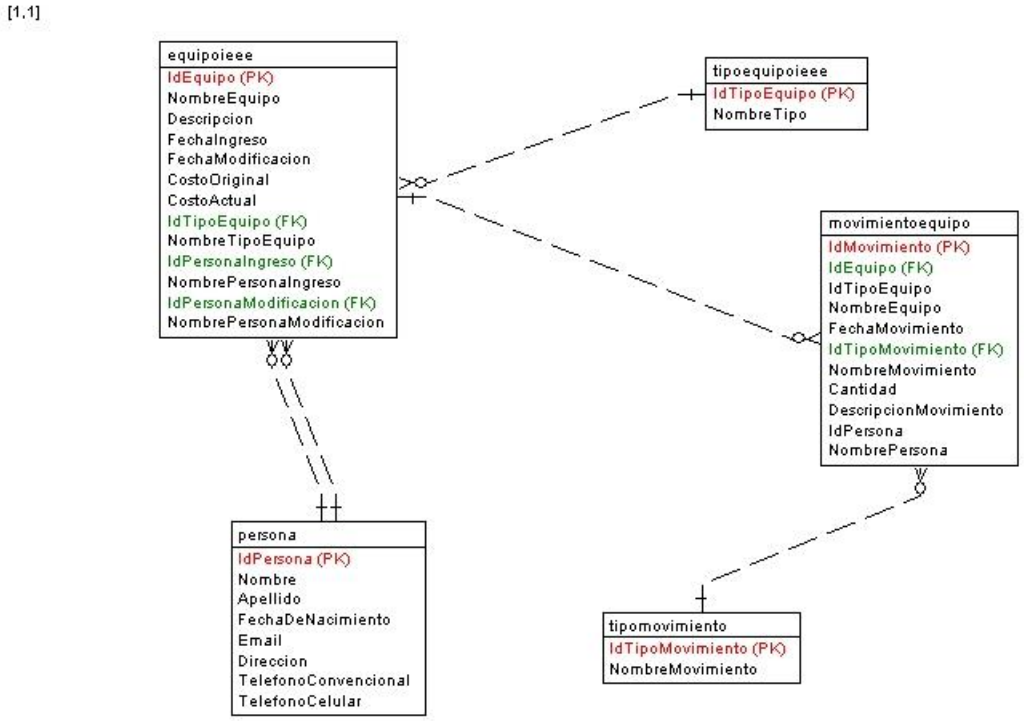


Figura 4.11 Diagrama de Datos que representa el Módulo de Inventario.

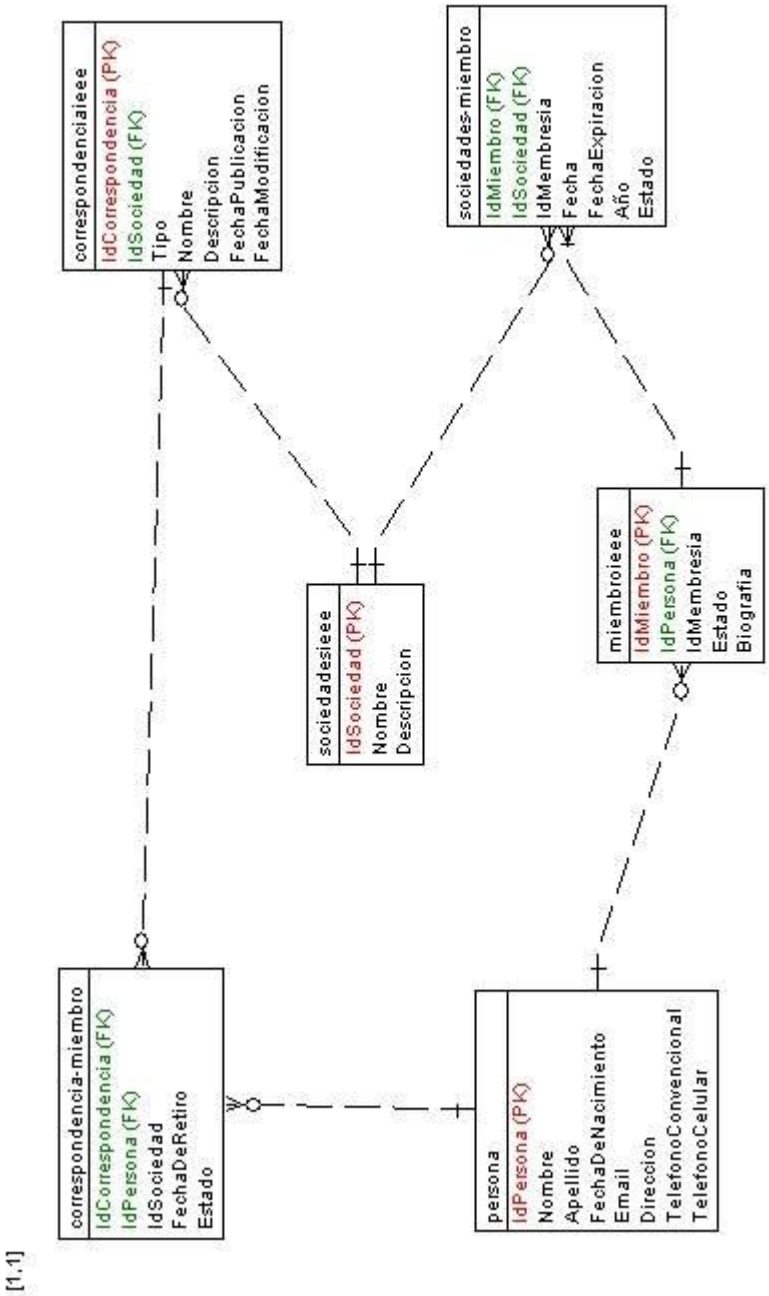


Figura 4.12 Diagrama de Datos que representa el Módulo de Correspondencia.

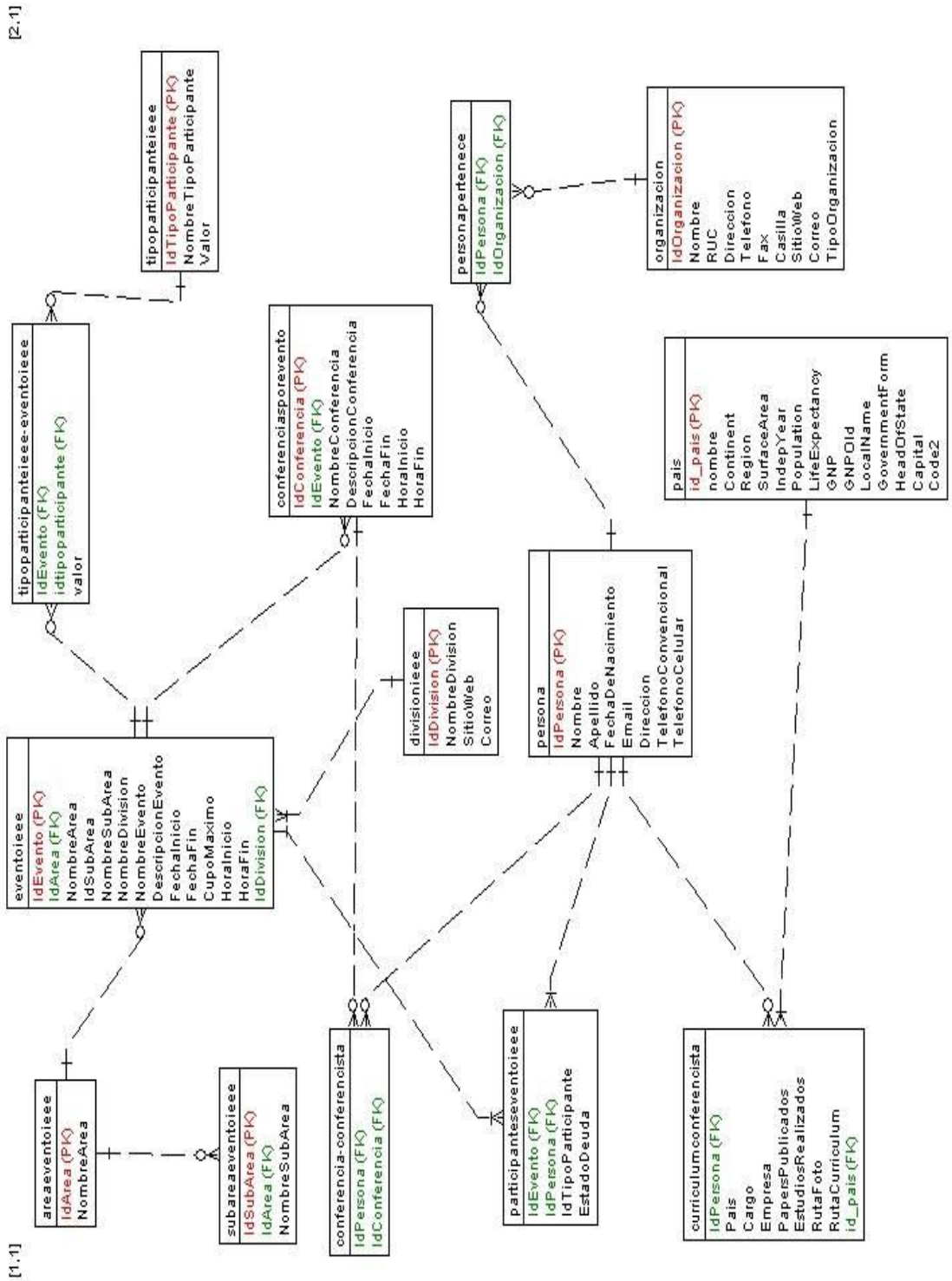


Figura 4.13 Diagrama de Datos que representa el Módulo de Eventos

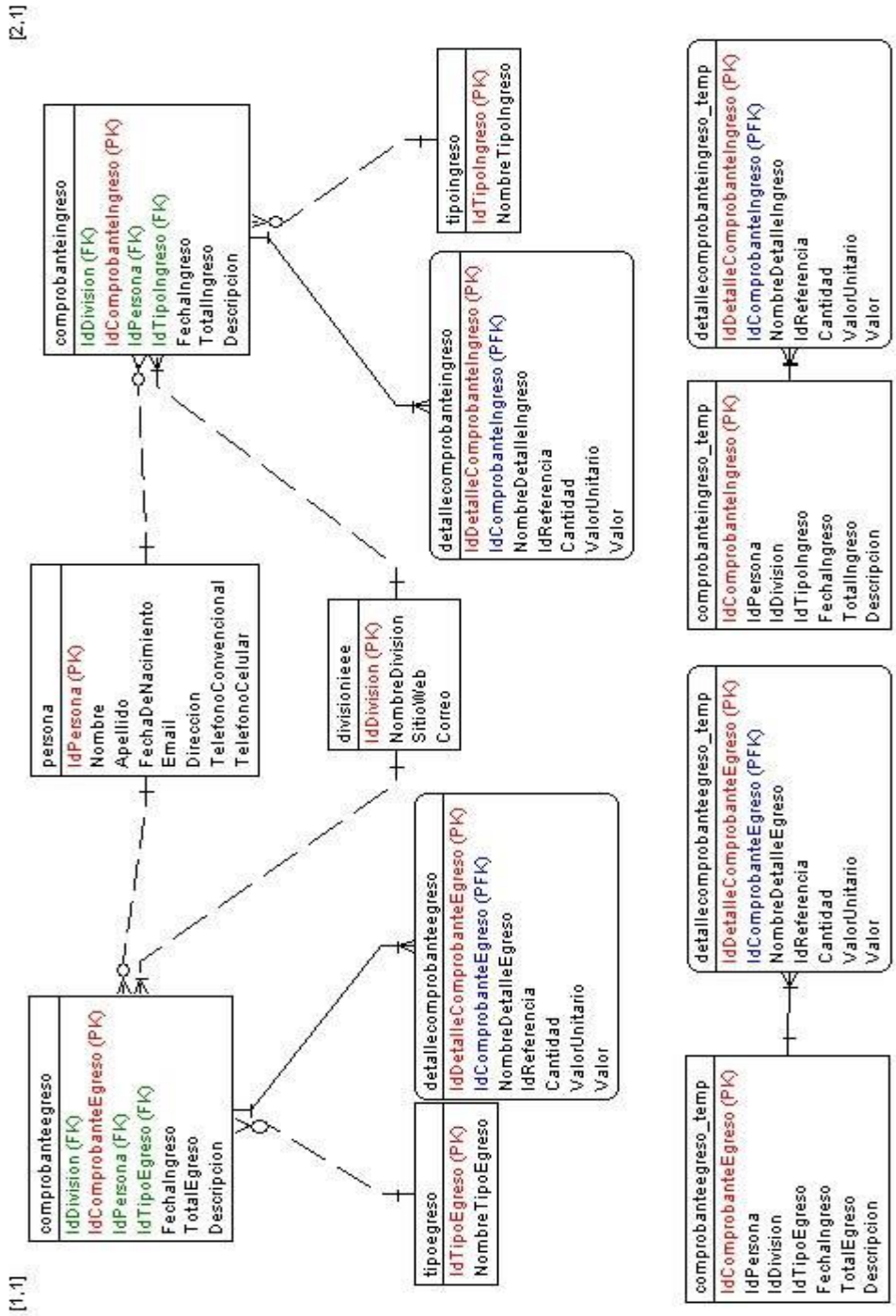


Figura 4.14 Diagrama de Datos que representa el Módulo Financiero.

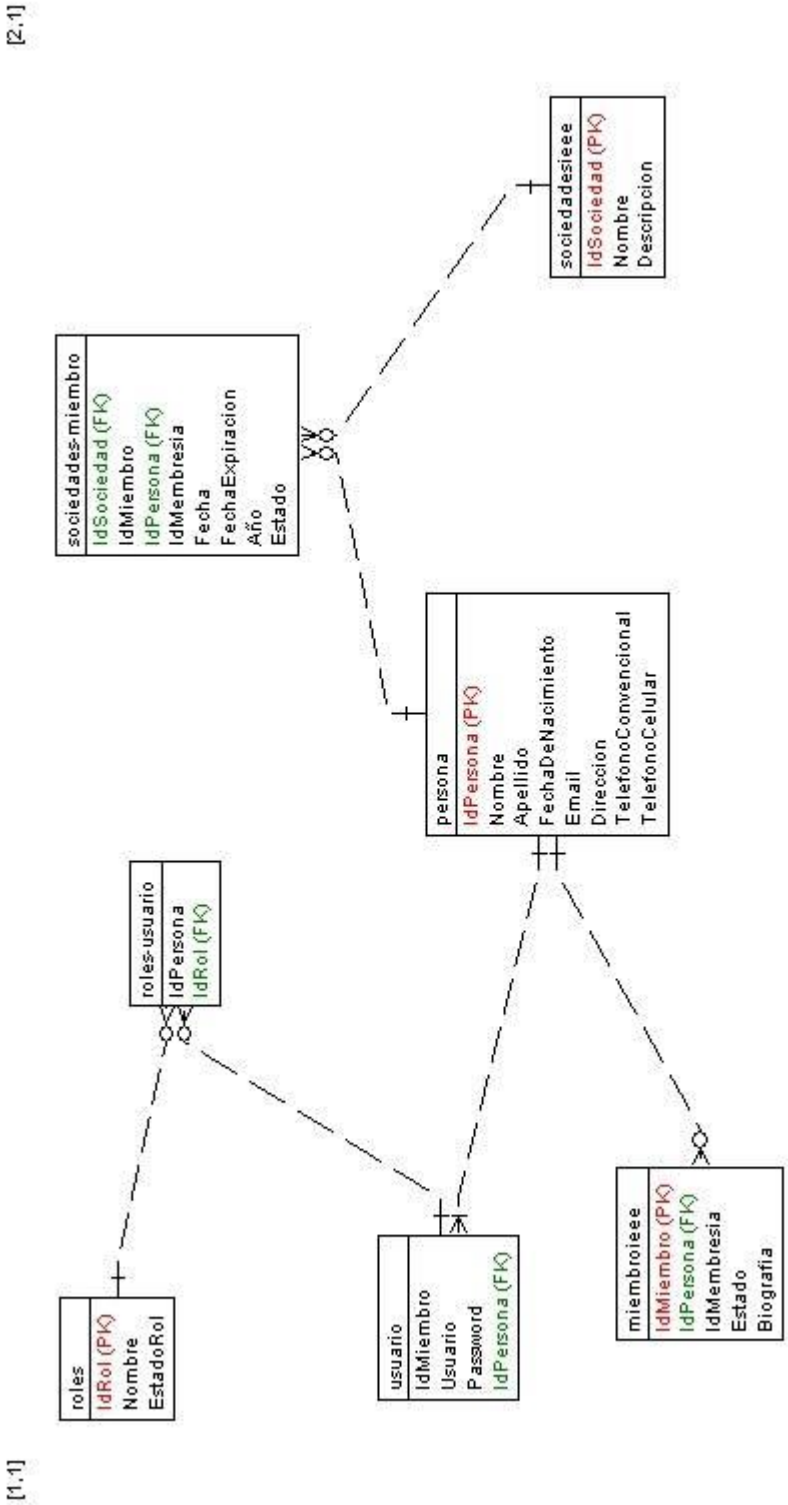


Figura 4.15 Diagrama de Datos que representa el Módulo de Membresía.

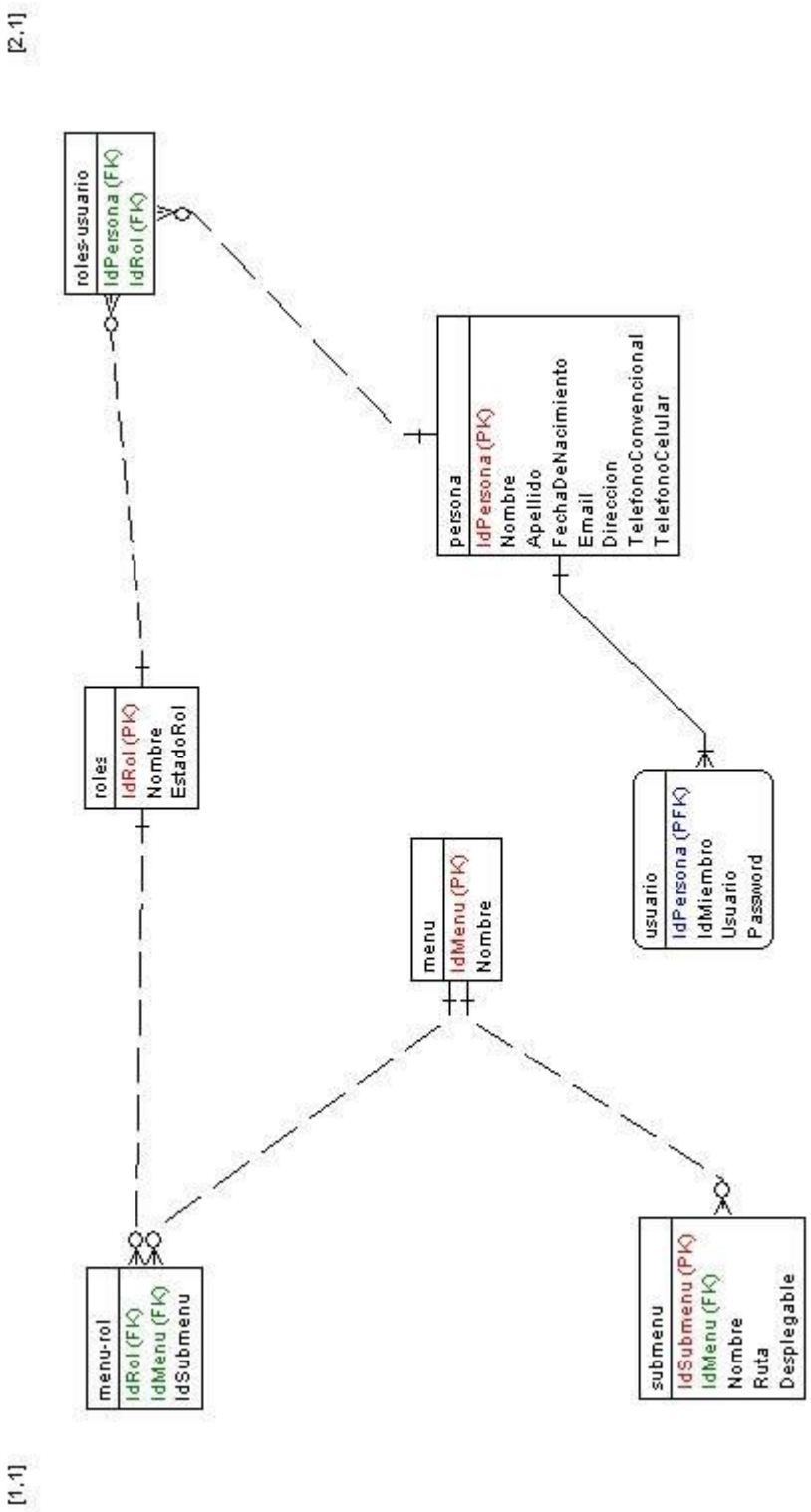


Figura 4.16 Diagrama de Datos que representa el Módulo de Roles.

