Planos de ejecución en Velneo V7

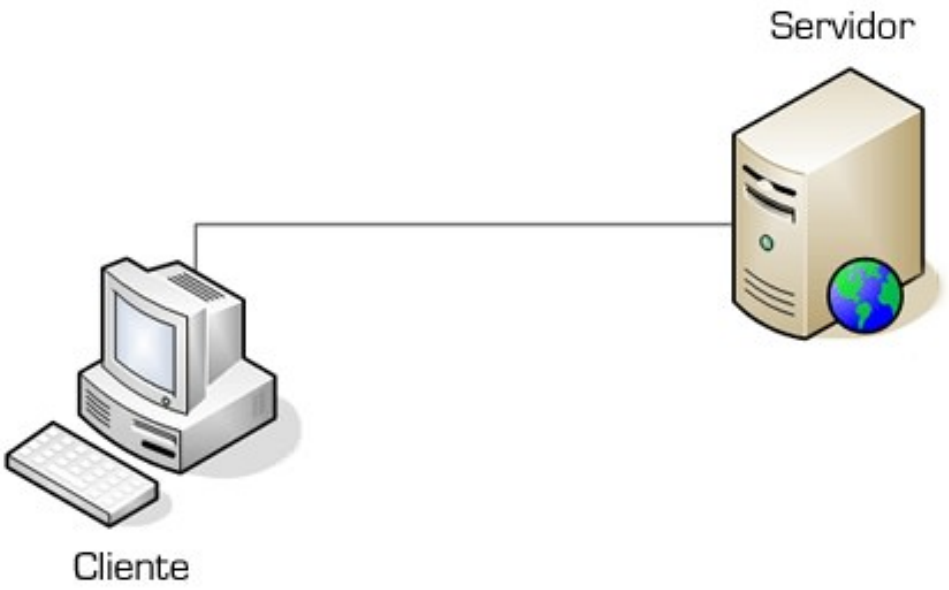
Por Jesús Arboleya

velneo®

Introducción

En este documento vamos a repasar los planos de ejecución en Velneo V7. Dominarlos nos permitirá optimizar la ejecución de nuestras aplicaciones. También repasaremos los modos que tenemos para forzar la ejecución en los diferentes planos.

Arquitectura Cliente/Servidor



Cuando ejecutamos nuestras aplicaciones en una arquitectura cliente-servidor como la de Velneo V7, las acciones se ejecutan en el servidor o en el cliente. Conocer qué se produce en cada una de ellas es imprescindible para plantear un metodología de ejecución optimizada. En la siguiente tabla se muestran ejemplos de diferentes objetos y sus planos de ejecución por defecto:

Objeto	**Se ejecuta en el…**
Alta, baja y modificación de registros	Servidor
Triggers y actualizaciones de tablas	Servidor
Regeneración de índices y área de datos	Servidor
Menús	Cliente
Objetos visuales: formularios, rejillas, etc.	Cliente
Localizadores de registros	Cliente
Procesos	Cliente o Servidor
Funciones	Depende desde donde son ejecutadas
Búsquedas	Depende desde donde son ejecutadas

1. Objetos que siempre se ejecutan en el servidor

Como podemos observar en la tabla anterior, hay tareas que siempre se ejecutan en el servidor. Todo lo relacionado con la base de datos: transacciones, regeneraciones de tablas, eventos de tabla o triggers y actualizaciones siempre se ejecutan en el servidor independientemente de que las operaciones de alta, baja, modificación o regeneración las ejecutemos en el cliente.

Aunque ejecutemos estos objetos desde el cliente, Velneo se encarga de enviar la operación desde el cliente al servidor para su ejecución. Y tras finalizar la transacción se encarga de enviar al cliente los datos que ya están grabados en la base de datos, a esto se le denomina refresco secundario.

Además, el servidor de Velneo se encarga automáticamente de enviar estas modificaciones a los usuarios conectados que estén haciendo uso de esos registros, a esto se le denomina refresco terciario y es el que produce el efecto de que la información modificada se refresque automáticamente a todos los usuarios que están ejecutando la aplicación al cabo de unos segundos, sin necesidad de programarlo.

