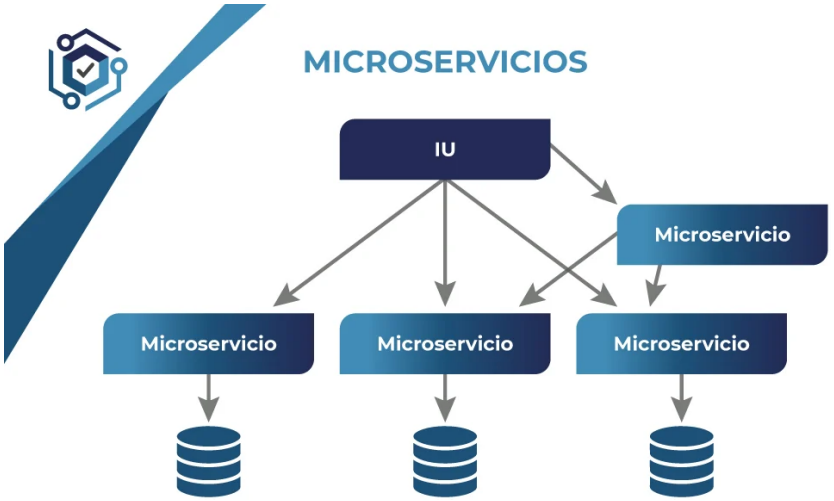
1. ¿En qué casos sería relevante e indispensable contar con este tipo de servicios?

Los **microservicios** son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes.

Con las *arquitecturas monolíticas*, todos los procesos están estrechamente asociados y se ejecutan como un solo servicio. Esto significa que, si un proceso de una aplicación experimenta un pico de demanda, se debe escalar toda la arquitectura. Agregar o mejorar las características de una aplicación monolítica se vuelve más complejo a medida que crece la base de código. Las arquitecturas monolíticas aumentan el riesgo de la disponibilidad de la aplicación porque muchos procesos dependientes y estrechamente vinculados aumentan el impacto del error de un proceso, ya que si se origina un error en un servicio, puede propagarse el error a los demás.

En cambio con una **arquitectura de microservicios**, una aplicación se crea con componentes independientes que ejecutan cada proceso de la aplicación como un servicio. Los servicios se crean para desempeñar una única función. Cada servicio es un componente en una arquitectura de microservicios y la comunicación entre ellos ocurre a través de APIs bien definidas. Cada uno de ellos se puede desarrollar, operar y escalar sin afectar el funcionamiento de otros servicios. Es decir, permite que cada servicio escale de forma independiente para satisfacer la demanda de la aplicación que respalda.



Además, cada servicio está diseñado para resolver un *único problema* específico. Si se aportará más código a un servicio a lo largo del tiempo, este se volverá más complejo, por lo cual, se podría dividir en servicios más pequeños y simplificados.

Los **microservicios** son útiles en una variedad de situaciones, generalmente para construir aplicaciones escalables, flexibles y resilientes. En los casos donde pueden ser útiles son:

A. Si una aplicación necesita escalar horizontalmente para manejar un mayor volumen de tráfico o procesamiento. Cada microservicio puede ser escalado de manera independiente, lo que permite agregar recursos solo donde se necesitan, en lugar de tener que escalar todo el sistema.
B. Si se busca facilitar el trabajo en equipo, los microservicios pueden simplificar el mantenimiento de una aplicación al permitir que se realicen actualizaciones y cambios de forma aislada. Esto significa que los desarrolladores pueden trabajar en actualizaciones y mejoras en partes específicas de la aplicación sin tener que preocuparse por el impacto en otras partes del sistema.

C. Son una buena opción para aplicaciones que se ejecutan en arquitecturas distribuidas, donde diferentes partes de la aplicación pueden ejecutarse en diferentes ubicaciones físicas o lógicas.

2. Más allá de haberse implementado a través de una arquitectura de Servidor HTTP. ¿Qué otra tecnología podría haberse utilizado para ejecutar tareas remotas?

Existen diferentes tecnologías que pueden utilizarse para ejecutar tareas de forma remota. Además de la arquitectura de servidor HTTP, se podría haber utilizado RPC (Remote Procedure Call) o comunicación mediante colas de mensajes.

- **Remote Procedure Call (RPC)** es un protocolo que permite a un programa en una computadora, solicitar un servicio de otro programa ubicado en otro proceso dentro de la red. Se utiliza para llamar a funciones o procedimientos de forma remota, de manera transparente y abstrayéndose de la complejidad de la comunicación en red.

La idea detrás de RPC es que su implementación permite que la llamada a procedimientos en otro proceso remoto, funciona de forma similar a una llamada de procedimiento local.

- La **comunicación mediante cola de mensajes** es una técnica para comunicar aplicaciones de forma asíncrona y distribuida a través de una cola de mensajes. En una arquitectura de cola de mensajes, los mensajes se envían a una cola centralizada, donde se almacenan temporalmente y se procesan en orden de llegada. De esta manera, los procesos que envían y reciben mensajes se comunican a través de ella.

3. Esta solución, si bien es escalable, presenta una limitante a nivel de “sincronismo” entre las partes. ¿Cómo se le ocurre que podría desacoplar las partes y hacer que la solución sea más escalable?

Cómo bien decimos, se puede implementar la comunicación entre partes utilizando colas de mensajes. En lugar de que los microservicios se comuniquen directamente entre sí a través de HTTP, se puede utilizar una cola de mensajes como intermediario. Esto permite que los procesos se comuniquen de manera asíncrona y reduzcan la dependencia entre ellos.

En nuestra implementación, la comunicación entre procesos es síncrona, ya que el proceso emisor realiza una petición y espera que el proceso receptor lo reciba, lo procese y envíe una respuesta. Recién ahí, sigue con su ejecución. Es decir que las partes están acopladas, ya que el cliente se encuentra conectado con el servidor hasta obtener el resultado.

En cambio, si pensamos una implementación donde la comunicación es asíncrona, los procesos funcionarían de forma independiente, ya que el proceso emisor envía un mensaje y continúa su trabajo sin esperar la respuesta inmediata del proceso receptor. La comunicación asíncrona es útil en sistemas distribuidos donde los procesos pueden estar separados geográficamente y tener diferentes velocidades de procesamiento.

Además, la comunicación asíncrona puede mejorar la tolerancia a fallos del sistema distribuido. En nuestro caso, el proceso cliente depende por completo del proceso servidor para obtener el resultado. Se encuentra bloqueado esperando la respuesta, por lo cual, no puede pedirle a otro proceso que realice esa misma operación como segunda alternativa.

Entonces, la comunicación asíncrona puede mejorar la tolerancia a fallos en el sentido que, un proceso puede enviar un mensaje a una cola de mensajes y, si más de un consumidor está suscrito a esa cola, uno podrá responder en lugar del otro en caso de que ocurran fallas. Esto permite que el proceso emisor no dependa de un único consumidor.

En resumen, si se requiere reducir el acoplamiento entre procesos y aumentar la tolerancia a fallos, el uso de cola de mensajes para implementar una comunicacion asincrona, puede ser una buena alternativa al uso de HTTP.

En resumen, si se requiere una comunicación asíncrona y tolerancia a fallos entre procesos, MQ puede ser una buena alternativa a HTTP.