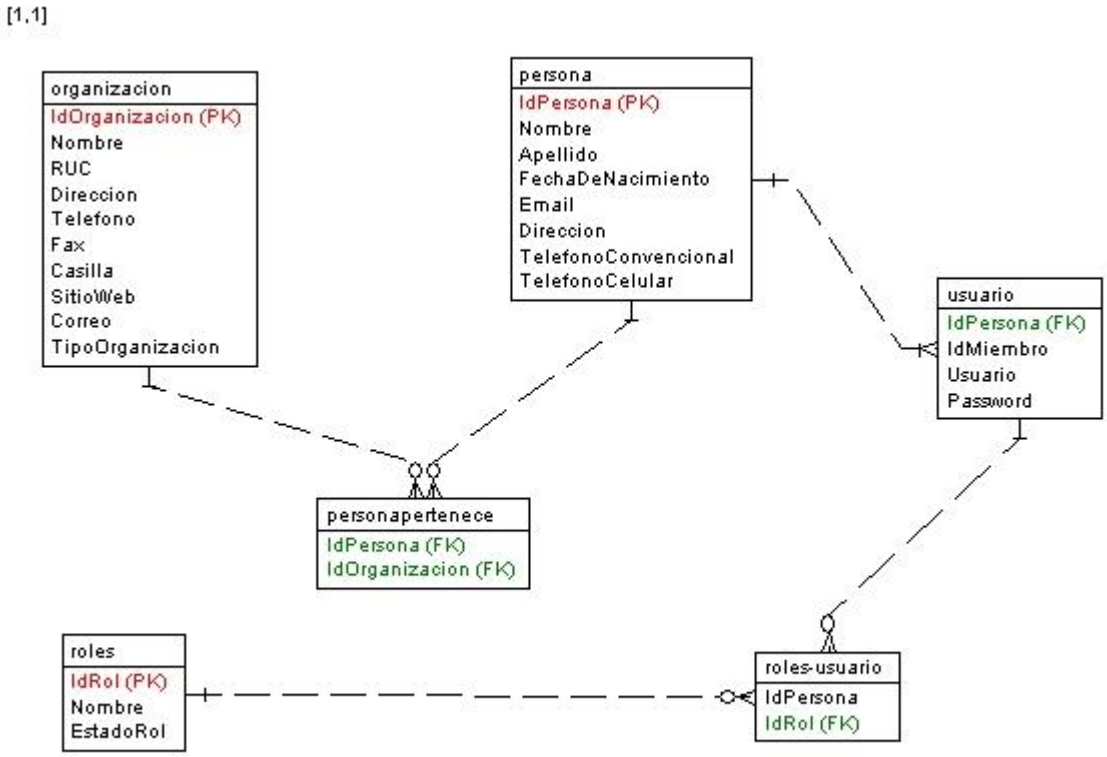


Figura 4.17 Diagrama de Datos que representa el Módulo de Usuarios.

CAPITULO 5

5. IMPLEMENTACIÓN Y PRUEBAS

5.1. Implementación

5.1.1. Menús dinámicos

El sistema RIEEEWEB cuenta con menús desplegables que permiten al usuario acceder a las diferentes operaciones implementadas en la aplicación. Estas opciones se cargan en el menú de acuerdo al rol que el usuario tiene asignado en el sistema.

El menú del sistema aparece en la parte superior del mismo y ha sido diseñado para tener un solo nivel de profundidad.

La figura 5.1 muestra el menú del sistema RIEEEWEB.

Figura 5.1 Menú Dinámico del sistema

Las opciones que se muestran en el menú se obtendrán para cada usuario dependiendo del rol que éste tenga en el sistema, ya que de acuerdo a ello existen opciones que ciertos usuarios pueden visualizar y otros no.

5.1.2. Módulo de roles

El módulo para la administración de roles del sistema sólo puede ser accedido por usuarios de tipo Administrador General y sirve para la creación de nuevos roles de acuerdo a las necesidades de la rama.

Para la creación de los roles tenemos clases de Java que implementan los patrones Data Object Value y Data Access Object descritos en secciones anteriores. La clase llamada RolesDVO es la que almacena los datos almacenados en la base de datos. Estos datos se obtienen con una clase llamada RolesDAO que implementa todos los métodos para manipular los roles del sistema.

Los roles nuevos pueden ser creados sólo en base a las opciones que ya se encuentran disponibles en el sistema, no se permite la creación de nuevas opciones para los menús.

Adicionalmente este módulo permite la eliminación de roles que a criterio del administrador ya no presten ninguna utilidad. La eliminación de los roles se la realiza de manera lógica más no física, es decir el estado del rol se cambia a inactivo. Cuando un usuario que posee un rol que ha sido inactivo, el sistema no permite al usuario ingresar con su usuario y contraseña pues el rol no existe ya para la aplicación.

La figura 5.2 muestra la interfaz de usuario de para la creación de un nuevo rol.

Figura 5.2 Creación de un nuevo rol en el sistema.

5.1.3. Módulo de Anuncios

Este módulo está habilitado para un usuario de tipo Administrador de Correspondencia y Membresías y permite que el usuario publique anuncios en el sistema para que éstos puedan ser vistos por todos los visitantes.

Los anuncios publicados aparecen al lado izquierdo de cada página en un pequeño espacio donde están rotando todos los anuncios publicados y que no han expirado. Cada anuncio que se publica tiene una fecha de expiración para que los anuncios sólo se publiquen hasta esa fecha y después dejen de publicarse de manera automática.

Cuando un anuncio es publicado, estos son insertados en la página a través de un método de JavaScript utilizando el modelo propuesto por AJAX. Cada página JSP tiene un método llamado onload que pertenece a la sección body, que representa el contenido principal del documento HTML. En este método se llama a la función que de manera asincrónica recolecta los anuncios que están disponibles y los coloca en la parte lateral de la página. Mediante un efecto implementado en JavaScript, los anuncios se van desplegando con intervalo de tiempo de fijado en la aplicación.

La figura 5.3 muestra un anuncio que ha sido publicado por el sistema y que puede ser visto por los usuarios.

Figura 5.3 Publicación de Anuncios

En esta sección aparece solamente una pequeña descripción del anuncio. Si se desea ver el anuncio completo sólo se tiene que seguir el enlace incluido en la descripción del mismo. Al dar clic sobre el anuncio el enlace lleva al usuario a una página donde puede ver el anuncio completo.

La figura 5.4 muestra el anuncio visto de manera completa.

Figura 5.4 Visualización completa de un anuncio.

De esta manera, a través de la publicación de anuncios se puede mantener informados a todos de cualquier noticia relacionada con la rama, especialmente la creación de nuevos eventos. Adicionalmente el sistema envía de manera automática después de cada publicación de anuncios un correo electrónico como aviso a todas las personas que tienen sus datos registrados en el sistema.

La implementación del envío de correo electrónico se la realiza mediante la librería JavaMail que tiene implementada toda la funcionalidad necesaria para enviar correo. Cuando se utiliza esta librería es necesario configurar la dirección de un servidor SMTP, así como las direcciones de correo electrónico de origen y de destino. En la aplicación implementada, la dirección del servidor y la dirección de correo electrónico de origen se leen del archivo de configuración web.xml

5.1.4. Módulo de Correspondencia

Este módulo está habilitado para los usuarios de tipo Administrador de Correspondencia y Membresías y permite registrar en el sistema toda la correspondencia que ingresa a la rama.

Este módulo tiene dos secciones que son:

- La sección que ve el administrador de Correspondencia
- La sección que ve el miembro IEEE

La primera sección le permite al administrador de correspondencia ingresar, modificar y entregar la correspondencia a cada miembro.

La segunda permite al miembro IEEE ver la correspondencia personal que ha llegado a la rama y que ha sido asignada a su buzón para acercarse a retirarla.

Al ingresar una correspondencia al sistema, esta sólo se asigna a aquellos miembros a los que les llegó y de acuerdo a qué sociedad IEEE envió la correspondencia. De igual manera al ingreso se envía un correo electrónico informativo a todos los miembros.

Además del aviso por medio del correo electrónico los miembros pueden ver su correspondencia en la sección correspondiente que aparece en la parte izquierda del sistema.

Al seleccionarla verán un listado de la correspondencia que tienen y que no ha sido retirada.

Para ingresar una nueva correspondencia se llena una clase de tipo CorrespondenciaDVO con los datos ingresados por el usuario. Esta clase se pasa luego a la clase que implementa el acceso a datos la cual se encarga de grabar los datos correspondientes y de relacionar la correspondencia con los usuarios respectivos, se almacena en el objeto request los datos que fueron grabados en la base para que estos sean despachados por el servlet llamado *ServletCorrespondencia* hacia una página donde el usuario recibe la retroalimentación de que la información fue almacenada satisfactoriamente.

La figura 5.5 muestra un listado de la correspondencia de un miembro de la Rama.

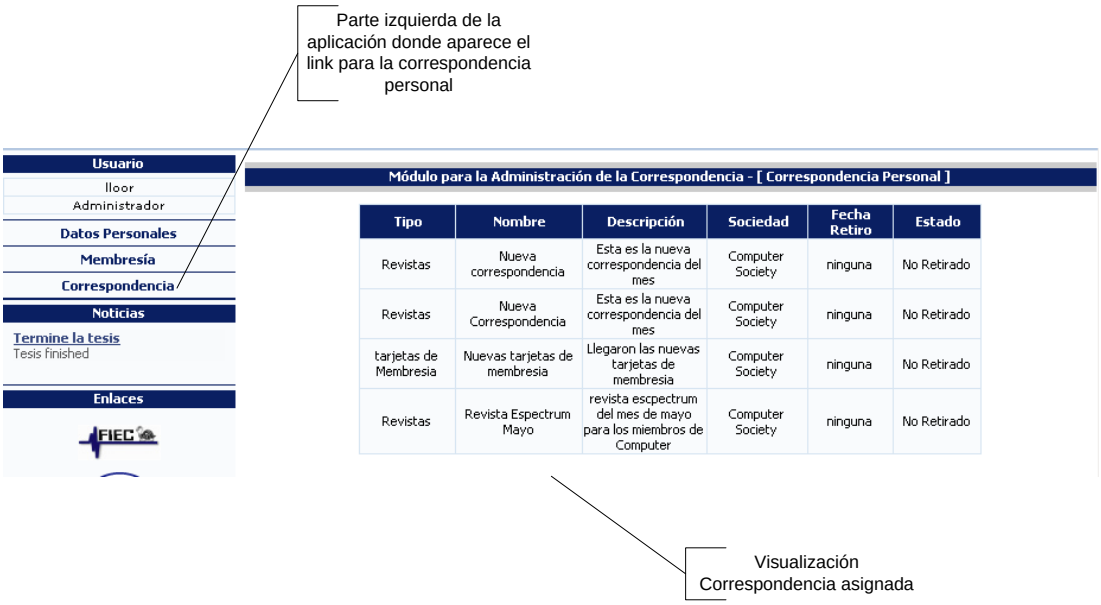


Figura 5.5 Visualización de Correspondencia Personal

Para la entrega de correspondencia existe una opción que permite que se entregue a cada miembro la correspondencia asignada y se marque en el sistema como entregada al miembro destinatario de la misma.

La figura 5.6 muestra cómo un Administrador de Correspondencia indica en el sistema que una correspondencia ha sido entregada.

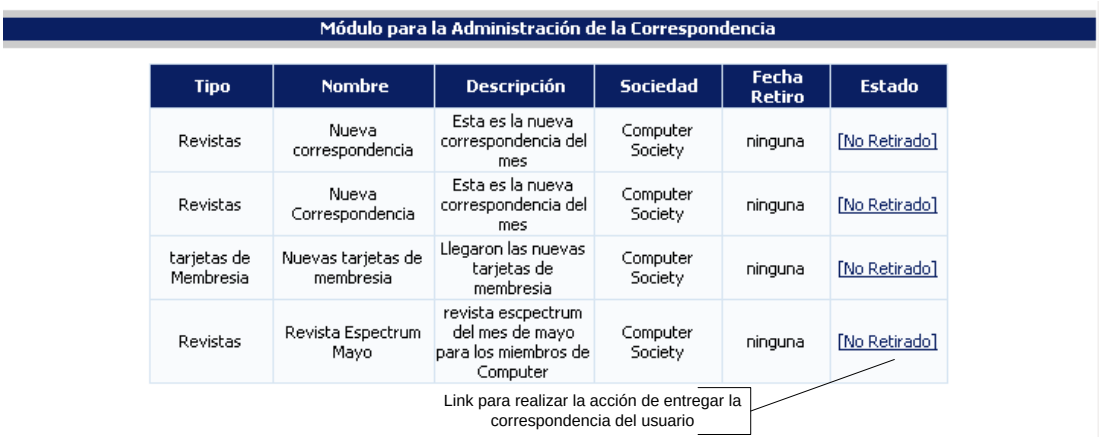


Figura 5.6 Entrega de correspondencia

5.1.5. Módulo de Inventario

Este módulo permite a un usuario de tipo administrador Financiero administrar todo el inventario físico de la rama.

El módulo de inventario permite ingresar nuevo inventario, así como modificarlo, además permite registrar todos los movimientos que se registren sobre cada ítem del inventario.

Para el registro de los movimientos de un ítem del inventario se emplean búsqueda de los ítems por medio de una ventana externa donde se muestran las coincidencias y se selecciona el ítem al cual se le va a registrar un movimiento en el inventario.

La figura 5.7 muestra una búsqueda de ítems de inventario. Al ingresar parte de la palabra que corresponde al ítem que se desea buscar se abre una ventana externa con las coincidencias para que el usuario seleccione el ítem que desea.

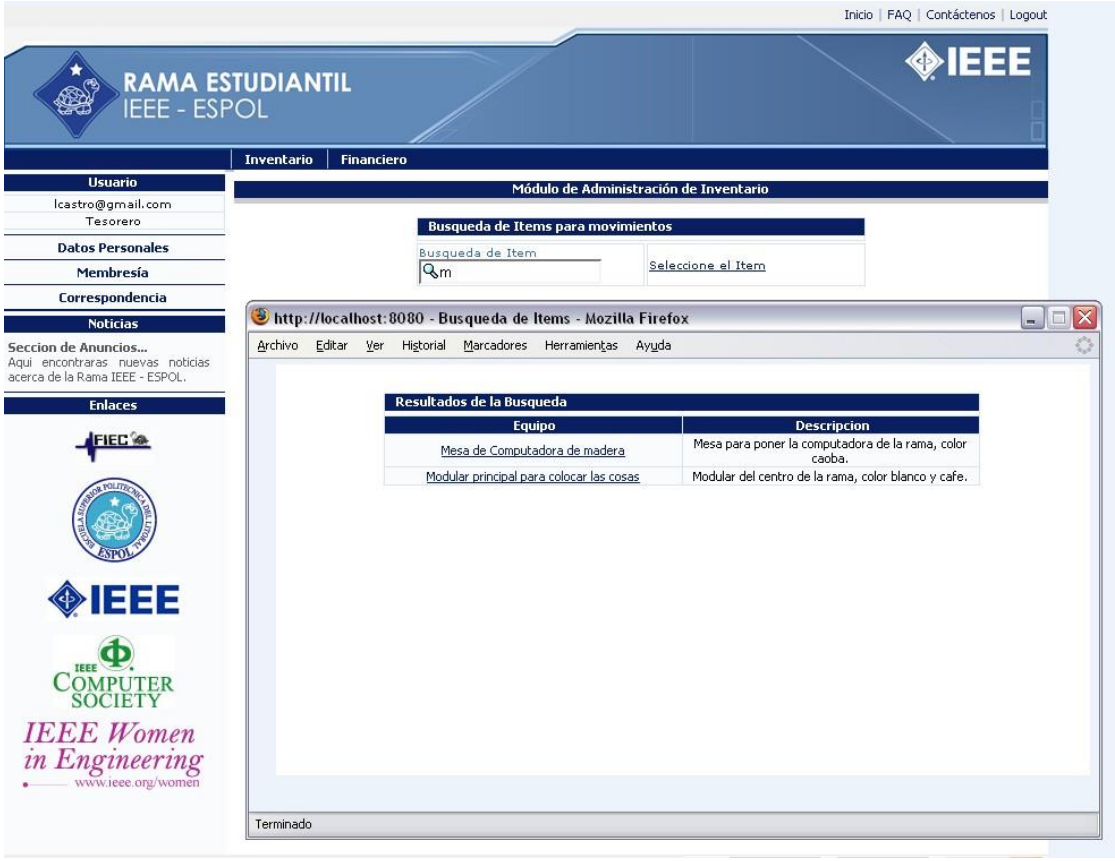


Figura 5.7 Búsqueda de ítems para registro de movimientos en el inventario

Algo relevante en este módulo de inventario es la creación de reportes que permiten ver listar los ítems que conforman el

inventario de la rama y todos y cada uno de los movimientos de los mismos. Esto servirá para tener un documento de referencia al hacer un conteo físico de los activos de la rama.

La figura 5.8 muestra un ejemplo de un reporte de inventario.