Todos los objetos que se programan en el proyecto de datos se ejecutan en el servidor. La mayoría de los objetos que se ejecutan en el servidor se programan y almacenan en los proyectos de datos. Sin embargo, hay algunos objetos que se pueden programar en el proyecto de datos o de aplicación y cuya ejecución se puede usar tanto en el cliente como en el servidor, estos objetos serán los que analizaremos en el apartado de objetos que se ejecutan en cliente o en el servidor.

2. Objetos que siempre se ejecutan en el cliente

Todos aquellos objetos de la capa de interfaz se ejecutan siempre en el cliente. El servidor nunca podrá mostrar un formulario, una rejilla ni ningún objeto que requiera interfaz. Para almacenar todos estos objetos usamos los proyectos de aplicación.

Esta capa cliente de nuestras aplicaciones es la encargada de interactuar con el usuario mostrando información o solicitándole datos. Por este motivo cuando creamos un formulario debemos entender que la interacción del usuario se hará en la capa cliente, sin embargo las transacciones con la base de datos siempre se realizarán en el servidor.

3. Objetos que se ejecutan en el cliente o en el servidor

Hay algunos objetos que se pueden declarar tanto en los proyectos de datos como en los proyectos de aplicación. ¿Cuál es la diferencia de hacerlo en uno o en otro? Pues que los objetos creados en los proyectos de datos podrán ser usados desde ese proyectos de datos o desde cualquier proyecto de datos o aplicación que lo herede, mientras que los objetos creados en los proyectos de aplicación pueden ser usados desde ese proyecto de aplicación o desde cualquier proyecto de aplicación que lo herede, pero nunca desde un proyecto de datos.

Recuerda que un proyecto de aplicación puede heredar proyectos de aplicación y proyectos de datos, pero un proyecto de datos sólo puede heredar proyectos de datos.

Por lo tanto, estos objetos que vamos a ver ahora en la mayoría de las ocasiones puede ser interesante crearlos en los proyectos de datos ya que se garantiza una reutilización mayor, y nos evitará mover el objeto desde el proyecto aplicación al proyecto de datos a posteriori.

¿Qué objetos pueden ser ejecutados tanto desde el cliente como desde el servidor?

- Fichero adjunto
- Variable global
- Constante
- Dibujo
- Esquema
- Función
- Búsqueda
- Cola de procesos
- Proceso

Vamos a repasar estos objetos y ver como se usan.

3.1 Fichero adjunto

Los ficheros adjuntos realmente es indiferente donde se guarden ya que en ejecución se almacenan en el directorio de caché de la aplicación. Por lo tanto lo recomendable es ubicarlos lo más próximos a los objetos que los usarán.

3.2 Variable global

Las variables globales sólo se pueden declarar en los proyectos de datos, pero tanto si son en disco como en memoria se pueden ejecutar indistintamente en el cliente y en el servidor. Sin embargo, hay tres aspectos muy importantes a tener en cuenta de las variables globales.

1. Las variables globales en disco cada vez que usan requieren que sea leído su valor de disco, esto significa que en el servidor son rápidas, pero usadas desde el cliente, cada vez que en una fórmula usamos una variable global estamos obligando al cliente a enviar una petición al servidor para obtener el valor actual de ese variable, pues no podemos olvidar que al ser en disco es compartida por todos los usuarios, y Velneo se encarga de garantizar que el valor siempre sea el mismo para todos los usuarios, incluso encargándose del bloqueo de la variable cuando su valor es modificado. Por lo tanto, debemos reducir al máximo el uso de variables globales en disco en la interfaz.

