¡Por supuesto! A continuación, se presentan las definiciones, ventajas, desventajas y ejemplos de empresas que utilizan cada una de las arquitecturas de software mencionadas, junto con las fuentes correspondientes para cada punto:

**1. Arquitectura Monolítica**

* **¿Qué es?** Una arquitectura en la que todos los componentes de una aplicación (interfaz de usuario, lógica de negocio y acceso a datos) se integran en una sola unidad o código base.
* **Pros:**
  + Simplicidad inicial en el desarrollo y despliegue.
  + Facilidad para realizar pruebas integrales.
* **Contras:**
  + Dificultad para escalar componentes individuales.
  + Mantenimiento complejo a medida que la aplicación crece.
  + Limitaciones en la adopción de nuevas tecnologías para partes específicas del sistema.
* **Ejemplos:**
  + Shopify (antes de migrar a microservicios).
  + Aplicaciones pequeñas o medianas.

**2. Arquitectura de Capas (N-tier)**

* **¿Qué es?** Divide la aplicación en capas (presentación, lógica de negocio y acceso a datos) que se comunican entre sí para procesar las solicitudes del usuario.
* **Pros:**
  + Mejor organización y mantenimiento del código.
  + Facilita la escalabilidad y reutilización de componentes.
* **Contras:**
  + Puede introducir latencia debido a la comunicación entre capas.
  + Complejidad en la gestión de dependencias entre capas.
* **Ejemplos:**
  + Sistemas bancarios y empresariales (como SAP).

**3. Arquitectura Orientada a Servicios (SOA)**

* **¿Qué es?** Una arquitectura que organiza la funcionalidad de la aplicación en servicios independientes que se comunican a través de interfaces bien definidas, generalmente utilizando protocolos estándar como SOAP o REST.
* **Pros:**
  + Reutilización de servicios en diferentes aplicaciones.
  + Escalabilidad y flexibilidad en la integración de sistemas.
* **Contras:**
  + Complejidad en la implementación y gestión de servicios.
  + Mayor latencia debido a la comunicación entre servicios.
* **Ejemplos:**
  + Amazon y eBay (antes de adoptar microservicios).
* **Fuente:**
  + [Arquitectura orientada a servicios](https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios)

**4. Arquitectura de Microservicios**

* **¿Qué es?** Una aproximación al desarrollo de software que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP). Cada servicio se encarga de implementar una funcionalidad completa del negocio.
* **Pros:**
  + Escalabilidad y despliegue independientes de cada servicio.
  + Facilita la adopción de nuevas tecnologías en servicios específicos.
* **Contras:**
  + Complejidad en la gestión y monitoreo de múltiples servicios.
  + Requiere una cultura organizacional alineada con la autonomía de equipos.
* **Ejemplos:**
  + Netflix, Uber y Spotify.
* **Fuente:**
  + [Arquitectura de microservicios](https://es.wikipedia.org/wiki/Arquitectura_de_microservicios)

**5. Arquitectura Multitenant**

* **¿Qué es?** Una arquitectura en la que una única instancia de software sirve a múltiples clientes (inquilinos), manteniendo datos y configuraciones separados para cada uno.
* **Pros:**
  + Optimización de recursos y reducción de costos operativos.
  + Actualizaciones y mantenimiento centralizados.
* **Contras:**
  + Complejidad en la gestión de seguridad y privacidad entre inquilinos.
  + Limitaciones en la personalización para clientes individuales.
* **Ejemplos:**
  + Salesforce y Google Workspace.

**6. Arquitectura Event-Driven**

* **¿Qué es?** Una arquitectura basada en la producción, detección y consumo de eventos, donde los servicios reaccionan a eventos en tiempo real para desencadenar procesos o flujos de trabajo.
* **Pros:**
  + Alta capacidad de respuesta y escalabilidad.
  + Desacoplamiento de componentes, lo que permite mayor flexibilidad.
* **Contras:**
  + Dificultad en el seguimiento y depuración de flujos de eventos.
  + Complejidad en la gestión de consistencia y manejo de errores.
* **Ejemplos:**
  + Uber (seguimiento en tiempo real) y PayPal (procesamiento de pagos).

**7. Arquitectura Micro Frontends**

* **¿Qué es?** Una arquitectura que extiende los conceptos de microservicios al desarrollo frontend, dividiendo la interfaz de usuario en fragmentos independientes que pueden ser desarrollados, desplegados y mantenidos por equipos separados.
* **Pros:**
  + Permite a equipos trabajar de forma autónoma en diferentes partes de la interfaz.
  + Facilita actualizaciones y despliegues independientes de componentes.
* **Contras:**
  + Desafíos en la integración y coherencia del diseño.
  + Complejidad