Reporte de Inventario Ingresado

Codigo	1	
Nombre Equipo	Mesa de	
Descripcion	Mesa para poner la computadora de la rama, color caoba.	
Fecha Ingreso	2007/04/14	
Fecha Modificacion	2007/04/14	
Costo Original	200.00	
Costo Actual	200.00	
Tipo Equipo	Muebles y Enseres	
Ingresado Por	Luis Enrique Loor Maspons	0920999406
Modificado Por	Luis Enrique Loor Maspons	0920999406

Codigo	2	
Nombre Equipo	Modular principal	
Descripcion	Modular del centro de la rama, color blanco y cafe.	
Fecha Ingreso	2007/04/14	
Fecha Modificacion	2007/04/14	
Costo Original	150.00	
Costo Actual	150.00	
Tipo Equipo	Muebles y Enseres	
Ingresado Por	Luis Enrique Loor Maspons	0920999406
Modificado Por	Luis Enrique Loor Maspons	0920999406

Figura 5.8 Reporte de Inventario

Los reportes generados en este módulo fueron realizados utilizando una herramienta de software libre llamada Jasper Reports. Esta es descrita más adelante en la sección de software utilizado.

Jasper Reports es una herramienta que acepta consultas SQL que se envían de acuerdo a los datos que se quieren mostrar.

A continuación se muestra la figura 5.9 donde se aprecia la consulta SQL para obtener los datos de inventario ingresado.

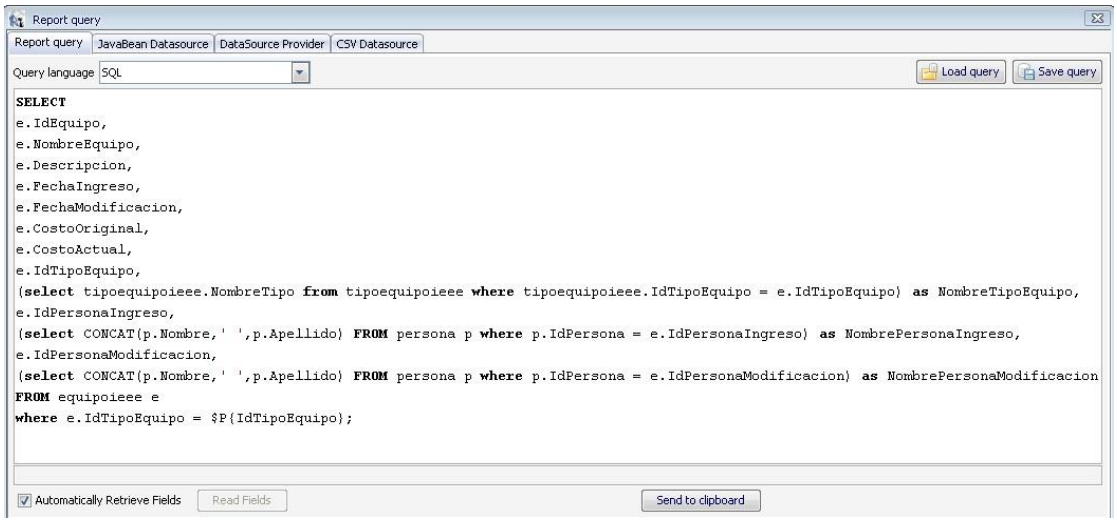


Figura 5.9 Consulta de Inventario en Jasper Reports

5.1.6. Módulo de Membresías

Este módulo permite a un usuario de tipo Administrador de Correspondencia y Membresías registrar los datos de las personas que quieren registrarse en el IEEE, así como las renovaciones posteriores de los miembros.

En el momento del registro se eligen las sociedades a las que quiere pertenecer el usuario y se le asigna el nombre de usuario y contraseña con la que puede ingresar al sistema.

Como las membresías, las sociedades IEEE tienen un tiempo límite de validez y deben expirar. Para este efecto existe una opción que permite inactivar las membresías de las sociedades que un usuario cualquiera ya tenga expiradas.

Para realizar el ingreso de los datos el servlet llamado *ServletMembresía* recoge los datos enviados desde la página donde se encuentra el formulario de ingreso de información. Una vez aquí el servlet crea tres instancias de clases DVO: PersonaDVO donde se almacena los datos de la persona, MiembroDVO que almacena los datos de la membresía y Sociedades-miembroCVO que almacena los datos de las

sociedades en las cuales se registra la persona. Cuando los objetos de valor han sido llenados se llama a una instancia de MiembroDAO que recoge las tres instancias de objetos anteriores y procede a realizar la operación de inserción en la base de datos.

Un aspecto importante de este módulo es que permite crear reportes de los miembros que se encuentran activos como inactivos y otro de los miembros activos por cada sociedad IEEE registrada en la rama. De esta forma las listas no se crearán de forma manual, sino que serán generadas por el sistema. En este módulo también se utiliza Jasper Reports para los reportes de los miembros de la rama.

5.1.7. Módulo de Eventos

Este módulo permite a un usuario de tipo Administrador General la creación de eventos para que los usuarios puedan usar la opción de registrarse en línea.

Para la creación de eventos se utilizó el siguiente esquema: un evento puede tener varias conferencias registradas así como una conferencia puede tener varios conferencistas asignados.

Este módulo se compone dos partes:

- La sección para el administrador
- La sección para el participante

La sección para el administrador es donde se crean los eventos, se agregan las conferencias y se asignan los conferencistas. Como se explicó anteriormente un evento puede tener varias conferencias por lo que el ingreso de las conferencias se realiza usando AJAX. Así se evita tener que refrescar la página en varias ocasiones. Cuando se ingresa una nueva conferencia, se muestran los datos de las conferencias ingresadas y un enlace que permite ingresar los conferencistas. Así se pueden ingresar varias conferencias para un mismo evento.

Para el proceso de crear un evento es necesario que el servlet llamado *ServletEventos* cree las siguientes instancias de objetos: EventoDVO, ConferenciaDVO, PersonaDVO, CurriculumConferencistaDVO y EventoDAO.

Para el ingreso del evento el servlet recoge los datos enviados en el objeto request y los almacena en el objeto correspondiente. Luego se invoca a un método implementado

en la clase EventoDAO que recibe como argumento un objeto EventoDVO e inserta los datos del mismo en la base de datos.

Como se expresa en párrafos anteriores, las conferencias se ingresan utilizando AJAX. Para esto es necesario que mediante una función implementada en JavaScript se invoque al *ServletEventos* para crear un documento XML que será devuelto como respuesta. Siguiendo el modelo propuesto por AJAX, en la página se define una función que manejará la respuesta asincrónica cuando ésta sea devuelta. Esta función actualizará la página utilizando la implementación del DOM en JavaScript.

La figura 5.10 muestra el formulario de ingreso de las conferencias y los datos de una conferencia que ya ha sido ingresada en la parte inferior del formulario.

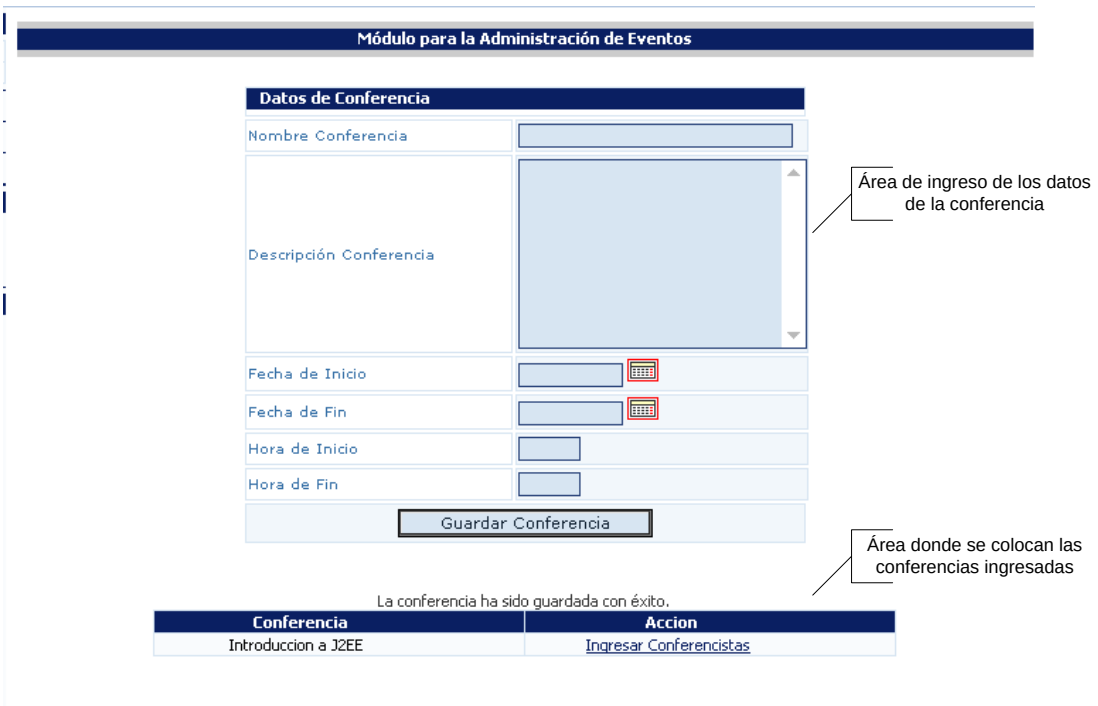


Figura 5.10 Ingreso de las conferencias con AJAX

De la misma manera el ingreso de los conferencistas se hace mediante *AJAX* ya que el comportamiento es igual al de las conferencias.

En la página principal de la sección Eventos, existe un menú donde se listan los eventos de cada capítulo de la rama. Aquí el usuario puede ver los detalles de los eventos así como registrarse en los mismos siempre y cuando haya cupo disponible. Al momento de hacer la consulta SQL para obtener los eventos a mostrar en la página JSP, se verifica que la fecha

de expiración no haya sido excedida; realizada esta verificación se obtienen los datos y el *ServletMembresias* envía los datos a una página JSP para que el usuario pueda ver los eventos disponibles.

Este módulo permite así mismo la creación de reportes como por ejemplo el reporte de credenciales del evento para los participantes registrados, un listado de los asistentes al evento y un resumen de los ingresos que dejó el evento.

La figura 5.11 muestra un reporte de los ingresos de un evento.

Reporte Ingresos Por Evento

Nombre del Evento

Fast Seeker

Tipo Participante	Costos	Registrados
Miembro IEEE	1	1
Estudiante Espol	2	0
Estudiante otra universidad	3	0
Profesional ESPOL	4	0
Profesional otra universidad	5	0

Total Ingresos: 1

Figura 5.11 Reporte de Ingreso por eventos

5.1.8. Módulo de Capítulos

Este módulo permite a un usuario de tipo Administrador General ingresar y modificar los datos de los capítulos que la rama vaya formando para la integración de sus miembros.

De manera similar al ingreso de las sociedades, se realiza la invocación a un servlet, en este caso *ServletCapitulos* que almacena los datos del nuevo capítulo a ser creado y los almacena en una instancia de una clase CapituloDVO. El servlet crea otra instancia pero esta vez de una clase CapituloDAO que contiene el método que inserta los datos en la base de datos. Así mismo los datos del nuevo capítulo se guardan en el objeto request para que sean despachados por el *servlet* a la página de retroalimentación del usuario.

5.1.9. Módulo de Sociedades

Este módulo permite al usuario de tipo Administrador General ingresar y modificar los datos de las nuevas sociedades IEEE que la rama desee promocionar a sus posibles miembros.

Para la creación de nuevas sociedades, se invoca un servlet llamado *ServletSociedades* el cual crea una instancia de la

clase SociedadesDVO. El *Servlet* recoge los datos y llena el objeto creado con la información, luego se crea una instancia de la clase SociedadesDAO la cual implementa el método que inserta los datos de la nueva sociedad en la base de datos. Los datos de la sociedad ingresada se guardan en el objeto request del servlet que despacha dicho objeto hacia una página de retroalimentación que indica el ingreso exitoso o no.

5.1.10. Módulo Financiero

Este módulo permite al usuario de tipo Administrador Financiero registrar todos los ingresos y egresos de la rama o separarlos de acuerdo a cada capítulo.

El registro de los ingresos y egresos utiliza AJAX para el registro de los detalles de cada transacción. Esto permite simular el registro de varios detalles en las transacciones en una sola página lo que da la apariencia de ser un solo formulario de ingreso. Este registro de ingreso se realiza mediante una llamada a una función de JavaScript que invoca de manera asincrónica al servlet llamado *ServletFinanciero*. A este servlet llegan los parámetros que son los datos del detalle

de la transacción y este los procesa para producir la salida que son los valores del detalle y el valor total de la misma.

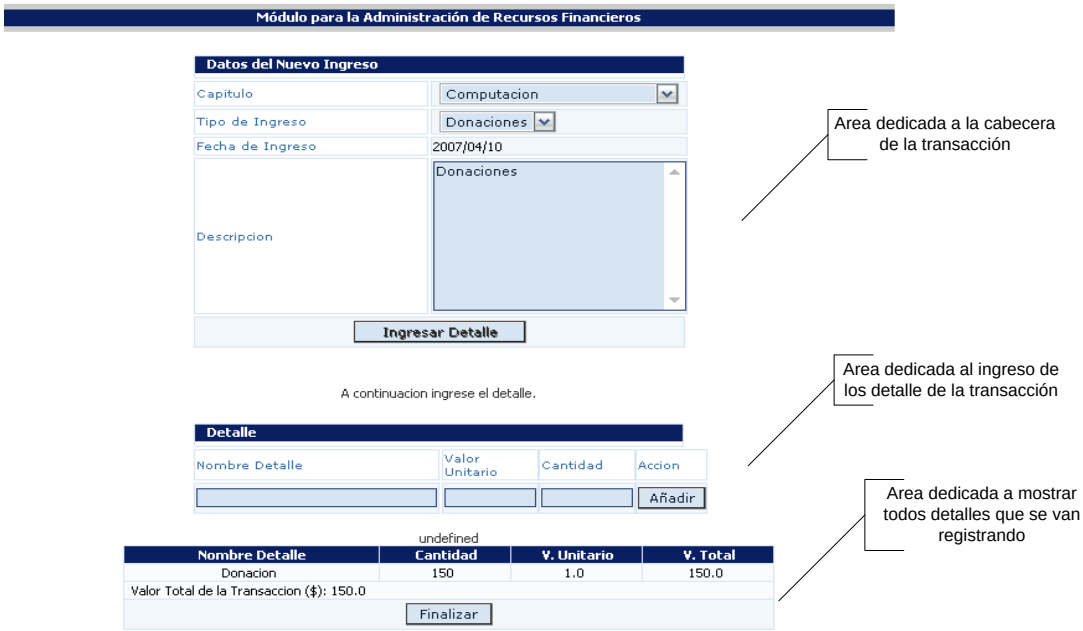


Figura 5.12 Formulario de registro de transacción

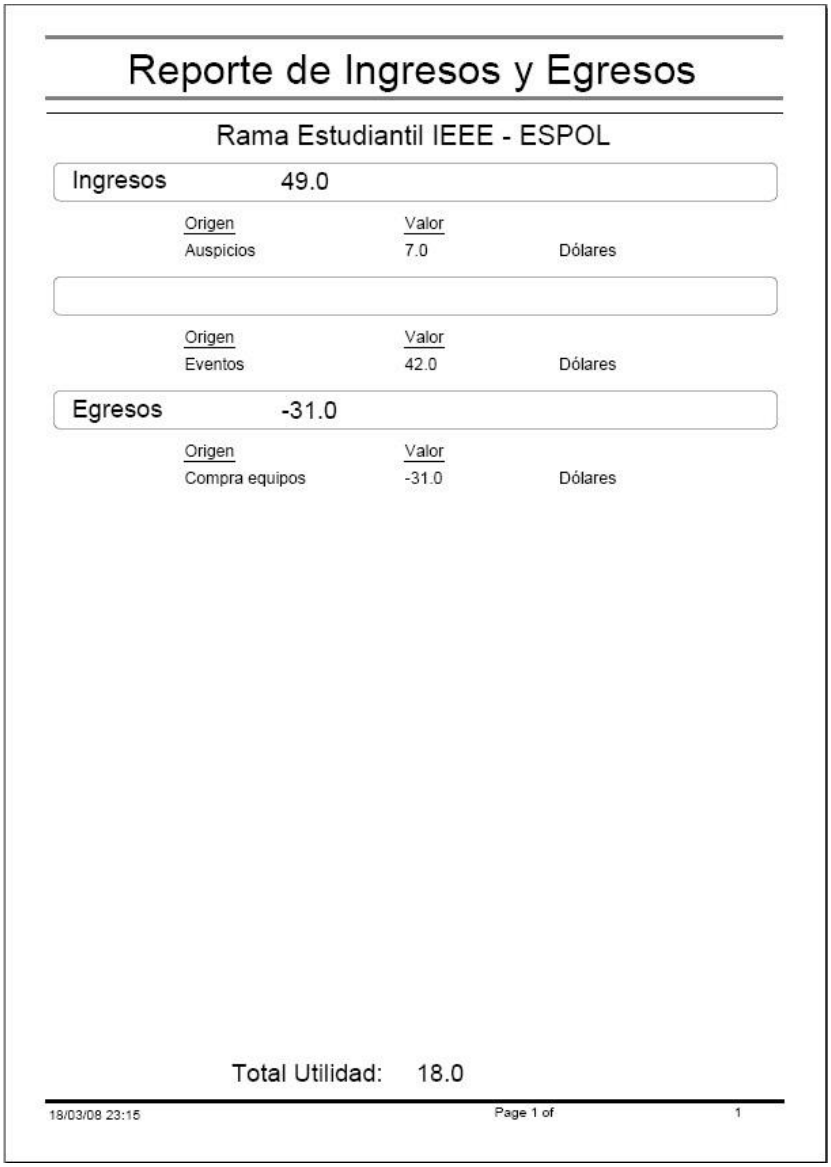
De igual manera se permite la modificación de transacciones ya sean de ingresos o egresos, para realizar estas operaciones también se utiliza *AJAX*. Se lo implementó de esta manera, debido a que en caso de modificaciones a las transacciones de detalle, estas se realizarían no sobre todo el conjunto de transacciones sino sobre unas cuantas. El uso de AJAX permite una interface más amigable para el usuario.

Los detalles de las transacciones se presentan en líneas que contienen la información en campos de textos editables para que el usuario ingrese los valores a cambiar. Una vez que el usuario ha terminado de editar los valores presiona el botón que tiene la etiqueta modificar para que se llame a una función de JavaScript que invoca asincrónicamente al servlet *ServletFinanciero* y este se encarga de crear un objeto de tipo FinancieroDAO para que éste modifique los datos.

Como el manejo financiero es de vital importancia para la rama, este módulo le permite al usuario la creación de reportes de utilidad de la rama en general o dividir los reportes por cada capítulo si así se lo requiere.

La generación de estos reportes se lo hace por período de tiempo por lo que es necesario elegir el rango de fechas en el cual se quiere realizar el reporte.

La figura 5.13 muestra un ejemplo de un reporte de ingresos y egresos.



Reporte de Ingresos y Egresos

Rama Estudiantil IEEE - ESPOL

Ingresos	49.0		
	Origen	Valor	
	Auspicios	7.0	Dólares
	Origen	Valor	
	Eventos	42.0	Dólares
Egresos	-31.0		
	Origen	Valor	
	Compra equipos	-31.0	Dólares

Total Utilidad: 18.0

18/03/08 23:15 Page 1 of 1

Figura 5.13 Ejemplo de un reporte de utilidad

5.1.11. Módulo de Usuarios

Este módulo le permite al usuario de tipo Administrador General cambiar los datos de los miembros y usuarios que tengan almacenados sus datos en el sistema.

