Remote Procedure Call con Enterprise JavaBeans

Juan Javier Malo Vega
jmalov@est.ups.edu.ec

***Resumen*— El presente documento describe el diseño e implementación de una aplicación cliente-servidor utilizando Java Enterprise Edition (Java EE), centrándose en las tecnologías de Enterprise JavaBeans (EJB), Java Naming and Directory Interface (JNDI), Swing, y Remote Procedure Call (RPC). Esta aplicación demuestra cómo se pueden emplear estas tecnologías integradas de Java EE para crear una solución empresarial robusta y escalable que gestione operaciones relacionadas con la gestión de usuarios.**

Palabras clave: RPC, EJB, cliente-servidor.

I. Introducción

En el ámbito de las aplicaciones empresariales, la necesidad de sistemas robustos, escalables y seguros es primordial. El desarrollo de tales aplicaciones exige el uso de tecnologías que no solo faciliten la implementación de funcionalidades complejas sino que también aseguren la integridad y confidencialidad de los datos procesados. Este informe presenta un análisis exhaustivo de una aplicación cliente-servidor implementada utilizando Java Enterprise Edition (Java EE), destacando el uso de Enterprise JavaBeans (EJB), Java Naming and Directory Interface (JNDI), Remote Procedure Call (RPC), y la interfaz gráfica de usuario Swing.

El sistema desarrollado se centra en la gestión de usuarios, permitiendo operaciones como la creación, consulta y administración de registros de usuarios a través de una interfaz cliente intuitiva y un servidor robusto. Este documento detalla la estructura del código, la interacción entre el cliente y el servidor, y los principios teóricos que fundamentan la arquitectura del sistema. El objetivo es proporcionar una visión clara del diseño y funcionamiento del sistema, subrayando cómo las tecnologías de Java EE se integran para ofrecer una solución empresarial eficaz.

II. Marco Teórico

A partir de esta sección, se desarrollan los contenidos del tema, de una forma ordenada y secuencial. Nótese que la sección debe ir organizada usando títulos como el anterior para cada tema nuevo incluido. Aparte, se incluyen subtítulos como el siguiente.

A. Enterprise JavaBeans (EJB)

EJB es una arquitectura de componentes del lado del servidor que simplifica el desarrollo de aplicaciones empresariales. Proporciona un sistema para construir componentes modulares y reutilizables que encapsulan la lógica de negocio de una aplicación.

- Session Beans: Manejan la lógica de negocio y pueden ser sin estado (stateless) o con estado (stateful). Los beans sin estado no mantienen información entre llamadas de método, mientras que los beans con estado sí. [1]
- Message-Driven Beans: Están orientados a mensajes y se utilizan para manejar asincronías en la lógica de negocio a través de JMS (Java Messaging Service).

B. Java Naming and Directory Interface (JNDI)

JNDI ofrece una API para acceder a una variedad de servicios de directorio. En Java EE, JNDI es utilizado para localizar EJBs y recursos de datos, lo que facilita la decoupling entre el código de la aplicación y la ubicación o acceso a los recursos empresariales.

C. Remote Procedure Call (RPC)

Es una técnica poderosa que permite a las aplicaciones cliente invocar procedimientos y métodos que se ejecutan en un servidor remoto como si fueran ejecutados localmente. En el contexto de Java Enterprise Edition (Java EE), RPC es un componente esencial para facilitar la comunicación entre el cliente y el servidor, permitiendo así un diseño de arquitectura distribuida en el que las tareas de procesamiento se pueden dividir eficazmente entre múltiples sistemas.

- RPC abstrae la complejidad de la red permitiendo que los desarrolladores se concentren en la lógica de negocio sin preocuparse por los detalles de la comunicación entre procesos. [3]
- Los clientes utilizan interfaces conocidas sin necesidad de conocer los detalles de la conexión de red o la ubicación física del servidor.
- Tanto el cliente como el servidor deben acordar una interfaz común.[4]

D. Swing

Swing proporciona un conjunto rico de componentes para la creación de interfaces gráficas de usuario (GUIs) en aplicaciones Java. Es parte de Java Foundation Classes (JFC) y ofrece una variedad de widgets como botones, listas, tablas, etc., así como la capacidad de personalizar su apariencia y comportamiento.

E. Modelo Cliente-Servidor

En este modelo, el cliente realiza peticiones a otro programa, el servidor, que le da servicio. Este modelo es base para muchas aplicaciones empresariales que distribuyen la carga entre los clientes y los servidores que manejan diferentes aspectos de la lógica de negocio, datos o presentación.[2]

F. Seguridad y Transacciones

Los componentes de Java EE están diseñados para trabajar en entornos distribuidos y heterogéneos, lo que asegura que las aplicaciones puedan escalar para manejar un aumento de carga y facilita la integración con otros sistemas y tecnologías.