2. Las variables globales en memoria son únicas para cada máquina. Esto significa que cada cliente (usuario) tendrá su variable global en memoria específica. El valor de esa variable es compartido por los procesos que se ejecutan en una aplicación de un equipo cliente específico, y no se comparten con el resto de usuarios. De la misma forma el servidor se comporta como un cliente más y tiene una instancia en memoria de las variables locales. Estas variables locales que se encuentran en el servidor son compartidas por todos los procesos que se ejecuten en el servidor, sean del usuario que sean, pero nunca se mezclan sus valores con los que existen en cada máquina cliente. De esta forma si, por ejemplo desde una máquina cliente modificamos el valor de la variable HORA a "16:30:00", la variable del resto de máquinas cliente no se verán afectadas, y tampoco cambiará el valor de la variable HORA que existe en el servidor.

3. Los valores de las variables en memoria no "viajan" entre el cliente y el servidor ni viceversa. Es decir, cada máquina es una isla, lo que nos garantiza que nunca se mezclan los valores de esas variables globales entre 2 máquinas, eso mismo debemos aplicarlo al servidor. Por eso cuando ejecutamos un proceso en el servidor, aunque lo lancemos desde un cliente, si dentro de ese proceso usamos el valor de la variable FECHA, el proceso una vez que se está ejecutando en el servidor obtendrá el valor de la variable FECHA que está en el servidor, pero no tendrá visibilidad del valor que había en la variable FECHA del cliente. Por este motivo para pasar valores desde un objeto que se está ejecutando en cliente a un proceso que se ejecute en el servidor usamos variables locales del proceso a las que les pasamos los valores deseados, utilizando los comandos de objeto: crear manejador de objeto, set y get variable local y disparar objeto. En resumen, las variables globales en memoria del servidor son compartidas por todos los procesos o transacciones que se ejecuten en el servidor, de todos los usuarios, mientras que en el cliente cada ejecución de la aplicación tendrá su instancia específica de la variable global en memoria. Incluso, si un usuario abre 2 veces la misma aplicación en la misma má-

quina, cada ejecución de la aplicación tendrá su instancia de la variable global en memoria y no será accesible para el resto de ejecuciones de la aplicación en la misma o en otras máquinas.

3.3 Constante

Es un objeto que se utiliza para almacenar valores estáticos. Mayormente se usa para almacenar cadenas de texto que queremos que sean fácilmente traducibles y que suelen utilizarse en fórmulas. La ventaja de poner las constantes en los proyectos de datos es que podemos usar su valor tanto en objetos de datos como en objetos de interfaz, aunque lo más lógico es ubicarlas próximas al objeto donde se utiliza.

3.4 Dibujo

Los dibujos a priori son un objeto que se usa en la interfaz de las aplicaciones, sin embargo también se utiliza en los esquemas de tablas y proyectos, por lo que es habitual encontrar objetos declarados en los proyectos de datos que son utilizados en los esquemas o también en proyectos de aplicación. Por lo tanto es a priori un objeto de interfaz y de parte cliente, aunque tiene utilidad documental en los proyectos de datos.

3.5 Esquema

Los esquemas cumplen una doble funcionalidad, por un lado es de gran utilidad durante el desarrollo de las estructuras de base datos y su documentación. Y cuando los esquemas están declarados en los proyectos de aplicación, es posible asignar a cada tabla un objeto de vista de datos permitiendo al usuario final realizar la navegación a través de la información de forma visual. Esta doble funcionalidad nos lleva en ocasiones a crear la estructura de datos en un esquema del proyecto de datos para luego duplicar o mover el esquema en el proyecto de aplicación y completarlo asignando en la propiedad objeto aquel que visualizará el usuario cuando navegue a través del esquema a cada tabla.

