¿Cómo definir un estilo arquitectónico para una aplicación?

Linda María Fernández Olvera
UPIICSA-IPN
Ingeniería de Diseño
lfernandezo1500@alumno.ipn.mx

Abstract – Este ensayo explica los diferentes estilos arquitectónicos, identificando sus características y las situaciones en las que es más conveniente usarlos.

Índice de Términos – estilos arquitectónicos, Cliente servidor, EDA, SOA, REST, Microservicios, Microkernel, P2P, monolítico.

I. INTRODUCCIÓN

Un estilo arquitectónico es una transformación que se impone al diseño de todo el sistema. El objetivo es establecer una estructura para todos los componentes del sistema.

El software construido para sistemas basados en computadora también tiene uno de muchos estilos arquitectónicos. Cada estilo describe una categoría de sistemas que incluye un conjunto de componentes (como una base de datos o módulos de cómputo) que realizan una función requerida por el sistema, un conjunto de conectores que permiten la "comunicación, coordinación y cooperación" entre los componentes, restricciones que definen cómo se integran los componentes para formar el sistema y modelos semánticos que permiten que un diseñador entienda las propiedades generales del sistema al analizar las propiedades conocidas de sus partes constituyentes.

La arquitectura del software de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos.

La arquitectura no es el software operativo. Es una representación que permite analizar la efectividad del diseño para cumplir los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software.

II. DESARROLLO

Un estilo arquitectónico establece un marco de referencia a partir del cual es posible construir aplicaciones que comparten un conjunto de atributos y características mediante el cual es posible identificarlos clasificarlos.

Comprender los estilos arquitectónicos nos colocan en una mejor posición como arquitectos, pues podemos identificar y diseñar soluciones que sigan determinados estilos arquitectónicos, además, podremos identificar que estilos se aplican mejor para resolver ciertos problemas, podremos identificar sus puntos fuertes y sus puntos débiles. Los estilos arquitectónicos más importante son:

- Monolítico
- Cliente-servidor
- Peer-to-peer (P2P)
- Arquitectura en capas
- Microkernel
- Service-Oriented Architecture (SOA)
- Microservicios
- Event Driven Architecture (EDA)
- Representational State Transfer (REST)

A. Estilo Monolítico

El estilo arquitectónico monolítico consiste en crear aplicaciones autosuficientes que contengan toda la funcionalidad necesaria para realizar la tarea para la cual fue diseñada, sin necesitar dependencias externas que complementen la funcionalidad, por lo tanto, sus componentes funcionan juntos, compartiendo recursos y memoria.

Todas las aplicaciones al inicio de la computación nacían con este estilo arquitectónico, sin embargo, con el tiempo y la aparición de internet existió la posibilidad de consumir servicios externos, llegaron las arquitecturas modulares que separaban el código, aún así las aplicaciones monolíticas siguen presentes casi donde son totalmente indispensables para mantener la operación de las empresas, donde el sistema necesita una autonomía total o no pueda conectarse a internet.

Los sistemas monolíticos pueden tener una serie de paquetes y un código claro donde cada paquete puede tener cierta parte de la funcionalidad y están desacoplados uno de otro. Sin embargo, al momento de compilarse el código todo se empaqueta en un solo software, toda librería que sea requerida será exportada como parte del archivo compilado de

salida.

En la figura 1 podemos ver el proceso de compilación de una aplicación monolítica en la cual todos los paquetes junto con las dependencias son compilados y dan como resultado un solo artefacto.

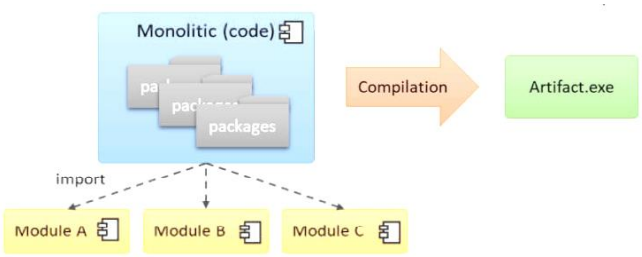

Figura 1. Compilación de aplicación Monolítica

B. Estilo Cliente - Servidor

Es uno de los estilos arquitectónicos distribuidos más conocidos, se forma por dos componentes llamado el proveedor y el consumidor, que son un servidor que brinda una serie de servicios o recursos los cuales son consumidos por el cliente o múltiples clientes.

