Material de lectura

Introducción a Desarrollo Web y Aplicaciones

Arquitectura Web distribuida

Para comenzar tengamos presente que hemos visto que una aplicación web tiene varios conceptos como arquitectura cliente servidor, Front End, Back End, Base de datos y Apis. Pero qué pasa cuando una aplicación sabemos que va a recibir muchas consultas de nuestra cliente.

Para poder dar respuesta a ello y antes de entender qué es una arquitectura distribuida hagamos el ejercicio de organizar cada concepto en su lugar y veamos cómo podemos distribuir nuestra aplicación.

Supongamos que nos juntamos varias compañeras del curso y tenemos un sistema web o aplicación web con la arquitectura Cliente / Servidor, en ella debemos organizar los conceptos que ya vimos como Front End, Back End, Base de datos y Api. Esta organización quedaría de esta manera:

Arquitectura Cliente / Servidor:

Cliente:
Front End: El código o programación que muestra información al navegador web llamado cliente.

Servidor:
Back End: El código o programación que ejecutas las acciones en el servidor y conecta con la base de datos.
Base de dato o DB: Donde se almacena la información.
Api: es quien nos permite conectar a otros sistemas.

¿Qué es una arquitectura centralizada?

Es cuando tenemos todos los elementos de nuestra aplicación web de arquitectura de Cliente / Servidor en un solo lugar equipo o servidor, es decir, tener el Back End, Front End, Bases de datos y APIs en el mismo equipo. Esto hace que en el caso de una falla del equipo toda nuestra aplicación también fallará.

¿Qué es la arquitectura distribuida?

Es tener la posibilidad y capacidad de separar nuestro sistema en distintos servidores de la red (sea red local o internet). Ya sabemos que cuando hablamos de arquitecturas estamos refiriéndonos a una estrategia de cómo construir nuestro sistema dependiendo de lo grande que sea, de las funcionalidades que tenga, esto es mas bien una forma de pensar en cómo escalar nuestro sistema para que soporte más usuarios o más transacciones.

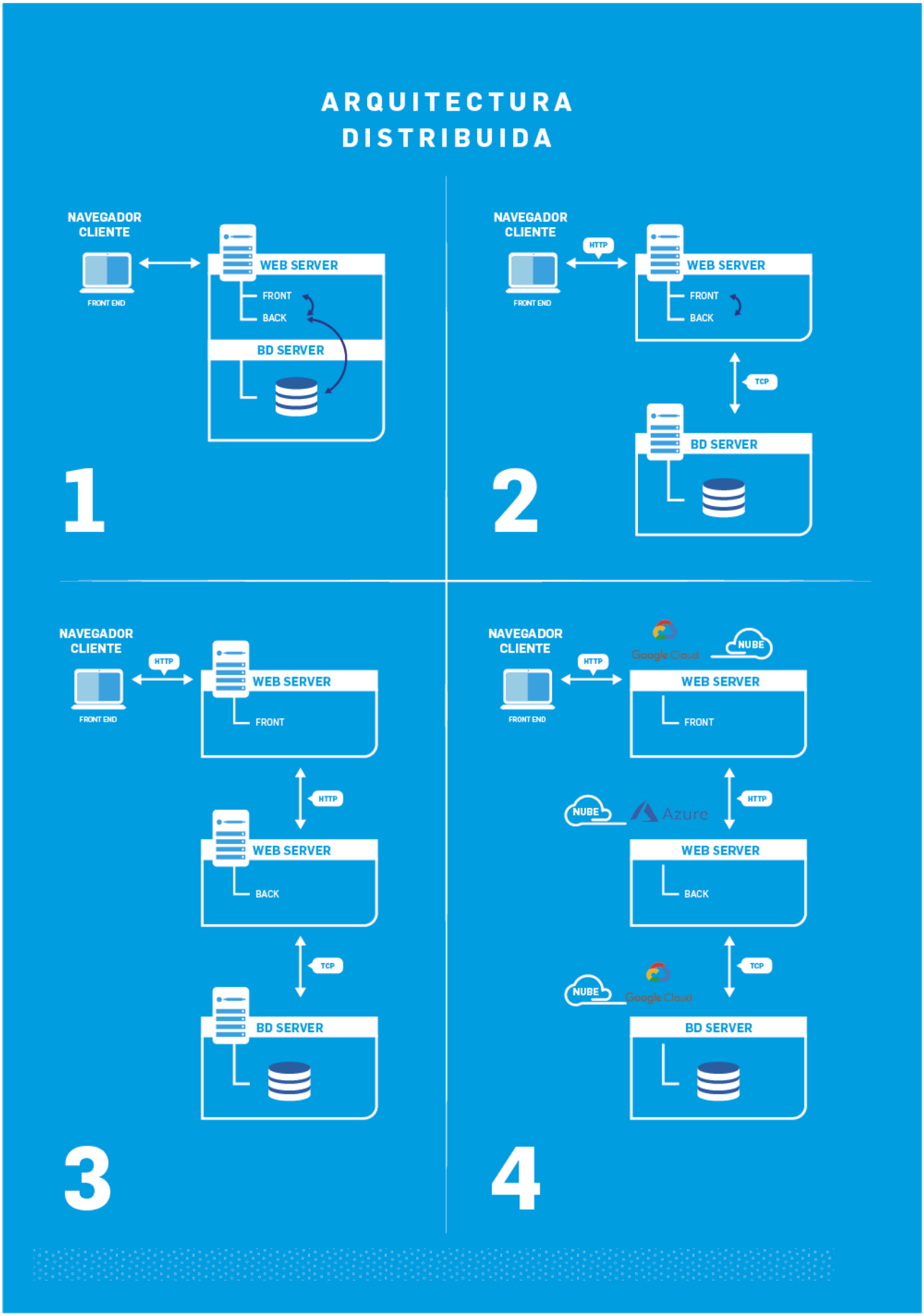
¿Pero cómo te das cuenta cuando una aplicación es distribuida?

