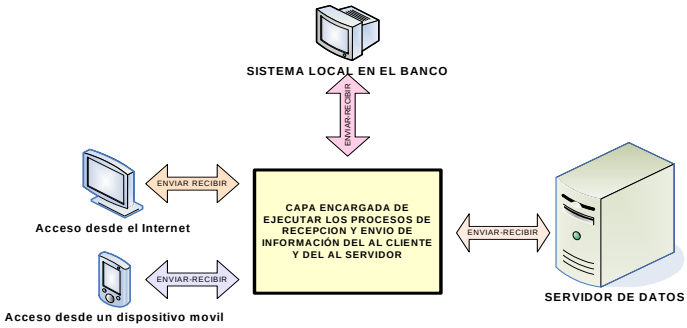
CAPITULO III

Análisis de Arquitecturas N-capas con J2EE

J2EE proporciona diferentes tipos de arquitecturas para el desarrollo de aplicaciones, cada una de estas muy funcionales dependiente al tipo de aplicación que se este construyendo o al criterio del desarrollador.

Este capitulo contiene una análisis de los posibles soluciones que se pueden presentar en el momento de desarrollar aplicaciones con J2EE



1. Posibles Escenarios
2. Soluciones Generales para la Arquitectura
3. Arquitectura Seleccionada

3.1 POSIBLES ESCENARIOS

J2EE es bastante flexible para aplicaciones que apoyan una variedad de tipos de cliente con el contenedor Web o el contenedor EJB, además, los diferentes tipos de escenarios de la aplicación J2EE permite establecer diversas configuraciones para que las aplicaciones puedan ejecutarse de manera más óptima.

3.1.1 Aplicaciones Multicapa

Las aplicaciones multicapa están diseñadas para que cada capa realice un trabajo específico sin conflicto con las demás, por ejemplo: el contenedor Web organiza los componentes de Web que casi se dedican exclusivamente a ocuparse de la lógica de presentación (JSP, Servlets), el contendor de EJB organiza los componentes de la aplicación que usan los recursos del almacén de datos para satisfacer las demandas de los componentes del contenedor Web; esta arquitectura es implícitamente escalable dependiendo del diseño de la aplicación y las necesidades del cliente. **[www001]**

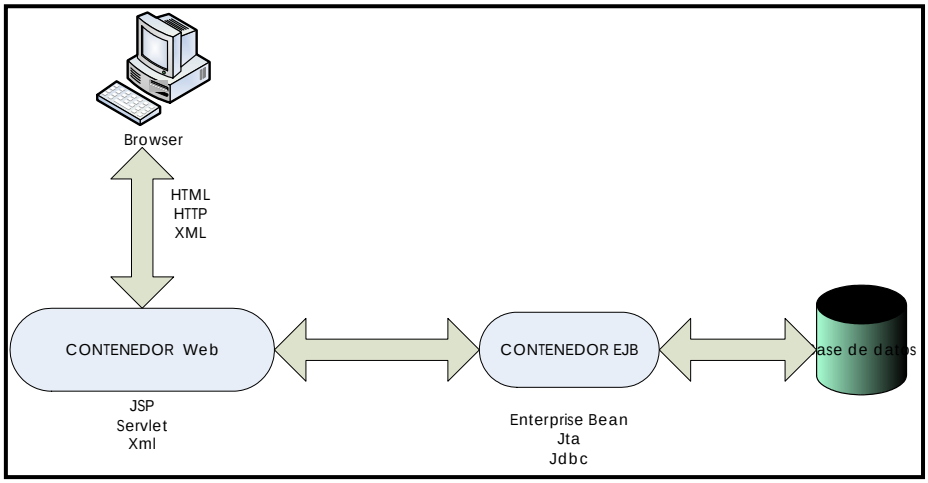


Figura 3.1: Aplicaciones Multicapas

El siguiente ejemplo muestra una arquitectura de J2EE basada en la necesidad de una empresa:

