Implementación de un portal web para la automatización del proceso de consultorías de mentores GOLD de la Región Latinoamericana del IEEE (R9), utilizando arquitectura Java 2 Enterprise Edition - J2EE y tecnología Ajax

Diana Cristina Vera Santana
Rubén Alejandro Lara Vásquez
Pedro Fabricio Echeverría Briones
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral
Guayaquil, Ecuador
diana.vera@ieee.org
rlara@fiec.espol.edu.ec
pechever@espol.edu.ec

Resumen

J2EE es una plataforma de programación para desarrollar y ejecutar software de aplicaciones con arquitectura de múltiples capas distribuidas. Se basa ampliamente en componentes modulares de software ejecutándose sobre un servidor de aplicaciones. Ajax es una técnica de desarrollo web para crear aplicaciones interactivas, se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. El presente artículo explica brevemente la los conceptos relacionados a la arquitectura J2EE, la tecnología Ajax y los patrones de diseños usados en la implementación de un portal web para la automatización del proceso de consultorías de mentores GOLD del la Región Latinoamérica de la IEEE. Se realiza una descripción del análisis, diseño y plan de pruebas utilizado para los módulos de Administración, Consultorías y Evaluaciones. Finalmente, se proponen las conclusiones y recomendaciones definidas luego de la realización del proyecto.

Palabras Claves: J2EE, AJAX, IEEE, GOLD, patrones, análisis, diseño, modelos, persistencia.

Abstract

J2EE is a programming platform for developing and executing applications software with multitier distributed architecture. It is based on a software modular components running over an application server. Ajax is a web development technique used to create interactive applications, running on the client, i.e. in the user's browser while maintaining communication with the server asynchronously in the background. This paper briefly explains the concepts related to architecture J2EE, Ajax technology and design patterns used in the implementation of a Web portal to automate the consultancies process of mentors in the GOLD Latin American Region of the IEEE
There is a description of analysis, design and a test plan used for the modules of Administration, Consulting and Evaluation. Finally, we proposed the conclusions and recommendations defined after the implementation of the project.

Key words: *J2EE, AJAX, IEEE, GOLD, Patterns, analysis, design, models, persistence.*

1. Introducción

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, con 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[1].

Dentro del IEEE existe un Grupo de Afinidad de Graduados de la última década llamado GOLD. El objetivo principal de este Grupo de Afinidad GOLD es ayudar a los jóvenes profesionales desde que son estudiantes hasta que ingresan al mundo laboral, para lo cual se ha planteado el programa *Peer to Peer*, que servirá como canal de comunicación e interacción entre varios usuarios. El programa ha sido diseñado para cubrir las interrogantes que existen en los miembros IEEE de las Secciones, Ramas Estudiantiles y Capítulos Estudiantiles de Latinoamérica, a través de la Mentoría de Profesionales IEEE.

Conociendo que la metodología orientada a objetos trabaja utilizando tres capas y tomando como referencia los requerimientos recibidos, se ha procedido a utilizar la arquitectura J2EE para el diseño de los módulos del portal, a través del uso de varios de sus patrones de diseño.

El presente artículo muestra en la primera parte una descripción de la arquitectura J2EE y los patrones que se utilizaron en el diseño de los módulos. Luego se explica cómo se aplicaron los patrones de diseño al análisis de los módulos, para finalmente demostrar a través de los resultados obtenidos de las pruebas esta aplicación.

2. Contenido

2.1. Descripción de la Arquitectura J2EE.

Java 2 Enterprise Edition o Java EE, es una plataforma de programación que permite generar contenido Web de manera dinámica y basada en herramientas de código abierto. J2EE es parte de la plataforma Java, para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de múltiples capas distribuidas, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones [2].

Las aplicaciones J2EE son generalmente consideradas aplicaciones de tres capas ya que se encuentran distribuidas principalmente en tres lugares: en las máquinas clientes (donde se encuentran las vistas con las que interactúan el cliente), la máquina del servidor J2EE (donde está toda la lógica del sistema), y la base de datos. En la figura 1 se muestra un detalle de lo que puede almacenar cada capa:

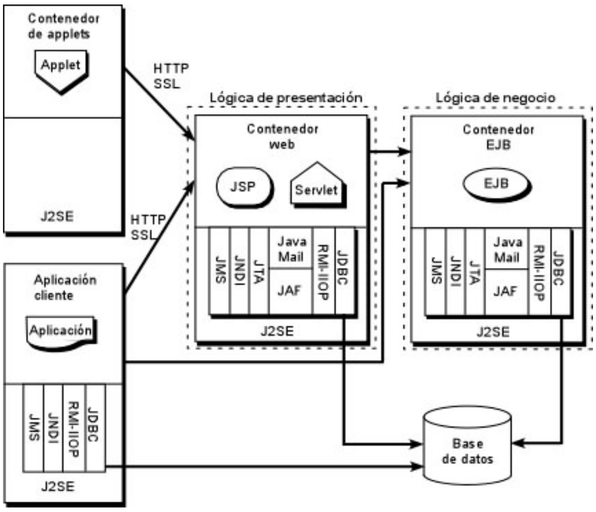

Figura 1. Aplicaciones multicapas

Existe una amplia gama de patrones de diseño propuestos para ser utilizados en el desarrollo de aplicaciones Web. Luego de realizar el análisis y diseño de la aplicación, hemos el uso de los siguientes patrones: Composite View (Vistas Compuestas), Data Transfer Object (Objeto de Transferencia a Datos), Data Access Object (Objeto de Acceso a Datos) y Model View Controller (Modelo Vista Controlador).

2.1.1 Objeto de acceso a Datos (DAO). Este patrón de diseño propone separar y encapsular toda la lógica de acceso a los datos que maneja la aplicación de la implementación, logrando de esta manera desligar la lógica de negocio y presentación del mecanismo de manejo del repositorio de datos que se utilice para almacenar la información. A continuación se muestra la figura 2 que muestra un diagrama de la implementación de un patrón DAO.

Cuando la capa de lógica de negocio necesite interactuar con la base de datos, va a hacerlo a través de la API que le ofrece DAO. Generalmente esta API consiste en métodos CRUD (Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar))

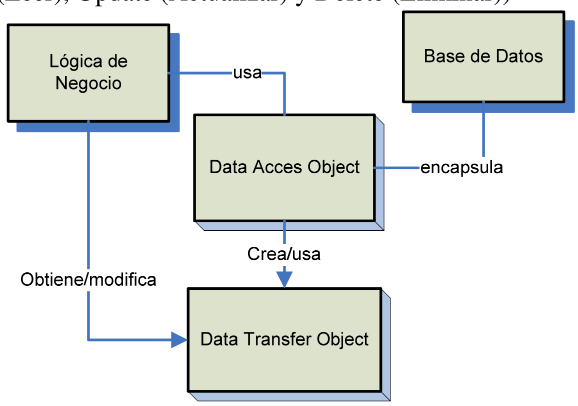

Figura 2. Data Access Object

2.1.2. Transferencia Data-Objeto (DTO). Los DTO o también denominados VO (Value Object). Son utilizados por DAO para transportar los datos desde la base de datos hacia la capa de lógica de negocio y viceversa. Podría decirse que un DTO es un objeto común y corriente, que tiene como atributos los datos del modelo, con sus correspondientes métodos de acceso (getters y setters).

2.1.3. Composite View (Vistas compuestas). Este patrón proporciona un mecanismo para combinar modularmente, las porciones atómicas de una vista completa, las páginas son construidas uniendo código en el formato dentro de cada vista [3]. Las JSP, Java Server Pages, y los servlets proporcionan mecanismos sencillos para poder incluir porciones de una página en otra (de modo estático o dinámico). Este esquema facilita el mantenimiento de las páginas.

2.1.4. Model View Controller (Modelo Vista-Controlador) MVC. Model View Controller (MVC) es uno de los modelos más recomendados para el diseño de aplicaciones interactivas. MVC separa detalles del diseño (persistencia, presentación, y control de los datos), disminuye la duplicación de código, centraliza el control de la aplicación y hace que los cambios o actualizaciones sean más fácilmente manejables [2].
Un diseño MVC puede centralizar el control de funcionalidades como el uso de seguridad, de sesión, y el flujo de la pantalla. MVC puede adaptarse a nuevas fuentes de datos, creando código que adapta la nueva fuente con las pantallas. En la siguiente figura 3 se muestra con mayor detalle la interacción entre los componente del esquema MVC.

