Curso de Programación Web .NET

Unidad 2

“Modelo Multicapas”

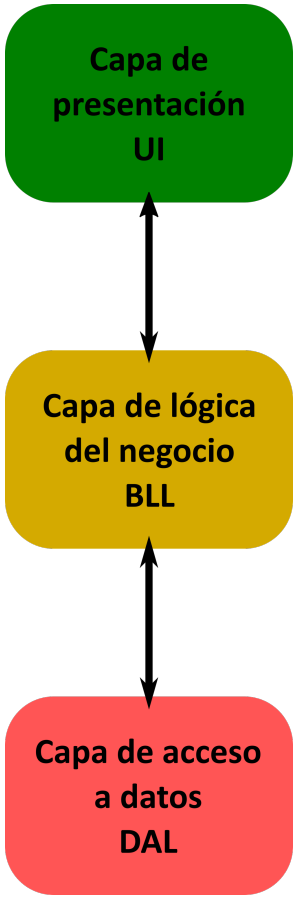
Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Índice

Programando en Capas

Introducción

El modelo de múltiples capas se refiere a cuando la lógica de la aplicación se divide en partes. Un número muy común de capas es 3: una para la interfaz de usuario, presentación o UI; otra para la lógica de negocio y otra para el acceso a los datos.



Separación Lógica vs Física

Cuando nos referimos a una separación lógica versus una física de los componentes de una aplicación nos estamos refriendo a conceptos relacionados pero distintos.

Separación Lógica

La separación lógica de la aplicación es expresado en el código. El diseño del mismo debe expresar claramente cuales son las capas que están siendo usadas.

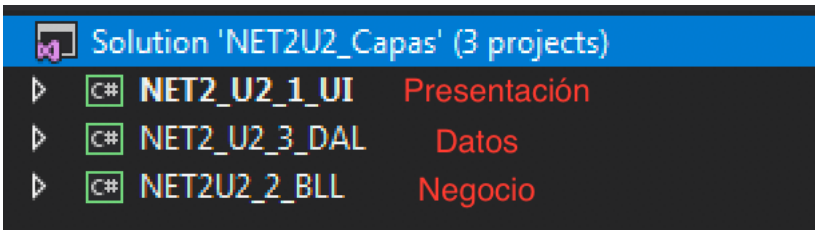
Cada una de esas capas debe tener en la solución:

- Su propio **namespace**.
- Su propio **Proyecto** y **assembly (exe, dll).**

Además las clases dentro de cada una de los proyectos no debe contener responsabilidades que no sean las que corresponden a su capa.

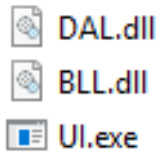
En Visual Studio se ve así:

Es muy común ver este **patrón** de 3 capas en muchos desarrollos, la próxima vez que abran una solución que no sea de ustedes y vean estas palabras BLL, DAL, UI o View van a saber que están trabajando multicapas. Reconocer este diseño tan común es un beneficio para entender otras aplicaciones.

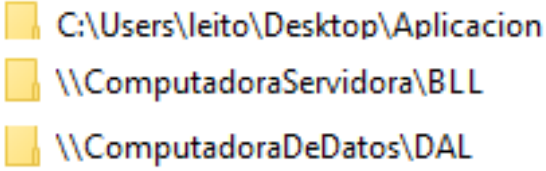


Separación Física

La separación física se refiere a la manera en que la aplicación es distribuida. Es posible tener una aplicación que contiene sus tres capas de datos, negocio y datos en una misma locación, como por ejemplo: una página web corriendo en mi máquina con 3 ensamblados (1 proyecto **ASP.NET** para la vista y otros 2 proyectos **Class library** para la capa de negocio y datos respectivamente) todo instalado en la misma carpeta o lo mismo vale para una aplicación escritorio:



Por otro lado también puedo tener cada capa distribuida en distintas locaciones físicas sea en la misma computadora o en computadoras separadas.

