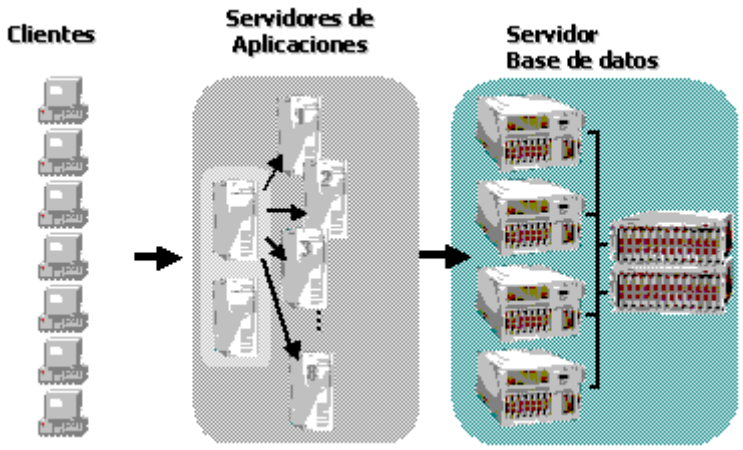
Aplicaciones distribuidas

El generador Java permite generar aplicaciones distribuidas, conocidas también como aplicaciones en tres capas.

A diferencia de las aplicaciones Cliente/Servidor tradicionales (dos capas), donde la ejecución de todo el código de la aplicación se realiza en el cliente, en una aplicación tres capas, se distribuye el código; ejecutando parte en el cliente y parte en el servidor de aplicaciones.



Servidores de Aplicaciones

Los servidores de aplicaciones se encargan de ejecutar el código que fue definido como remoto en una aplicación distribuida.

Cuando el cliente requiere algo del servidor (por ejemplo datos), se comunica con el servidor de aplicaciones solicitándole que ejecute determinado proceso remoto con tales parámetros. El servidor de aplicaciones dispara dicho proceso, quien le devolverá los resultados y éste se comunicará con el cliente para enviárselos.

Los clientes se comunican con el servidor de aplicaciones a través de algún protocolo de comunicaciones

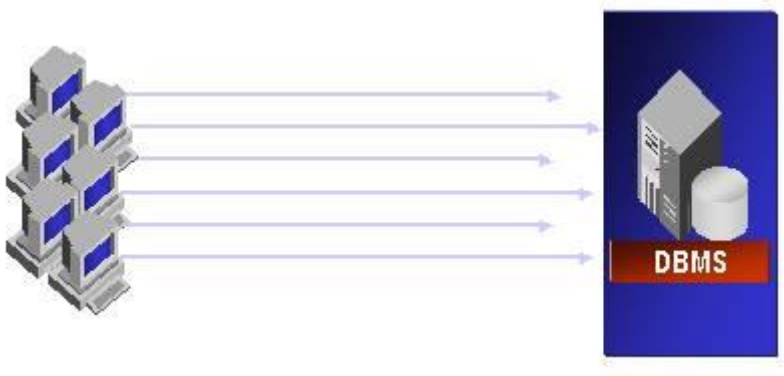
Protocolos de comunicaciones

Los soportados por el generador son Remote Method Invocation (**RMI**), Common Object Request Broker Architecture (**CORBA**), Distributed Component Object Model (**DCOM**) y Hypertext Transfer Protocol (**HTTP**).

Existen distintos tipos de servidores de aplicaciones que pueden ser utilizados para la ejecución de las aplicaciones distribuidas. Dependiendo del protocolo a utilizar en la aplicación, se usará uno u otro tipo de servidor de aplicaciones. En caso de RMI, CORBA y DCOM se utiliza el **Servidor de aplicaciones GeneXus**, mientras que si se utiliza HTTP como protocolo, el servidor de aplicaciones será uno externo al generador, y servirá cualquiera que tenga un **Motor de Servlets**; por ejemplo: RESIN, Jakarta-tomcat, Websphere, Jrun, etc.

Pool de conexiones

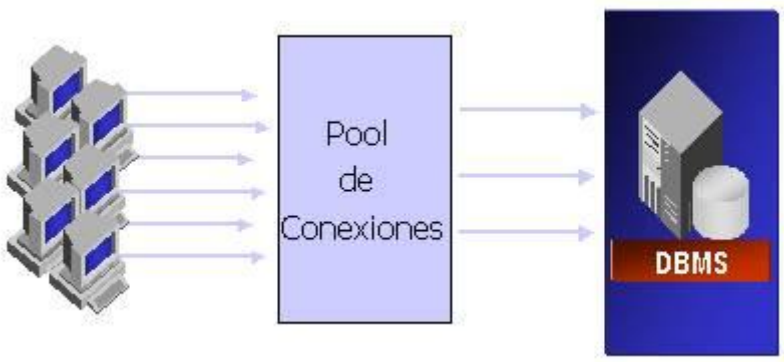
En las aplicaciones cliente/servidor tradicionales, se mantiene una conexión a la base de datos por cada cliente final, esto es, si tengo 100 clientes finales tendré 100 conexiones a la base de datos.



Esto es aceptable para aplicaciones con un número de clientes relativamente pequeño, pero es inconveniente cuando la cantidad de clientes es grande, o impredecible (como en aplicaciones en Internet), por varias razones:

- La carga al servidor de base de datos es mayor cuanto más conexiones abiertas haya.
- En la mayoría de los DBMSs se debe comprar una licencia de uso por cada usuario conectado.
- No es óptimo, dado que la mayoría de las conexiones están sin hacer nada una parte importante del tiempo.

La solución a estos problemas es permitir el compartir conexiones a la base de datos entre los diferentes clientes de la aplicación, de modo de que existan menos conexiones físicas a la base de datos que clientes.



La idea general de un pool de conexiones, es que cada vez que un cliente necesita una conexión (esto es, cada vez que prepara un cursor, lo ejecuta, ejecuta una sentencia directamente, etc.), se la solicita al Pool. Esta conexión pertenece al cliente hasta que de algún modo se determina que no la necesita más, y en ese momento se le puede asignar la conexión a otro cliente.

Si un cliente necesita una conexión y no existe ninguna disponible, se queda esperando hasta que alguna conexión se libere.

Otra ventaja es que las conexiones se mantienen abiertas, por lo que los clientes se ahorran el tiempo de conexión al DBMS al iniciar la aplicación.