Los datos de los miembros que se pueden cambiar son los que se registran durante el proceso de ingreso como miembro IEEE de la rama. Se pueden cambiar todos los datos excepto lo que corresponde a las sociedades IEEE en las que se encuentra registrado, para este efecto se encuentra la renovación de las membresías en el módulo de membresías.

Los datos de los usuarios son los datos de las personas que se han registrado en los eventos y que pueden cambiarlos por actualización o por mal ingreso de los mismos a la hora de haberse registrado.

Para modificar los datos previamente se realiza una búsqueda del usuario. Dicha búsqueda se hace mediante el ingreso de las iniciales del apellido de la persona y al ejecutar la acción se ejecuta una función de JavaScript que invoca al servlet llamado *ServletUsuarios* que realiza la consulta a la base de datos haciendo uso de una instancia de la clase UsuarioDAO que implementa un método para obtener los datos de los usuarios. Una vez que se listan los usuarios se escoge el que se quiere actualizar y se invoca al mismo servlet para que obtenga todos

los datos y los guarde en el objeto request y puedan luego ser entregados a la página correspondiente. Una vez entregados a la página esta se encarga de visualizar los datos y de colocarlos en modo editable para que se modifiquen aquellos datos que requieran ser modificados. Una vez que se ha terminado la edición los datos son entregados nuevamente al servlet para que este invoque nuevamente a la instancia de la clase UsuarioDAO y llame el método que actualiza los datos del usuario en la base de datos.

5.2. Uso de AJAX en las interfaces del sistema

5.2.1. Tablas Dinámicas

La creación de tablas dinámicas nos permite mostrar en forma tabular nuevos datos ingresados en diferentes lugares de la aplicación de acuerdo a alguna opción escogida por el usuario. Un ejemplo de este uso en la aplicación, es la creación de tablas de conferencias ingresadas durante el proceso de creación de un evento.

Dado que un evento puede tener varias conferencias, el usuario necesitaría varias páginas para la creación del mismo. En una página ingresaría los datos del evento y en otro conjunto de

páginas ingresaría la información de cada conferencia con sus respectivos conferencistas.

Para evitar la utilización de tantas páginas diferentes se utilizó AJAX en conjunto con los métodos provistos por el DOM para ingresar los datos del evento y de cada una de sus conferencias de una manera asincrónica. Después que se ingresa cada conferencia se genera una tabla dinámica que muestra las conferencias a medida que van siendo asignadas a un evento. Toda esta funcionalidad se logra en una sola página facilitando al usuario la interacción con el sistema.

La figura 5.14 muestra el formulario de ingreso de las conferencias para un evento. En la parte inferior se puede apreciar la tabla dinámica con el nombre de una conferencia que ha sido previamente ingresada en el sistema.

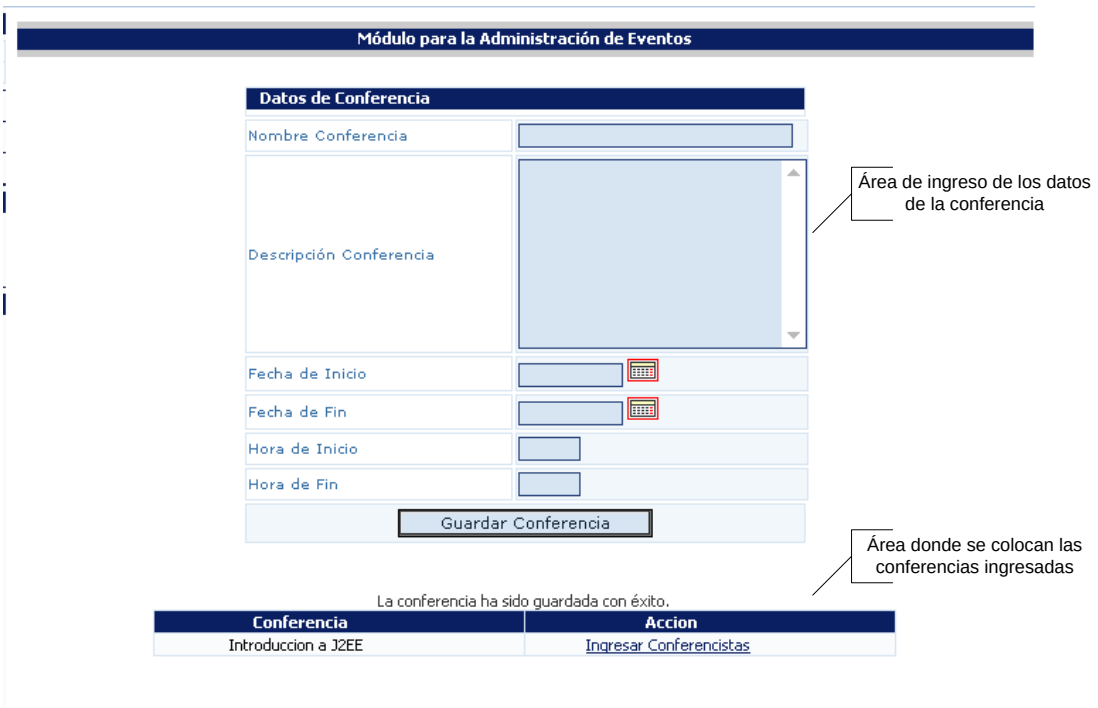


Figura 5.14 Creación de tablas dinámicas

Para formar la tabla dinámica del ejemplo anterior la aplicación hace uso de funciones JavaScript que de manera asincrónica llaman a la ejecución del servlet llamado *ServletEventos*. El servlet ejecuta las acciones necesarias para ingresar los datos de la conferencia y luego obtiene todas las conferencias ingresadas para formar un XML que será devuelto a la función de JavaScript que llamó al servlet para que ésta procese el XML.

A continuación se presenta la estructura del XML formado por el servlet.

```
<conferencia>
  <idevento>1</idevento>
  <idconferencia>1</idconferencia>
  <nombreconferencia>Voz sobre IP</nombreconferencia>
  <descripcionconferencia>Tecnologia para la transmision de voz sobre el protocolo de HTTP</descripcionconferencia>
  <fechainicio>28/02/2008</fechainicio>
  <fechafin>28/02/2008</fechafin>
  <horainicio>10:00</horainicio>
  <horafin>12:00</horafin>
</conferencia>
```


Figura 5.15 XML que contiene los datos de conferencias ingresadas

Una vez que el XML es retornado, mediante propiedades del DOM obtenemos los elementos que queremos mostrar en pantalla y formamos la tabla dinámica.

A continuación se muestra una fracción de código donde vemos como mediante DOM podemos añadir objetos a la página web ya formada.

```
cadena += "<table class=\"usuario\">";
    cadena +="<tr>";
        cadena +="<td class=\"largeheader\">Cedula</td>";
        cadena +="<td class=\"mediumheader\">Nombre</td>";
        cadena +="<td class=\"largeheader\">Accion</td>";
    cadena +="</tr>";

for(loop=0; loop<conferencista.length; loop++)
{
    cadena +="<tr>";
        cadena +="<td class=\"medium\">"+conferencista[loop].childNodes[0].childNodes[0].nodeValue+"</td>";
        cadena +="<td class=\"large\">"+conferencista[loop].childNodes[1].childNodes[0].nodeValue;

    cadena +="</tr>";
}

cadena += "</table><br />";

document.getElementById("mensaje").innerHTML = "Los datos del conferencista han sido guardados con exito.";
document.getElementById("tablaconferencistas").innerHTML = cadena;
```

Figura 5.16 Código JavaScript que utiliza DOM para modificar una página y añadir una tabla dinámica.

5.2.2. Búsquedas Dinámicas

Las búsquedas de datos dentro de la aplicación son constantes ya que se necesita de ellas para efectuar operaciones sobre los mismos. Una búsqueda común dentro de una aplicación Web requiere una página con un formulario y luego una página donde se muestran los resultados de la búsqueda. Si bien es cierto este enfoque no es malo, no es menos cierto que dificulta el trabajo de realizar búsquedas de manera frecuente. AJAX resuelve este problema pues el resultado de las búsquedas se puede mostrar en la misma página y dado que se reciben los datos asincrónicamente no hay necesidad de refrescar la página entera.

Un ejemplo de este uso dentro de la aplicación es la búsqueda de miembros para renovar las membresías, donde la búsqueda se realiza por las iniciales del apellido y de manera asincrónica se devuelven los resultados a la página evitando el tener que recargar toda la página para mostrar los resultados.

La figura 5.17 muestra cómo funcionan las búsquedas con AJAX. Se introduce el texto que va a ser objeto de la búsqueda y presiona el botón buscar. Se devuelve un listado con las coincidencias de la búsqueda.

Figura 5.17 Ejemplo de búsquedas con AJAX

El usuario puede ingresar el número de caracteres que considere necesario para realizar la consulta de los miembros de la rama. Al presionar el botón buscar se invoca una llamada

asincrónica al servlet llamado *ServletMiembros* que recoge los caracteres enviados como parámetro. El servlet se encarga de instanciar la clase MiembroDAO de la cual se utiliza el método que busca a los miembros por el apellido. Al realizar la consulta y obtener los datos se arma la estructura XML que será devuelta a la función de JavaScript que invocó la acción del servlet.

La figura 5.18 muestra la estructura XML que construye el servlet.

```
<persona>
  <idpersona>0920999406</idpersona>
  <nombre>Luis Loor</nombre>
  <email>lloor@espol.edu.ec</email>
  <idmiembro>IEEE0001</idmiembro>
</persona>
```

Figura 5.18 XML que contiene los datos de la búsqueda de un usuario.

Otro tipo de búsqueda dinámica dentro de la aplicación es el autocompletar texto. La técnica de autocompletar nos permite cambiar dinámicamente una lista de elementos conforme se cambia el campo de texto que sirve para filtrar una búsqueda.

Un ejemplo del uso de esta técnica en la aplicación, es la búsqueda de Conferencistas para las conferencias que se dictan en un evento. A medida que se va ingresando las iniciales del apellido del conferencista, que es el campo por el cual se realiza la búsqueda, se van mostrando en una lista todos los posibles conferencistas que coinciden con la consulta que se ha enviado a realizar al servidor.

La figura 5.19 muestra la acción de autocompletar. Se observa el cuadro de texto donde se ingresan las iniciales del apellido del conferencista y a continuación, en la parte inferior del campo de texto, van apareciendo los nombres de los conferencistas cuyos apellidos coinciden con el texto ingresado.

Figura 5.19 Ejemplo de Auto completar en la aplicación

Cada vez que se ingresa una letra dentro del campo de texto, se ejecuta la acción onkeyup() definida para ese campo. Esta función envía el texto ingresado hacia el servlet llamado *ServletEventos* para que este se encargue de crear un objeto de tipo EventoDAO para que llamando al método que obtiene los datos, forme una estructura XML que es devuelta a la función onkeyup() para que forme la lista con los resultado de las coincidencias.

A continuación se muestra la estructura del documento XML.

```
<curriculumconferencista>
  <idpersona>0920999406</idpersona>
  <nombre>Luis</nombre>
  <apellido>Loor</apellido>
  <fechadenacimiento>23/02/1983</fechadenacimiento>
  <email>lloor@espol.edu.ec</email>
  <direccion>Alborada 1 etapa</direccion>
  <telefonocasa>2270697</telefonocasa>
  <telefonocelular>097791383</telefonocelular>
  <pais>Ecuador</pais>
  <cargo>Ingeniero de Desarollo</cargo>
  <empresa>Metromovil</empresa>
  <papers />
  <estudios>Tercer Nivel, ESPOL</estudios>
</curriculumconferencista>
```

Figura 5.20 XML que contiene los datos de un conferencista

5.2.3. Formularios Dinámicos

La creación dinámica de formularios permite obtener datos del servidor y presentarla al usuario a manera de un formulario.

Un ejemplo de este uso en la aplicación, es la creación de formularios dinámicos en el registro de personas en un evento. Una persona que ya se ha registrado en un evento, mantiene su información almacenada en la base de datos y la puede buscar para proceder con el registro en el evento; esta información que el usuario consulta, se muestra a manera de un formulario de sólo lectura para indicar al usuario que su información se encuentra disponible y que puede seguir con su registro en el evento.

La figura 5.21 muestra la creación de un formulario dinámico. El usuario ingresa su número de cédula y pulsa el botón buscar. En la parte inferior se muestra el formulario con información de la persona a la que pertenece el número de cédula que se ingresó.

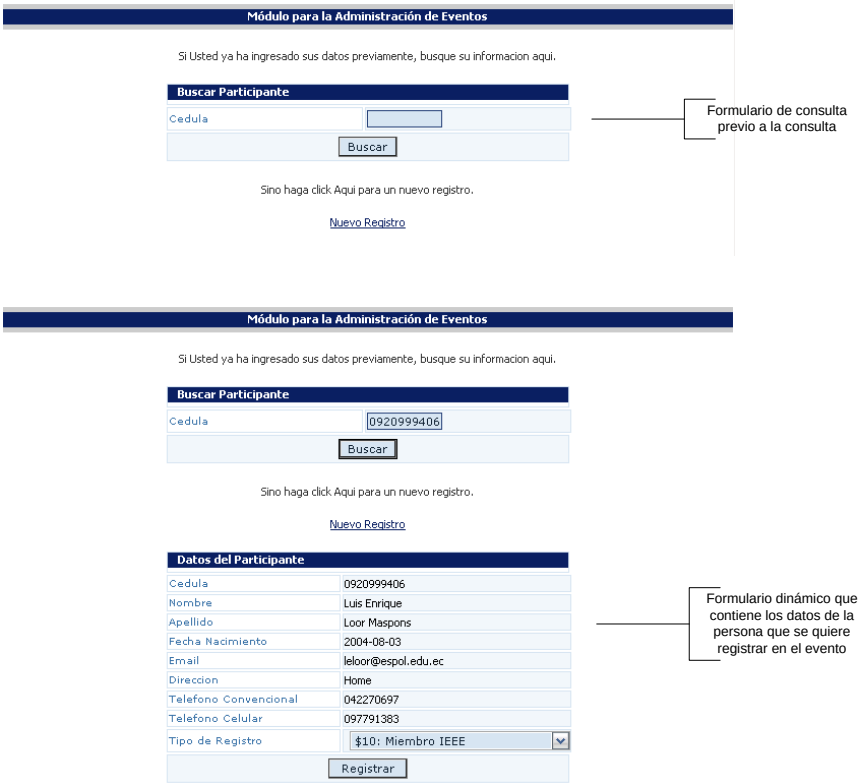


Figura 5.21 Creación de Formularios dinámicos

Los datos pueden haber sido almacenados en un registro de un evento previo o por que el usuario se ha registrado como miembro IEEE. Para la consulta el usuario ingresa su número cédula y pulsa sobre el botón buscar el cual llama a una función implementada en JavaScript que invoca la ejecución de un servlet de manera asincrónica. El servlet que se llama es el *ServletUsuario* que se encarga de instanciar la clase UsuarioDAO que implementa los métodos para el acceso a los datos de los usuarios. De la clase UsuarioDAO se llama al

método que busca un usuario por su cédula. Una vez obtenidos los datos el servlet arma la estructura XML que será devuelta a la función de JavaScript que invocó la ejecución del servlet.

La siguiente figura muestra la estructura XML que retorna el servlet.

```
<participante>
  <idpersona>0920999406</idpersona>
  <nombre>Luis</nombre>
  <apellido>Loor</apellido>
  <fechadenacimiento>23/02/1983</fechadenacimiento>
  <email>lloor@espol.edu.ec</email>
  <direccion>Alborada 1 etapa</direccion>
  <telefonocasa>2270697</telefonocasa>
  <telefonocelular>097791383</telefonocelular>
</participante>
```

Figura 5.22 XML que contiene los datos de un participante.

Cuando la función de JavaScript recibe el XML, elimina el formulario de ingreso de los datos y en su lugar, valiéndose de funciones del DOM crea un formulario donde los campos son de sólo lectura con lo cual el usuario confirma que sus datos ya se encuentran en el sistema y puede proceder con su registro.

5.2.4. Dobles Combos

Los combos son muy utilizados en las aplicaciones Web y nos sirven para seleccionar información. El ejemplo típico del uso de

dobles combos es un combo principal con ciertos países y otro combo donde se cargarán las provincias dependiendo del país que se encuentre seleccionado en el primer combo.

La estrategia del uso de dobles combos usando *AJAX* nos permite cambiar dinámicamente el contenido de un ComboBox de acuerdo al cambio que se haya producido en otro sin necesidad de refrescar la página completa.

Esta técnica se usa en la aplicación, en el módulo de Eventos durante el proceso de creación de un nuevo evento. Para cualquiera de los capítulos de la rama, existe la opción de escoger el área de estudio y el sub-área de estudio a la cual va estar dirigidas las conferencias del evento. Esta opción de escoger el área y una sub-área de un evento se ha implementado usando la técnica de doble combo con *AJAX*.

Al cambiar la opción seleccionada en el combo que contiene las áreas del evento se invoca una función implementada en JavaScript que de manera asincrónica envía como parámetro el identificador del área del evento. El Servlet llamado es el *ServletEventos* quien crea una instancia de la clase EventoDAO que permite el acceso a datos. La clase EventoDAO invoca al

método que busca las sub-áreas que están relacionadas con un área de estudio. Cuando se han obtenido los datos, con ellos se forma una estructura XML que se devuelve a la función de JavaScript que invocó originalmente al *Servlet*.

La siguiente figura muestra la estructura XML donde se encuentran los datos de las sub-áreas de un evento.

```
<subarea>
  <idsubarea>1</idsubarea>
  <idarea>2</idarea>
  <nombresubarea>Programacion orientada a objetos</nombresubarea>
</subarea>
```

Figura 5.23 XML con los datos de sub-áreas de un evento.

Para colocar los datos de las sub-áreas se hace uso de métodos del DOM para obtener una referencia al ComboBox, limpiar la información que haya tenido previamente y finalmente cargar los datos del XML obtenido.

A continuación se muestra una fracción de código que visualiza la utilización de los métodos del DOM para agregar la información al combo box de las sub-áreas de un evento.

```
var subarea = responseXML.getElementsByTagName("subarea");
var select = document.getElementById("opcSubAreaEvento");
select.disabled = false;
clearElement(select);

for(loop=0; loop<subarea.length; loop++)
{
    var opc = document.createElement("OPTION");
    opc.text = subarea[loop].childNodes[2].childNodes[0].nodeValue;
    opc.value = subarea[loop].childNodes[0].childNodes[0].nodeValue;
    select.options.add(opc);
}
```

Figura 5.24 Código JavaScript que manipula un combobox mediante los métodos del DOM.

La figura 5.25 muestra dos imágenes. En la primera se pueden apreciar el listado de las áreas de estudio (sin seleccionar ninguna) y el listado de las sub-áreas de estudio (completamente vacío). En la segunda imagen se muestra seleccionada el área de computación y se observa que el listado de las sub-áreas se llenado con los valores correspondientes.

