Patrones de arquitectura de software 5 patrones: Arq. En capas,

Arq. Orientada a eventos, Arq. microkernel, Arq. Basada en espacio, Arq. En micro servicios

**Arq. En capa:** La arquitectura basada en capas se enfoca enla distribución de roles y responsabilidades deforma jerárquica garantizando una forma muyefectiva de separación de responsabilidades.El rol indica el modo y tipo de interacción conotras capas, y la responsabilidad indica lafuncionalidad que está siendo desarrollada.

**Arq. Orientada a eventos***: Event-driven architecture* o EDA, es un patrón de [arquitectura software](https://es.wikipedia.org/wiki/Arquitectura_software) que promueve la producción, detección, consumo de, y reacción a [eventos](https://es.wikipedia.org/wiki/Evento_estad%C3%ADstico).

Un evento puede ser definido como "un cambio significativo en un estado". Por ejemplo, cuando un consumidor compra un coche, el estado del coche pasa de "se vende" a "vendido". La arquitectura del sistema del vendedor de coches debe tratar este cambio de estado como un evento, cuyo suceso puede ser conocido en otras aplicaciones en la arquitectura. Desde una perspectiva formal, lo que es producido, publicado, propagado, detectado o consumido es un mensaje (típicamente asíncrono) llamado notificación del evento, y no el evento en sí mismo, el cuál es el cambio de estado que disparó la emisión del evento. Los eventos no viajan, solamente ocurren. Por otro lado, el término evento es frecuentemente usado para denotar el mensaje de notificación en sí mismo, lo cual puede llevar a algún tipo de confusión.

Este patrón arquitectónico puede ser aplicado por el diseño e implementación de aplicaciones y sistemas que transmitan eventos entre componentes software que estén emparejados libremente y servicios. Un sistema dirigido por eventos está compuesto típicamente de emisores de eventos (o agentes) y consumidores de eventos (o "sink" en inglés). Los consumidores tienen la responsabilidad de llevar a cabo una reacción tan pronto como el evento esté presente. La reacción puede o no puede ser completamente proporcionada por el consumidor en sí mismo. Por ejemplo, el consumidor debe tener solamente la responsabilidad de filtrar, transformar y reenviar el evento a otro componente o debe proporcionar una reacción propia a algún evento.

Construir aplicaciones y sistemas alrededor de una arquitectura dirigida por eventos permite a estas aplicaciones y sistemas ser construidos de una manera que facilita un mayor grado de reacción, debido a que los sistemas dirigidos por eventos están, por el diseño, más normalizados para entornos no predecibles y asíncronos.

**Arq Microkernel**: El patrón Microkernel se aplica a sistemas de software que deben estar habilitados para adaptarse a requerimientos cambiantes del sistema. Separa un núcleo de funcionalidad mínima de la funcionalidad extendida y de partes específicas al cliente. También sirve como un socket para conectores en estas extensiones y coordinar su colaboración.

Las funciones centrales de un SO son controladas por el núcleo (kernel) mientras que la interfaz del usuario es controlada por el entorno (shell). Por ejemplo, la parte más importante del DOS es un programa con el nombre "COMMAND.COM" Este programa tiene dos partes. El kernel, que se mantiene en memoria en todo momento, contiene el código máquina de bajo nivel para manejar la administración de hardware para otros programas que necesitan estos servicios, y para la segunda parte del COMMAND.COM el shell, el cual es el interprete de comandos.

**Arq. Basada en espacio:** es una rama de la [ingeniería de software](https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) que enfatiza la [separación de asuntos](https://es.wikipedia.org/w/index.php?title=Separaci%C3%B3n_de_asuntos&action=edit&redlink=1), *separation of concerns* (SoC), por lo que se refiere a la funcionalidad de amplio rango disponible a través de un sistema de software dado. Es un acercamiento basado en la reutilización para definir, implementar, y componer [componente de software](https://es.wikipedia.org/wiki/Componente_de_software) [débilmente acoplados](https://es.wikipedia.org/w/index.php?title=D%C3%A9bilmente_acoplado&action=edit&redlink=1) en sistemas. Esta práctica persigue un amplio grado de beneficios tanto en el corto como el largo plazo, para el software en sí mismo y para las organizaciones que patrocinan tal software.

Los [ingenieros de software](https://es.wikipedia.org/wiki/Ingeniero_de_software) consideran los componentes como parte de la plataforma inicial para la [orientación a servicios](https://es.wikipedia.org/w/index.php?title=Orientaci%C3%B3n_a_servicios&action=edit&redlink=1). Los componentes juegan este rol, por ejemplo, en [servicios de web](https://es.wikipedia.org/wiki/Servicio_web) y, más recientemente, en las [arquitecturas orientadas a servicios](https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios) (SOA), por el que un componente es convertido por el servicio web en un servicio y consiguientemente hereda otras características más allá de las de un componente ordinario.

**Arq. En micro servicios:** Tradicionalmente los servicios de TI se han desarrollado utilizando una arquitectura monolítica, respondiendo a cada necesidad del cliente con la construcción de una solución, conocida como entregable. Las desventajas de este tipo de arquitectura son bastante conocidas, siendo una de ellas la dificultad de reaccionar de forma rápida y flexible a los cambios.

Los Microservicios a menudo combinado con tecnologías de contenedores tales como Docker, se están utilizando para agilizar el desarrollo de servicios y responder de forma ágil a las necesidades del negocio, combinado con modelos de trabajo como DevOps, que rompe las barreras de comunicación y colaboración entre los equipos de desarrollo y operación, equilibrando y agilizando los procesos de despliegue y gestión.

En la conferencia se abordará a detalle la arquitectura basada en microservicios, sus ventajas de escalabilidad y flexibilidad tecnológica, y sus complejidades que no todas las organizaciones están en posición de enfrentar, tales como la entrega continua, infraestructura automatizada, el aseguramiento de la calidad, entre otros. Se discutirán los requerimientos previos y necesarios para la migración o implementación de una arquitectura basada en microservicios, bajo ejemplos y escenarios reales sobre cómo alcanzarlos.