3.6 Función

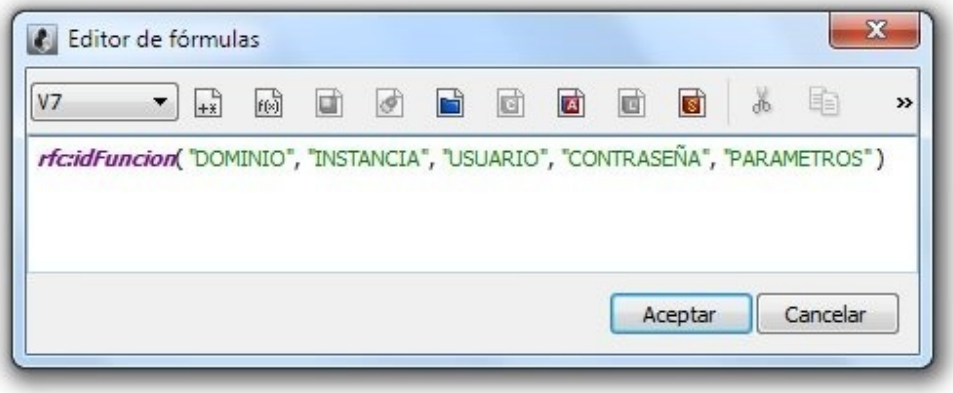
La función es un objeto de gran uso ya que es muy flexible al permitir que sea usado en cualquier fórmula, por lo que es posible usarlo prácticamente en cualquier parte de nuestra aplicación. Desde el contenido inicial de un campo, la condición de una actualización, hasta la condición de activo o visible de cualquier control.

Es un objeto realmente útil y potente, aunque debemos tener claro dos cosas de las funciones. La primera es que la función siempre se ejecuta en el plano donde es lanzado, por lo tanto si una función es ejecutada desde el servidor su ejecución se realizará en el servidor y si es lanzada desde el cliente su ejecución se realizará en el cliente.

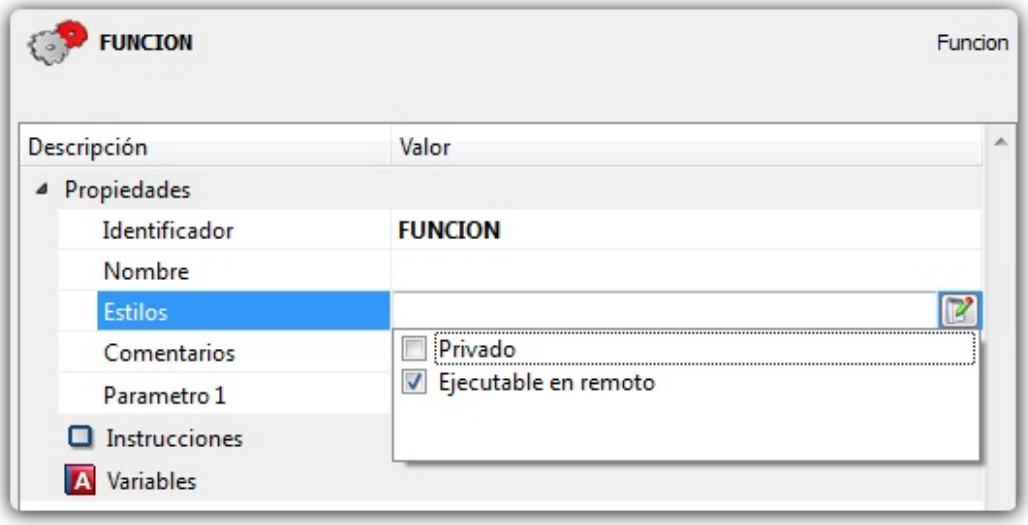
Lo segundo que debemos saber de una función es que si realiza operaciones de base de datos y transacciona, dicha transacción será independiente aunque exista otra transacción en curso. Esto debemos tenerlo en cuenta para aprovecharlo cuando nos interese y evitarlo en el caso de que deseemos que no se genere una transacción independiente, en este último caso deberíamos sustituir la función por un proceso ejecutado con el comando "Disparar objeto" del grupo de comando de instrucción de objeto.

Mención aparte requieren las funciones remotas. A diferencia de lo comentado hasta ahora las funciones remotas siempre se ejecutan en el servidor. Las funciones remotas tienen tres características importantes:

La primera es que pueden ser lanzadas desde cualquier fórmula mediante la siguiente sintaxis:



La segunda es que sólo son ejecutables remotamente las funciones que tengan activo el check de ejecutable remotamente.