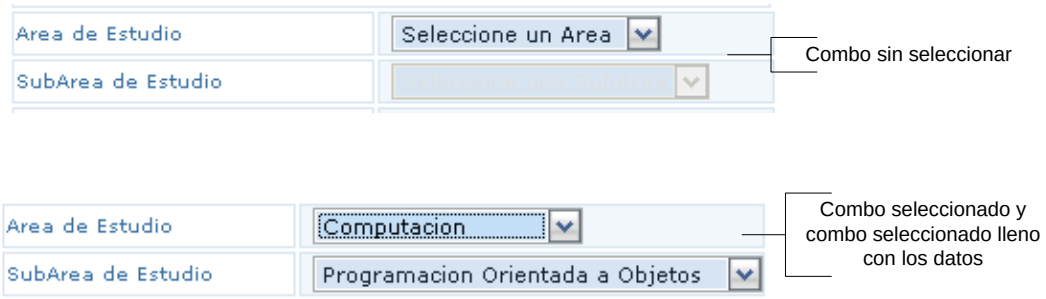


Figura 5.25 Dobles Combos con Ajax

5.3. Plataforma

5.3.1. Hardware

Para el desarrollo del proyecto de tesis se utilizaron los siguientes recursos de hardware:

- Procesador Intel Pentium 4 de 2.4 Ghz.
- Memoria RAM de 512 MBytes.
- Disco duro de 40 GBytes.
- Tarjeta de red.

Para el desarrollo del sistema se necesitan al menos los siguientes requerimientos de disco duro para alojar las aplicaciones.

- 7 MB de espacio en disco para el servidor web Apache Tomcat

- 36 MB de espacio en el disco para el Enterprise Manager de MySql
- 200 MB de espacio en el disco para el motor de base de datos MySql.
- 200 MB de espacio en el disco para Eclipse versión 3.2
- 240 MB de espacio en el disco para la instalación de MyEclipse versión 5
- 50 MB de espacio en el disco para JasperReports e IReports.

Sumando el espacio requerido por estas aplicaciones tenemos un total de 733 MB que es lo que debemos tener de espacio libre en nuestro disco duro.

Acerca de las herramientas de desarrollo hablaremos en la siguiente sección.

En cuanto a la utilización de recursos tenemos que en procesamiento y requerimientos de memoria RAM, Eclipse es la aplicación va a ocupar mayor parte de la memoria RAM (alrededor del 45% según el administrador de tareas de Windows XP). La memoria restante será ocupada por las demás aplicaciones. Solo entre Eclipse, Apache y MySQL

tendremos ocupado cerca del 95% de la memoria por lo que abrir otros programas puede hacer que la computadora de desarrollo se vuelva lenta.

5.3.2. Software

Para el desarrollo del proyecto se usaron las siguientes herramientas de software:

- Servidor Web Apache Tomcat v 5.5.23
- Motor de base de datos MySQL Server 5.0
- EMS (Enterprise Manager) Interfaz visual para manejo de base de datos MySQL
- Eclipse 3.2 como entorno de desarrollo.
- Jasper Reports

A continuación describiremos cada una de las herramientas de software utilizadas.

Servidor Web Apache Tomcat

Apache Tomcat es sin lugar a duda unos de los contenedores Web más populares que existen como contenedor de servlets y JSP. La elección de Tomcat a pesar de existir otras soluciones para los mismos requerimientos, se basa en que este es un

software de libre distribución fácilmente integrable con varios entornos de desarrollo, además su popularidad hace que en la actualidad las últimas versiones de Tomcat tengan un gran rendimiento y que existan tanto versiones para Linux como para Windows.

La figura 5.26 muestra la página de inicio de Tomcat cuando se ha instalado correctamente en el servidor.

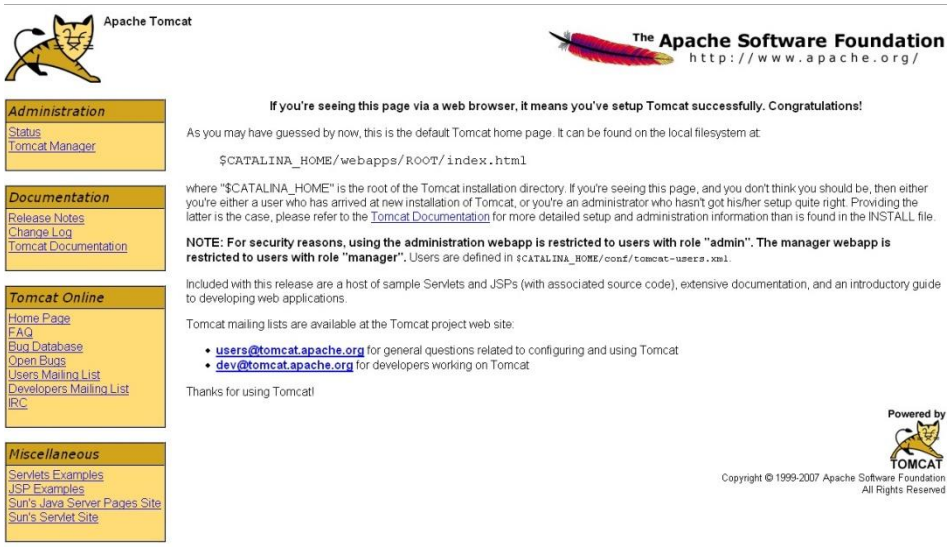


Figura 5.26 Servidor Web Apache Tomcat

Motor de base de datos MySQL Server 5.0

Para el almacenamiento de datos se escogió MySQL Server 5.0 ya que es un software de libre distribución.

Además de ser de libre distribución MySQL tiene otras características que lo convierten en un motor de base de datos muy utilizado en aplicaciones Web que se desarrollan bajo J2EE.

Entre las ventajas de usar MySQL están:

- Es software libre.
- Es altamente escalable tanto en la cantidad de datos que puede almacenar (orden de los terabytes por tabla) como en el número de usuarios concurrentes que puede atender.
- Es un motor de base de datos que viene tanto en versiones para Linux como para Windows lo que lo hace portable entre plataformas.
- A partir de la versión 5.0 se incluye la creación de procedimientos almacenados.

EMS (Enterprise Manager) Interfaz visual para manejo de base de datos MySQL

Si bien es cierto MySQL es un software gratuito, este hecho hace que no ofrezca todas las facilidades al usuario en cuanto a la programación ya que se la hace a través de consola lo cual

no facilita el trabajo. Sin embargo existen soluciones que integran MySQL a una interfaz visual que hace más fácil y amigable la administración de base de datos.

Para este fin tenemos EMS que ofrece a los usuarios una interfaz amigable para realizar operaciones de administración sobre una base de datos MySQL.

La figura 5.27 muestra la pantalla de inicio de EMS.

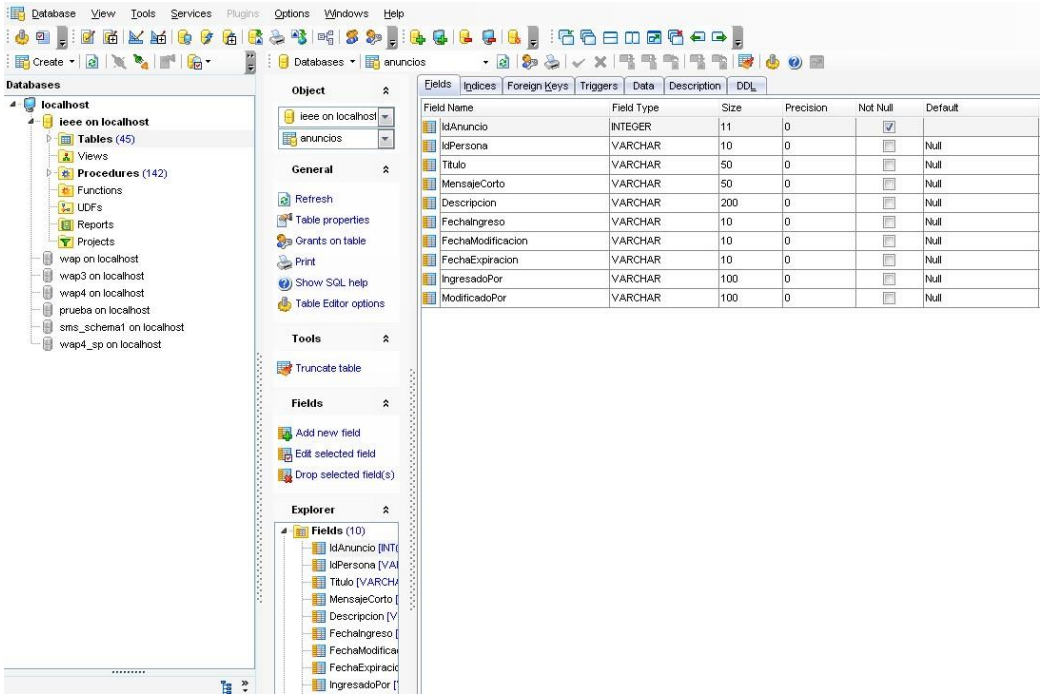


Figura 5.27 EMS como interfaz gráfica para MySQL

Eclipse 3.2

Existen muchos entornos de desarrollo que permiten el desarrollo de aplicaciones Web entre ellas tenemos NetBeans, WebSphere, Eclipse, JDeveloper entre otros.

Eclipse es un IDE diseñado para crear aplicaciones en Java y es ampliamente difundido entre los programadores de aplicaciones Web ya que tiene a su disposición un sin número de plugins que le otorgan mayor funcionalidad.

Eclipse es de libre distribución por lo que cualquiera puede descargarlo y usarlo. Cualquier versión de Eclipse que uno descargue no trae consigo un plugin para el desarrollo de aplicaciones Web por lo que hay que descargar uno e instalarlo. Para J2EE existen varios plugins pero se decidió usar uno que se llama MyEclipse v 5.0.

MyEclipse v 5.0 trae muchas utilidades para J2EE siendo las más importantes un validador de Javascript, validador de DOM, pre visualización de HTML en Internet Explorer y Firefox y un validador de requerimientos AJAX, siendo este último de gran

utilidad ya que como se ha explicado a lo largo de este documento, el sistema RIEEEWEB usa *AJAX* en muchas partes de la aplicación.

Otra opción para escoger Eclipse como entorno de desarrollo es que se puede integrar con el servidor Web Apache Tomcat de manera que se puede configurar la aplicación usando el mismo IDE

La figura 5.28 muestra el entorno de desarrollo de Eclipse.

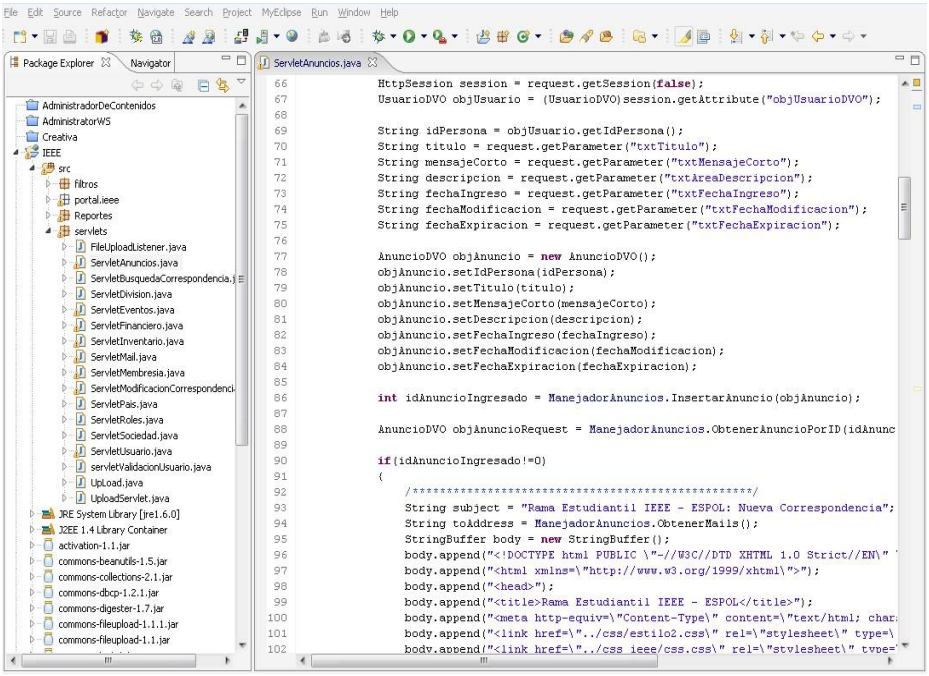


Figura 5.28 Entorno de desarrollo Eclipse con plugin MyEclipse

Jasper Reports

Jasper Reports es una herramienta desarrollada en Java que permite la creación de informes en diferentes tipos de formato como PDF, CVS, HTML, XML entre otros [15].

Con Jasper Reports se pueden realizar consultas a una base de datos y presentar los datos obtenidos a manera de informe en cualquiera de los formatos soportados. Para poder trabajar de una manera sencilla con Jasper Reports existe un editor visual llamado iReports [16] que permite de una manera fácil la creación y edición de los reportes complejos como gráficos de barra, imágenes y subreportes. Tanto Jasper Reports e iReports son herramientas de libre distribución.

A continuación se muestra una imagen de la edición de un reporte de Jasper Reports con el editor visual de iReports.

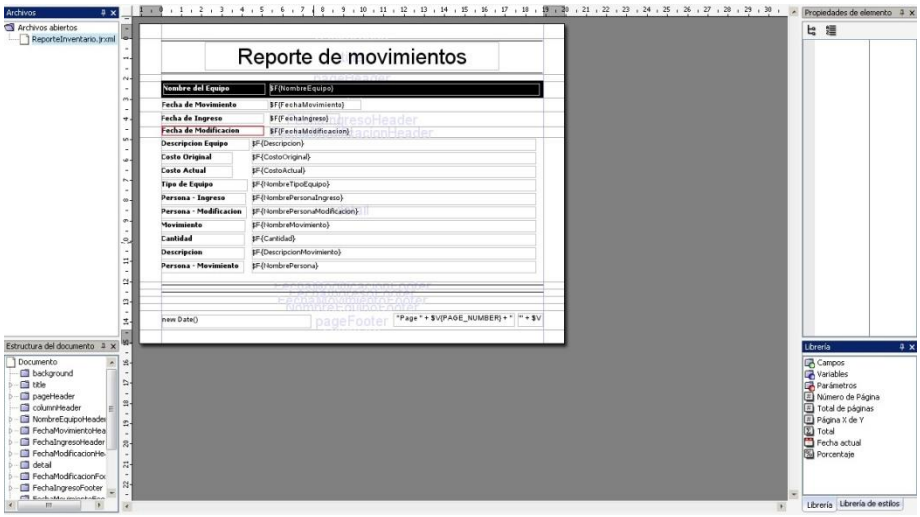


Figura 5.29 Entorno gráfico para la creación de los reportes.

5.4. Seguridad

5.4.1. Autorización

El acceso a los recursos del sistema está limitado según los roles de cada usuario, de esta manera se protege al sistema de acceso no autorizados a recursos.

El sistema consta de recursos que son privados y otros que son públicos. Los recursos que son públicos pueden ser accedidos por cualquier visitante al portal, mientras que los recursos privados están restringidos a los miembros de la rama que cuentan con un usuario y contraseña en el sistema.

Además de proteger los recursos que son privados mediante el acceso con un usuario y contraseña, al momento de almacenar

la contraseña del usuario en la base de datos, ésta no se guarda como un texto plano, sino que se le aplica una función de encriptación llamada MD5 que cifra dicha clave y guarda el texto cifrado. Así, si alguien tiene acceso a la base de datos no puede usar la contraseña de otra persona para entrar al sistema puesto que está encriptada.

EL sistema para autorizar el acceso a un recurso privado, verifica que el usuario haya iniciado sesión con su usuario y contraseña. Si cualquier usuario intenta acceder a un recurso mediante al dirección del recurso (ejemplo http://dominio/usuario/modificarUsuario.jsp) sin haber ingresado al sistema, el sistema identifica esto como un acceso no autorizado y redirige la petición hacia la página de ingreso al sistema.

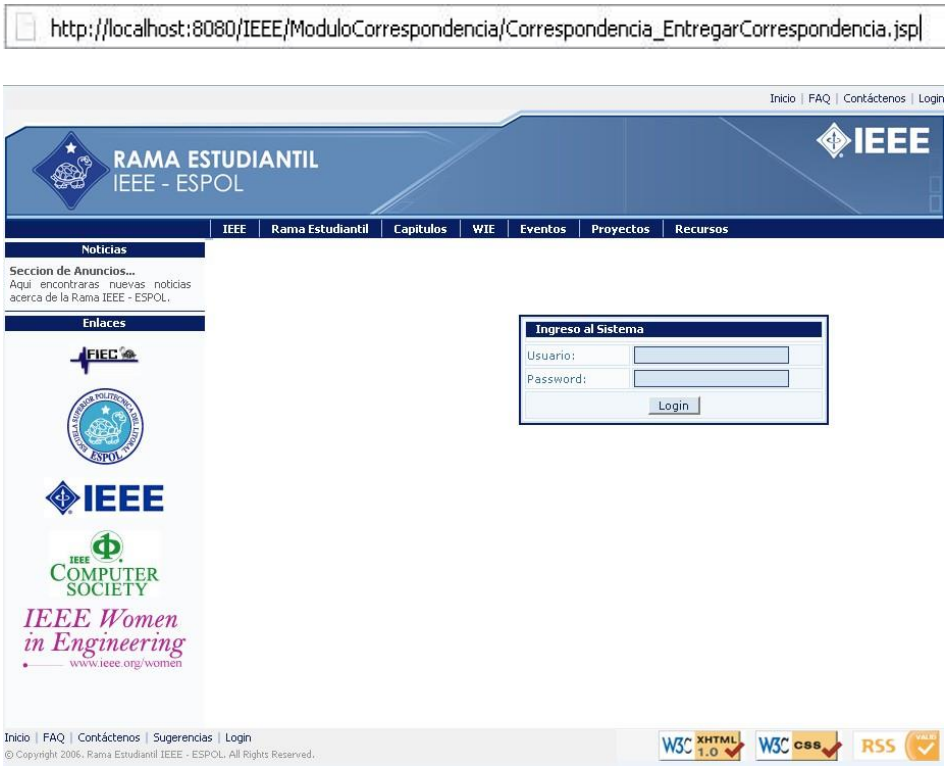


Figura 5.30 Página de ingreso del sistema.

Como se puede apreciar en la figura 5.30 al querer ingresar a una página directamente sin haber iniciado sesión, el sistema no autoriza el acceso al recurso y se envía al usuario a la página de ingreso.

Así mismo si se intenta acceder a un recurso y el rol del usuario no permite visualizar dicho recurso, el sistema muestra un mensaje indicando que no se puede visualizar el rol.

Figura 5.31 Mensaje de Recurso no autorizado.

5.4.2. Roles

En el sistema se definieron los siguientes roles:

- Administrador General
- Administrador de Correspondencia y Membresías
- Administrador Financiero
- Publico

A continuación se detallan los privilegios asignados a cada uno de los roles.

Administrador General: Este rol del sistema le permite al usuario tener acceso a las siguientes funcionalidades:

- Módulo de Roles: Este módulo le permite crear nuevos roles en el sistema de acuerdo a las opciones hábiles en el sistema y las necesidades de la Rama Estudiantil y su voluntariado.