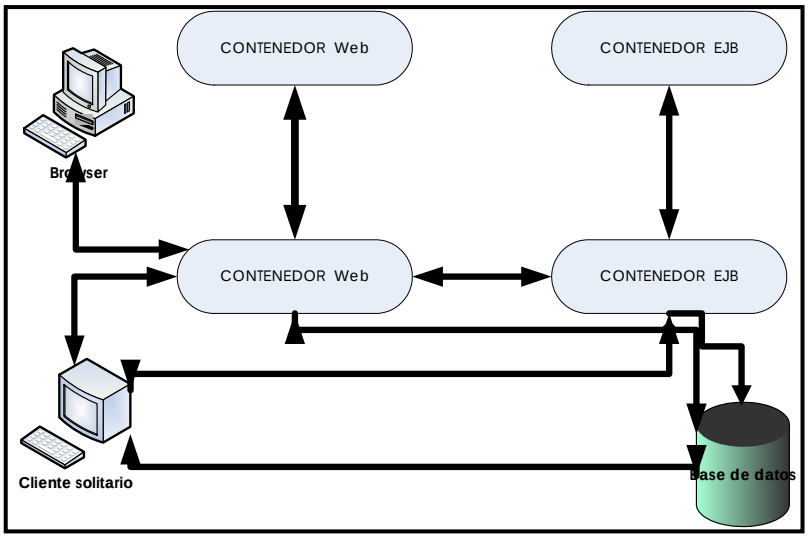


Figura 3.2: Comunicación entre Contenedores Web y EJB

La aplicación de la figura 3.2 es multicapa que usa ambos un contenedor de Web y un contenedor de EJB. Los requisitos existentes para diseñar una aplicación idéntica a este modelo serían:

- ✓ La necesidad de hacer cambios rápidos y frecuentes a la aplicación.
- ✓ La necesidad de dividir la aplicación a lo largo de las líneas de presentación y la lógica de negocio para aumentar el modelamiento.
- ✓ La necesidad de simplificar el proceso de asignar los recursos humanos adecuadamente entrenados para lograr la tarea de desarrollo.
- ✓ La necesidad de tener diseñadores familiarizados con las aplicaciones de lado del cliente.
- ✓ La necesidad de tener el vocabulario necesario para comunicar la lógica comercial a otros equipos teniendo relación con los factores humanos y la estética de la aplicación
- ✓ La habilidad de congregar en las aplicaciones una gran variedad de componentes de diferentes fuentes, incluso independientemente de el lenguaje que usa la lógica comercial.

- ✓ La habilidad de desplegar los componentes transaccionales en múltiples plataformas de software y de hardware independientemente de la tecnología base de la aplicación.

Es decir que este modelo promueve el posible crecimiento de la aplicación, a través de la reutilización de código.

3.1.2 Cliente Autosuficiente

El Cliente autosuficiente puede acceder a un almacén de datos directamente si necesidad de establecer comunicación con el contenedor Web o el EJB. Esto se lo hace por medio de un JDBC o un conector directo. **[www001]**

El cliente puede estar programado en un lenguaje como java o en cualquier otro tipo de lenguaje. De esta forma la presentación como la lógica comercial se encuentran integradas en la aplicación de cliente.