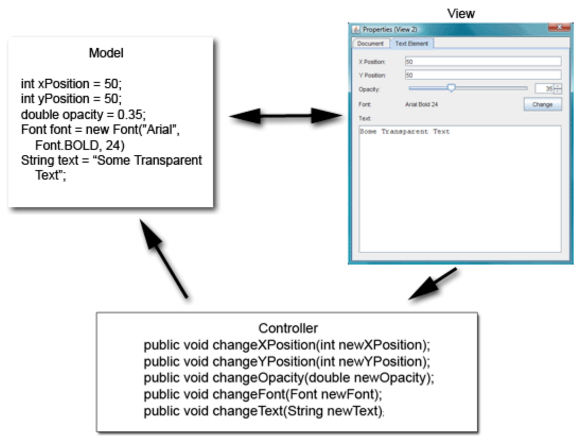

Figura 3. Aplicación en Java utilizando MVC

2.3. Modelo de persistencia de datos

El modelo utilizado para la persistencia de datos es Hibernate. Hibernate es un poderoso objeto de alto rendimiento / persistencia relacional y servicio de consulta. Hibernate permite desarrollar clases persistentes siguiendo lenguaje orientado a objetos - incluyendo la asociación, herencia, polimorfismo, la composición y las colecciones. Hibernate permite expresar consultas en su propia extensión de SQL portátil (HQL), así como en SQL nativo, o con un API de ejemplo o criterio orientados a objetos.
En las siguientes imágenes podemos comparar el modo de conexión usado por estas 2 opciones y la diferencia entre los mismos:

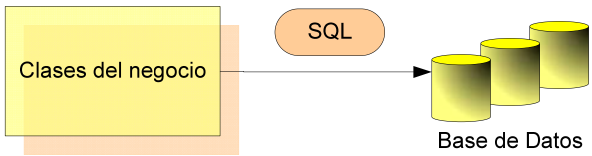

Figura 4. Sentencias SQL directas

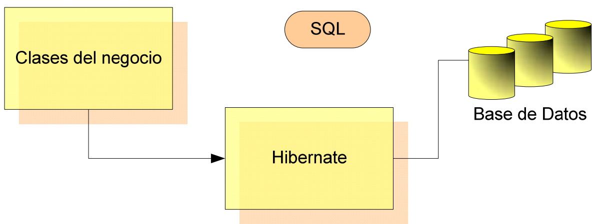

Figura 5. Capa de persistencia Hibernate

2.4. Análisis de los módulos

En esta sección analizaremos lo realizado en el análisis de los módulos que se implementaron. El sistema fue dividido en los siguientes módulos para su implementación:

- Módulo de Administración
- Módulo de Consultorías
- Módulo de Evaluaciones

Como el esquema a utilizar es el Modelo-Vista-Controlador (MVC), este serviría de guía para la implementación de cada módulo, considerando aquellos objetos que son parte del modelo, de las vistas y controladores de peticiones. La implementación resultaría más manejable puesto que se sigue el patrón seleccionado para su diseño.
Luego solo había que identificar aquellas funcionalidades que integraban los módulos para finalmente tener el sistema completo.

2.5. Diseño de los módulos

2.5.2. Modelo Vista Controlador. A continuación se muestra la figura 6 que presenta un ejemplo de la adaptación del MVC a la implementación de los módulos al portal. En este caso, se refiere al ingreso de una actividad.

El usuario a través de la vista Act_Crear_Actividad.jsp, hace un ingreso de los datos de una pregunta y la vista hace una invocación al script AcriptActividad.js, este a su vez invoca al servlet ServletActividad.java quien llama al controlador TblActividadController.java, quien a su vez, llama al modelo para interactuar con la clase TblActividad.java para poder acceder a la base de datos a través del controlador

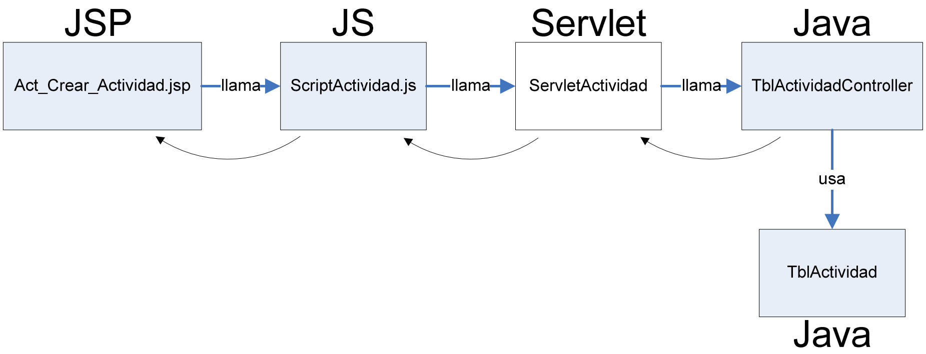

Figura 6. Diagrama de secuencia de creación de Actividad

2.5.1. Vista Compuesta. Este patrón fue utilizado para componer las vistas resultantes que serán visibles para los usuarios.

En la aplicación Peer to Peer usamos este patrón para el encabezado (header.jsp en las páginas mostradas antes de la autenticación y header_sesion.jsp en las páginas mostradas luego de la autenticación exitosa al sistema), para el pie de página (footer.jsp) y para la menú lateral (lateral.jsp). Esto quiere decir que las páginas JSP están compuestas por al menos 3 páginas adicionales.