- Módulo de Sociedades: Este módulo le permite al usuario ingresar nuevas sociedades de la IEEE para ofrecer a los usuarios.
- Módulo de Divisiones: Este módulo le permite al usuario ingresar nuevos capítulos a la Rama según la expansión de la misma.
- Módulo de Eventos: Este módulo permite la creación de eventos para que puedan ser vistos por los visitantes del sistema y puedan también registrarse en los mismos.

Administrador Financiero: Este rol del sistema le permite al usuario que lo posea tener acceso a las siguientes funcionalidades:

- Módulo Financiero: Este módulo le permite al usuario registrar todos los ingresos y egresos de la Rama para así tener un mejor control de los recursos necesarios para que la Rama IEEE – ESPOL funcione a la normalidad.
- Módulo de Inventarios: Este módulo le permite al usuario llevar un control de todos los recursos físicos que posee

la Rama, tanto de los existentes así como de aquellos que se van adquiriendo.

Administrador de Correspondencia y Membresías: Este rol del sistema le permite al usuario que lo posea tener acceso a las siguientes funcionalidades:

- Módulo Membresías: Este módulo permite que el usuario puede registrar a nuevos usuarios en las sociedades IEEE que ofrece la Rama; así como la renovación de miembros ya registrados.
- Módulo de Anuncios: Este módulo le permite al usuario publicar noticias en el sitio Web de la Rama, así los visitantes podrán estar al tanto de las últimas novedades de la Rama.
- Módulo Correspondencia: Este módulo permite al usuario registrar nueva correspondencia que la IEEE envía a sus miembros, de esta manera cada miembro tiene conocimiento de su correspondencia a través del sistema.

Publico: Este rol básicamente no tiene ninguna de las funcionalidades descritas anteriormente ya que está destinado a usuarios que no son miembros voluntarios de la rama. Este rol del sistema sólo tiene activo la sección en la parte izquierda del sistema donde puede revisar sus datos personales, de membresía y su correspondencia personal; por lo demás el menú de opciones que muestra la aplicación es el mismo con el cual inicia.

5.4.3. Filtros

En muchas ocasiones nos encontramos con acciones que son repetitivas y deben realizadas por cada página JSP o servlet que tengamos en nuestra aplicación, como por ejemplo el control de seguridad de la aplicación.

Lo óptimo sería encontrar una manera de que el contenedor de servlets (en nuestro Tomcat) antes de pasar el control al servlet o la página JSP intercepte las peticiones y ejecute dichas acciones repetitivas. Esta es la función de los filtros.

Los filtros [17] no son más que clases que ejecutan acciones antes de pasar la petición al servlet. Cuando una acción llega a

un servlet se ejecutan los filtros (configurados en la aplicación) antes de pasar el control al servlet. Una vez que el filtro captura la petición, éste puede acceder y modificar al objeto request y el objeto response, lanzar excepciones o continuar con la ejecución de otros filtros.

Lo único que debemos hacer para llamar a un filtro es crear la clase de Java que implemente de la interfaz Filter y configurarlo en el archivo web.xml de la aplicación. En este archivo se realiza un mapeo ente los filtros que se invocarán y los servlets o urls relacionados a esos filtros. A continuación se muestra la configuración del filtro llamado *FilterAutenticacion* en el archivo web.xml de la aplicación.

```
<filter>
    <filter-name>FilterAutenticacion</filter-name>
    <filter-class>filtros.FilterAutenticacion</filter-class>
</filter>
<filter-mapping>
    <filter-name>FilterAutenticacion</filter-name>
    <url-pattern>/ModuloAnuncios/*</url-pattern>
</filter-mapping>
```

Figura 5.32 Configuración de un filtro en el archivo web.xml

Como podemos ver en la figura 5.32 la configuración de un filtro consta de dos partes: la declaración del filtro y el mapeo del recurso a filtrar.

La declaración va dentro de la etiqueta <filter>. Dentro de esta etiqueta tenemos dos etiquetas más <filter-name> y <filter-class>, la primera contiene el nombre del filtro y la segunda la clase de Java que implementa el filtro.

El recurso a filtrar está dentro de la etiqueta <filter-mapping>. Dentro de esta etiqueta también tenemos dos etiquetas más <filter-name> y <url-pattern>. La primera etiqueta es el nombre del filtro y la segunda contiene la ruta del recurso al cual se va a aplicar el filtro.

En el ejemplo de la figura 5.30 vemos que el nombre del filtro es FilterAutenticacion y que se va a aplicar a los recursos que están dentro de la carpeta /ModuloAnuncios.

En RIEEEWEB se utiliza un filtro que por cada llamada a una página JSP verifica si dicha página está entre los accesos que tiene el usuario puesto que un usuario podría querer ingresar a páginas que se encuentran restringidas para el rol que posee.

El filtro actúa de manera que cada vez que se invoca a una página JSP, se verifican los accesos que tiene el usuario de acuerdo a su rol y si la página solicitada está permitida. Si el permiso es concedido se permite la visualización de la página

solicitada; caso contrario se muestra una página indicando al usuario que el recurso solicitado no está autorizado para ser visto

La figura 5.33 muestra el funcionamiento del filtro para la verificación de los accesos a páginas no autorizadas.

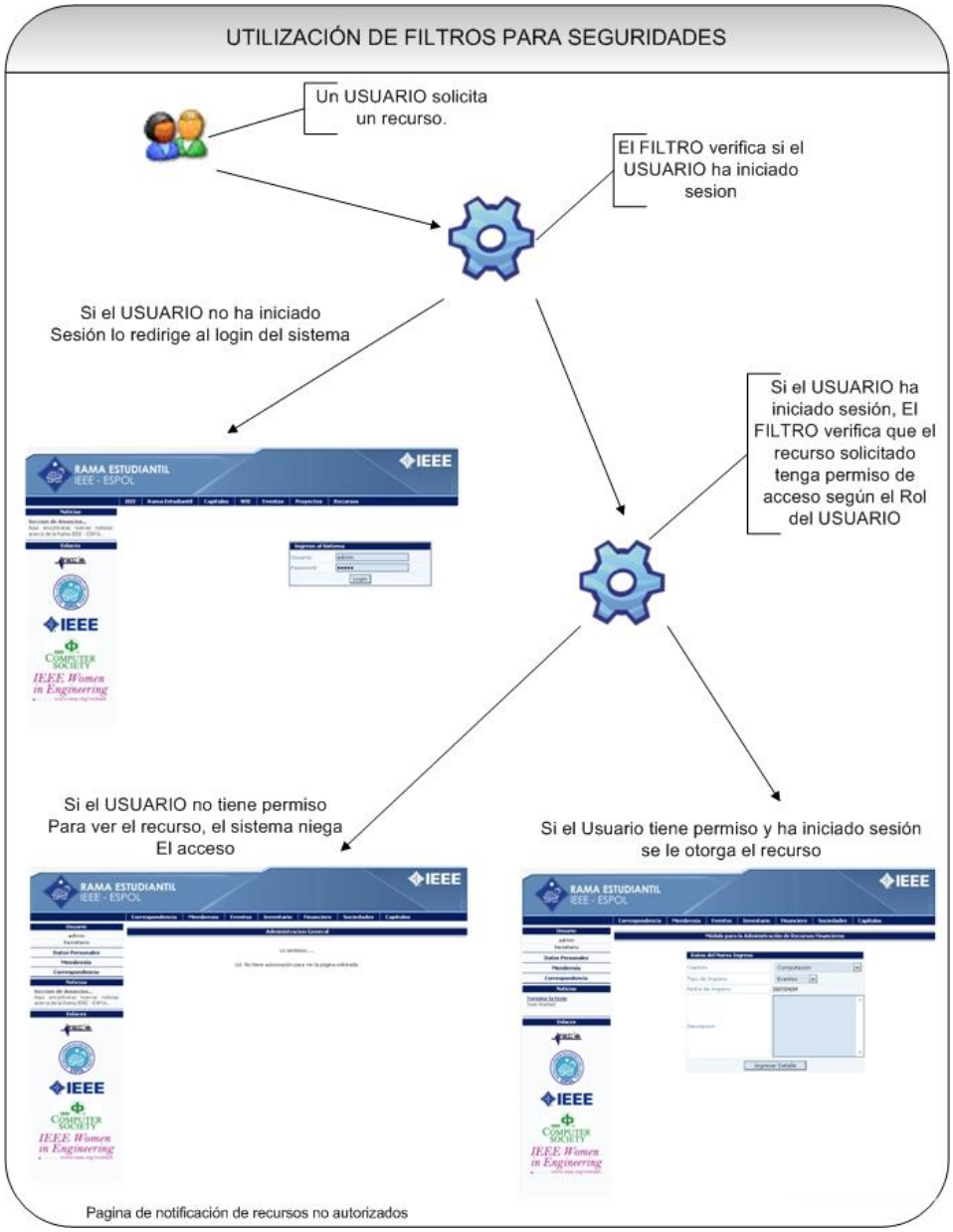


Figura 5.33 Funcionamiento un filtro

5.5. Configuraciones

Existen ciertas configuraciones que hay que realizar para el correcto funcionamiento de la aplicación:

- Para el envío de correo electrónico hay que copiar las librerías mail.jar y activation.jar en el directorio common/lib de Tomcat.
- Para la generación de reportes es necesario copiar dentro del directorio WEB-INF/lib de la aplicación las librerías itext.jar y jasperreports-1.2.8.jar. Además de esto es necesario tener definido un directorio donde van a ser colocados los archivos de Jasper Reports de los reportes para que la aplicación los pueda leer.
- Para la subida de archivos es necesario copiar las librerías commons-fileupload-1.1.jar, commons-logging.jar y fileupload-ext.jar en directorio WEB-INF/lib de la aplicación.

5.6. Pruebas

5.6.1. Plan de Pruebas

Para realizar las pruebas del sistema se establecieron algunos escenarios que reflejaran las funcionalidades del sistema para que puedan ser evaluadas por los miembros de la Rama estudiantil que probaron el sistema. Para las pruebas del sistema se tomaron en cuenta los procesos más críticos como son la publicación de eventos, registros de miembros, la publicación de correspondencia y el ingreso de valores económicos.

A continuación se describen los escenarios para las pruebas.

Escenario de prueba 1: Publicación de Eventos

- El usuario tiene que ingresar al sistema con el rol de Administrador General.
- En el menú de Eventos elegir la opción de ingresar eventos.
- En la página de ingreso de los datos del evento, ingresar la información del evento.
- En la página de ingreso de conferencias, ingresar la información de las conferencias.
- En la página de ingreso de conferencistas, ingresar la información de los conferencistas.

Escenario de prueba 2: Registro de un participante en un evento.

- En la página principal, buscar el menú Eventos y elegir entre los varios capítulos de la rama.
- En la página de los eventos del capítulo, elegir de entre los eventos disponibles.

- En el formulario de registro ingresar los datos personales y la información requerida en el formulario.

Escenario de prueba 3: Confirmación de Registros.

- El usuario debe ingresar al sistema con rol de Administrador General.
- En el menú Eventos buscar la opción de Confirmar registros.
- En la página de Confirmación de registros buscar los eventos por el año.
- Entre los eventos encontrados elegir de cual se van a confirmar los registros.
- De entre los participantes registrados confirmar algunos participantes.

Escenario de prueba 4: Ver reportes de Eventos.

- El usuario debe ingresar al sistema con rol de Administrador General.
- En el menú Eventos buscar la opción de Ver Reportes.
- Buscar los eventos por el año.
- Seleccionar la opción ver reportes de participantes.
- Seleccionar la opción ver reportes de ingresos del evento.

Escenario de prueba 5: Registro de miembros en la Rama Estudiantil.

- El Usuario debe ingresar al sistema con rol de Administrador de Correspondencia y Membresías.
- En el menú de Membresías buscar la opción de ingresar miembros.
- En la página de registro llenar el formulario de ingreso con los datos personales.
- Una vez completado los datos de ingreso proceder a pulsar el botón ingresar para finalizar el proceso.

Escenario de prueba 6: Reportes de Miembros de la rama

- El usuario debe ingresar al sistema con rol de Administrador de Correspondencia y Membresías.
- En el menú de membresías buscar la opción ver reportes
- Escoja la opción ver reportes de miembros activos
- Escoja la opción ver reportes de miembros inactivos.

Escenario de prueba 7: Ingresar Correspondencia

- El usuario debe ingresar al sistema con el rol de Administrador de Correspondencia y Membresías.

- En el menú de correspondencia buscar la opción de Ingresar correspondencia.
- En el formulario de ingreso, llenar los datos requeridos.
- De entre las lista de usuarios que aparece, escoger a aquellos a quienes se les quiere asignar la correspondencia.
- Para finalizar dar clic en “Ingresar”.

Escenario de prueba 8: Entregar correspondencia.

- El usuario debe ingresar al sistema con el rol de Administrador de Correspondencia y Membresías.
- Buscar al miembro IEEE al que se le va a entregar la correspondencia.
- De entre el listado de la correspondencia aquella que se va a entregar y haga clic en “No Entregado”.

Escenario de prueba 9: Registrar Ingreso Económico.

- El usuario debe ingresar al sistema con el rol de Administrador Financiero.
- En el menú de Financiero buscar la opción de “Nuevo Ingreso”.

- En el formulario de ingreso, agregar la cabecera de información.
- Agregar tantos detalles del ingreso como se necesite.

Escenario de prueba 10: Generar reporte de Utilidades.

- El usuario debe ingresar al sistema con el rol de Administrador Financiero.
- En el menú de Financiero buscar la opción de ver reportes.
- De entre las opciones elegir “Ver reporte de utilidades”.

5.6.2. Resultado de las Pruebas

A continuación se presentarán los resultados obtenidos luego de hacer las pruebas en el desarrollo de los módulos.

Resultado de las pruebas del escenario de prueba 1

Prueba número:	1
Nombre de la prueba:	Publicación de un evento.
Número de Intento	**Resultados**
1	Para grabar un evento es necesario ingresar el área y sub área del evento, nombre del evento, descripción, capítulo de la rama que organiza el evento, las fechas de inicio y fin, las horas de inicio y fin, el cupo máximo de personas registradas y los valores que cada tipo de participante al evento tiene que cancelar.

	Al momento de guardar los datos, los campos correspondientes a la descripción del evento y las horas de inicio y fin del evento no fueron llenados por lo que el sistema alertó al usuario de que dichos campo no pueden quedar vacíos.
2	Se llenaron los campos de la hora de inicio y fin del evento con los valores 8 y 12, lo cual corresponde a un formato incorrecto por lo que el sistema notificó al usuario que debía ingresar la hora en el formato (hh:mm)

Prueba número:	2
Nombre de la prueba:	Publicación de conferencias.
Número de Intento	**Resultados**
1	Para ingresar conferencias es necesario llenar la siguiente información. • Nombre de la conferencia • Descripción • Fecha y hora de inicio y fin de la conferencia Los campos de la hora de inicio y fin fueron llenados con los valores 08 y 10, lo cual corresponde a un formato incorrecto por lo que el sistema notificó que se debía ingresar la hora en el formato (hh:mm)

Prueba número:	3
Nombre de la prueba:	Asignación de conferencistas a las conferencias de los eventos.
Número de Intento	**Resultados**
1	Para asignar un conferencista hay que llenar los siguientes campos: • Datos Personales (Común para cualquier transacción de registro). • País • Cargos • Empresa donde labora • Estudios realizados • Papers publicados • Archivo con su curriculum y una foto. Cuando un conferencista ha sido ingresado

	previamente, estos datos están disponibles y no es necesario volverlos a ingresar, solamente es necesario buscar al conferencista por las iniciales de su apellido y todos sus datos serán cargados al sistema. Existe una opción de autocompletar donde se busca a los conferencistas, esta opción no fue identificada durante el ingreso de los conferencistas.
2	De un listado de conferencistas previamente ingresados en el sistema, se pidió al usuario que buscara uno en particular (Ing. Cristina Abad). El usuario ingresó el nombre del conferencista para buscarlo con lo cual no se mostraron coincidencias puesto que la búsqueda se la realiza por el apellido del conferencista.
3	Al registrar todos los datos que son necesarios para asignar conferencistas, se ingresó una dirección de correo inválida en el campo que corresponde al email de contacto del conferencista. El sistema notificó al usuario que la dirección de correo no era correcta.

Prueba número:	4
Nombre de la prueba:	Confirmación de registros.
Número de Intento	**Resultados**
1	El usuario vio el listado de los participantes registrados en un evento. Confirmó el registro de algunos y pudo ver la credencial del participante. Sin embargo no sabía qué hacer con ella puesto que no había una impresora disponible para imprimir. Se guardó un archivo con las credenciales de los registros confirmados en una carpeta temporal.

Resultado de las pruebas del escenario de prueba 2

Prueba número:	1
Nombre de la prueba:	Registro de participantes en un evento
Número de Intento	**Resultados**
1	El usuario primero intentó buscar sus datos por

	medio de su cédula de identidad. No se encontraron sus datos registrados.
2	Para el registro de los datos del participante s necesario ingresar los siguientes datos: • Datos Personales (Común para todo tipo de transacción de registro). • Si trabaja, los datos de la empresa donde labora. En los datos de la empresa no es necesario ingresar el RUC de la empresa, pero si se lo ingresa éste debe tener 13 dígitos. El usuario ingresó los datos personales correctamente, sin embargo en los datos de la empresa donde labora ingresó un RUC de menos de 13 dígitos (XXXXXXXXX001), por lo cual el sistema notificó al usuario que el RUC debe tener 13 dígitos.

Resultado de las pruebas del escenario de prueba 3

Prueba número:	1
Nombre de la prueba:	Confirmación de registros
Número de Intento	**Resultados**
1	El usuario buscó los eventos que se habían publicado en el presente año digitando el año en formato de 4 dígitos (2008) e intentó realizar la consulta. El sistema no realizó la consulta e indicó al usuario que tenía que escoger un capítulo de la rama para buscar los eventos.
2	El usuario buscó el evento tanto por el año de publicación como por el capítulo de la rama. En la lista de participantes registrados, procedió a confirmar los registros de algunos participantes dando clic en el link correspondiente.

Durante la realización de las pruebas en este escenario se sugirieron pequeños cambios a realizar al sistema. A continuación se detalla el cambio sugerido para este escenario de prueba.

Cuando se realiza la confirmación de un registro, el sistema da la opción imprimir la credencial del participante. Esto se lo realizaba por cada participante al cual se le confirmaba el registro. El cambio solicitado es que ahora solo existirá un solo botón que imprimirá todas las credenciales de los participantes a los cuales se les haya confirmado el registro.

La figura 5.34 muestra como se realizaba la impresión de las credenciales antes del cambio solicitado.

Nombre	Estado	Modificar	Confirmar Registro
Luis Enrique Loor Maspons	Confirmado	$2Estudiante ESPC	Ver Credencial
Vanesa Camacho	Confirmado	$7Profesional otra	Ver Credencial

Figura 5.34 Impresión de las credenciales antes del cambio

La figura 5.35 siguiente muestra como se realiza la impresión de las credenciales después del cambio.

