**Actividad: Investigación de arquitecturas, por GitHub**

**Integrantes:** Tiare Acuña, Ricardo Rodríguez, Maximiliano Rubilar.

Investigar y desarrollar la siguiente actividad sobre diferentes arquitecturas de software, donde deberás encontrar y escribir sus definiciones, empresas que las utilizan, y casos de estudio relevantes.

Una vez investigado comentar con tus compañeros.

**Completar lo siguiente:**

* ¿Qué es?: Definición y explicación con lo que entiendes de la arquitectura.
* Beneficios y desafíos enfrentados (ventajas y desventajas).
* Empresas que la utilizan: Ejemplos de empresas que han adoptado esta arquitectura o técnica y cómo la implementan.

**Temas a Investigar:**

1. Arquitectura Monolítica
2. Arquitectura de Capas (N-tier)
3. Arquitectura Orientada a Servicios (SOA)
4. Arquitectura de Microservicios
5. Arquitectura Multitenant
6. Arquitectura Event-Driven
7. Arquitectura Micro Frontends

**Recomendaciones para la Investigación:**

* Uso de Herramientas de Colaboración: Utilizar plataformas como Google Docs, Miro y Trello para coordinar y compartir información.
* Asignación de Tareas: Dividir los temas entre los miembros del grupo de manera equitativa.

**Desarrollo:**

**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**:

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

**Desafíos**:

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

**Empresas que la Utilizan**:

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

🤖🦝🤖

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

**Definición:**

La Arquitectura de Capas (N-tier) organiza una aplicación en diferentes capas donde cada capa tiene una función específica. Generalmente las capas incluyen la presentación, la lógica de negocio y el acceso a datos.

**Explicación:**

Es como dividir una aplicación en partes donde cada parte tiene su propio rol esto permite modificar capas sin afectar a las demás lo que facilita el mantenimiento y mejoras.

**Beneficios:**

* Puedes modificar o añadir partes sin afectar el sistema.
* Es más fácil actualizar y corregir errores en cada capa de forma independiente.
* Las capas pueden ser realizadas en otros proyectos.
* Diferentes equipos pueden trabajar en distintas capas de manera simultánea.

**Desafíos:**

* La estructura de varias capas puede hacer que la aplicación sea más difícil de manejar.
* La comunicación entre capas puede reducir la velocidad de la aplicación.
* Requiere más tiempo y recursos para implementarla y mantenerla.

**Empresas que la Utilizan:**

* La utilizan empresas grandes para desarrollar aplicaciones escalables y eficientes

en sistemas grandes y complejos.

* se utiliza en empresas de alto rendimiento que necesitan manejar grandes cantidades de datos y aumentar sus servicios con facilidad.

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

**Definición:**

La Arquitectura Orientada a Servicios (SOA) es un enfoque de diseño de software donde las funciones del sistema se dividen en servicios independientes, que se comunican entre sí a través de interfaces definidas.

**Explicación:**

Cada función es un servicio independiente, estos servicios interactúan entre sí para formar un sistema lo que permite modificar o agregar servicios sin afectar al resto del sistema.

**Beneficios:**

* Los servicios se pueden escalar por separado.
* Se pueden agregar o modificar servicios fácilmente.
* Cada servicio es independiente, lo que facilita los cambios y mantenimiento.

**Desafíos:**

* Gestionar múltiples servicios es complicado.
* La comunicación entre servicios puede afectar la velocidad.
* Mayor exposición a ataques si no se implementan medidas adecuadas.
* Si la red falla, afecta el sistema completo.

**Empresas que la Utilizan:**

* Las empresas lo utilizan para la optimización del rendimiento: Al tener servicios independientes, pueden optimizar cada parte del sistema, asegurando un rendimiento eficiente y sin interrupciones.
* También se utiliza para tener flexibilidad en el desarrollo: Cada servicio se puede modificar o actualizar independientemente sin interrumpir el resto de la infraestructura.