La tercera característica y principal es que se permite la ejecución de la función no sólo en el servidor al que estamos conectados, además se puede lanzar la función remota contra cualquier servidor. Esto es realmente útil para la gestión de aplicaciones con datos distribuidos, para la realización de sincronizaciones e incluso para el intercambio de datos entre aplicaciones que pueden estar ejecutándose en el mismo o en distinto servidor.

3.7 Búsqueda

Las búsquedas pueden ser programadas tanto en los proyectos de datos como de aplicación y pueden ser ejecutadas en el cliente o en el servidor, a continuación veremos como.

Es fundamental conocer como funciona este objeto para optimizar nuestras aplicaciones de forma correcta. Aunque existen otras maneras de localizar información, el objeto búsqueda es el más usado para encontrar registros de nuestras tablas, por lo tanto es debemos conocer y entender que una búsqueda está compuesta por uno o varios componentes, y que cada uno de los componentes se encarga de buscar registros de la tabla en uno de sus índices.

Por ejemplo: Supongamos una búsqueda en la tabla FACTURAS con 2 componentes: El primero buscará por el índice CLIENTE y nos devolverá todas las facturas del código de cliente que le resolvamos, el segundo buscará por el índice FECHA entre las fechas que le resolvamos. El resultado de la búsqueda del primer componente se cruzará con el resultado obtenido en el segundo componente, de esa forma obtendremos las facturas de un cliente entre fechas.

Si lanzamos está búsqueda en el cliente vClient ¿Cómo se realiza su ejecución?

1. La aplicación ejecutada en vClient solicitará al servidor la ejecución del primer componente de la búsqueda.
2. La lista de registros resultantes será devuelta por el servidor al cliente.
3. A continuación el cliente solicitará al servidor la ejecución del segundo componente de la búsqueda.
4. El servidor devolverá la nueva lista de registros.
5. El cliente se encargará de cruzar ambas listas.

Los pasos 1, 2, 3 y 4 requieren el establecimiento de un socket entre el cliente y el servidor así como 2 peticiones, una por cada componente de búsqueda y los correspondientes envíos de la lista de registros del servidor al cliente. Por este motivo es tan importante optimizar la búsqueda para que se ejecute en el servidor y así evitemos este tránsito de información.

Si lo ejecutamos en el servidor, la gran ventaja es que aunque se ejecutan los mismos pasos la velocidad de ejecución es la más óptima ya que se evitan las solicitudes y la transferencia de información entre el cliente y el servidor.

Si la búsqueda sólo tiene un componente no se apreciará diferencia entre ejecutarla en el cliente o ejecutarla en un proceso lanzado en el servidor. Sin embargo, para 2 o más componentes de búsqueda si es conveniente optimizar su ejecución en el servidor. Cuantos más componentes se ejecuten de la búsqueda más se notará la optimización.

Un error muy común que se cometía al programar la ejecución de búsquedas utilizando la función "Disparar objeto" estaba derivado de que el parámetro "Modo de ejecución" permitía seleccionar el valor "3º plano: Servidor (síncrono)". A partir de la versión 7.15, este parámetro asumirá el valor "No aplicable" para todos los objetos, incluida la búsqueda, salvo para los procesos. Por lo tanto, aunque ejecutemos una búsqueda con "Disparar objeto" y especifiquemos el valor "3º plano", realmente este parámetro no se aplica y por lo tanto se ejecutará en el cliente tal y como acabamos de comentar.

Para ejecutar una búsqueda en el servidor es necesario lanzarla desde un proceso que esté siendo ejecutado en el servidor. Por lo tanto debemos crear un proceso que será ejecutado en "3º plano" y dentro de ese proceso ya podremos lanzar la búsqueda que se ejecutará con el mejor rendimiento.