El cliente solo es una capa para representar los datos y detonar acciones que modifican el estado del servidor.

En esta arquitectura el cliente y el servidor son desarrollados como dos aplicaciones diferentes, de tal forma que cada una se desarrolla de forma independiente pero siempre respetando el mismo protocolo de comunicación. Como se muestra en la Figura 2.

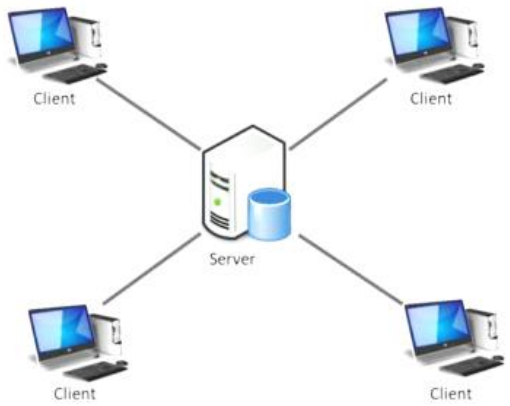

Figura 2. Arquitectura Cliente - Servidor

La idea de separar el cliente del servidor radica en centralizar la información y separar las responsabilidades, por lo tanto, el servidor es el único que tiene acceso a los datos.

Es ideal para aplicaciones en tiempo real y aplicaciones centralizadas de alto rendimiento.

C. Estilo Peer – to – peer

El estilo Red entre iguales (P2P) es una red de computadoras donde todos los dispositivos conectados a la red actúan como cliente y servidor al mismo tiempo. No es necesario un servidor central que administre la red, si no que todos los nodos de la red pueden comunicarse entre sí.

Esta arquitectura permite conectarse a otras computadoras de la red para consumir sus recursos expuestos por los nodos de la red, pero al mismo tiempo funciona como un servidor, lo que permite que otros nodos se conecten a nuestro software para leer los recursos propios. Entre más computadoras se conecten a la red el poder de procesamiento se agrega, lo que lo vuelve una red con capacidad de escalamiento.

Existen dos clasificaciones dependiendo la centralización. Los puros estructurados (Figura 3). (busca los recursos basados en tablas Hash distribuidas) o no estructurados (no existe un servidor central ni redes).

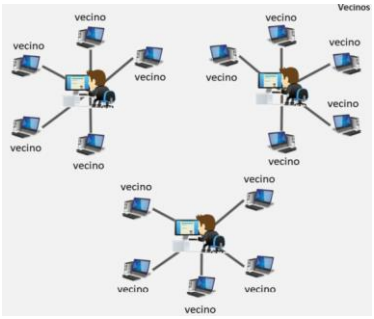

Figura 3. Sistema P2P no estructurado

Y las arquitecturas P2P hibridas, también conocida como centralizadas por tener un servidor central que sirve de enlace entre los nodos de la red (Figura 4).

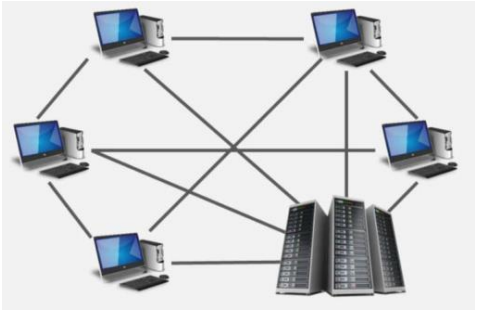

Figura 4. Arquitectura P2P centralizada

Ideal para compartir archivos, streaming, procesamiento distribuido, criptomonedas y aplicaciones descentralizadas.