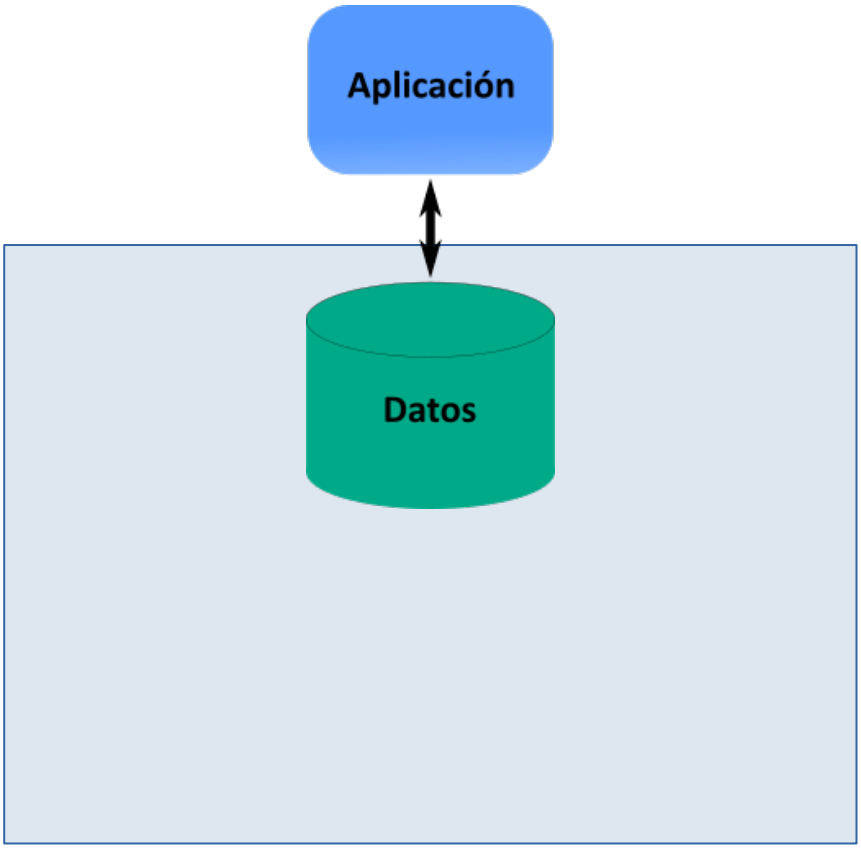


Como vimos se puede dividir lógicamente una aplicación sin separarla físicamente, pero no funciona en ambos sentidos: es **imposible** separar físicamente una aplicación sin separarla de manera lógica.

Separar capas físicamente en distintas computadoras nos permite que múltiples equipos puedan trabajar en distintas partes de una aplicación grande de manera simultanea.

Modelo Monolítico

Las aplicaciones simples comienzan típicamente con pocas capas, agregar capas a aplicaciones simples a



veces puede ser demasiado trabajado agregado sin ningún beneficio.

El dibujo de arriba representa a una aplicación en una computadora que puede ser una PC, un servidor o un smartphone. Los datos pueden estar guardados en la memoria de la aplicación o puede ser un archivo en una carpeta o una base de datos local como Access.
Esta aplicación es ideal para un solo usuario ya que contiene todo lo que necesita en la misma computadora. Además las habilidades requeridas para un desarrollador de este tipo de aplicaciones son simples, por ejemplo un programador que solo sabe C# puede crear esta aplicación usando proyectos consola o winforms desentendiéndose de SQL, HTML, XML, XAML, Javascript etc. Más tarde la aplicación podría refactorizada o re-escrita para agregar las capas.

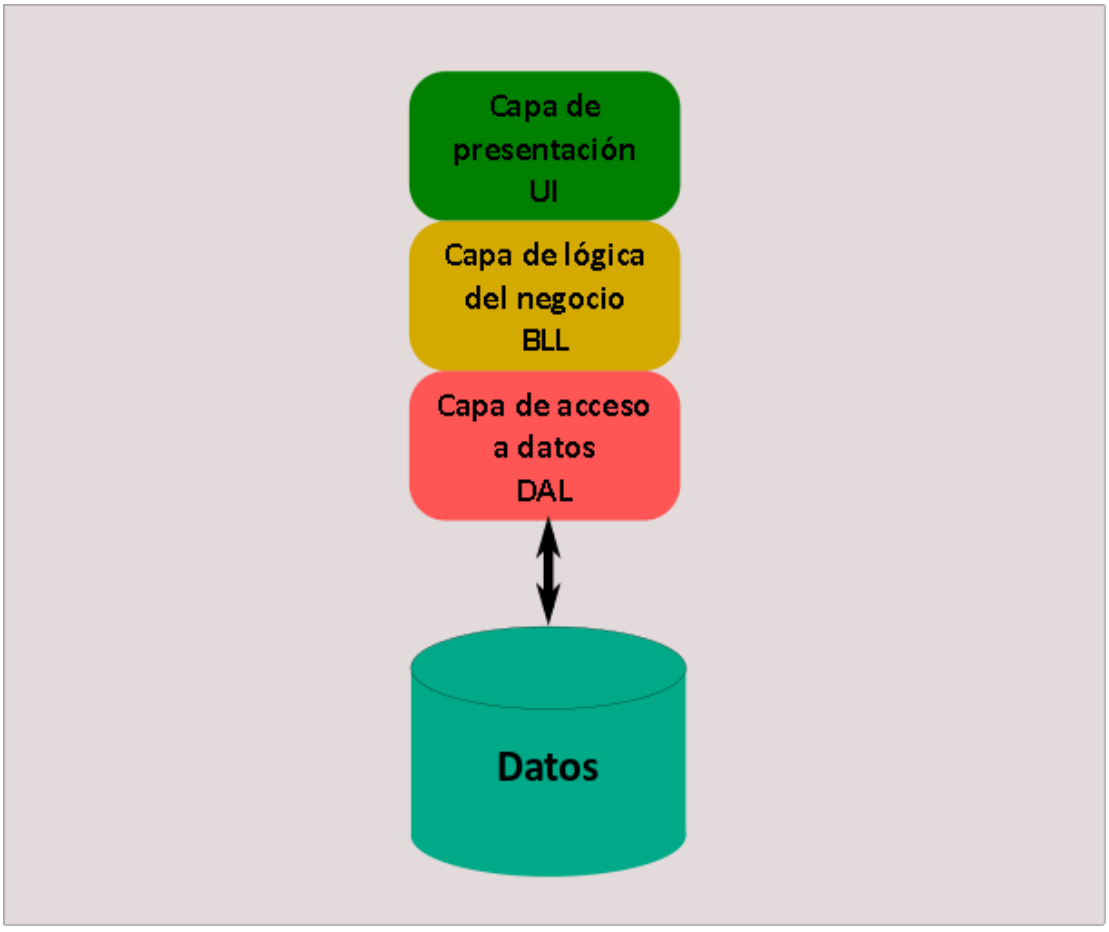
Los datos no son necesariamente una capa lógica en la aplicación, pero es una de las primeras cosas que se separa físicamente. Por ejemplo, una base de datos generalmente se instala en un servidor separado y administrada de manera independiente de las aplicaciones que la utilizan.