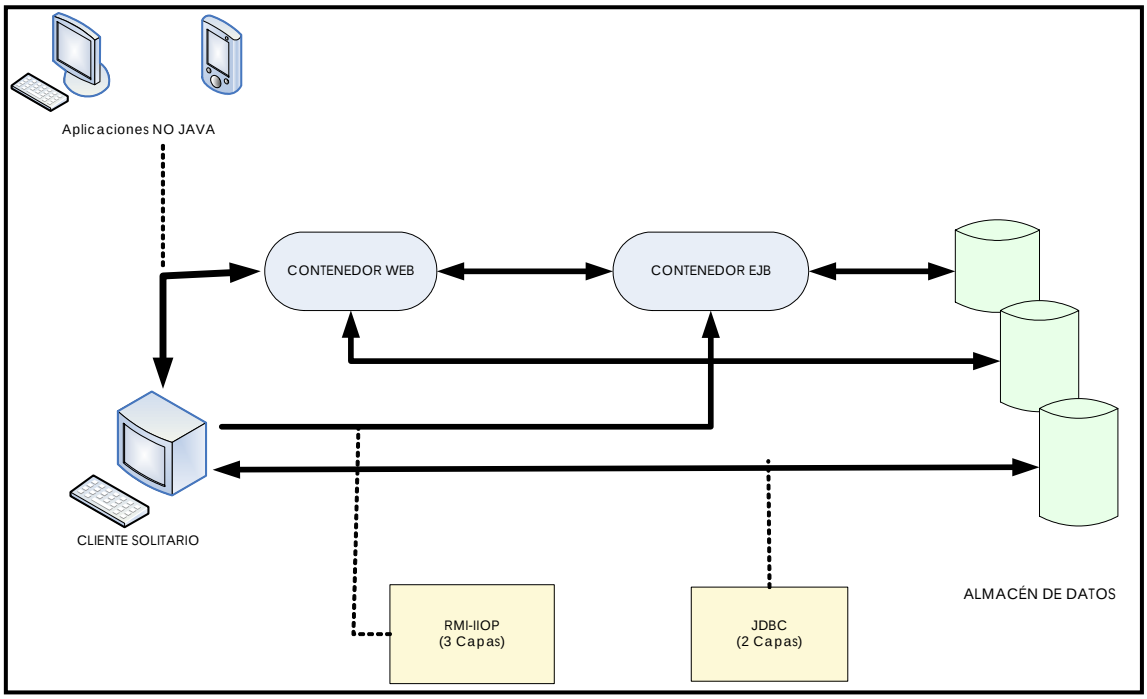


Figura 3.3: Cliente Solitario

3.1.3 Aplicaciones centradas en la Web

Las aplicaciones centradas en la Web mantienen a la capa de presentación y la capa de negocio juntas. El cliente debe estar programado ya sea con Servlets, JSP o algún caso con XML. Estos tendrán acceso a la base de datos por medio de un JDBC. **[www001]**

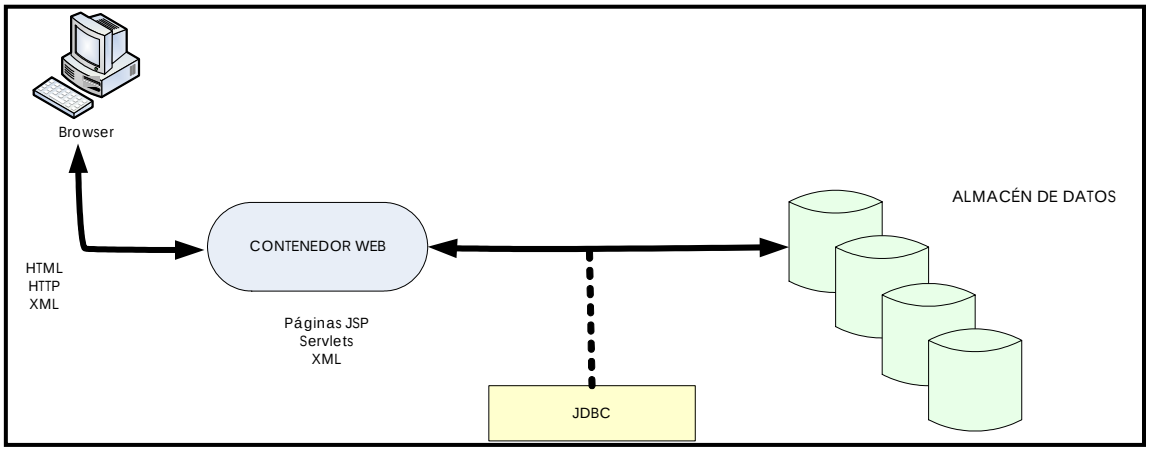


Figura 3.4: Aplicación Centralizada

La figura 3.5 muestra esquemáticamente los componentes del contenedor Web el cual mantiene comunicación con el almacén de datos por medio de un JDBC u otro tipo de conector.