En este punto depende de que estés mirando el proyecto o la aplicación, te compartimos solo 2 enfoques:

Como usuario: No te darías cuenta porque si el sistema está distribuido funciona como un conjunto único y sincronizado.
Como Programador: Cuando te asignen a un proyecto o cliente en base a preguntas concretas podrás ir conociendo cómo se implementó o distribuyo el sistema, pero te compartimos algunas formas en las que puedes entender que estás frente a una arquitectura distribuida:

1. ***Por la Líder del Proyecto:*** Cuando se comienza a trabajar en un proyecto generalmente la líder del mismo hace una explicación del tipo de aplicación con la que se está trabajando, además de indicarnos en qué parte del proyecto estaremos trabajando.
2. ***Por el perfil asignado***: Cuando nos asignan el trabajo en una empresa nos especifican si trabajaremos en el Front End, en el Back End o en ambos Full Stack, de esa manera podemos inferir que la arquitectura es distribuida, igualmente siempre es mejor preguntar para estar seguros.
3. ***Por un diagrama***: Generalmente se utilizan diagramas de aplicación para documentar un sistema, en el se puede ver la separación del sistema y si esta distribuido en 1 o varios servidores.

En los siguientes diagramas o esquemas de una aplicación que fue creada y pensada en forma distribuida separando el código en Front End , Back End. Podemos ver cómo se puede ir escalando o distribuyendo en distintos servidores y que en cualquier caso seguirá funcionando.



Analicemos juntos los escenarios de la imagen anterior:

Escenario 1: En este caso podemos ver que la aplicación está separada en Front End, Back End y bases de datos están en el mismo servidor. Notemos que la aplicación fue diseñada de forma modular o separada (para poder distribuirla) todas las partes del sistema están en un mismo servidor, es decir, en caso de fall del servidor afecta a todo el sistema.

Escenario 2: En este caso podemos ver que se ha separado la base de datos y el sistema sigue funcionando porque el desarrollador Back End escribió el código pensando en una arquitectura distribuida. Pero la parte del Front End y Back End aun están en un mismo servidor.

Ayuda

Texto a voz

Escenario 3: En este caso podemos ver que cada parte del sistema Front End, Back End y Base de datos esta en un servidor diferente. Con esto comenzamos a ver los beneficios del diseño con arquitectura distribuida en los sistemas.
Escenario 4: En este caso podemos ver que cada parte del sistema está en la nube de distintas empresas y nuestro sistema sigue funcionando por su diseño modular o distribuido.

¿Hasta dónde puedo modularizar o distribuir mi sistema?

El cómo distribuir el sistema es algo que se analiza en el diseño de la aplicación o se va cambiando a medida que el sistema va creciendo, normalmente cuando llegamos a un trabajo las aplicaciones ya están funcionando y necesitan de nuestro conocimiento para mantenerlas y agregarles mejoras.

