**Patrones Describir 2 Cules son las fortalezas, contexto, donde usarlo**

# Arquitectura en capas

## Descripción del patrón

Componentes dentro del patrón de arquitectura en capas están organizados Capas horizontales, cada capa desempeñando un papel específico en La aplicación.

No especifica la cantidad de capas, la mas común consiste en 4 consisten en cuatro capas estándar: presentación, negocios, persis-tence, y base de datos.

Cada capa papel especifico y responsabilidad , cada capa es una abstracción de lo que debe realizar, no debe preocuparse de las otras capas sino de su función especicfica

## Fortalezas

Una de las potentes características del patrón de arquitectura en capas es la *separación de preocupación* entre los componentes. Componentes dentro Una capa específica sólo se ocupa de la lógica que pertenece a esa capa. Fácil de mantener debido a las interfaces definidas y alcance limitado de los componenetes.

## Debilidades

Sumidero antipatron capasa sin hacer nada, se pueden ver estos casos pero se debe tener en cuenta el porcentaje de solicitudes que tienen este comportamiento. 80-20 es bueno. Si esto no se cumple y se invierte se debe tener en cuenta hacer capas abiertas.

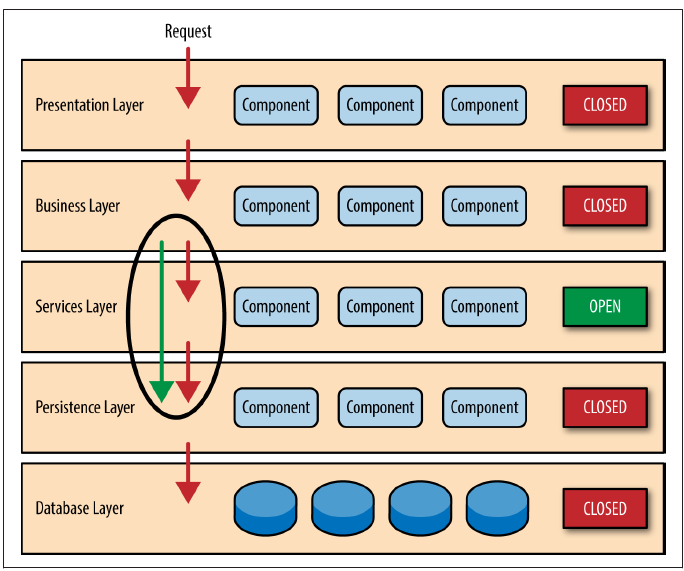
## Clave

Cada una de las capas es marcada como cerrada, Una capa cerrada significa que como una solicitud Se mueve de capa a capa, debe pasar por la capa justo debajo de ella Para llegar a la siguiente capa por debajo de esa.

No se pregunta directamente por un concepto clave conocido como *capas de aislamiento.*

El concepto de capas de aislamiento significa que los cambios Capa de la arquitectura generalmente no afectan o afectan a los componentes En otras capas. Cada capa es indepenediente.

Si se quiere incluir otra capa como una de servicios para que se acceda desde la presentación, no tiene sentido que las otras capas también pasen por haym entonces se hace una capa **abierta.**



Aprovechar el concepto de capas abiertas o cerradas ayuda a definir la relación entre capas en la arquitectura

## Ejemplo

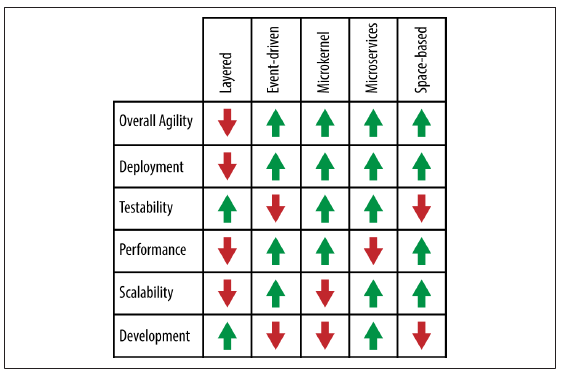
Recuperar la información de un individuo

La pantalla acepta la solicitud, cuando la recibe la lleva al delegado del cliente que es el que sabe que modulo de la capa de negocio pueden procesar esta información y que parámetros recibe (Contrato), el objeto cliente del negocio agrega toda la información necesaria de la solicitud llama al Dao cliente en la capa de persistencia quiene ejecutan las sentenciaas para recuperar los datos y devolverlos al objeto cliente del negocio este recibe y agrega los datos y envía al delegado cliente que luego los pasa a la pantalla para mostrar la información.

