**Arquitectura Monolítica**

**Definición:**

La arquitectura monolítica es un modelo de diseño de software donde todos los componentes y funcionalidades de una aplicación se integran en un único bloque o unidad. Este enfoque centraliza el desarrollo, despliegue y mantenimiento del software.

**Explicación con tus palabras:**

Es como tener todo el código de una aplicación en un solo proyecto grande. Cada funcionalidad, desde el manejo de usuarios hasta el procesamiento de pagos, está en un solo lugar.

**Beneficios**:

1. Desarrollo y despliegue sencillos.
2. Facilidad para la depuración y las pruebas.
3. Menor latencia de comunicación entre componentes.

**Desafíos**:

1. Dificultad para escalar horizontalmente.
2. Mantenimiento complejo a medida que la aplicación crece.
3. Implementación de cambios y actualizaciones más riesgosa.

**Empresas que la Utilizan**:

1. Algunas startups y pequeñas empresas que buscan una solución rápida y sencilla.
2. Aplicaciones tradicionales y heredadas de empresas más grandes.

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

**Definición:** La arquitectura de capas, también conocida como N-tier, es un modelo de diseño de software que divide una aplicación en varias capas o niveles, cada uno con responsabilidades específicas. Cada capa interactúa solo con las capas adyacentes, lo que facilita la organización, escalabilidad y mantenimiento del sistema.

Obtenido en ChatGPT.

**Explicación con tus palabras:**

Imaginemos una persona vistiéndose, donde los pies son una capa, los calcetines la segunda y los zapatos la tercera, yo no puedo interactuar con los zapatos sin antes haber interactuado con los calcetines, de hecho, si se puede, pero no se debería, en la arquitectura en capas es similar, se basa en una arquitectura vertical donde cada capa interactúa con la predecesora o sucesora que trabajan en conjunto con un fin, como ejemplo vestirse correctamente como lo es en este caso.

**Beneficios:**

1. Capas independientes lo que facilita la gestión.
2. Fácil desarrollo.
3. Facilidad para realizar pruebas unitarias.
4. Cada modificación realizada no afecta a otra capa.
5. Facilidad para implementar medidas de seguridad.

**Desafíos:**

1. Latencias entre interacción de capas.
2. Aumento en consumo de recursos.
3. Puede llegar a existir la dependencia de capas.
4. No promueve el desarrollo ágil.
5. Lentitud al comienzo del proceso de desarrollo.

Empresas que la utilizan:

1. Empresas donde se requiera un desarrollo rápido.
2. Utilizada por desarrolladores con poca experiencia, recomendado para realizar tu primer proyecto.

**Arquitectura Orientada a Servicios (SOA)**

**Definición:**

La Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) es un estilo de diseño de software en el que los componentes de una aplicación se dividen en servicios independientes y autónomos. Cada servicio realiza una función específica y se comunica con otros servicios a través de interfaces bien definidas.

Obtenido en ChatGPT.

**Explicación con tus palabras:**

Pensemos en una institución educacional, donde el director les pide a los profesores a cargo que realicen un informe sobre un método de enseñanza ya inventado y lo apliquen durante este año, cada profesor funcionaria de forma independiente. A pesar de que todos trabajen de manera diferente, todos poseen un fin común, pudiendo interactuar con otros profesores y comparando sus metodologías y/o trabajando en equipo.

**Beneficios:**

1. Reutilización deservicios, ayudando a mejorar los tiempos de creación de la aplicación empresarial, ya que, se prioriza la implementación de servicios ya creados que la creación de nuevos servicios.
2. Mejora la escalabilidad y flexibilidad, ya que, en cada servicio se puede escalar de manera independiente según sus necesidades.
3. Obtenemos un mantenimiento eficiente**,** nos encontramos con una programación “modular” de servicios, estos pueden ser mantenidos de manera independiente sin interrumpir el funcionamiento del resto.

**Desafíos:**

1. Gestión compleja, ya que, a medida que se agregan más servicios para atender nuevas demandas, la gestión se vuelve cada vez más compleja. Cada nuevo servicio trae componentes que deben ser monitoreados, coordinados y mantenidos, lo que hace más difícil controlar y gestionar el sistema en su conjunto.

**Empresas que la utilizan:**

1. Empresas que desarrollen aplicaciones empresariales o plataformas compleja.
2. Instituciones que interactúen con gran volumen de datos, transacciones o servicios.
3. Empresas de telecomunicaciones.
4. Entre otras.

**Arquitectura de Microservicios**

**Definición:**

La arquitectura de microservicios es un enfoque de diseño donde una aplicación se descompone en pequeños servicios independientes que pueden ser desarrollados, desplegados y escalados de manera autónoma.

Obtenido en ChatGPT.

**Explicación con tus palabras:**

Imaginemos un restaurante donde la cocina está dividida en varias secciones, un área de carnes, un área de sopas y otra de pastas, cada área ofrece un servicio al restaurante, cada una de esta tiene un rol definido, pero todas trabajando en conjunto y así otorgarle un servicio de calidad a los clientes, gracias a esto el cliente solo observa la carta y pide a gusto independiente de su requerimiento y lo que conlleva.

**Beneficios:**

1. Mejor Escalabilidad, debido a que cada microservicio puede escalar dependiendo de sus necesidades.
2. Mejoramiento en tiempos de desarrollo, debido a que cada microservicio es independiente al otro, este puede ser desarrollado o modificado sin intervenir otro.
3. Resiliencia entre microservicios, en caso de fallas en un microservicio, este se puede intervenir sin afectar a otro.