D. Estilo de Arquitectura en Capas

Es una de las más utilizadas, no solo por la simplicidad sino porque es utilizada por defecto cuando no sabemos que arquitectura emplear.

Consta en dividir la aplicación con la intención de que cada capa tenga un rol muy definido, sin embargo no se definen cuantas capas debe tener la aplicación sino más bien se centra en la separación de la aplicación en capas.

Todas las capas están de manera horizontal, por lo tanto cada capa solo puede comunicarse con la capa que está inmediatamente por debajo, como se muestra en la Figura 5. Si desea comunicarse con capas más separadas tendrá que hacerlo por las intermedias.

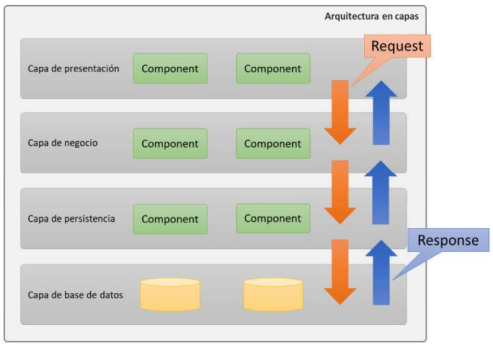

Figura 5. Comunicación entre capas

Es ideal para sistemas versátiles, el entorno empresarial, aplicaciones web y como arquitectura de uso general.

E. Estilo Microkernel

También conocido como arquitectura plug-in, permite crear aplicaciones extensibles mediante la cual es posible agregar nueva funcionalidad mediante la adición de pequeños plugins que extienden la funcionalidad del sistema. Las aplicaciones se dividen en dos tipos de componentes, en sistema Core (central) y los plugins o módulos del sistema que son componentes periféricos con acciones añadidas al componente core, como se muestra en la Figura 6. Por lo tanto, solo puede haber un componente central y muchos plugins.

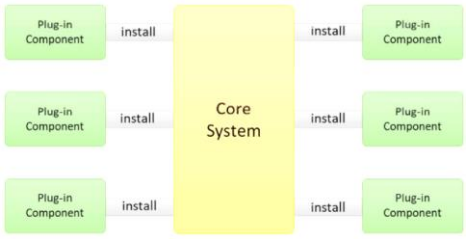

Figura 6. Estructura de un Microkernel

No son sistemas fáciles de desarrollar pues se necesitan crear aplicaciones capaces de agrandar dinámicamente su funcionalidad a medida que nuevos plugins son instalados.

Es ideal para plataformas para desarrollar aplicaciones o como productos que están empaquetados.

F. Estilo Service-Oriented Architecture (SOA)

Esta arquitectura busca resolver los problemas de comunicación entre las diferentes tecnologías, incentivando a las empresas a crear servicios interoperables que se aprovechan en toda la organización. Estos servicios deben ser construidos utilizando estándares abiertos para que puedan ser consumidos por cualquier aplicación sin importar la tecnología que se usa.

En su estructura todas las aplicaciones están diseñadas para ser integradas con otras aplicaciones, por lo que deben exponer como servicios todas las operaciones o funciones que tienen, como en la Figura 7, donde los sistemas consumen los servicios disponibles de la red.

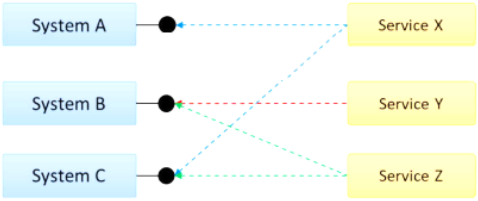

Figura 7. Consumo de servicios

El estilo permite la encapsulación, pues permite ocultar la complejidad del sistema detrás de una serie de servicios de alto nivel. También promueve la construcción de servicios que hagan una sola tarea lo que los hace más reutilizables.

Este estilo puede ser usado con otras arquitecturas, es muy flexible en su implementación, pero es ideal para aplicaciones que procesan o enviar masivas cantidades de datos y cuando el performance de la aplicación es importante.