Para ilustrar mediante código jsp (java server page) el uso del patrón, podemos apreciar en la figura 7 como se implementa la composición de vistas.

Como podemos notar a través de la sentencia include, se invoca a cada una de las partes que conforman la página principal.

```
        <div id="cuerpo_lateral">
            <jsp:include page="../lateral.jsp" />
        </div>
    <p class="clear"> </p>
        <div id="cuerpo_pie">
            <jsp:include page="../pie.jsp" />
        </div>
    </div>
</div>
    <div id="pie">
        <jsp:include page="../standares.jsp" />
    </div>
```

Figura 7. Código para componer páginas dentro de otras

2.6 Plan de Pruebas

Para desarrollar las pruebas del sistema se utilizarán la clasificación de pruebas por defecto y entre ellas las pruebas de caja negra.

Las pruebas funcionales o de caja negra son un enfoque para llevar a cabo pruebas donde estas se derivan de la especificación del programa o componente. El sistema es una "caja negra" cuyo comportamiento sólo se puede determinar estudiando las entradas y salidas relacionadas.

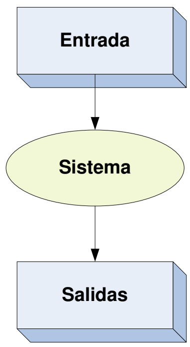

Figura 8 Flujo de pruebas de funcionalidades más representativas

Para realizar las pruebas se pidió la colaboración de voluntarios IEEE tener los roles de administrador GOLD y estudiantes. Se definió que se utilizaría 5 actividades para desarrollar las pruebas, cada actividad con 3 categorías asociadas y cada categoría con un mentor.

Para pruebas a, c, d, e, g, h, i, j se logró el 100% de efectividad entre lo que se esperaba que ocurriera y lo que realmente ocurrió.

Para las pruebas b y f, el sistema respondió como se esperaba, con la novedad que no se consideraron para los estudiantes pre-condiciones en el uso de estas funcionalidades, puesto que para crear una consultoría o actividad, el estudiante debió haber finalizado su actividad anterior para que el sistema le permita utilizar estas funcionalidades exitosamente.

3. Conclusiones y recomendaciones

Luego del desarrollo de este artículo, podemos concluir que:

Existe mayor beneficio al desarrollar una aplicación distribuida en capas usando patrones de diseño de J2EE, pues facilita el mantenimiento de la aplicación.

Se utilizó un framework de persistencia que mejoró la comunicación con la base de datos, pensamos que las aplicaciones que en su código utilizan comunicación directa con el motor de base de datos pueden llegar a ser más lentas que aquellas que usan una capa intermedia que realice este trabajo.

Al implementar el módulo de consultorías, se logró automatizar el almacenamiento de la comunicación entre 2 miembros IEEE en calidad de mentoría, que son tan necesarios en la comunidad Latinoamericana del IEEE.

Con el fin de incrementar la base de conocimiento, se recomienda promover el uso del sistema para aprovechar sus funcionalidades y mantener una comunicación interactiva entre estudiantes y mentores.

Es recomendable establecer métricas para medir la calidad del producto de software que se está desarrollando.

Se recomienda que si se va a añadir funcionalidad al portal se considere seguir utilizando el uso de patrones de la arquitectura J2EE.

Recomendamos aumentar funcionalidad a la aplicación, brindando otros servicios a los miembros estudiantiles tales como base de conocimiento, suscripciones RSS, múltiples idiomas, etc.

Para mayor seguridad en la información almacenada, se recomienda realizar actualizaciones de las herramientas en el servidor frecuentemente con nuevos parches.

Implementar una capa adicional a las vistas y al modelo para tener el sistema en varios idiomas, actualmente solo soporta el español porque será implementado para Latinoamérica.

Diseñar un módulo de datamining para analizar el liderazgo e todos los proyectos tanto de estudiantes como profesionales

4. Agradecimientos

Este artículo fue realizado gracias al apoyo del Comité GOLD de la Región Latinoamericana del IEEE y del Comité de actividades estudiantiles de la R9 2008-2009.

5. Referencias

[1] IEEE Webmaster, About IEEE, http://www.ieee.org/web/aboutus/home/index.html, consultado el 25 de septiembre de 2008.

[2] Java 2 Platform, Enterprise Edition (J2EE) Overview, http://java.sun.com/j2ee/overview.html ,consultado el 12 de Agosto de 2009.

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

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

[5] Java 2 Platform, Enterprise Edition (J2EE) Overview, http://java.sun.com/j2ee/overview.html