Cuando un aplicación se hace más grande, compleja y con más usuarios necesitamos seguir modularizando el sistema, dado que no nos alcanza con separar en Front End, Back End y Base de datos. En esta situación ya debemos pensar en modularizar o separar algunas funcionalidades del sistema, algunos motivos pueden ser:

Por alta demanda: Cuando el sistema tiene una funcionalidad que es compleja, consume mucho recurso del servidor o es muy demandada por distintas partes del sistema.
Por interconexión: Cuando un sistema tiene funcionalidades que se necesita dar acceso a otros sistemas para consumir ese proceso, función o datos.
Por segregación de roles: Cuando es necesario separar roles o funciones por motivos de seguridad o aspectos técnicos, también puede ser porque negocio lo requiere, por ejemplo si se decide por seguridad separar el proceso de autenticación del sistema para reforzar la seguridad.
Por escalamiento: Cuando las proyecciones indican que en un periodo de tiempo la demanda aumentará considerablemente, será necesario agregar más servidores en la red con la misma funcionalidad para que satisfaga la demanda.

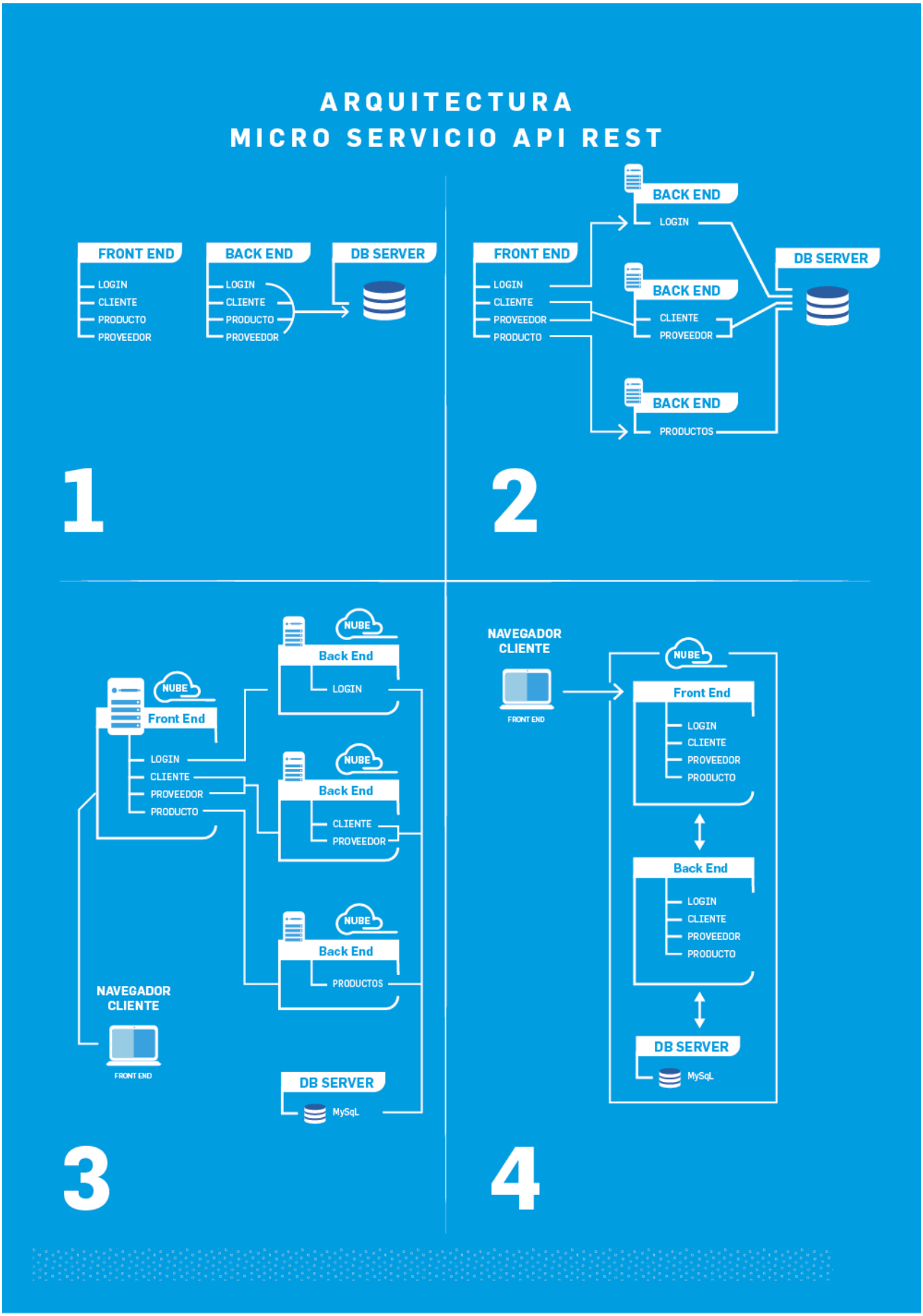
Existen varias **formas de separar estas funcionalidades que llamaremos API REST o Microservicios**, si bien a medida que avancemos iremos aprendiendo más sobre las API REST, ahora veremos cómo la arquitectura distribuida puede aplicarse para pasar de un gran sistema que tiene todo en un solo lugar a separarlo en pequeñas y que todo siga funcionando.

