

## LÓGICA DEL NEGOCIO DE LA EMPRESA EN UNA CAPA INTERMEDIA

### Resumen / Abstract

El trabajo que se presenta sugiere dos maneras de afrontar el desarrollo de un sistema informático para la empresa en un ambiente colaborativo. Teniendo en cuenta la dinámica actual y las necesidades de mantenimiento inmediato a los sistemas automatizados. Se comparan los modelos de dos y tres capas para la arquitectura cliente-servidor haciendo especial énfasis en las ventajas del modelo de tres capas por su facilidad de extensión a diferentes entornos de desarrollo de la aplicación cliente sin necesidad de grandes variaciones en los procesos de manipulación de los datos.

*The work that is presented suggests two ways to confront the development of an informative system for the company in multi-user environment. Keeping in mind the current dynamics and the necessities of immediate maintenance to the automated systems. The models of two and three layers for the client-server architecture are compared making special emphasis in the advantages of the three layers model for its extension easiness to different development's environment of the client application without necessity of big variations in processes of data's manipulation.*

### Palabras claves / Key words

Cliente-servidor, modelo de tres capas, modelo de dos capas, lógica del negocio

*Client-serves, three layers model, two layers model, business logic*

## INTRODUCCIÓN

Cuando se habla de conexión entre el mundo de los negocios y el mundo de las computadoras se destacan dos conceptos: las reglas de negocio y los requerimientos del negocio.

Todo sistema automatizado trata de reflejar parte del funcionamiento del mundo real, para automatizar tareas que de otro modo serían llevadas a cabo de manera ineficiente, o bien, no podrían realizarse. Para ello, es necesario que cada aplicación refleje las restricciones que existen en el negocio que está modelando, de forma que se imposibilite la realización de acciones no válidas. A las reglas que debe seguir la aplicación para garantizar esto se les llama reglas de negocio. Ejemplos de tales reglas son: no permitir crear facturas pertenecientes a clientes inexistentes, controlar que el saldo negativo de un cliente nunca sobrepase cierta cantidad, etcétera.

Los requerimientos del negocio, por su parte, son restricciones, usualmente impuestas por la propia compañía o entidad, que está utilizando para ayudarse a operar en su ambiente particular de negocio.

Los requerimientos y reglas del negocio definen una línea para los requerimientos de la aplicación y proporcionan una guía para los desarrolladores. En términos prácticos, estos elementos de negocio son metas que los desarrolladores se esfuerzan por conocer para sus aplicaciones.

---

**Yoelys Ronda Amador**, Ingeniera Informática, Instructora, Centro de Estudios de Ingeniería y Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría (ISPJAE), Ciudad de La Habana  
e-mail: yoelys@ceis.ispjae.edu.cu  
**Kadir Cobas Santos**, Ingeniero Informático, CEIS, ISPJAE  
**Daymel Marrero Viñas**, Ingeniero Informático, CEIS, ISPJAE, Ciudad de La Habana

Juntos, las reglas y los requerimientos de negocio proporcionan un entorno para desarrollar aplicaciones que pueden ayudar a la compañía a alcanzar sus metas.<sup>1</sup>

Este trabajo propone una solución para enfrentar la automatización de las actividades de la empresa de manera que se cumplan los requerimientos y reglas del negocio en un medio multiusuario y que los cambios de estos no constituyan un freno en el correcto desempeño de los sistemas informativos minimizando el esfuerzo requerido para mantener una aplicación en un ambiente colaborativo.

## MODELO CLIENTE-SERVIDOR

El esquema cliente-servidor "es un modelo de computación en el que el procesamiento requerido para ejecutar una aplicación o conjunto de aplicaciones relacionadas se divide entre dos o más procesos que cooperan entre sí".<sup>2</sup> Usualmente, la mayoría del trabajo pesado se hace en el proceso llamado servidor y el o los procesos clientes solo se ocupan de la interacción con el usuario (aunque esto puede variar).<sup>3</sup>

Los principales componentes del esquema cliente-servidor son entonces los **clientes**, los **servidores** y la **infraestructura de comunicaciones**.

Los **clientes** interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.

Los **servidores** proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallas, y otros aspectos afines.

Para que los clientes y los servidores puedan comunicarse se requiere una **infraestructura de comunicaciones**, la cual proporciona los mecanismos básicos de direccionamiento y transporte.

### Modelo cliente-servidor de dos capas

Las aplicaciones cliente-servidor clásicas o de dos capas como su nombre lo indica agrupan la lógica de presentación (interfaz) y la lógica de aplicación en la máquina cliente y acceden a fuentes de datos compartidos a través de una conexión de red que se encuentran en el servidor de datos.