**Arquitectura** **de Microservicios**

**Definición:**

La arquitectura de microservicios es un estilo de desarrollo de software en el que una aplicación se divide en múltiples servicios pequeños e independientes. Cada uno de estos servicios se encarga de una funcionalidad específica y se comunica con los demás a través de APIs, lo que permite mayor flexibilidad y escalabilidad.

**Explicación con mis palabras:**

En lugar de tener una aplicación gigante, se crean pequeños servicios especializados que trabajan juntos y pueden actualizarse de forma independiente.

**Beneficios:**

* Cada microservicio puede crecer de forma independiente sin afectar a los demás.
* Se pueden usar diferentes tecnologías según las necesidades de cada servicio.
* Si un servicio falla, no necesariamente afecta a toda la aplicación.
* Los equipos pueden trabajar en paralelo en diferentes servicios sin depender unos de otros.

**Desafíos:**

* Se necesita una buena gestión para coordinar la comunicación entre los microservicios.
* Como cada servicio tiene su propia base de datos, puede ser complicado sincronizar información.
* Se requiere una infraestructura más avanzada para manejar la comunicación y despliegue de múltiples servicios.

**Empresas que la utilizan:**

* **Netflix:** Divide su sistema en microservicios para manejar recomendaciones, transmisión de contenido y autenticación de usuarios de forma separada.
* **Amazon:** Utiliza microservicios para su plataforma de e-commerce, permitiendo gestionar pedidos, pagos y envíos de manera eficiente.
* **Uber:** Ha migrado de una arquitectura monolítica a microservicios para manejar mejor su crecimiento y la cantidad de usuarios conectados.

**Arquitectura** **Multitenant**

**Definición:**

La arquitectura multitenant es un modelo en el que una misma aplicación o infraestructura de software es compartida por varios clientes (**tenants**), pero cada uno tiene su propia configuración y datos aislados de los demás.

**Explicación con mis palabras:**

Significa que una misma plataforma puede servir a muchas empresas o usuarios sin que sus datos se mezclen.

**Beneficios:**

* En lugar de tener múltiples servidores para cada cliente, se comparte la infraestructura, reduciendo costos.
* Las mejoras y cambios se aplican a todos los usuarios al mismo tiempo.
* Se pueden agregar nuevos clientes sin necesidad de desplegar nuevas instancias de la aplicación.

**Desafíos:**

* Es fundamental asegurarse de que los datos de cada cliente estén correctamente separados.
* Aunque se puede adaptar, es más difícil ofrecer soluciones completamente personalizadas a cada usuario.
* La base de datos y la gestión de usuarios deben estar bien estructuradas para evitar conflictos entre clientes.

**Empresas que la utilizan:**

* **Salesforce:** Plataforma CRM que permite a distintas empresas gestionar sus relaciones con clientes desde una misma infraestructura.
* **Google Workspace:** Ofrece servicios como Gmail y Drive a múltiples empresas dentro de un mismo entorno compartido.
* **Shopify:** Permite a diferentes negocios tener su propia tienda en línea sin necesidad de una infraestructura separada para cada uno.

**Arquitectura** **Event-Driven**

**Definición:**

Es un enfoque de diseño de sistema donde las interacciones y el flujo de datos están impulsados por **eventos**. Un evento es cualquier cambio de estado o acción que sucede en un sistema, como un usuario realizando una compra, un sensor registrando un cambio de temperatura, o una aplicación enviando una notificación

**Explicación con mis palabras:**

Como cuando un cliente realiza una compra, el productor puede genera un mensaje que puede activar de diferentes modos

**Beneficios:**

* Alta escalabilidad, ideal para poder manejar grandes volúmenes de datos y solicitudes simultaneas, como en aplicaciones en tiempo real o sistemas loT.
* Respuesta inmediata los eventos son procesados en el momento que ocurren, lo que permite una experiencia de usuario más fluida y reactiva.
* Desacoplamiento productores y consumidores están independientes entre sí, lo que facilita la evolución de los sistemas sin afectar a otros componentes.
* Flexibilidad es bastante fácil generar nuevos consumidores de eventos sin necesidad de modificar el producto o canales que están existentes.
* Tolerancia a los fallos si un consumidor llegara a fallar el sistema puede seguir generando eventos, y esos eventos pueden ser procesados posteriormente.