**Desafíos:**

1. Infraestructura compleja,
2. Dificultad para lograr una coherencia de datos entre microservicios.
3. Costo de implementación elevado, dependiendo de cuantos microservicios serán utilizados, esto demanda un desembolso de dinero mayor a comparación de otras arquitecturas.

**Empresas que la utilizan:**

1. Empresas tecnológicas.
2. Plataformas de comercio electrónico.

Entre otras empresas de gran impacto.

Esta arquitectura esta pensada en empresas que buscan mejorar la escalabilidad, flexibilidad, disponibilidad y adaptación de su aplicación.

**Arquitectura Multitenant**

**Definición:**

La arquitectura multitenant es un modelo donde una única instancia de software sirve a múltiples clientes o "inquilinos". Cada cliente tiene su propio espacio de trabajo, pero todos comparten los mismos recursos y la misma infraestructura.

Obtenido en ChatGPT.

**Explicación con tus palabras:**

Una pantalla de un restaurante

El contenido generado por IA puede ser incorrecto.

Supongamos que tengo un espacio en donde puedo ofrecer diferentes tipos de arriendo a mis locatarios, en uno donde ofrezco todo el local completo, todo independiente, seguridad, ganancias, gastos, cuentas, etc.. , otro donde subdivido el local en 2 pero trabajan con nombres diferentes y la tercera opción es que comparten todo el mismo local, donde los ingresos van a la misma cuenta y se diferencian con un identificador las ganancias de cada uno.

Un ejemplo de esto podría ser un patio de comidas, con diferentes locales y con sistemas diferentes.

Otro ejemplo podría ser un salón de belleza con diferentes emprendedores trabajando en el mismo local, donde solo se comparte el salón.

Y el otro ejemplo es una peluquería donde contrato varios peluqueros, donde todos trabajan en conjunto, pero cada uno tiene identificado cuanto gano por día.

Por ende, en esta arquitectura deben considerar 3 aspectos cruciales:

1. Nivel de aislamiento.
2. Como restauraremos la información.
3. Nivel de encriptación.

Considerando esto, se puede elegir entre optar por una base de datos para cada organización, esquemas separados por cada organización o un esquema compartido por las organizaciones, dentro las cuales cada una tiene sus ventajas y desventajas.

**Beneficios:**

1. **Base de datos independiente por tenant:** 
   1. Alto nivel de seguridad.
   2. Flexibilidad en datos.
   3. Aislamiento total.
2. **Schema por tenant:**
   1. Bajo costo.
   2. Aislamiento moderado.
   3. Escalabilidad.
3. **Shared schema:**
   1. Eficacia en recursos.

**Desafíos:**

1. **Base de datos independiente por tenant:** 
   1. Costo.
2. **Schema por tenant:**
   1. Aislamiento reducido.
   2. gestión.
3. **Shared schema:**
   1. Separación de datos.
   2. Ecalabilidad limitada.

**Empresas que la utilizan:**

Empresas de plataformas o software de nube.

Plataformas de tiendas en línea.

Empresas con contenido multiusuario.

Etc..

**Arquitectura Event-Driven**

**Definición:**

La arquitectura orientada a eventos (Event-Driven Architecture, EDA) es un patrón de diseño donde los sistemas reaccionan a eventos o sucesos, como cambios en el estado de los datos o la llegada de nuevos datos, para activar una acción o proceso.

Obtenido en ChatGPT.

**Explicación con tus palabras:**

Imagina que quieres ir de viaje, pero tienes que esperar un evento para poder viajar, el conocido pago del mes, donde se genera el evento “pago”, pudiendo realizar este viaje a gusto, pero lamentablemente no sabes como llegar a la ubicación, y tienes que esperar que se genere otro evento “Recibir ubicación”, te quedas ahí en el terminal esperando a que algún familiar te responda y te diga cómo llegar al hotel.

**Beneficios:**

1. Promueve el desacoplamiento de componentes.
2. Facilita la escalabilidad.
3. Flexibilidad.
4. Promueve la resiliencia.
5. Respuesta en tiempo real.

**Desafíos:**

1. Complejidad en la implementación y desarrollo.
2. Trazabilidad.

**Empresas que la utilizan:**

Empresas de mensajería.

Sistemas de análisis de tiempo real.

**Arquitectura Micro Frontends**

**Definición:**

La arquitectura de micro frontends es un enfoque donde el frontend de una aplicación se divide en partes independientes (micro frontends), cada una de las cuales puede ser desarrollada, desplegada y gestionada de manera autónoma, pero todas juntas forman la interfaz completa.

Obtenido en ChatGPT.

**Explicación con tus palabras:**

Considera esta arquitectura como un completo, por un lado, haces las vienesas, por otro lado, el tomate y por otro la palta, donde tenemos como base el pan, al tener todo listo, unes las piezas y logras tener un rico completo, el cual se puede observar en diferentes formas por mientras que lo vas preparando y/o comiendo.

Pudiendo visualizar de forma independiente cada ingrediente si así lo deseas.

**Beneficios:**

1. Desarrollo independiente de las aplicaciones.
2. Promueve la escalabilidad en caso de ser requerida a futuro.
3. Flexibilidad tecnológica.

**Desafíos:**

1. Integración de componentes.
2. Rendimiento.

**Empresas que la utilizan:**

1. **Spotify:** Utiliza micro frontends para dividir su aplicación web en secciones independientes.
2. **IKEA:** Ha adoptado micro frontends para permitir que diferentes equipos trabajen en partes separadas de su sitio web