Estas aplicaciones de dos capas trabajan bien en aplicaciones a escala de departamentos con un modesto número de usuarios, una base de datos sencilla y una red segura y rápida (figura 1).<sup>4</sup>

La ventaja que presenta este tipo de aplicaciones es que los datos están centralizados. Esta centralización beneficia a la empresa pues es más fácil compartir los datos, se simplifica la generación de reportes y se proporciona consistencia en el acceso a los datos.

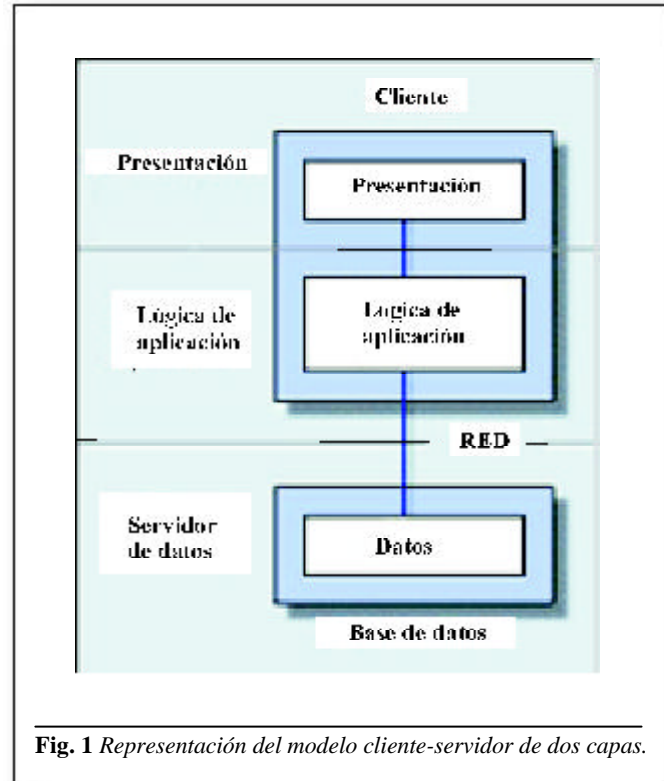


Fig. 1 Representación del modelo cliente-servidor de dos capas.

A continuación se enumeran las principales desventajas que presentan este tipo de aplicaciones:

- Dificiles de mantener: Esto viene dado por el hecho de que son difíciles de mantener las reglas de negocio de la lógica de aplicación ya que estas están programadas en cada cliente y esto implica que cualquier cambio tiene que ser redistribuido en todos los clientes.
- Se compromete la confidencialidad: Al tener programada la lógica de aplicación en el cliente este tiene a su disposición todas las reglas de negocio de la empresa.
- Están estrechamente limitadas a una fuente de datos: Los clientes casi siempre están configurados para acceder a una base de datos en particular por lo que mover los datos a una base de datos diferente se hace realmente complicado.

Para mejorar este tipo de aplicaciones se introdujeron los llamados procedimientos almacenados (del inglés *stored procedure* (SP)) los procedimientos almacenados son funciones de software precompiladas que se ejecutan dentro de la propia base de datos.

Una de las principales ventajas que trae consigo el uso de estos procedimientos almacenados es que, como se localizan en la propia base de datos pueden procesar la información en su misma fuente, sin tener que transferir a través de la red. Por ejemplo: se supone que usted desea calcular el promedio de notas de un estudiante. A la hora de efectuar este cálculo si no se emplean los procedimientos almacenados hay que transmitir los datos de todas las notas del estudiante por la red hasta el cliente y una vez allí coger todos los datos, hacer el cálculo y mostrar el

resultado al cliente. Ahora, si se usa un procedimiento almacenado llamado, por ejemplo, "spCalcularPromedio", lo único que se debe hacer desde el cliente es ejecutar el procedimiento y este, en el servidor, calcula el promedio y le envía la respuesta al cliente.

Las desventajas de usar esta forma de trabajo radican en que SQL (Structured Query Language), que es el lenguaje en el que se programan estos procedimientos almacenados, no es tan rico como el resto de los lenguajes de programación tradicionales, como Basic, C++, Java, Pascal, etc., además, estos tienen que ejecutarse en la misma máquina en la que estén los datos lo cual limita la escalabilidad de la aplicación.<sup>4</sup>

### MODELO CLIENTE-SERVIDOR DE TRES CAPAS

Como es la propia empresa la encargada de obligar a que se cumplan algunas reglas de negocio, es conveniente encontrar la manera de centralizar la gestión de estas reglas en un único lugar, de modo que todo el código necesario no se haya de duplicar en cada una de las aplicaciones.<sup>3,5</sup> La solución puede ser crear una aplicación que se encargue de llevar a cabo estas tareas, de modo que todos los clientes pidan o envíen información a la misma, no al gestor de base de datos en el servidor: a este solo accederá la nueva aplicación, que conforma una nueva capa dentro de un sistema Cliente-Servidor, la capa intermedia o middle-tier, con lo que nuestro sistema ha pasado de ser un sistema cliente-servidor convencional a ser un sistema con tres capas (*three-tiered*). En la figura 2 se muestra uno de ellos.