**Desafíos:**

* Complejidad de diseño: Requiere un mayor esfuerzo significativo para poder definir eventos, productores, consumidores y los intermediarios de comunicación
* Dificultad para depurar y monitorear: Dado que los eventos son asíncronos y pueden desencadenar múltiples procesos, que pueden llegar a rastrear errores o problemas que pueden llegar a ser complicados
* Latencia y rendimiento: Pueden ser un método bastante rápido, pero al usar mucho el uso intermediario de bróker puede provocar latencia en algunos casos
* Orden y consistencia: Garantizar que los eventos se procesen en el orden correcto para evitar duplicados que pueden ser un desafío
* Seguridad y fiabilidad: Proteger los eventos en tránsito y garantizar que lleguen a sus destinos que pueden requerir una infraestructura robusta

**Empresas que la utilizan:**

* Empresas de tecnologías o software como Netflix, Amazon y Google que tienen que manejar datos en tiempo real
* Empresas de E- commerce, como Shopify o mercado libre
* Sistemas que usan dispositivos conectados como dispositivos domésticos inteligentes
* Gaming: Plataformas de videojuegos en línea que tienen que sincronizar acciones en tiempo real como eventos en multijugador

**Arquitectura Micro Frontends**

**Definición:**

Es un enfoque de desarrollo aplicaciones web que aplica los principios de los microservicios al Frontend. En lugar de solo construir una única interfaz de usuario monolítica, este se divide en múltiples micros frontend independientes, cada uno encargado de una parte especifica de la aplicación.

**Explicación con mis palabras:**

Es prácticamente como una plataforma de comercio electrónico con diferentes secciones como catálogo de productos para comprar.

**Beneficios:**

* Equipos diferentes pueden trabajar en distintas partes de la aplicación sin tener que interferir con el otro.
* Permite el poder escalar tanto en el desarrollo (en múltiples equipos) como la infraestructura.
* Es posible manejar diferentes frameworks o herramientas para cada micro frontend, según las necesidades o habilidades del equipo.
* Al estar dividido en módulos pequeños es más fácil identificar y solucionar los problemas sin afectar a toda la aplicación.
* Los micros frontend pueden implementarse de forma independiente, reduciendo el riesgo de errores al actualizar la aplicación.
* Es mucho más sencillo añadir nuevas funcionalidades o cambiar partes especificas si tocar toda la aplicación.

**Desafíos:**

* Consistencia en la interfaz del usuario (UI): Garantiza una experiencia de usuario coherente entre los diferentes micro frontend puede ser complicados.
* Integración: Combinar múltiples micros frontend en una aplicación única la cual puede generar complejidad técnica, especialmente con el enrutamiento y la comunicación entre módulos.
* Sobrecargar las herramientas: Usar diferentes tecnologías puede complicar la configuración pruebas y monitoreos en el sistema.
* Rendimiento: Al tener que cargar múltiples frontend puede aumentar el tiempo de carga inicial de la aplicación.
* Coordinación entre equipos: Aunque sea independiente, sigue siendo necesario coordinar decisiones globales como el diseño o los estándares técnicos.
* Gestión del estado compartido: Compartir datos o estados entre micro frontend sin poder acoplarlos puede ser un desafío técnico importante.

**Empresas que la utilizan:**

* Spotify: Utiliza micro frontends para su plataforma modular, permitiendo un desarrollo ágil y la entrega rápida de nuevas funciones.
* Amazon: Modulariza su plataforma de comercio electrónico para mantener el código y lanzar características más rápido.
* Facebook: Implementa micro frontend para gestionar diferentes secciones de su interfaz, como el feed de noticias, notificaciones y anuncios.
* PayPal: Mejora la escalabilidad y mantenibilidad de sus aplicaciones web mediante esta arquitectura.
* IKEA: Usa micro frontends para su plataforma en línea, asegurando una experiencia de usuario fluida y actualizaciones rápidas.
* Starbucks: Adopta esta arquitectura para optimizar su sistema de pedidos en línea y manejar un código base en crecimiento.