Nombre	Estado	Modificar	Confirmar Registro
Luis Enrique Loor Maspons	Confirmado	$2: Estudiante ESF	[Registro Confirmado]
Vanesa Camacho	Confirmado	$7: Profesional otr	[Registro Confirmado]
Mostrar Credenciales			

Figura 5.35 Impresión de las credenciales después del cambio

Resultado de las pruebas del escenario 4

Prueba número:	1
Nombre de la prueba:	Ver reportes de los eventos.
Número de Intento	**Resultados**
1	El usuario buscó los eventos por el año y por el capítulo de la rama; esto en base a la experiencia en la prueba del escenario 3. Se consultó el reporte de participantes al evento y el de los ingresos del evento.

Resultado de las pruebas del escenario 5

Prueba número:	1
Nombre de la prueba:	Registro de miembros de la rama estudiantil.
Número de Intento	**Resultados**
1	El usuario ingresó en la página de registro los datos personales y de membresía del nuevo usuario. En los datos de la membresía del usuario no se ingresó el número de membresía (campo alfa numérico de 8 caracteres) del nuevo usuario por lo que el sistema notificó al usuario que ese campo era requerido.
2	El usuario ingresó los datos personales y de membresía correctamente y procedió a guardar los datos. Sin embargo no seleccionó ninguna sociedad para registrar al nuevo miembro, el sistema envió un mensaje de que no había escogido ninguna sociedad para registrar.
3	El usuario ingresó todos los datos requeridos por el sistema y se mostró un mensaje que indicó que el registro se había realizado de manera exitosa.

Resultado de las pruebas del escenario 6

Prueba número:	1
Nombre de la prueba:	Reportes de miembros de la rama.
Número de	**Resultados**

Intento	
1	El usuario procedió a realizar la consulta de los miembros activos de la rama estudiantil. El resultado fue satisfactorio.
2	El usuario procedió a realizar la consulta de los miembros inactivos de la rama. El resultado fue satisfactorio.
3	El usuario procedió a realizar la consulta de los miembros de la activos de la rama pero agrupándolos por capítulos.

Durante la realización de las pruebas en este escenario se sugirió el cambio que se detalla a continuación.

Para las consultas de los miembros existen tres enlaces que realizan los tres tipos diferentes de consultas: los miembros activos, los miembros inactivos y los miembros agrupados por cada capítulo de la rama. El cambio que se implementó fue que ahora existe un sólo enlace para la realizar las consultas y existen varios filtros que se aplican a la consulta. Es decir, ahora el usuario escoge qué tipo de reporte de usuarios quiere visualizar (activos o inactivos) y escoge de qué capítulo de la rama quiere ver los miembros.

La figura 5.36 muestra cómo estaba formado antes el esquema de reportes de los miembros de la rama.

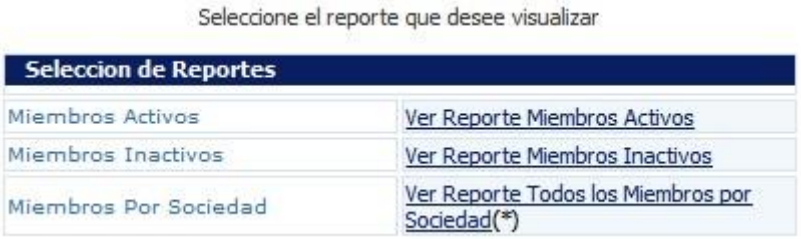


Figura 5.36 Esquema anterior de reportes de los miembros de la rama.

La figura 5.37 muestra el nuevo esquema para realizar los reportes de los miembros de la rama. Como se puede apreciar, ahora sólo se escoge la sociedad de la cual se quiere obtener el reporte y el estado de los miembros.

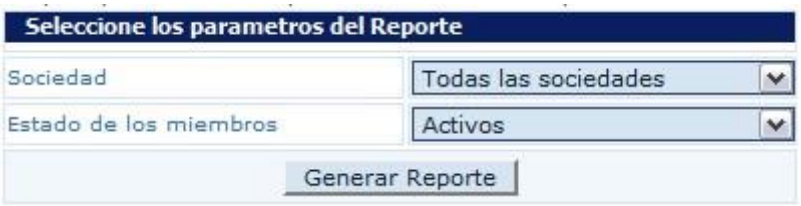


Figura 5.37 Esquema actual de reportes de los miembros de la rama.

Resultado de las pruebas del escenario de prueba 7

Prueba número:	1
Nombre de la prueba:	Ingreso de correspondencia.
Número de Intento	**Resultados**
1	El usuario ingresó en el formulario los datos de la correspondencia. Se llenaron todos los campos y se procedió con el ingreso de la correspondencia. Sin embargo no se escogieron a los miembros

	de la rama cuya correspondencia había llegado, por lo que el sistema mostró un mensaje de error indicando que para asignar la correspondencia era necesario escoger a los miembros.
2	El usuario ingresó los datos de la correspondencia y asignó los usuarios que debían retirar dicha correspondencia. El sistema ingresó la nueva correspondencia y mostró un mensaje indicando que se había ingresado la correspondencia correctamente.

Resultado de las pruebas del escenario de prueba 8

Prueba número:	1
Nombre de la prueba:	Entrega de correspondencia.
Número de Intento	**Resultados**
1	Para entregar la correspondencia, primero es necesario buscar al miembro IEEE para poder registrar que la correspondencia ha sido entregada. Para buscar a los miembros existe campo con autocompletar donde se ingresa las iniciales del apellido. El usuario no ingresó ninguna letra para realizar la búsqueda del miembro de la rama al que se iba a entregar la correspondencia. El sistema mostró un mensaje que indicaba que se debía ingresar al menos una letra para realizar la consulta de miembros de la rama.
2	El usuario ingresó la letra “l” para realizar la búsqueda. De entre la lista de usuarios que apareció escogió uno y el sistema listó la correspondencia pendiente de entrega. En este punto el usuario no identificó que tenía que dar clic en la frase “No entregado” para que el sistema marque a la correspondencia como entregada al usuario.
3	El usuario marcó como entregada la correspondencia al usuario. El sistema eliminó de la lista de correspondencia pendiente

	aquella que se marcó como entregada.

En este escenario se sugirió un cambio. Puesto que no fue intuitivo para el usuario que al dar clic en “No entregado” para que el sistema marcara la correspondencia como retirada por el miembro de la rama, se cambió la etiqueta que decía “No entregado” por otra que dice “Entregar correspondencia”.

La figura 5.38 muestra la etiqueta anterior donde para entregar la correspondencia se tenía que dar clic en la etiqueta “No entregado”.

Tipo	Nombre	Descripción	Sociedad	Fecha Retiro	Estado
Tarjetas de Membresia	Membresia Computer	Tarjetas de Membresia de Computer	Computer Society	ninguna	[No Retirado]
Sobres	Información personal	Sobres con información personal	Computer Society	ninguna	[No Retirado]
Sobres	Invitación	Sobre de invitación a evento	Computer Society	ninguna	[No Retirado]

Figura 5.38 Entrega de correspondencia antes del cambio

La figura 5.39 siguiente muestra el cambio de la etiqueta “No entregado” por “Entregar Correspondencia” para que sea más intuitiva al usuario.

Tipo	Nombre	Descripción	Sociedad	Fecha Retiro	Estado
Tarjetas de Membresia	Membresia Computer	Tarjetas de Membresia de Computer	Computer Society	ninguna	Entregar Correspondencia
Sobres	Información personal	Sobres con información personal	Computer Society	ninguna	Entregar Correspondencia
Sobres	Invitación	Sobre de invitación a evento	Computer Society	ninguna	Entregar Correspondencia

Figura 5.39 Entrega de correspondencia después del cambio

Resultado de las pruebas del escenario de prueba 9

Prueba número:	1
Nombre de la prueba:	Registro de Nuevo Ingreso Económico.
Número de Intento	**Resultados**
1	El usuario intentó ingresar el valor unitario como una cantidad separada los decimales con coma, el valor ingresado fue 10,25. Sin embargo la separación de los decimales se la realiza con un punto. El sistema mostró el mensaje de error indicando que el formato de la cantidad ingresada no era válido.
2	El usuario terminó de ingresar los detalles del ingreso y pulsó el botón finalizar. El sistema en ese momento registró la transacción y mostró una opción para se pueda imprimir el comprobante físico del ingreso. El usuario guardó el archivo del comprobante en una carpeta de la PC donde se encontraba realizando las pruebas.

Prueba número:	2

Nombre de la prueba:	Registro de Nuevo Egreso Económico.
Número de Intento	**Resultados**
1	En base a la experiencia previa en el registro de los ingresos, el registro de los egresos no conllevó presentó problema. Cuando el sistema le mostró el archivo del comprobante físico de la transacción, el usuario guardó dicho archivo en una carpeta de la PC donde se encontraba.

Resultado de las pruebas del escenario de prueba 10

Prueba número:	1
Nombre de la prueba:	Ver reportes de utilidades de la rama.
Número de Intento	**Resultados**
1	Para realizar el reporte el usuario debe escoger un rango de fechas. El usuario escogió las fechas entre las cuales se debe obtener el reporte. El sistema mostró un mensaje indicando que para visualizar los reportes se debía proporcionar las fechas de inicio y fin del período.
2	El usuario ingresó las fechas entre las cuales quería obtener el reporte de utilidades de la rama y pudo obtener el documento donde se mostraban los detalles

CONCLUSIONES Y RECOMENDACIONES

- Aplicar J2EE en el desarrollo del portal administrativo de la Rama Estudiantil IEEE – ESPOL permitió implementar una aplicación robusta y de bajo costo, tomando en consideración que está dirigido a una Rama Estudiantil cuyos líderes son todos voluntarios, el costo y la eficiencia es aún más importante.
- El uso de patrones de diseño permitió implementar una aplicación mucho más eficiente puesto que se promueve la reutilización de código y se facilita la tarea de mantenimiento de la aplicación.
- La utilización de AJAX permitió la implementación de interfaces más fáciles de usar para tareas comunes como búsqueda o ingreso de datos. La actualización de las páginas de forma asincrónica permite tener como resultado una aplicación más eficiente en consumo de ancho de banda y en el tiempo de respuesta para el usuario.
- El portal permitirá simplificar la gestión de los procesos administrativos de la Rama IEEE – ESPOL, y llevar un registro digital de los eventos organizados, con sus respectivos asistentes y conferencistas.
- Es importante que se promocione la utilización del portal desarrollado para la Rama IEEE – ESPOL, puesto que constituirá una herramienta eficiente de difusión de los eventos organizados por cada capítulo de la Rama así como de los beneficios de ser miembro de la misma.

- La utilización de CSS en el diseño de las páginas dio como resultado un portal web que será fácil de rediseñar puesto que bastará con cambiar las hojas de estilo utilizadas para cambiar la apariencia completa de todas las páginas del sistema.
- Aplicar estándares de diseño Web en el desarrollo de la aplicación, permitió separar los estilos del contenido, lo cual aumenta la velocidad de navegación y facilita el mantenimiento de las páginas que conforman el portal.
- Este proyecto de tesis abarcó los procesos de mayor importancia dentro de la Rama Estudiantil en la actualidad, por lo cual recomiendo que se analicen futuras necesidades, para que éste sistema sirva como base para proyectos de tesis que adhieran más funcionalidades.

ANEXOS

ANEXO A: DESCRIPCIÓN DE LOS CASOS DE USO

A continuación se describen los casos de uso de sistema.

Caso de Uso 1: Ingresar al Sistema

Descripción:

Permite que un usuario ingrese en el sistema.

Notas:

El usuario puede ser cualquier persona que haya sido registrada en el sistema como miembro de la IEEE.

Escenarios:

1.1 Ingreso exitoso al sistema.

1.2 Ingreso no exitoso debido por usuario o contraseña incorrectos.

1.3 Ingreso no exitoso por error en el servidor.

Actores: Administrador de Correspondencia y Membresías, Administrador General, Administrador Financiero, Miembro IEEE.

Caso de Uso 2: Publicar Anuncios

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia publicar un anuncio en el sistema para que sea visto de manera pública.

Notas:

Los anuncios que se publican tienen una fecha de expiración que es la fecha en la cual el anuncio publicado dejará de ser visualizado.

Escenarios:

2.1 Ingreso exitoso del nuevo anuncio.

2.2 Ingreso no exitoso debido a problemas con la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 3: Modificar Anuncios

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia modificar los datos de anuncios ingresados.

Notas:

El sistema permite la modificación de anuncios que ya han expirado.

Escenarios:

3.1 Modificación exitosa del anuncio.

3.2 Modificación no exitosa debido a errores con la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 4: Ingresar Correspondencia

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia ingresar nueva correspondencia que haya llegado a la rama.

Notas:

La correspondencia se ingresa dependiendo del tipo de sociedad a la que pertenece la correspondencia y se asigna sólo a los miembros de dichas sociedades.

Escenarios:

4.1 Ingreso exitoso de la correspondencia.

4.2 Ingreso no exitoso debido a errores con la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 5: Modificar Correspondencia

Descripción:

Permite a un usuario de tipo Administrador de correspondencia modificar los datos de la correspondencia que ha sido ingresada.

Notas:

Para modificar la correspondencia hay que buscar la correspondencia de acuerdo a un rango de fecha en las que ésta puede haber sido ingresada.

Escenarios:

5.1 Modificación exitosa de los datos de la correspondencia.

5.2 Modificación no exitosa debido a errores en la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 6: Entregar Correspondencia

Descripción:

Permite a un usuario de tipo Administrador de correspondencia entregar la correspondencia a los usuarios que vayan a retirarla.

Notas:

Para entregar la correspondencia hay que buscar la correspondencia que tiene asignada cada miembro para poder realizar la entrega de la misma.

Escenarios:

6.1 Entrega exitosa de la correspondencia

6.2 Entrega no exitosa debido a que el usuario no tiene ninguna correspondencia asignada.

6.3 Entrega no exitosa debido a errores con la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 7: Ver correspondencia Personal

Descripción:

Permite a cualquier usuario del sistema sin importar su rol, ver qué correspondencia ha llegado a su buzón.

Notas:

Para poder ver la correspondencia, el usuario debe haber ingresado al sistema con su usuario y contraseña.

Escenarios:

7.1 Visualización exitosa de la correspondencia personal.

7.2 Visualización no exitosa de la correspondencia personal debido a que no hay ninguna correspondencia asignada.

7.3 Visualización no exitosa debido a errores con la base de datos.

Actores: Administrador de Correspondencia y Membresías, Administrador General, Administrador Financiero, Miembro IEEE.

Caso de Uso 8: Crear Roles

Descripción:

Permite a un usuario de tipo Administrador General crear nuevos roles según las opciones que ya se encuentran habilitadas.

Notas:

Sólo se pueden crear roles usando a las opciones que ya existen en el sistema, no se permite crear nuevas opciones para los nuevos roles.

Escenarios:

8.1 Ingreso exitoso del nuevo rol.

8.2 Ingreso no exitoso del nuevo rol debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 9: Eliminar Roles

Descripción:

Permite a un usuario de tipo Administrador General eliminar roles que a su criterio no tienen más cabida en el sistema.

Notas: Cuando un rol queda inactivo, los usuarios que tenían dicho rol asignado pasan a estar en un estado de inactividad.

Escenarios:

9.1 Eliminación exitosa del rol seleccionado.

9.2 Eliminación no exitosa debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 10: Ingresar Inventario

Descripción:

Permite a un usuario de tipo Administrador Financiero ingresar ítems del inventario de la rama.

Notas:

El tipo de inventario a ingresar puede ser cualquiera de los tipos registrados en el sistema.

Escenarios:

10.1 Ingreso exitoso del nuevo ítem del inventario.

10.2 Ingreso no exitoso del nuevo ítem del inventario debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 11: Modificar Inventario

Descripción:

Permite a un usuario de tipo Administrador Financiero modificar los datos de los ítems que conforman el inventario de la rama.

Notas:

Para buscar los ítems a modificar hay que ingresar el tipo de ítem a buscar y un rango de fechas según su fecha la fecha de ingreso del ítem.

Escenarios:

11.1 Modificación exitosa del ítem.

11.2 Modificación no exitosa debido a que el ítem no fue encontrado en la búsqueda.

11.3 Modificación no exitosa debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 12: Ingresar Movimientos de Inventario

Descripción:

Permite a un usuario de tipo Administrador Financiero ingresar movimientos sobre el inventario existente en la rama.

Notas:

Para registrar los movimientos sobre inventario hay que buscar los ítems mediante una ventana de búsqueda por el nombre del ítem.

Escenarios:

12.1 Ingreso exitoso del movimiento del inventario.

12.2 Ingreso no exitoso del movimiento debido a que el ítem no fue encontrado en la búsqueda.

12.3 Ingreso no exitoso del movimiento debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 13: Generar Reporte de Ítems de Inventario

Descripción:

Permite a un usuario de tipo Administrador Financiero generar un reporte en formato PDF de los ítems con sus respectivos movimientos.

Notas:

Para generar el reporte hay que establecer el rango de fechas para los movimientos así como el tipo de movimiento y el tipo de ítem.

Escenarios:

13.1 Generación del reporte exitosa.

13.2 Generación del reporte no exitosa debido que no se han establecido correctamente los parámetros para filtrar el reporte.

13.3 Generación del reporte no exitosa debido a que la búsqueda según los parámetros establecidos no produjo resultados.

13.4 Generación del reporte no exitosa debido a problemas con los archivos de JasperReports necesarios para la creación del mismo.

13.5 Generación no exitosa del reporte debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 14: Registrar Ingreso Financiero

Descripción:

Permite a un usuario de tipo Administrador Financiero registrar un ingreso financiero.

Notas:

Para registrar un ingreso hay que establecer si el ingreso pertenece a la rama en general o algún capítulo de la misma.

Escenarios:

14.1 Registro exitoso del ingreso financiero.

14.2 Registro no exitoso del ingreso financiero debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 15: Modificar Ingreso Financiero

Descripción:

Permite a un usuario de tipo Administrador Financiero modificar un ingreso financiero.

Notas:

Para modificar un ingreso financiero hay que establecer el tipo de transacción (ingreso) y escribir el número de comprobante de la transacción.

Escenarios:

15.1 Modificación exitosa del ingreso financiero.

15.2 Modificación no exitosa debido a que no se encontró ningún ingreso financiero con el comprobante ingresado.

15.3 Modificación no exitosa debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 16: Generar Comprobante de Registro de Ingreso

Descripción:

Permite a un usuario de tipo Administrador Financiero obtener un comprobante de registro de un ingreso financiero.

Notas:

Para obtener este comprobante el usuario debe haber registrado un ingreso económico.

Escenarios:

16.1 Generación exitosa del comprobante de ingreso.

16.2 Generación no exitosa debido a que el usuario no ha registrado ningún ingreso financiero.

16.3 Generación no exitosa debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 17: Registrar Egreso Financiero

Descripción:

Permite a un usuario de tipo Administrador Financiero registrar un egreso financiero.

Notas:

Para registrar un egreso financiero hay que establecer si pertenece a la rama en general o a algún capítulo.

Escenarios:

17.1 Ingreso exitoso del egreso financiero.

17.2 Ingreso no exitoso del egreso financiero debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 18: Modificar Egreso Financiero

Descripción:

Permite a un usuario de tipo Administrador Financiero modificar un egreso financiero.

Notas:

Para modificar un ingreso financiero hay que establecer el tipo de transacción (egreso) y escribir el número de comprobante de la transacción.