En java : Pantalla en JSF,delegado cliente como el bean admin, negocio EJB, objetos de base de datos POJOS

En .net : ASP de .net c# como el negocio y el acceso en ADO.

## Análisis



Despliegue debe hacerse en horas libres o fines de semana.

Escalabilidad: puede dividir las capas en implementaciones físicas pero es costoso escalar. O replicar nodos.

# Arquitectura impulsada por eventos

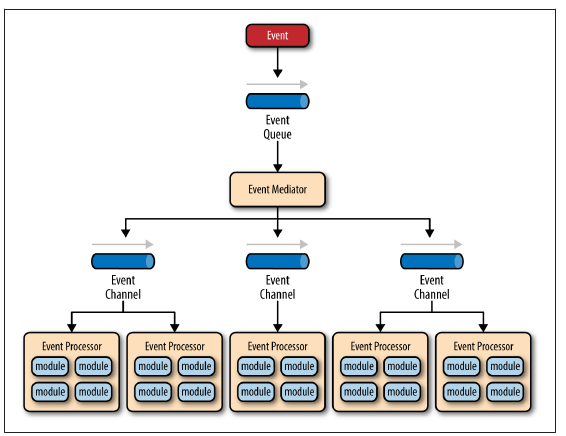
Es un patrón de arquitectura asíncrono para producir aplicaciones altamentete escalabales- también es altamente adapatable y se puede usar para pequeñas aplicaciones asi como para grandes y complejas. Tiene componenetes altamentese desacopadlos y componentes de procesamiento de eventos de porposito único que reciben y procesan eventos de forma asincrónica.

Consiste en dos topologías un mediador y un corredor.La tpologia de mediador es comumnete usada para orquestar multiples pasos dentro de un evento a través de un mediador central mientras que el corredor es usado cuando se quiere encadenar eventos juntos si un mediador central. Por que difieren en estas dos ¿

## Topología mediador

Es util para eventos con multiples pasos que requieren un nivel de dirección para procesar el evento.

Hay 4 componentes principals dentro de la topologia de mediardor : Colas de eventos, un mediador de eventos, canales de eventos y procesadores de sucesos.



El patrón no especifica la implementación del componente de cola de eventos, puede ser cola de mensajes un endpoint de un servicio web., combinación

Dentro de este patron hay 2 tipos de eventos.

Evento inicial : Recibido por el mediador

Envento de procesamiento : generado por el mediador y recibido por los componentes de eventos de proceso.

El componente mediador es responsable de orquestar los pasos contenidos en el evento inicial. No realiza la lógica para procesar el evento sino que conoce los pasos necesarios para hacerlo.

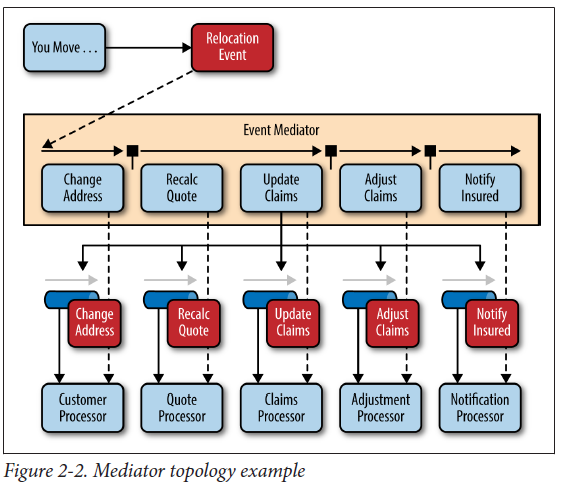
Los canales de eventos son usados por el mediador para para pasar de forma asíncrona eventos con cada paso del evento inicial por el evento de proceso.