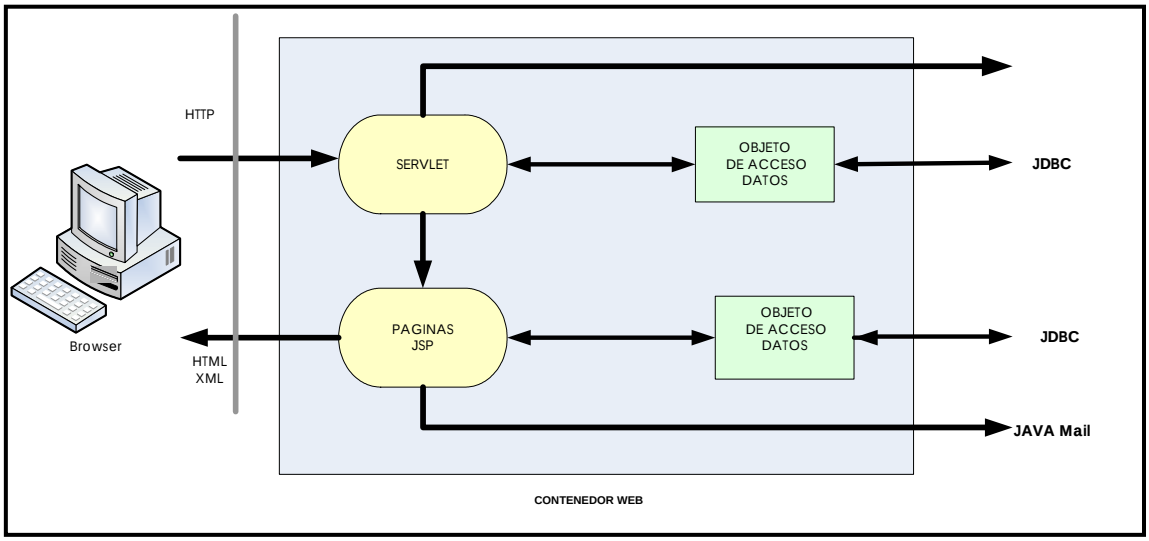


Figura 3.5: Contenedor Web

3.1.4 B2B (Business to Business)

El escenario B2B dispone en su arquitectura la utilización identificada de la capa de presentación y la de negocios, es decir que la una no esta inmersa en la otra.

El escenario B2B a más de la programación JAVA se hace necesario la interacción con XML para mantener un corrector flujo de datos entre las diferentes capas. Además dispone de la capacidad de comunicarse con otro tipo Servicios Web a través de RMI. **[www001]**