3.8 Cola de procesos

Una cola de procesos es un objeto que se combina como su nombre indica con objetos proceso. No tiene importancia si la cola de procesos se programa en el proyecto de datos o de aplicaciones. El objetivo de las colas de procesos es disponer del control de orden de ejecución de los procesos cuando son ejecutados en segundo plano, en el cliente. Esto lo analizaremos a continuación cuando repasemos los planos de ejecución.

3.9 Proceso

El proceso es el único objeto que podemos programar para que se ejecute en el cliente o en el servidor. Si necesitamos ejecutar el proceso desde objetos del proyecto de datos, deberemos programar el proceso en el proyecto de datos y podremos usarlo tanto en los proyectos de datos como en los de aplicación. Sin embargo, si creamos el proceso en el proyecto de aplicación, sólo podremos ejecutarlo desde los proyectos de aplicación.

Independientemente de donde esté programado su ejecución dependerá de como lo lancemos. Es muy habitual que un proceso origen (llamador) ejecute otro proceso (destino). En la siguiente tabla se muestra que posibilidades de ejecutar un proceso existen en función de donde está ejecutándose el proceso llamador.

Proceso llamador ejecutándose en…	1º plano	2º plano	3º plano
1º plano	Sí	Sí	Sí
2º plano	No	Sí	Sí
3º plano	No	No	Sí

Para que un proceso pueda ejecutar otro proceso en diferente plano tenemos 2 posibles comandos de instrucción: Ejecutar proceso y Disparar objeto. En ambos comandos disponemos de la propiedad "Modo de ejecución" que nos permite especificar si el proceso a ejecutar se lanzará en 1º, 2º o 3º plano.

En la tabla anterior vemos que hay opciones no disponibles, esto significa que por ejemplo si desde un proceso que se está ejecutando en 3º plano (servidor) ejecutamos otro proceso con "Ejecutar proceso" o "Disparar objeto”, aunque especifiquemos la opción "1º plano" o "2º plano" en el modo de ejecución el proceso siempre será lanzado en “3º plano” (servidor).

3.10 Un último ejemplo

Otro ejemplo que muestra las diferencias de ejecución entre el cliente y el servidor es el comando de instrucción "Mensaje". Si lo ejecutamos en el cliente se mostrará un cuadro de diálogo con el mensaje, si lo ejecutamos en el servidor se mostrará en la lista de mensajes del sistema visibles desde vAdmin.

Planos de ejecución en Velneo V7

Existen 3 planos de ejecución:

Modo de ejecución…	Se ejecuta en el…
1º plano	Cliente
2º plano	Cliente
3º plano	Servidor

1º plano

Es el plano de ejecución interactiva del cliente, el hilo principal de ejecución con el que interactúa el usuario de la aplicación. Cuando arranca la ejecución de un aplicación se realiza en primer plano. Cuando lanzamos un proceso en primer plano mientras está en ejecución el usuario no podrá seguir trabajando y deberá esperar a que finalice, lo más habitual es que en esos momentos esté visible el cursor de espera.

Este es el plano en el que se ejecutan todos los objetos de interfaz: formularios, rejillas, informes, etc. Es un plano de ejecución para los botones, toolbars, opciones de menú, manejadores de evento y también habitual para la ejecución de procesos de corta duración o aquellos en los que tengamos garantizado que los datos están disponibles en caché.

Recuerda que siempre que estés en primer plano, por defecto salvo que lo especifiques de forma explícita en el modo de ejecución seguirás ejecutando los procesos en primer plano.

Como veíamos en la tabla anterior, desde un proceso en “1º plano” podemos ejecutar otro proceso en “1º plano” si queremos que se ejecute de forma síncrona en el cliente, en “2º plano” si queremos que se ejecuta de forma asíncrona en cliente o en “3º plano” si deseamos que se ejecute de forma síncrona en el servidor.

2º plano

Este plano también se ejecuta en el cliente. Para conseguir ejecutar un proceso en segundo plano debemos utilizar una de la siguientes opciones:

1. Comando de instrucción: Ejecutar proceso con modo 2º plano.
2. Comando de instrucción: Disparar objeto con modo 2º plano.
3. Acción en stock para toolbar o menú: Ejecutar proceso en 2º plano.