G. Interoperabilidad y Escalabilidad

RPC permite que un programa invoque procedimientos (métodos) en un espacio de direcciones diferente, que en el

contexto de Java EE, típicamente sería en un servidor remoto. Esto es fundamental en arquitecturas cliente-servidor donde el cliente necesita acceder a servicios que se ejecutan en el servidor.

III. Práctica Realizada

Todos los párrafos deben tener intentos o tabulaciones en la primera línea. También, todos los párrafos deben estar alineados de forma justificada y hacia la izquierda.

A. Server

El sistema está compuesto por las siguientes clases e interfaces principales, ubicadas en los paquetes model, controller y service:

- Usuario: Es la entidad que representa un usuario en el sistema. Esta clase es serializable y se mapea a una tabla users en la base de datos. Incluye atributos como cédula, nombre, correo, celular, automóvil, y tipo de sangre.

- ServicioUsuario: Esta clase es un EJB sin estado que implementa tanto ServicioUsuarioLocal como ServicioUsuarioRemoto. Gestiona operaciones de ingreso y lista para la entidad Usuario, aprovechando la gestión de persistencia proporcionada por el EntityManager guardamos en la base de datos por defecto de Wildfly.

- ServicioUsuarioLocal: Interfaz local que declara los métodos crearUsuario y listarUsuarios. Estos métodos son utilizados por componentes que se ejecutan en el mismo entorno de ejecución, lo que proporciona un acceso más eficiente y rápido.

- ServicioUsuarioRemoto: Interfaz remota que declara métodos idénticos a la interfaz local. Esta interfaz facilita la interacción con el servicio desde componentes cliente que operan en un entorno distribuido, permitiendo el acceso a través de la red.

- Controlador: Es un bean gestionado que se encarga de la interacción con la vista y el modelo en el contexto de una solicitud. Utiliza ServicioUsuarioLocal para realizar operaciones como la creación de un nuevo usuario.

Funcionalidades y Operaciones:

- Creación de Usuario: El método crearUsuario permite registrar nuevos usuarios en la base de datos. Esta operación se realiza a través de la clase ServicioUsuario, que utiliza el EntityManager para persistir la entidad Usuario.

- Listado de Usuarios: El método listarUsuarios genera una consulta JPQL para obtener todos los usuarios registrados en la base de datos y los devuelve en forma de lista.

B. Cliente

El cliente está construido utilizando Java Swing para la interfaz gráfica y se comunica con el servidor a través de una interfaz remota definida por ServicioUsuarioRemoto.

- Main (main.Main): Esta clase contiene la lógica principal del cliente y la interfaz de usuario. Se encarga de inicializar la GUI y manejar eventos para la creación y listado de usuarios.

- Interfaz de Usuario: Utiliza Swing para ofrecer una interfaz gráfica que permite a los usuarios introducir datos como cédula, nombre, correo, celular, auto y tipo de sangre, y realizar operaciones como registrar y listar usuarios.

- Inicialización y Configuración JNDI: El cliente inicializa una conexión JNDI para localizar el servicio remoto ServicioUsuarioRemoto mediante la especificación del factory y la URL del proveedor.

Funcionalidades del Cliente

- Registro de Usuario: Permite a los usuarios ingresar datos en los campos proporcionados y registrar un nuevo usuario en el sistema a través del método crearUsuario del servicio remoto. La GUI proporciona retroalimentación directa sobre el éxito o fallo de la operación.

- Listado de Usuarios: Ofrece la capacidad de listar todos los usuarios registrados en el sistema, mostrando la información detallada de cada usuario en un área de texto.

Proceso de Operación

- Inicialización de Servicios: Al iniciar, el cliente establece una conexión con el servidor utilizando configuraciones JNDI y recupera la referencia al servicio remoto ServicioUsuarioRemoto.

- Interacción con la GUI: A través de la interfaz gráfica, los usuarios pueden realizar acciones como registrar y listar usuarios. Los botones y campos de texto están diseñados para ser intuitivos y fáciles de usar.

- Manejo de Excepciones: El cliente maneja excepciones adecuadamente, mostrando mensajes de error en la interfaz gráfica y permitiendo a los usuarios corregir los datos de entrada antes de reiniciar operaciones.