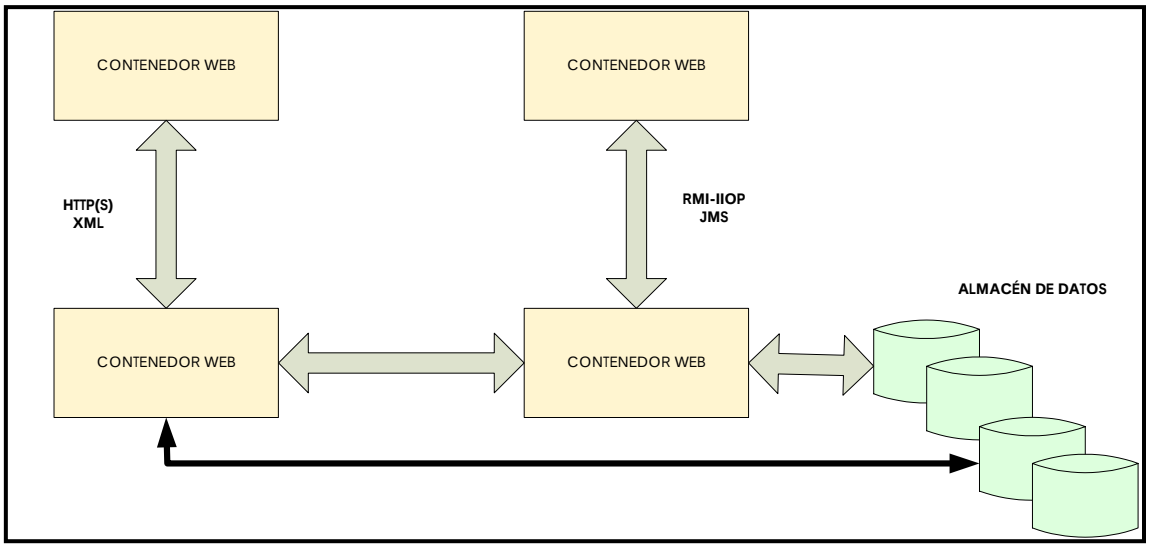


Figura 3.6: B2B

Las posibilidades en B2B nos permiten establecer comunicación con el Almacén de Datos directamente a través del Contenedor Web o también del contendor Web al contenedor EJB y este Almacén de Datos.

3.2 SOLUCIONES GENERALES PARA LA ARQUITECTURA

Si bien los diferentes escenarios de J2EE presentados pueden dar a pensar que son las únicas formas de crear aplicaciones, pues no lo son, las aplicaciones de J2EE antes de ser dependiente de la arquitectura son muchos mas dependientes de las necesidades de la aplicación y del cliente.

Las necesidades del cliente son las que impondrán que tipo de escenario va ha ser el utilizado o aún más si se va ha combinar las características de un escenario con el de otro.

A continuación se van ha presentar algunas necesidades de aplicación y sus posibles soluciones:

Una empresa que esta orientada al sector bancario, además de necesitar un sistema de cliente local, necesita de un portal para servicios bancarios al cliente por la Web y par dispositivos móviles.

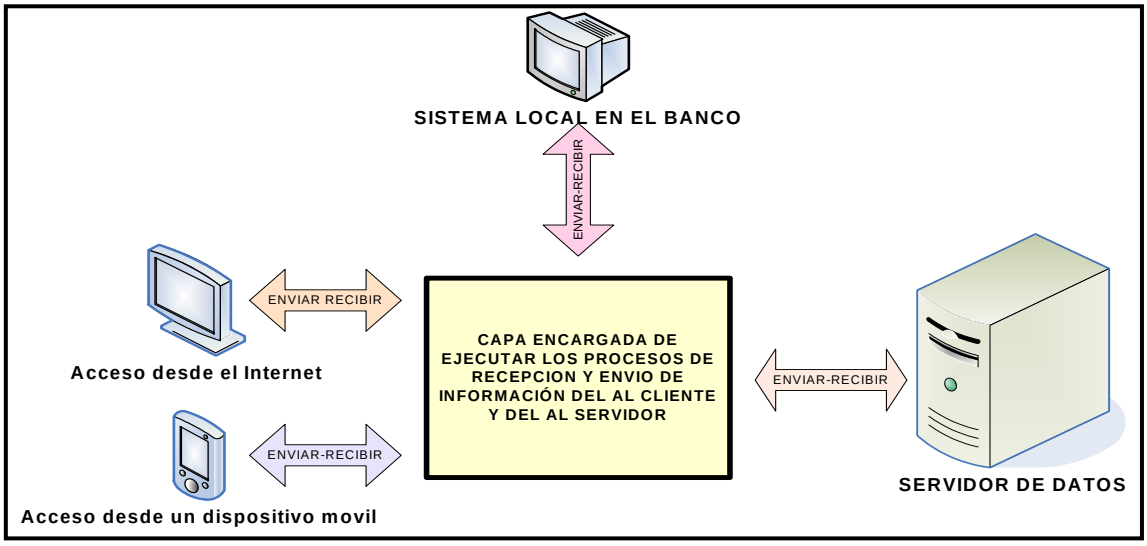


Figura 3.7: Sistema Bancario

El problema planteado presenta un sin número de soluciones, pero, la solución más óptima es aquella que nos permita, desde el punto de vista del desarrollador:

- ✓ La re-utilización de código, es decir, optimizar el código para no estar haciendo aplicaciones en un 100% diferentes entre el cliente local, Web.
- ✓ La capacidad de dar un ***mantenimiento*** al sistema sin tener la necesidad de deshabilitarlo por completo.
- ✓ Mantener la seguridad de los datos, es decir la presentación lo más alejada del acceso a datos.
- ✓ La capacidad de reducir al máximo los errores de accesos y sobre carga de los servidores. Esto es muy importante ya que si bien sabemos en tiempo de diseño los accesos no son muchos, una vez implantado el sistema a la realidad los accesos serán muy numerosos.