Decíamos que en primer plano sólo existe un hilo de ejecución, es decir es una ejecución síncrona. Sin embargo, el 2º plano nos abre la puerta a la ejecución asíncrona de tal forma que si desde un proceso ejecutamos varios en segundo plano, todos los procesos en 2º plano podrán estar ejecutándose a la vez. Esta característica es muy valiosa para determinadas funcionalidades como por ejemplo:

- Pasar datos de tablas en memoria a tablas en disco.
- Cálculos de informes.
- Impresión de informes.
- Envío de emails.
- Etc.

Esto nos permite, por ejemplo, que un formulario de grabación de pedidos cuando el usuario acepta, lance un proceso en 2º plano que estará encargado de grabar los datos en la tablas, realizar cálculos, imprimir el pedido e incluso enviarlo por correo, todo en 2º plano, lo que significa que el usuario tras pulsar el botón aceptar podrá comenzar a grabar el siguiente pedido sin tener que esperar a que se realicen las acciones comentadas.

Puede darse el caso de que no nos interese que todos los procesos que lanzamos en segundo plano se ejecuten a la vez porque tengan dependencia. Si programamos piezas de código reutilizables, como por ejemplo si lanzamos en segundo plano un proceso de facturación que realiza primero la selección de albaranes a facturar, luego el cálculo de las facturas sobre los albaranes seleccionados y finalmente la impresión de las facturas calculadas, debemos asegurarnos de que los 3 procesos se ejecuten en ese orden y uno tras otro, nunca a la vez.. Esto se puede conseguir de dos formas:

- Ejecutando un proceso en 2º plano que se encargará de ejecutar los 3 procesos en 1º plano y en ese orden. Al hacerlo así, aunque los ejecutamos en primer plano, en realidad seguirán ejecutandose en el mismo plano del proceso lanzador, es decir, 2º plano pero de forma síncrona. Si los ejecutamos en 2º plano entonces no conseguiríamos nuestro objetivo ya que se ejecutarían los 3 a la vez.

- Usando una cola de ejecución de procesos. Es la forma más elegante de ejecutar los 3 procesos, lo que hacemos es lanzar los procesos en 2º plano pero en la misma cola, por ejemplo COLA_FACTURACION. De esta forma la cola se encargará de ir ejecutando en 2º plano cada uno de los procesos y cada vez que termine uno de ellos lanzará el primer proceso que se encuentre en espera de dicha cola.

Lo más lógico es que no incluyamos en los procesos en 2º plano llamadas a objetos con interfaz, ya que el proceso queda parado a la espera de que el usuario interactúe. La opción más recomendable es que un proceso en primer plano se encargue de pedir al usuario toda la información necesaria, la almacene en tablas, variables globales o locales y que a continuación lance él proceso o procesos en 2º plano para su ejecución en paralelo o en cola.

Desde un proceso en 2º plano es posible ejecutar procesos en 3º plano. En ese caso el proceso en 2º plano queda a la espera de que finalice la ejecución del proceso en 3º plano para continuar con su ejecución. Esta forma de ejecutar los procesos es realmente útil cuando estamos en primer plano y deseamos lanzar un proceso para que se ejecute de forma optimizada en el servidor sin que paralice la interfaz del usuario. El flujo es bastante sencillo:

1. El proceso en primer plano ejecuta el proceso en 2º plano. El usuario puede seguir trabajando.
2. El proceso en segundo plano lanza el proceso en 3º plano y queda a la espera.
3. El proceso en 3º plano se ejecuta en el servidor.
4. Cuando finaliza continúa ejecutándose el proceso en 2º plano.

Recuerda que los procesos en 2º plano son un gran recurso para optimizar tus aplicaciones y mejorar la experiencia de usuario.