La lógica de negocio se encuentra en el componente de evento de procesamiento

Los eventos de procesamiento son contenidos en ellos mismos, independiente, altamente desacoplado que realizan una tarea específica en el sistema. Cada componente de evento de proceso debería realizar una única tarea de negocio.

Un mediador se puede implementar con un Hub

### Ejemplo



## Topología Intermediario

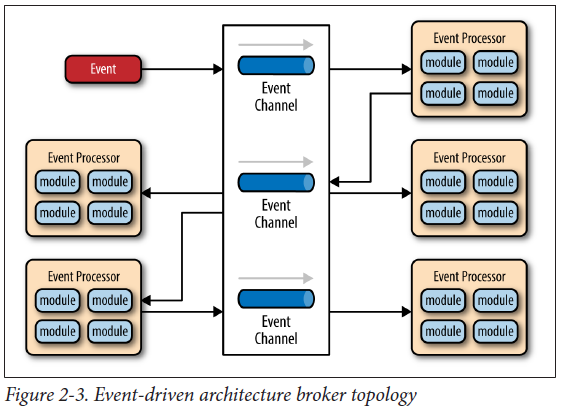
En esta topología no hay un mediador central , el flujo de mensajes se distribuye a través de los componentes de proceso de eventos similar a una cadena. Es útil cuando se tiene un flujo de procesamiento de eventos simples que no necesitan orquestación.

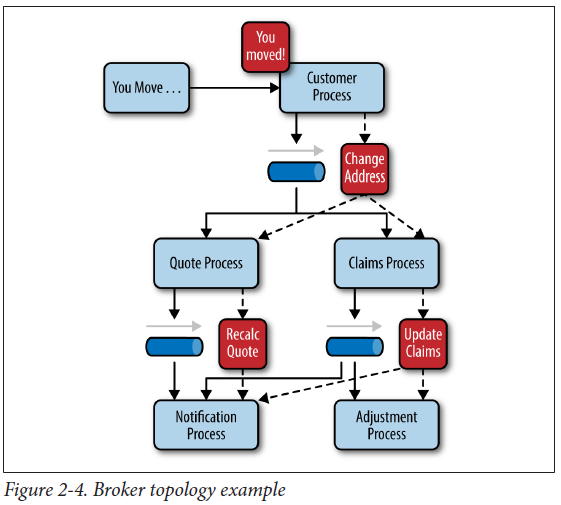
Hay dos tipos de componentes

Broker

Procesador de eventos

No hay un mediador central, cada componente de procesador de eventos es responsable de procesar un evento y publicar un nuevo evento indicando que la acción fue desarrollada.





## Consideraciones

Es realativamente complejo de implementar por su carácter asincrónico distribuido, Al implementar este patrón, debe abordar varios problemas de arquitectura distribuida, como la disponibilidad remota de procesos, la falta de capacidad de respuesta y la lógica de reconexión de intermediarios en caso de un fallo de intermediario o mediador.

Una consideración a tener en cuenta al elegir este patrón de arquitectura es la falta de transacciones atómicas para un solo proceso de negocio. Porque el componente de procesador de eventos son altamente desacoplados y distribuidos, es muy difícil mantener una unidad de transacción a través de ellos.

# Arquitectura MicroKernel

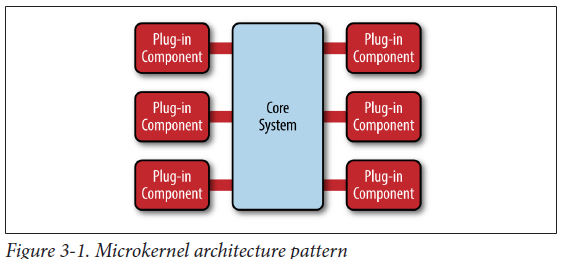
Es un patrón natural para la implementación de aplicaciones basadas en productos. Este patrón permite añadir características adicionales a la aplicación como pulg-ins, proporsionando extensibilidad asi comoaislamiento.