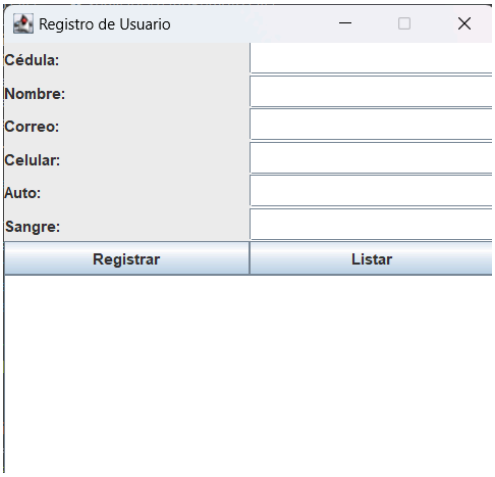

Fig. 1 Interfaz de usuario

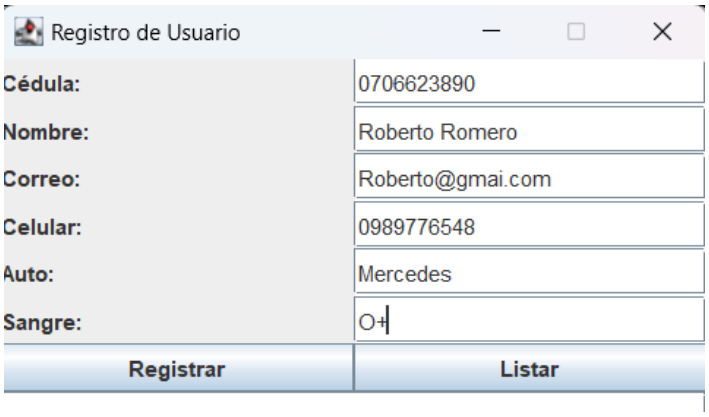

Fig. 2 Ingreso datos de usuario

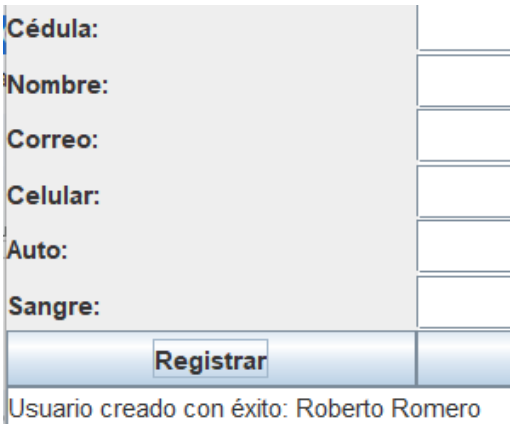

Fig. 3 Comprobación de usuario ingresado

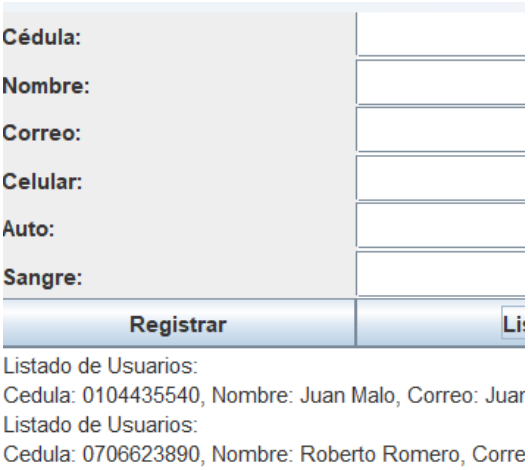

Fig. 4 Listando los usuarios guardados en la base de datos

IV. Conclusiones

La combinación de teoría y práctica en el desarrollo de aplicaciones cliente-servidor utilizando Java Enterprise Edition (Java EE) ilustra un enfoque robusto y sofisticado para resolver problemas complejos en entornos empresariales. La teoría proporciona los fundamentos y principios que guían el diseño y la implementación de sistemas distribuidos, mientras que la aplicación práctica de estas teorías en herramientas como Enterprise JavaBeans (EJB), Java Naming and Directory Interface (JNDI), Swing, y Remote Procedure Call (RPC) demuestra la efectividad de Java EE en la creación de aplicaciones escalables, seguras y mantenibles. RPC se centra en la capacidad de un sistema para ejecutar código a través de una red como si fuera local. En la práctica, RPC en Java EE facilita la comunicación entre el cliente y el servidor, permitiendo operaciones complejas y gestión de datos a través de interfaces bien definidas, lo cual es esencial para la coherencia y escalabilidad de la aplicación. JNDI permite a las aplicaciones cliente-servidor localizar y llamar a servicios EJB de manera eficiente, lo cual es crucial para la integridad y el rendimiento del sistema.

V. Referencias

[1] Monson-Haefel, R. (2001). Enterprise JavaBeans (3rd Edition). O'Reilly Media.

[2] Tanenbaum, A. S., & Van Steen, M. (2007). Distributed Systems: Principles and Paradigms (2nd Edition). Prentice Hall.

[3] Birrell, A. D., & Nelson, B. J. (1984). Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, 2(1), 39-59.

[4] Corbin, J. R. (1991). The Art of Distributed Applications: Programming Techniques for Remote Procedure Calls. Springer-Verlag.