3º plano

Es el recurso estrella en la optimización de aplicaciones con Velneo V7. Lanzar procesos en 3º plano supone en muchos casos un más que notable ahorro de transferencia de información entre el cliente y el servidor y el aprovechamiento de las capacidad de procesamiento de las máquinas servidores, lo que se traduce en un gran ahorro de tiempo que aporta mejoras en la experiencia de usuario, cuando somos usuarios no nos gusta esperar.

No debemos caer en el error de creer que todos los procesos deben ejecutarse en 3º plano para obtener el mejor rendimiento ya que eso no es cierto. Cuando desde el cliente necesitamos realizar un cálculo basado en información que ya está disponible en la caché del cliente, no interesa ejecutarlo en el servidor ya que sería más lento al tener que establecer un socket entre el cliente y el servidor, realizar la llamada al proceso y esperar la respuesta del servidor tras finalizar la ejecución del proceso. Por lo tanto, deberíamos partir de la base de que los procesos, por defecto se ejecuten en primer plano e iremos optimizando nuestra aplicación a medida que lo requieran sus funcionalidades. Por este motivo, es de gran ayuda programar y probar nuestras aplicaciones en Cloud ya que nos permitirá encontrar aquellas funcionalidades que se ejecutan con más lentitud para optimizarlas, algo que puede pasar desapercibido si lo ejecutamos en local.

¿Qué procesos son candidatos a ser ejecutados en el servidor?
Todos los procesos que recorran listas de registros realizando altas, bajas o modificaciones, así como para la ejecución de búsquedas con varios componentes, son candidatos a ser ejecutados en tercer plano. En definitiva, cualquier proceso que al ser ejecutado en el servidor nos evite una importante transferencia de información.

Otro aspecto a tener en cuenta es la duración del proceso que se ejecuta en 3º plano. No hay que olvidarse de que un servidor puede estar ejecutando simultáneamente múltiples procesos en 3º plano. Si ejecutamos procesos que requieren muchos recursos podemos llegar a repercutir en el rendimiento del servidor. Para evitarlo es recomendable evitar la ejecución de procesos que realicen largas transacciones, lo que produce bloqueos de registros más largos y además genera un fichero de transacciones de gran tamaño que se gestiona con peor rendimiento. Si necesitamos lanzar un proceso que realice la modificación de millones de registros, en la mayoría de las ocasiones será más óptima dividir esa transacción en transacciones mucho más pequeñas. El punto óptimo se debe buscar en base al número de registros a procesar en cada transacción, tanto penaliza un transacción excesivamente grande como un número elevado de transacciones. Por ejemplo, para modificar 2 millones de registros podemos optar por lanzar 40 transacciones de 50.000 registros. Eso siempre será mejor que hacer una única transacción que modifique 2 millones de registros, y como comentaba tampoco es óptimo generar 100.000 transacciones de 200 registros ya que crear y destruir cada transacción consume recursos y tiempo.

Ya hemos repasado en los planos anteriores como se pueden lanzar un proceso en 3º plano:

1. Comando de instrucción: Ejecutar proceso con modo 3º plano.
2. Comando de instrucción: Disparar objeto con modo 3º plano.

También hemos repasado que cualquier proceso que es lanzado desde el servidor siempre se ejecutará en 3º plano.

Otro aspecto a tener muy en cuenta es el que comentamos al repasar las variables globales en memoria, donde indicamos que no “viajan” entre el cliente y el servidor y para pasar información usaremos los comandos de instrucción del grupo de objeto que nos permitirán pasar a un proceso un registro o lista de registros como origen del mismo así como alimentar todas las variables locales del proceso que necesitemos para pasarle la información. Debemos recordar en este punto que cualquier proceso con destino ficha o lista nos devolverá tanto en el cliente como en el servidor el registro o lista de registros que añadamos a su salida.

Una gran ventaja de los proceso en 3º plano es que al ser ejecutados en el servidor tiene un acceso muy rápido a la base de datos y también a las variables globales en disco, por lo que no debemos preocuparnos de su uso en este plano de ejecución.