## Descripción

Consta de dos tipos de componentes de arquitectura:

Core System y Modulos Plug-in.

El core system tradicionalmente contiene solamente la minima funcionalidad requerida para hacer el Sistema operacional. Muchos sistemas operativos implementean microkernel, desde una perspectiva de aplicación de negocio el core system es frecuentemente copmo la lógica genereal de negocio, casos especiales procesamiento condicional complejo.



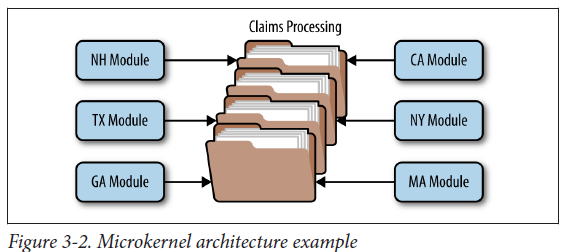
Los modulos pulg-in son stand- alone, componenetes independientes que contiene procesamiento especializado o extienden el core system para adicional capacidades de negocio. Se debe manatener minima la comunicación entre plug ins.

El core system debe conocer cual modulo esta disponible y como obtenerlos, una forma comuin de implementar es un registro de plugin. Este registro contiene nombre, contrat, comunicación..

## Ejemplo

IDE Eclipse, se descarga el básico yb se puede ir agregando plug-in o navegadores de internet

La pila de folders representa el core system este contiene la lógica de negocio básica requerida, cada plug-in contiene una regla especifica para este estado. La clave es que la reglas y procesamiento es separado del core y puede ser agregado,removido o cambiado con poco o ningún efecto sobre el resto del core system o de otros plug in.



## Consideraciones

Algo grande que pensar de este patrón es que puede ser embebido o usado como parte de otro patrón de arquitectura, por ejemplo si resuelve un problema particular puede embebirlo en microservicio asoi como el event-driven.

Para aplicaciones basadas en productos el microkernel debería ser la primera opción de arquitectura particularmente para productos que tiene reléase para adicionar características todo el tiempo y quieren control sobre las características que el usuario tiene.

# Arquitectura Microservicios

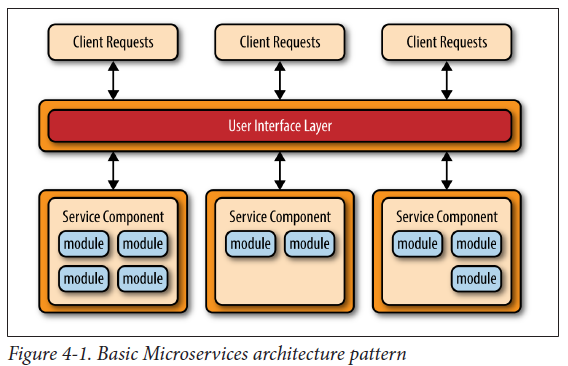
Esta ganando terreno rápidamente como alternativa viable de arquitecturas orientadas a servicios y aplicacione smonoliticas.

## Decripción

Sin importar la tipología hay cosas comunes, el primero concepto es la nocion de unidades desplegadas por separado, cada componenete del microservicio es desplegada como una unidad separada, looq ue permite un despliegue mas fácil.

El concepto mas importante es la nocion de un componenete de servicio. Es mejor pensar sobre componenetes de servicios que sobre un servicio. el cual puede variar la granulariadad

Un componente de servicio contiene uno o mas modulos que representanun aunica función o una porción independiente de una amplia aplicación de negocio. Uno de los grandes desafíos es diseñar sobre un buen nivel de granularidad.



Otro concepto clave dentro de esta arquitectura es que es distribuido, significa que todos los componenetes dentro de la arquitectura están completamente desacoplados.