En la arquitectura de tres capas la presentación, la lógica de aplicación y los elementos de datos están conceptualmente separados. Los componentes de la capa de presentación manejan la interacción con el usuario y realizan las peticiones del cliente a los componentes de la capa intermedia. Los componentes de la capa intermedia, manipulan la lógica de negocio y hacen las peticiones a la base de datos (figura 3).

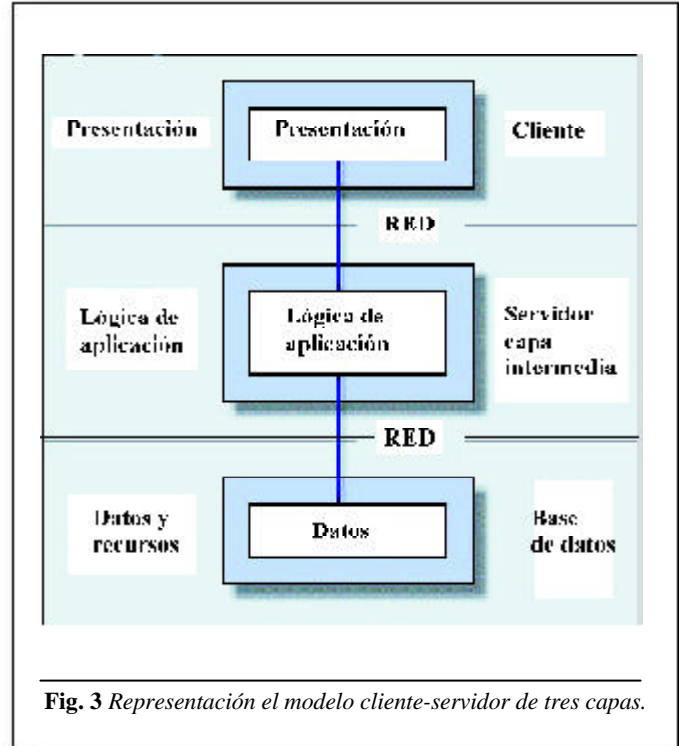


Fig. 3 Representación el modelo cliente-servidor de tres capas.

Una de las diferencias claves entre el modelo tradicional cliente-servidor de dos capas y el de tres capas es el uso de lo que es conocido como *middleware*. En la industria de la computación *middleware* es un término general que define a cualquier aplicación que sirve para "unir" o mediar entre dos aplicaciones que usualmente trabajan independientes. Algunos de los más populares tipos de *middleware* son los monitores de transacciones (transactions mMonitors) y los ORB (object reques brokers).<sup>6</sup>

**Monitores de transacciones (transactions monitors):** Los monitores de Transacciones trabajan prácticamente como policías del tráfico ya que estos se colocan entre los clientes y el servidor.

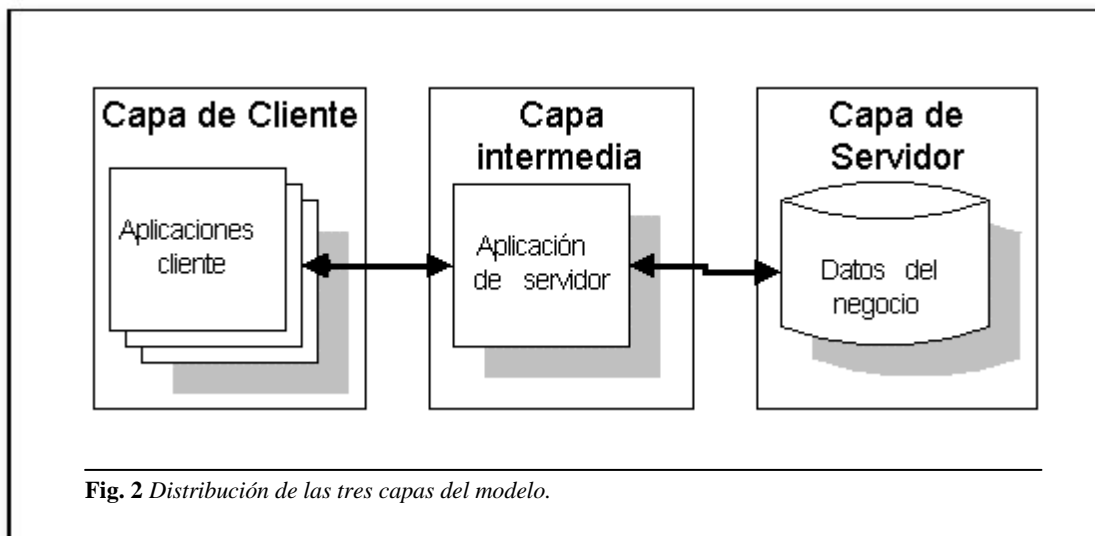


Fig. 2 Distribución de las tres capas del modelo.