La solución presentada es muy semejante al escenario B2B, debido a que se hace necesario la utilización de EJB para poder reutilizar el código para todas las áreas que necesitan ser resueltas.

La lógica de presentación se mantiene alejada de la de datos, esto da la seguridad necesario y necesitada por el cliente.

Otro ejemplo: Una empresa de insumos necesita ir un paso más adelante en el mercado, por lo que se hace necesario un Portal Web para darse a conocer en el mundo, además sus empleados facturan por medio de dispositivos portátiles (***PALM*** o PDA`s), estos datos no son ingresados en línea debido al alto costo de mantenerlos permanentemente conectados a la Web, sino, son ingresados por infrarrojo o cable en el servidor desde una Terminal.

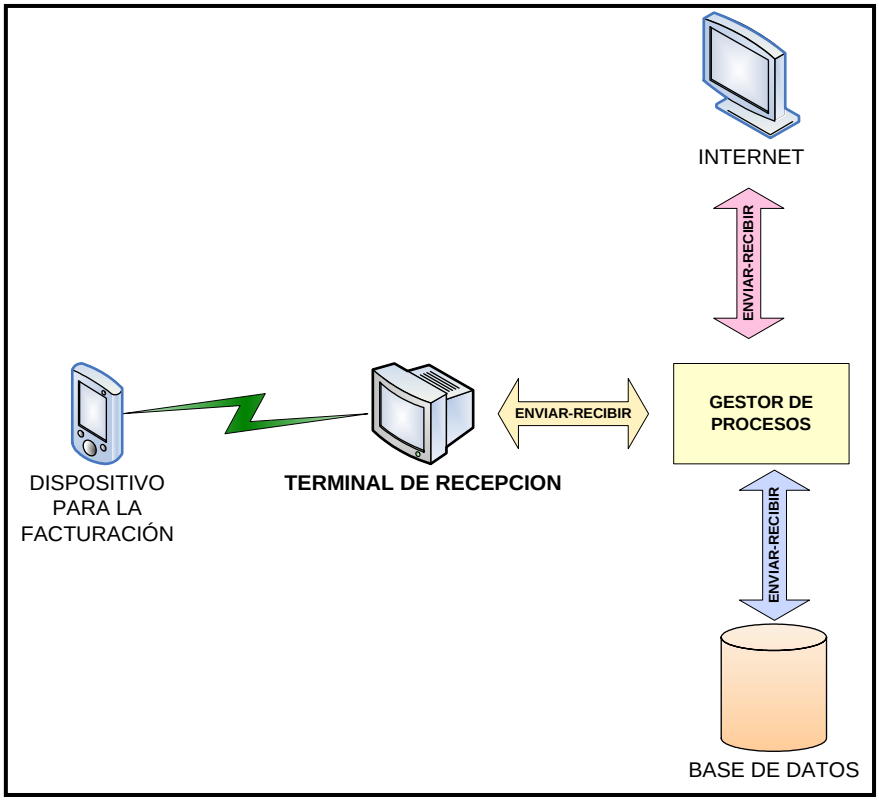


Figura 3.8: Sistema con Dispositivos Móviles

Al analizar una posible solución para este problema puedo decir que:

- ✓ Se hace necesaria la implementación de un portal Web con una seguridad moderada debido a que solo es un sitio de Información donde no implica por ningún motivo transacción de nivel monetario.
- ✓ Se hace necesario implementar una aplicación que pueda correr en los dispositivos portátiles, que accedan a una Base de Datos temporal ubicada en el dispositivo.
- ✓ Y por último una aplicación de cliente que permita transportar los datos del dispositivo móvil al Servidor.

Se ve que se hace necesario como en el ejemplo anterior mantener un esquema de reutilización de código para no tener que rehacer funciones y procedimientos.