Normalmente tiene dos principales fuentes : aplicaciones monolíticas usando capas y aplicaciones distribuidas a través de arquitectura orientada a servicios.

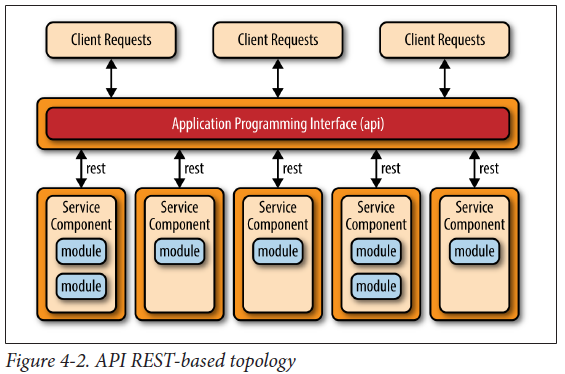
Los componentes de servicios pueden ser desarrollados, probados y desplegados de forma individual.

## Topologías

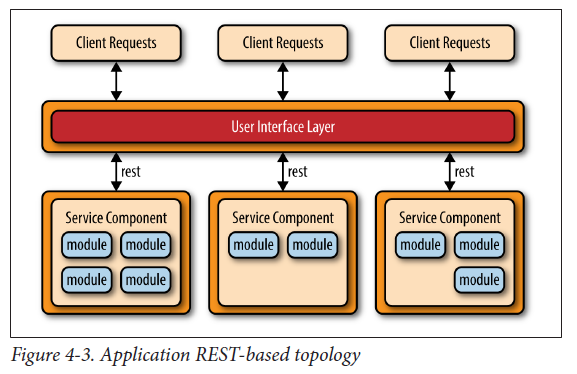
3 principales y mas comunes

API REST-based, Aplicación REST- based y mensajes centralizados.

API Rest-based : Util para sitios web que exponent servicios individuales autonomos pequeños, atraves de algún tipo de API. : Se compone de servicios pequeños que contiene uno o dos modulos con funcones especificas de negocio, típicamente se accede mediante una interfaz basada en REST implementado a través de una API.

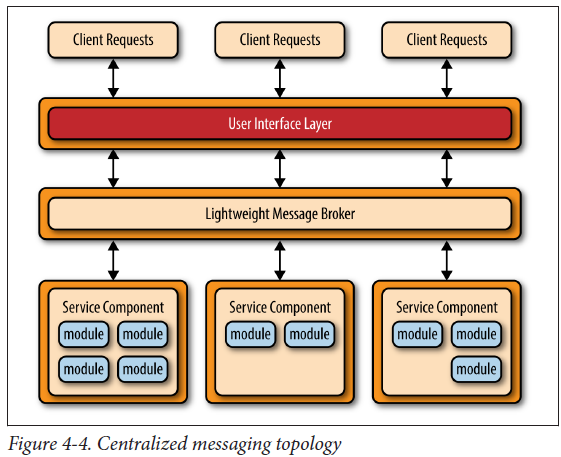


REST aplicación : Difiere de la de API en el enfoque que es basado en que las solicitudes se reciben a través de las pantallas de las aplicaciones empresariales web



Esta tipología es común para pequeños o medianas apliacciòn que tienen bajo grado de complejidad.

Mensajes centralizado: no se debe confundir con SOA



## Evitar dependencias y orquestación

Desafío determinar el nivel correcto de granularidad.

## Consideraciones

Ventaja: Se pueden hacer despliegues en tiempo real

# Arquitectura basada en el espacio

La mayoría de aplicaciones basadas en web tienen el mismo flujo de solicitud: una solicitud desde un navegador va al web server, luego una aplicación de servidor, luego finalmente aun servidor de base de datos. Este patrón trabaja bien para un conjunto pequeño de usuarios, el cuello de botella empieza a aparecer como la carga de usuarios incrementa, primero en la capa del web server, luego en la capa del servidor de aplicaciones y finalmente en la capa de base de datos. La respuesta usual es escalar el servidor web….