Para ejemplificar tomaremos un sistema que tiene lo siguiente:

Front End: Todos fue diseñado pensando en una arquitectura distribuida, las funcionalidades son las siguientes.
Login: Se conecta con el Back End para validar el usuario y clave.
Cliente: Se conecta con el Back End para consultar, editar, crear y eliminar clientes
Producto: Se conecta con el Back End para consultar, editar, crear y eliminar productos
Proveedor: Se conecta con el Back End para consultar, editar, crear y eliminar proveedores

Back End: Todos fue diseñado pensando en una arquitectura distribuida, las funcionalidades son las siguientes,
Loguin: Recibe la petición, consulta la base de datos y valida si el usuario.
Cliente: Recibe la petición, consulta la base de datos y devuelve los datos de cliente.
Producto: Recibe la petición, consulta la base de datos y devuelve los datos del producto.
Proveedor: Recibe la petición, consulta la base de datos y devuelve los datos del proveedor.

Ahora veamos en un diagrama o esquema cómo estas funcionalidades se pueden ir distribuyendo en distintos servidores:



Analicemos juntos los escenarios de la imagen anterior:

Escenario 1: En esta caso tenemos el sistema que esta en 3 servidores para cada separación

1. Front End: Todas las funcionalidades están juntas y en el mismo servidor.
2. Back End: Todas las funcionalidades están juntas y en el mismo servidor.
3. Base de datos: Todos los datos en una base de datos y en el mismo servidor.

Ayuda

Texto a voz

Escenario 2: En este caso **podemos ver que cada parte del sistema está distribuido en 5 servidores** y el sistema sigue funcionando porque el desarrollador Back End escribió el código pensando en una arquitectura distribuida.

1. Font End: Todas las *funcionalidades están juntas* y en el mismo servidor.
2. Back End: La funcionalidad de *login está en un servidor* exclusivo.
3. Back End Las funcionalidades de *cliente y proveedor están en un servidor*.
4. Back End: La funcionalidad *producto está en un servidor* exclusivo.
5. Base de datos: Todos los datos en una base de datos y en el mismo servidor.

Escenario 3: En este caso **podemos ver que cada parte del sistema está distribuido en 5 servidores en nubes diferente**s y el sistema sigue funcionando porque el desarrollador Back End escribió el código pensando en una arquitectura distribuida.

1. Font End: Todas las *funcionalidades están juntas* y en el mismo servidor de la nube de Argentina.
2. Back End: La funcionalidad de *login está en un servidor de la nube* de España.
3. Back End Las funcionalidades de *cliente y proveedor están en un servidor* de la nube de Canadá.
4. Back End: La funcionalidad *producto está en un servidor en la nube* de Nueva Zelanda.
5. Base de datos: Todos los datos en una base de datos y en un servidor propio.

Escenario 4: En este caso **podemos ver que el sistema está en la misma nube,** a pesar que las funcionalidades están todas juntas como el en escenario 1 y nuestro sistema sigue funcionando.

1. Font End: Todas las *funcionalidades están juntas* y en el mismo servidor de la nube de Argentina.
2. Back End: Todas las *funcionalidades están juntas* y en la nube de Argentina.
3. Base de datos: Todos los datos en una base de datos y en la nube de Argentina.

Como hemos visto hay varios niveles de como modularizar, distribuir o separar en capas una aplicación con sus funcionalidades, en algunos cosos la separación solo puede ser Font End, Back End. Pero existen otras formas de separar en partes mas pequeñas la aplicación y eso lo hacemos con la ayuda de las APIs REST.

Actividad previa

◄ Arquitectura de las aplicaciones Web

Siguiente actividad

Elementos de la arquitectura Web ►

Resumen de retención de datos

Ayuda

Texto a voz