Como fue dicho la seguridad es moderada, pero en ningún motivo nula.

Los esquemas presentados reúnen las características del escenario Cliente Único y Aplicación centralizada en la Web, e incluso como vemos B2B. En pocas palabras muchos caminos para la solución.

La aplicación que irá en el dispositivo móvil necesariamente tendrá que tener las características de Cliente Único. Debido a que como fue dicho va ha utilizar temporalmente un almacenamiento local.

La aplicación que irá en el Terminal en cambio puede ser del tipo de conexión por JDBC. Este será el encargado de comunicarse con el dispositivo móvil y a su vez conectarse al servidor de datos y migrar la información

Para finalizar a aplicación Web será de cliente centralizada, tal vez se pueda optimizar el código con el aumento de EJB, pero al final se tendrá el mismo resultado.

Como último ejemplo: Una empresa necesita crear una nueva aplicación que interactúe con un sistema diseñado en un lenguaje y plataforma diferente.

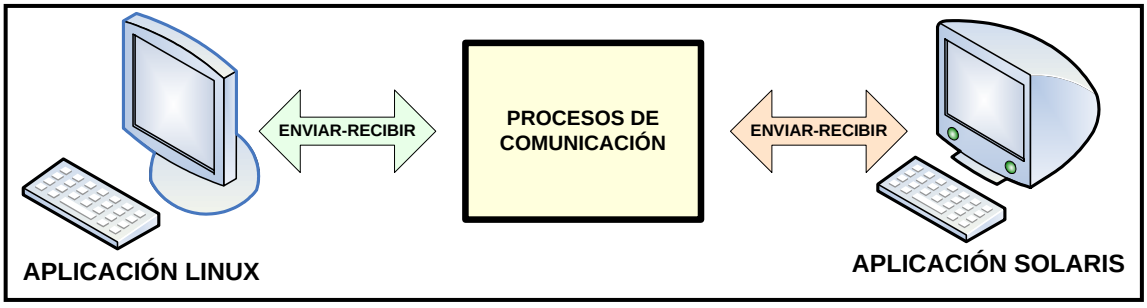


Figura 3.9: Aplicaciones en diferentes Plataformas

Este problema nos refiere directamente a la utilización de EJB ya que se puede diseñar algunos para que interactúen con el sistema y estos puedan ser llamados por la nueva aplicación.

Se pude observar que los sistemas que son diseñados no nos orientan necesariamente a la Web a pesar de ser el mejor candidato.

3.3 ARQUITECTURA SELECCIONADA

El escenario B2B es el que mejor se adapta a los requerimientos del sistema a desarrollar. La gran modulabilidad entre la capa de presentación y la capa de negocios hacen que esta se ha seleccionada para el desarrollo de la aplicación.

La necesidad de mantener la capa de negocios separada de la presentación hace que tanto el escenario del Cliente Autosuficiente como la de Aplicaciones centradas en la Web, no sean las más óptimas, además de:

- ✓ El cliente autosuficiente se restringe demasiado a la plataforma de cliente, y mantiene todo en una sola aplicación esto hace que se dificulte hacer algún cambio sin correr el riesgo de alterar el acceso a datos.
- ✓ Las aplicaciones centradas en Web, a pesar de ser una opción bastante aceptable ya que mantiene al cliente independiente del acceso a datos como a la presentación, no se la ha tomado en cuenta por la razón expuesta al inicio de mantener la capa de presentación y de negocios separada.
- ✓ Además ninguna de estas presenta la escalabilidad que presenta la arquitectura del escenario B2B.

Al mantener la capa de presentación separada de la de negocios nos permite expandir las posibilidades de la aplicación, de ser estrictamente de plataforma o de ser estrictamente de Web a ser una aplicación que puede ser implementable en cualquier tipo de plataforma inclusive la móvil sin tener que modificar la capa de negocios que va ha ser la única que interactúa directamente con la Información.

La capa de presentación puede ser modificada independientemente sin correr el riesgo que el acceso a la información resulte alterado en el transcurso.