le los datos y controlan el paso de las consultas o actualizaciones desde los clientes a la base de datos adecuada. Por otro lado dichos monitores mantienen abiertas solo aquellas conexiones a a base de datos que son necesarias, lo que trae consigo un mejor rendimiento para el trabajo de la base de datos.

**Object request brokers (ORB):** Los ORB caen en una categoría diferente a los Monitores de Transacciones. Estos proporcionan transparencia en la ubicación de los módulos y servicios de la aplicación. Esto significa que cuando una aplicación necesita interactuar con un proceso servidor no tiene que saber en qué lugar de la red se encuentra dicho proceso. La única ubicación que conoce la aplicación es la de su ORB *client stub* que se encuentra en la misma máquina que la aplicación.

El ORB *client stub* conoce dónde están corriendo todos los procesos servidores o en que máquina tienen que correr. Si un proceso en particular no está corriendo cuando una aplicación cliente solicita, el ORB inicia dicho proceso en una de las máquinas para las cuales el proceso está configurado para correr. La aplicación cliente no conoce en qué servidor el proceso solicitado está corriendo o si aún lo está. El ORB mantiene una pista de todos los procesos servidores para las aplicaciones clientes.

A continuación se enumeran algunas ventajas de las aplicaciones de tres capas:<sup>7,8</sup>

- Los componentes de la aplicación pueden ser desarrollados en cualquier lenguaje general lo que posibilita que el grupo de desarrolladores no se centre en el uso de un solo lenguaje.
- Los cambios que se realicen en la lógica del negocio afectan, en la mayoría de los casos, solamente a la capa intermedia.
- Los componentes están centralizados lo que posibilita su fácil desarrollo, mantenimiento y uso.
- Los componentes de la aplicación pueden estar esparcidos en múltiples servidores permitiendo una mayor escalabilidad.
- Los problemas de limitación para las conexiones a las bases de datos se minimizan ya que la base de datos solo es vista desde la capa intermedia y no desde todos los clientes. Además que las conexiones y los *drivers* de las bases de datos no tienen que estar en los clientes.
- Los componentes de aplicación de la capa intermedia pueden ser asegurados centralmente usando una infraestructura común. Se pueden conceder o denegar los permisos componente a componente simplificando la administración.<sup>4</sup>

## CONCLUSIONES

El empleo del modelo cliente/servidor de tres capas facilita el mantenimiento de la aplicación. Aunque se ha planteado que la lógica del negocio, el cliente y el servidor de datos estén físicamente separados, se garantiza un funcionamiento similar en el caso que se decida ubicar en la máquina servidor los procedimientos relacionados con la lógica del negocio. La ventaja de disponer de una aplicación cliente que no tiene que conocer necesariamente los elementos de la lógica del negocio facilitan la diversidad de lenguajes de programación empleados para esta capa y su posterior reutilización en interfaces cliente diferentes. [3]

## REFERENCIAS

1. **AGULLÓ SOLIVERES, PEDRO.:** "Desarrollo Cliente/Servidor: ubicación de las reglas del negocio", Revista *Profesional para*

*programadores*, 1997.

2. **Datapro Client/Server Analyst, Client Server Computing: Emerging Trends, Solutions and Strategies**, 1994.
3. **RODRÍGUEZ, ALBERTO:** "Diseño de aAplicaciones Three Tier", 12/12/01
4. **CDROM Mastering Distributed Application Design and Development version 1.0**, Microsoft Corporation, 1998.
5. **PLANK, RICARDO:** "Tecnologías para el desarrollo de aplicaciones en la web (Parte II)", 12/12/01.
6. **Web Deployment Application with Microsoft Visual Basic 5.0**, McGraw Hill, 1999.
7. **ANTELO JIMÉNEZ, ERICK:** "Diseño de aplicaciones en tres capas con Windows DNA", 11/02/02.
8. **CORCHUELO GIL, RAFAEL:** "Desarrollo de aplicaciones en tres capas", 11/02/02

## REVISTAS CIENTÍFICAS DEL ISPJAE



- <http://intranet/ediciones/default.htm>
- [www.ispjae.edu.cu/ediciones/default.htm](http://www.ispjae.edu.cu/ediciones/default.htm)