En este caso vamos a necesitar utilizar SQL agregándole complejidad a nuestro sistema, pero vamos a ganar seguridad en los datos de la aplicación y la posibilidad de optimizar la base de datos.

Modelo Multicapas

No es algo extraño encontrarnos con un diseño desestructurado con toda la lógica embebida con la UI y la capa de acceso a datos accediendo a una base de datos remota, especialmente entre hobbistas o para aplicaciones que son muy simples.

Sin embargo esto no es un diseño recomendable ya que para casos donde tengo más de una aplicación accediendo a la base de datos voy a tener que estar actualizando todas las aplicaciones de manera sincronizada para que puedan interactuar con la base de datos. Lo cual no es una tarea simple.



Esto va a hacer que un cambio en la base de datos o en las capas de negocio no afecten a mis usuarios, salvo que sea un cambio muy grande. También me va a ayudar a aplicar técnicas de mejora de performance o cache.

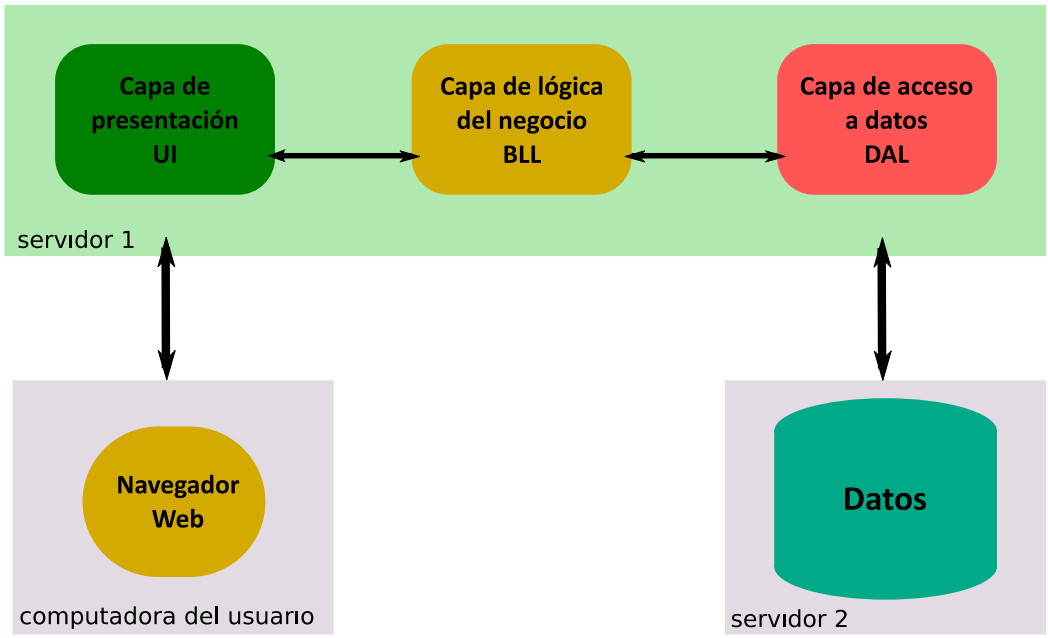
Separación física en la práctica

La separación física en la práctica de todas las capas no nos brinda muchas ventajas y nos puede dar dolores de cabeza en el futuro.

El problema de separar las capas físicamente es que vamos a perder las facilidades que nos brinda la plataforma como por ejemplo: agregar como referencia otro ensamblado, algo que necesitaremos para que las capas se comuniquen entre sí.

Otro patrón que se nos va a hacer difícil implementar en un modelo de separación física es el de **inyección de dependencias**, patrón que nos permite desacoplar las distintas capas por medio del uso de interfaces. Para poder llevarlo a la práctica necesitamos referencias a los ensamblados y al estar en distintas locaciones agregar ese referencias va a ser un trabajo más complejo.

Un modelo recomendado es tener las 3 capas en un mismo contenedor físico o servidor probablemente sera un servidor web y luego tener la base de datos en otro servidor separado para poder administrar sus recursos independientemente:



Mantener las cosas simples a veces es lo más complicado.