ESTA arquitectura esta específicamente diseñada para direccionar y resolver los problemas de escalabilidad y concurrencia

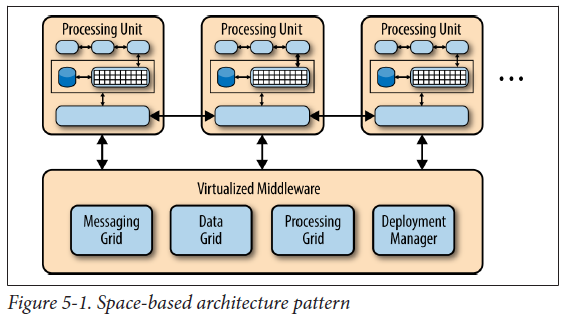
Este patrón también llamado patrón de arquitectura de la nube, la alta escalabilidad es alcanzada removiendo las restricciones en la base de datos central y usando grillas de datos replicada en su lugar. Los datos de la aplicación son mantenidos en memoria y replicados entre todas las unidades de procesamiento activas. Las unidades de proceso pueden ser dinámicamente iniciadas y apagadas como la carga de usuarios crece y decrece.

Con lo que se aborda la escalabilidad variable. Debido a que no existe una base de datos central, el cuello de botella de la base de datos se elimina, proporcionando escalabilidad casi infinita dentro de la aplicación.

La mayoría de aplicaciones que implementa este patrón son web sites que reciben una solicitud desde un navegador y realizan una acción corta, eje: un sitio de subastas.

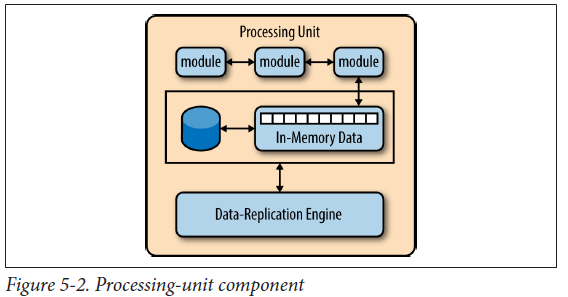
Hay 2 principales componentes

Unidad de procesamiento y Middleware virtualizado



## Unidad de procesamiento

La unidad de procesamiento contiene los componentes de la aplicación incluye componentes web asi como el backend de la lógica de negocio, contiene los modulos de la aplicación en una grilla de memoria y un almacenamiento asíncrono opcional persistente, también contiene un middlware virtualizado para replicar los cambios de datos hecho por una unidad de procesamiento a otra unidad de procesamiento activa.



## Middleware Virtualizado

**El middleware virtualizado** maneja gestión interna y comunicaciones, contiene componentes que controlan varios aspectos de datos sincronización y manejo de solicitudes. Este incluye los componentes grid mensajeria, grid datos, grid procesamiento y administrador de despliegue.

### Grilla de mensajería

Determina cual componente de procesamiento esta disponible para recibir la solicitud y la remite

### Grilla de datos

Es el mas importante puesto que interactua con la maquina de replicación de datos para administrar y actualizar la información entre unidades de proceso para que tengan lainformación disponible.

### Grilla de procesamiento

Es un componente opcional que adminitra la distribución de solicitudes cuando hay multiples unidades de proceso

### Administrador de implementación

Administra el encendido y apagado dinamico de las unidades de proceso basado en las condiciones de carga. Y enciende nuevas unidades de proceso cuando la carga incrementa, este es un componenete critico para alcanzar la escalabilidad necesaria.

## Consideraciones

Es un complejo y costoso patrón para implementar, es una buena decisión para aplicaciones web pequeñas con carga variable. Sin embargo esto no es adecuado para tradicionales Aplicaciones de bases de datos relacionales a gran escala con grandes cantidades de datos operativos.

Aunque no requiere almacenamiento centralizado, se incluye comúnmente uno para realizar la cargagrilla de datos de memoria