Escenarios:

18.1 Modificación exitosa del egreso financiero.

18.2 Modificación no exitosa del egreso financiero debido a que no se encontró ningún egreso financiero con el número de comprobante ingresado.

18.3 Modificación no exitosa debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 19: Generar Comprobante de Registro de Egreso

Descripción:

Permite a un usuario de tipo Administrador Financiero obtener un comprobante de registro de un egreso financiero.

Notas:

Para poder obtener el comprobante de egreso el usuario debe haber registrado un egreso financiero.

Escenarios:

19.1 Generación exitosa del comprobante de egreso.

19.2 Generación no exitosa debido a que no se ha registrado ningún egreso financiero.

19.3 Generación no exitosa debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 20: Generar Reporte de Utilidades de la Rama

Descripción:

Permite a un usuario de tipo Administrador Financiero obtener un reporte de utilidades de la rama.

Notas:

Para poder generar el reporte de utilidades, se deben haber registrado ingresos y egresos.

Escenarios:

20.1 Generación exitosa del reporte de utilidades.

20.2 Generación no exitosa debido a que no existen registros de ingresos y egresos.

20.3 Generación no exitosa debido a errores en la base de datos.

Actores: Administrador Financiero.

Caso de Uso 21: Generar Reporte de Utilidades por cada Capítulo

Descripción:

Permite a un usuario de tipo Administrador Financiero obtener un reporte de ingresos y egresos por cada capítulo de la rama.

Notas:

Para poder generar el reporte es necesario que se seleccione el capítulo de la rama del cual se quiere obtener el reporte.

Escenarios:

21.1 Generación exitosa del reporte de las utilidades por capítulo.

21.2 Generación no exitosa debido a que no se han registrado ingresos y egresos en el capítulo seleccionado para el reporte.

Actores: Administrador Financiero.

Caso de Uso 22: Registrar nuevos Miembros a la Rama IEEE – ESPOL

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia registrar nuevos miembros IEEE a la rama, así como registrar las sociedades a las que se registra.

Notas:

En el momento del registro al nuevo miembro se le asigna un usuario y contraseña y su rol en el sistema para que después pueda ingresa en el mismo.

Escenarios:

22.1 Ingreso exitoso de los datos del nuevo miembro.

22.2 Ingreso no exitoso de los datos del nuevo miembro debido a problemas con la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 23: Renovar las membresías de los miembros

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia renovar las membresías a los miembros que deseen renovar la suscripción a la IEEE.

Notas:

Para poder renovar las membresías, se debe buscar al miembro de acuerdo a las iniciales de sus apellidos para luego ver las membresías

a las que se desea suscribir y aquellas a las que estaba suscrito y desea renovar.

Escenarios:

23.1 Renovación exitosa de la membresía.

23.2 Renovación no exitosa de la membresía debido a que no se encontró ningún miembro en la búsqueda previa a la renovación.

23.3 Renovación no exitosa debido a errores en la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 24: Generar Reporte de Miembros activos e inactivos

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia generar un reporte de los miembros activos e inactivos de la rama.

Notas:

Para poder generar este reporte deben existir miembros registrados en el sistema.

Escenarios:

24.1 Generación exitosa del reporte de los miembros activos e inactivos.

24.2 Generación no exitosa del reporte debido a que no existen miembros registrados en el sistema.

24.3 Generación no exitosa debido a errores en la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 25: Generar Reporte de todos los miembros activos por cada capítulo de la rama

Descripción:

Permite a un usuario de tipo Administrador de Correspondencia generar un reporte de todos los miembros activos por cada capítulo de la rama.

Notas:

Para poder generar este reporte debe haber miembros registrados y activos en el sistema.

Escenarios:

25.1 Generación exitosa del reporte de todos los miembros activos por cada capítulo de la rama.

25.2 Generación no exitosa del reporte debido a que no existen miembros registrados o activos en el sistema.

25.3 Generación no exitosa del reporte debido a errores en la base de datos.

Actores: Administrador de Correspondencia y Membresías.

Caso de Uso 26: Modificación de los datos de los miembros

Descripción:

Permite a un usuario de tipo Administrador General modificar los datos de los miembros registrados en el sistema.

Notas:

Para poder modificar los datos hay que buscar a los miembros por medio de las iniciales de los apellidos.

Escenarios:

26.1 Modificación de los datos de los miembros exitosa.

26.2 Modificación de los datos de los miembros no exitosa debido a que no se encontró ningún miembro.

26.3 Modificación de los datos de los miembros no exitosa debido a errores en la base de datos.

Actores: Administrador de membresías.

Caso de Uso 27: Crear Eventos

Descripción:

Permite a un usuario de tipo Administrador General crear nuevos eventos para que sean promocionados por la Rama.

Notas:

Escenarios:

27.1 Creación exitosa del evento.

27.2 Creación no exitosa del evento debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 28: Modificar Eventos

Descripción:

Permite a un usuario de tipo Administrador General la modificación de los datos de un evento que haya sido creado.

Notas:

Para modificar los datos de un evento hay que buscar los eventos de acuerdo al año de creación.

Escenarios:

28.1 Modificación exitosa de los datos del evento.

28.2 Modificación de los datos del evento no exitosa debido a que no hay eventos creados para el año buscado.

28.3 Modificación de los datos del evento no exitosa debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 29: Registrar participantes en los eventos creados

Descripción:

Permite a cualquier persona que visite el portal Web registrarse en los diferentes eventos que promociona la rama.

Notas:

Para registrarse en los eventos los participantes deben ingresar sus datos personales.

Escenarios:

29.1 El registro del participante se realiza de manera exitosa.

29.2 El registro del participante no se realiza de manera exitosa debido a que no existe cupo disponible para el evento seleccionado.

29.3 El registro del participante no se realiza de manera exitosa debido a errores en la base de datos.

Actores: Administrador General, Administrador Financiero, Administrador de Membresía, Usuario público, Miembro IEEE.

Caso de Uso 30: Confirmar registro de los participantes

Descripción:

Permite a un usuario de tipo Administrador General confirmar el tipo de registro de los participantes a un evento.

Notas:

Para confirmar el registro de los participantes es necesario buscar el evento del cual se quiere hacer la confirmación por medio del año de creación del evento.

Escenarios:

30.1 Confirmación de registro exitosa.

30.2 Confirmación de registro no exitosa debido a que no se encontró ningún evento en la búsqueda previa.

30.3 Confirmación no exitosa debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 31: Generar credenciales de los eventos

Descripción:

Permite a un usuario de tipo Administrador General generar las credenciales de los eventos para cada participante.

Notas:

Para generar las credenciales de los eventos es necesario previamente haber confirmado el registro de los participantes.

Escenarios:

31.1 Generación de la credencial exitosa.

31.2 Generación de la credencial no exitosa debido a que no se había confirmado el registro previamente.

31.3 Generación de la credencial no exitosa debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 32: Generar Reporte de ingresos por cada evento

Descripción:

Permite a un usuario de tipo Administrador General generar un reporte de los ingresos resultantes de un evento.

Notas:

Para poder generar el reporte de los ingresos por evento es necesario que se hayan confirmado los registros de los participantes.

Escenarios:

32.1 Generación del reporte de ingresos por eventos exitosa.

32.2 Generación del reporte de ingresos por eventos no exitosa debido a que no han confirmado los registros de los participantes.

32.3 Generación del reporte de ingresos por eventos no exitosa debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 33: Crear nuevas sociedades para el registro de los miembros

Descripción:

Permite a un usuario de tipo Administrador General ingresar nuevas sociedades a la Rama para ofrecerlas a los estudiantes.

Notas:

Escenarios:

33.1 Ingreso de los datos de la nueva sociedad IEEE exitosa.

33.2 Ingreso de los datos de la nueva sociedad IEEE no exitoso debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 34: Modificar sociedades

Descripción:

Permite a un usuario de tipo Administrador General modificar los datos de las sociedades IEEE ya creadas.

Notas:

Para poder modificar los datos de las sociedades IEEE ya creadas hay que escoger la sociedad a modificar de un listado donde aparecen todas las sociedades IEEE.

Escenarios:

34.1 Modificación de los datos de la sociedad IEEE exitosa.

34.2 Modificación de los datos de la sociedad IEEE no exitosa debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 35: Crear nuevos capítulos de la Rama

Descripción:

Permite a un usuario de tipo Administrador General crear nuevos capítulos para la rama.

Notas:

Escenarios:

35.1 Ingreso de los datos del nuevo capítulo de la rama exitoso.

35.2 Ingreso de los datos del nuevo capítulo de la rama no exitoso debido a errores en la base de datos.

Actores: Administrador General.

Caso de Uso 36: Modificar los capítulos de la Rama

Descripción:

Permite a un usuario de tipo Administrador General modificar los datos de los capítulos creados.

Notas:

Para poder modificar los datos de los capítulos se debe seleccionar el capítulo de un listado donde aparecen todos los capítulos que se han creado para la rama.

Escenarios:

36.1 Modificación exitosa de los datos.

36.2 Modificación no exitosa debido a errores en la base de datos.

Actores: Administrador General.

ANEXO B: VALIDACIÓN DEL CÓDIGO XHTML

Como se expuso en el capítulo 3 XHTML es un estándar que se usa en la actualidad para el desarrollo de páginas web. En dicho capítulo se explicaron sus ventajas y cuáles son las reglas que se siguen para validar código XHTML.

A continuación se muestra un ejemplo que muestra como las páginas del portal desarrollado son válidas según el validador XHTML de la W3C.

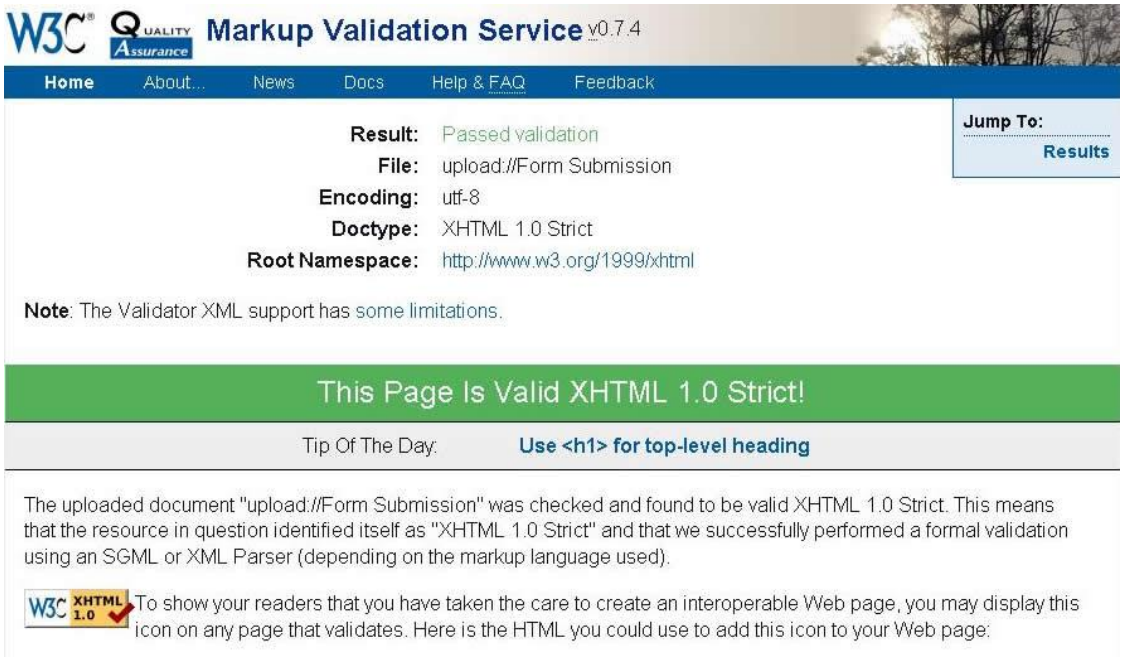


Figura B.1: Imagen que muestra la validación de una página XHTML en modo estricto.

Sin embargo hay que tomar en cuenta que a la hora de validar el código, el validador de la W3C al encontrar el carácter “&” nos dirá que hay un error debido a que en vez de “&” debe estar “&”. A la hora de validar el código no se tomó en cuenta los enlaces puesto que ellos contienen el carácter “&” que sirve para enviar parámetros en una dirección. Si reemplazamos “&” por “&” los enlaces del portal quedarían inválidos, por esto es que a la hora de la validación no se consideraron los enlaces.

ANEXO C: USO DE JSTL (Java Standart Tag Library)

Como se explicó en el capítulo 4, las páginas JSP sólo muestran información al usuario de acuerdo a peticiones que éste realiza. La información que se muestra en las páginas es enviada a través de objetos de Java que deben ser recogidos dentro la página. Para recoger los objetos se debe escribir código Java, lo cual no es muy recomendado en las páginas JSP, entre otras razones porque la depuración de errores es muy compleja. Para evitar escribir código dentro de las páginas existe un proyecto de Apache llamado JSTL (Java Standart Tag Library) que permite acceder a objetos y funciones de Java sin necesidad de escribir scriptleps ni directivas jsp en la página.

Para usar JSTL en las páginas JSP es necesario hacer una referencia a las librerías de JSTL. Para esto usamos la directiva taglib.

La siguiente figura muestra como se realiza la referencia a JSTL dentro de las páginas JSP.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

Figura C.1: Referencia a JSTL dentro de una página JSP.

Además de esto es necesario incluir dentro del proyecto las librerías estándar.jar y jstl.jar

Para hacer referencia a los objetos de java sólo se necesita invocar a los objetos de la siguiente manera:

```
${requestScope.objEvento.fechaInicio}
```

Figura C.2: Invocación de los objetos a través de JSTL

El código JSTL de la figura anterior nos quiere decir que del objeto request obtengamos el objeto objEvento y de éste mostremos el atributo fechaInicio.

ANEXO D: IMPLEMENTACIÓN DE MVC EN LOS SERVLETS

Como se habló en el capítulo 4, uno de los patrones usados es el MVC (model view controller), que se encarga de procesar las peticiones de los usuarios y mostrar la información requerida. Para poder implementar MVC a través de servlets, es necesario que en ellos exista un identificador para las peticiones de los usuarios, por esto en cada servlet existe una variable llamada *acción* que es la que recoge la petición del usuario que viene en el objeto request.

```
String accion = request.getParameter("accion");
```

Figura D.1: Variable que recoge el parámetro “accion”

Una vez que el servlet captura la petición del usuario establece por medio de sentencias condicionales cuál es la acción que el usuario ha solicitado realizar y en base a esto realizar las operaciones necesarias.

```
//==================================================
// Buscar Inventario
//==================================================

if(accion.equals("buscarinventario"))
{
    int opcTipoEquipo = Integer.parseInt(request.getParameter("opcTipoEquipo"));
    String txtFechaInicio = request.getParameter("txtFechaInicio");
    String txtFechaFinal = request.getParameter("txtFechaFinal");
    InventarioDVO objArrayInventario[] = null;
```

Figura D.2: Implementación del MVC en un servlet a través de la sentencia condicional if.

Como se puede apreciar en la figura anterior, a través de la sentencia condicional if se pregunta si la acción ha realizar es “buscarinventario”.

Con esta técnica se implementan los servlets controladores que interactúan en el modelo MVC. En el sistema existe un servlet controlador por cada módulo desarrollado así se puede tener un sobre los módulos de la aplicación.

ANEXO E: ARCHIVOS DE CONFIGURACIÓN

Dentro de la aplicación existe un archivo de configuración llamado web.xml donde se realizan varias configuraciones según las necesidades del aplicativo.

En el archivo web.xml de la aplicación desarrollada existen las siguientes configuraciones necesarias para el funcionamiento correcto del sistema:

- Configuración de los servlets: Cada servlet de la aplicación se encuentra registrado en este archivo para que puedan ser invocados.

```
- <servlet>
    <description>This is the description of my J2EE component</description>
    <display-name>This is the display name of my J2EE component</display-name>
    <servlet-name>ServletPais</servlet-name>
    <servlet-class>servlets.ServletPais</servlet-class>
  </servlet>
```

Figura E.1: Configuración de un servlet dentro del archivo web.xml.

- Configuración de filtros: Para que se aplicar los filtros se debe especificar el recurso a filtrar.

```
- <filter>
    <filter-name>FilterAutenticacion9</filter-name>
    <filter-class>filtros.FilterAutenticacion</filter-class>
  </filter>
- <filter-mapping>
    <filter-name>FilterAutenticacion9</filter-name>
    <url-pattern>/ModuloRoles/*</url-pattern>
  </filter-mapping>
```

Figura E.2: Configuración de filtro dentro del archivo web.xml.

- Configuración de páginas de error: Cuando ocurre algún tipo de error en la aplicación el sistema muestra páginas de error personalizadas que se encuentran descritas en el archivo web.xml

```
- <error-page>
    <error-code>404</error-code>
    <location>/General/RecursoNoEncontrado.jsp</location>
  </error-page>
```

Figura E.3: Configuración de páginas de error personalizadas dentro del archivo web.xml.

- Configuración del tiempo de la sesión: El tiempo durante el cual la sesión de los usuarios es válida se encuentra descrito en minutos dentro del archivo.

```
- <session-config>
    <session-timeout>120</session-timeout>
  </session-config>
```

Figura E.4: Configuración de tiempo de expiración de sesión dentro del archivo web.xml.

BIBLIOGRAFÍA

[1] Institute of Electrical and Electronics Engineers, http://www.ieee.org.

[2] Construir Aplicaciones EJB con JBoss, Lomboz y Eclipse, http://www.programacion.com/java/tutorial/jap_aplic_jboss/3/

[3] [Alur et al., 2003] D. Alur, J. Crupi and D. Malks, *Core J2EE Patterns: Best Practices and Design Strategies – Second Edition*. Sun Microsystems Press, ISBN:0-13-142246-4, June 2003.

[4] AJAX, http://es.wikipedia.org/wiki/AJAX

[5] AJAX: Un nuevo acercamiento a las aplicaciones web, http://www.maestrosdelweb.com/editorial/ajax/

[6] World Wide Web Consortium (W3C), http://www.w3c.org

[7] Cascading Style Sheet, http://www.w3.org/Style/CSS/

[8] W3C XHTML Working Group, http://www.w3.org/MarkUp/

[9] Historia de los patrones, http://www.1x4x9.info/files/patrones/html/online-chunked/ar01s02.html

[10] Java en castellano. Aplicación de Patrones J2EE en un Caso de Estudio,

http://www.programacion.com/java/articulo/inukisoft/index.html

[11] Floyd Marinescu, Value Objects, VO Factory, and Generic Attribute Access, parte del libro “EJB Design patterns” , 2001.

[12] Design Patterns: Data Access Object, http://java.sun.com/blueprints/patterns/DAO.html

[13] Core J2EE Patterns Best Practices and Design Strategies, http://www.corej2eepatterns.com/Patterns2ndEd/CompositeView.htm

[14] XHTML 1.0 Strict, W3C markup Validator, http://www.w3.org/MarkUp/

[15] Jasper Reports Open Source http://www.jasperforge.org/sf/projects/jasperreports

[16] Ireports Open Source. http://www.jasperforge.org/sf/projects/ireport

[17] Filtros en java, Oscar González Moreno, http://www.samelan.com/oscargonzalez/doc/java_filters.pdf