Entendiendo la programación en 3 (o más) capas

¿Qué es la programación por capas?

Es una metodología para desarrollar software.

¿Cuál es el objetivo?

Que cada una de las partes que compone una aplicación informática se encuentre separada y no interfiera con las demás partes.

A cada una de esas partes se le llama «capa».

¿Cuáles son las ventajas de la programación por capas?

En las aplicaciones administrativas/contables/empresariales generalmente hay varios módulos o subsistemas que interactúan entre ellos y algunos de los informes que proveen son complejos. Al tener a cada módulo claramente separado de los demás se puede terminarlo más rápido y en caso de encontrarle algún error, se puede determinar rápidamente donde se encuentra ese error y corregirlo sin temor de que interfiera con algún otro módulo.

En síntesis: podremos programar más rápido, estaremos más organizados, cometeremos menos errores, y esos errores serán fáciles de encontrar y de corregir.

Además, facilita el trabajo en equipo porque cada programador (o conjunto de programadores) puede dedicarse a una tarea específica, sin riesgo de que lo que uno hace afecte negativamente a otro.

Otra ventaja: las capas (o parte de ellas) pueden reutilizarse. Por ejemplo, si creamos una entidad llamada CLIENTE, podremos usar a esa entidad en cualquier aplicación que requiera datos de clientes. Lo mismo sería para la entidad VENTAS, la entidad COBRANZAS, la entidad PAGOS, la entidad BANCOS, la entidad PAISES, etc. Las escribimos una vez y las podemos usar muchas veces, sabiendo que siempre funcionarán bien.

¿Cuáles son las capas principales?

Aunque hay aplicaciones que pueden tener muchas capas, las 3 principales son:

- Presentación
- Lógica del Negocio
- Acceso a Datos

De aquí en adelante se las abreviará como: **CP** (Capa de Presentación), **CLN** (Capa de Lógica del Negocio), y **CAD** (Capa de Acceso a Datos)

Captura 1.

Como se puede ver en la ***Captura 1.***, solamente la **CAD** (Capa de Acceso a Datos) se comunica directamente con la Base de Datos.

¿Qué se coloca en la CP (Capa de Presentación)?

Todo lo que el usuario puede ver. Eso incluye a formularios, menús, reportes, y cualquier otra cosa que sea visible para el usuario.

También pueden realizarse validaciones sencillas, y que no requieren acceso a la Base de Datos. Por ejemplo:

- que la Fecha de la Venta no sea posterior al día de hoy
- que en la Venta siempre exista un Número de Factura
- que la cantidad vendida no sea negativa
- que el importe cobrado no sea negativo
- etc.

¿Qué se coloca en la CLN (Capa de Lógica del Negocio)?

Las validaciones complejas, las entidades, y los procesos. Aquí se valida que los

datos ingresados por el usuario sean correctos, y también que sólo pueda realizar operaciones permitidas. Además, para cada entidad aquí se colocan todas las acciones que ella puede realizar. La idea es representar a cada entidad en su totalidad, con todos los datos y todas las rutinas que se necesiten.

Para que se encuentren en esta capa, las validaciones deben ser complicadas o deben requerir el acceso a la Base de Datos. Por ejemplo:

- el código del cliente debe existir
- no se le puede vender a crédito a un cliente moroso a un cliente solamente se le puede cobrar una Factura que tiene saldo
- no se pueden aceptar devoluciones pasadas las 72 horas de la venta
- no se hacen descuentos a los productos que están en oferta
- etc.

En el caso de las entidades, por ejemplo: EMPLEADO. Necesitaremos:

- Una rutina que grabe los datos de un Empleado
- Una rutina que obtenga los datos de un Empleado, sabiendo su Identificador
- Una rutina que obtenga los datos de todos los Empleados
- Una rutina que devuelva el Identificador del Empleado enviándole su Número de Legajo (o Código del Empleado)

Algunos ejemplos de entidades: productos, clientes, proveedores, empleados, órdenes de compra, facturas de venta, cobranzas realizadas, pagos efectuados, etc.

¿Qué se coloca en la CAD (Capa de Acceso a Datos)?

Los comandos que se comunican directamente con la Base de Datos. Por ejemplo:

- Conectarse a la Base de Datos
- Desconectarse de la Base de Datos
- Iniciar una Transacción
- Finalizar una Transacción
- Llamar a una vista (vista de BD, no es la capa de presentación)
- Ejecutar un procedimiento almacenado

Y ahora viene algo muy importante: no dependemos del origen de los datos. Eso significa que en nuestra aplicación si queremos podemos usar tablas .DBF, o un motor SQL tal como **Firebird**, MySQL, MariaDb, PostgreSQL, SQL Server, etc. El que se nos ocurra. No importa. Nuestra aplicación funcionará igualmente bien con cualquiera de ellos. Será aquí, en la **CAD**, donde escribiremos los comandos que se comunicarán con la Base de Datos y esos sí pueden ser específicos de nuestro origen de datos, pero la **CP** y la **CLN** no se verán afectadas.

Por ejemplo, en **Firebird** para ejecutar un procedimiento almacenado se escribe: ***EXECUTE PROCEDURE MiProcedimientoAlmacenado*** y en SQL Server se escribe: ***EXEC MiProcedimientoAlmacenado***. Como ves, los comandos aunque realizan la misma tarea tienen nombres distintos. Entonces en la **CAD** escribimos el comando correcto para nuestro motor SQL y listo. Ya está. Las otras capas ni se enteraron que cambiamos nuestro origen de datos. Pero les sigue funcionando muy bien todo.

¿Y por qué ni la **CP** ni la **CLN** se enteraron de que cambiamos el origen de datos?

Porque ellas reciben un ***cursor***. No se involucran en ningún momento con el envío o la recepción de comandos a la Base de Datos. Esa es tarea exclusiva de la **CAD**. Entonces, como trabajan con cursores, les resulta totalmente indiferente si nuestro origen de datos es **Firebird**, MySQL, MariaDb, SQL Server, Oracle, o el que sea.

Y esto, como te puedes imaginar, es una gran ventaja.

¿A qué se llama cohesión?

A que los módulos trabajen de forma coordinada, para alcanzar su objetivo de forma rápida y eficaz.

Lo ideal es que tengan una alta cohesión.

¿A qué se llama acoplamiento?

A la dependencia que existe entre los módulos. Ya habíamos visto que lo ideal es que los módulos sean totalmente independientes unos de otros. En caso de que sea imposible evitar la dependencia, hay que tratar de que sea lo mínimo posible.

Lo ideal es que tengan un acomplamiento muy bajo.

¿Cómo se implementa la programación por capas?

Separando cada capa de las demás. La comunicación entre capas se realiza mediante procedimientos claros y bien establecidos. Algo importante a recordar es que cada capa puede estar compuesta por muchos archivos, pueden ser decenas e inclusive cientos, dependiendo de la complejidad de la aplicación.

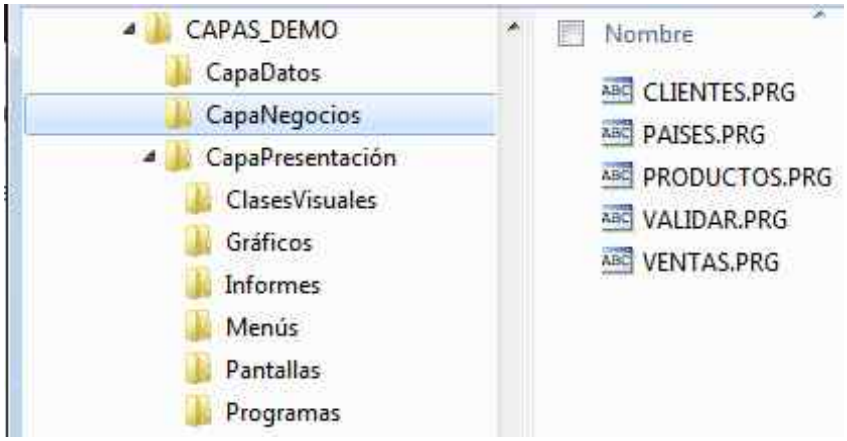
Lo más conveniente y lo más recomendable es que se utilicen **clases**. Las **clases** por su propia naturaleza mantienen encapsulados a los datos y a las rutinas que manejan a esos datos, y eso es lo ideal cuando se programa en capas. Recuerda, no es obligatorio usar **clases**, pero sí es altamente recomendable usarlas.

Estructura de las carpetas

Cuando no se programa por capas, lo normal es que tengamos varias carpetas: Clases, Gráficos, Informes, Menús, Pantallas, Programas, etc.

Cuando se programa por capas, es más conveniente tener una carpeta para cada capa, algo como:

Captura 2.



Captura 3.

Importante:

Ni la **CLN** (Capa de Lógica del Negocio) ni la **CAD** (Capa de Acceso a Datos) deben mostrarle mensajes al usuario. Si por ejemplo una de ellas descubre un error entonces debe enviar un número de error o un mensaje de error, para que la **CP** (Capa de Presentación) se lo muestre al usuario. No es tarea ni de la **CLN** ni de la **CAD** mostrar mensajes.

Además, ni la **CP** (Capa de Presentación) ni la **CLN** (Capa de Lógica del Negocio) deben saber cuales son realmente los nombres de las **tablas**, de las **vistas**, o de los **procedimientos almacenados**. La **CP** y la **CLN** reciben cursores, la obtención de esos cursores y conocer cuales son sus nombres reales en la Base de Datos es competencia exclusiva de la **CAD** (Capa de Acceso a Datos).