G. Estilo de Microservicios

Es uno de los estilos más populares, consiste en crear pequeños componentes de software que solo hacen una tarea, son autosuficientes y permiten evolucionar de forma independiente del resto de los componentes.

Un microservicio es un pequeño programa que se especializa en una tarea, por lo tanto, los microservicios son altamente cohesivos, pues toda la operación está relacionada con resolver un único problema.

Es contraria a la arquitectura monolítica, como se muestra en la Figura 8, se muestra un API Gateway que es un componente que se posiciona de cara a los microservicios para servir como puerta de entrada, control de acceso y la funcionalidad que se expone a la red pública.

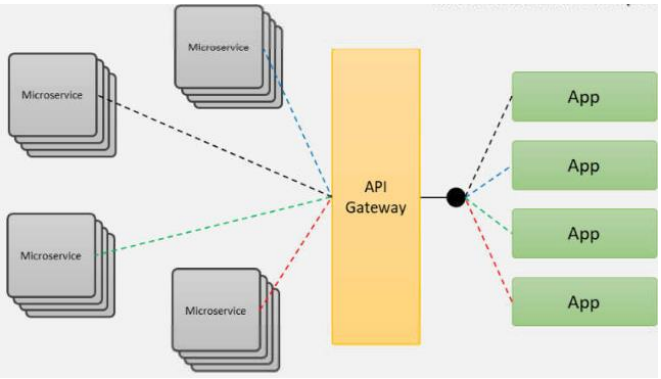

Figura 8. Arquitectura de microservicios

Es ideal para aplicaciones pensadas para tener un gran escalamiento, aplicaciones grandes que se complica ser desarrolladas por un solo equipo de trabajo y proyectos en los que se busca agilidad en el desarrollo.

H. Estilo Event Driven Architecture (EDA)

Es una arquitectura asíncrona y distribuida, pensada para crear aplicaciones altamente escalables, los componentes establecen una comunicación de forma síncrona, se obtiene una respuesta y se procede con el siguiente paso dado por un evento como se muestra la Figura 9.

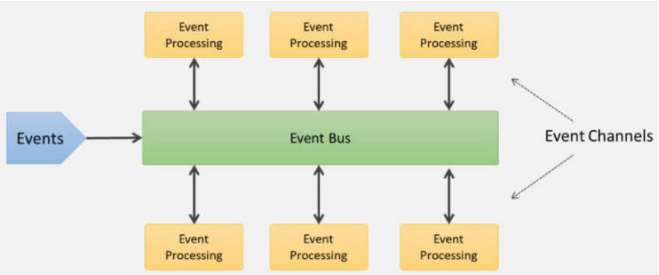

Figura 9. Arquitectura EDA

Un evento lo podemos definir como un cambio significativo en el estado de la aplicación, ya sea por un cambio de datos o alguna acción.

Existen dos topologías diferentes para implementar la arquitectura.
La topología con mediador (Figura 10) se utiliza para procesar eventos complejos, donde el éxito de la operación requiere de una serie de pasos definidos y un orden de ejecución exacto (Orquestación: coordinar pasos y secuencias de ejecución).

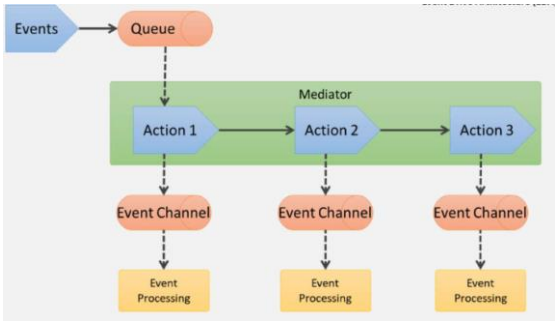

Figura. Orquestación con mediador

La topología con Broker (Figura 11) sirve para procesar eventos simples, en los cuales no es necesario orquestar una serie de pasos para llegar al objetivo.

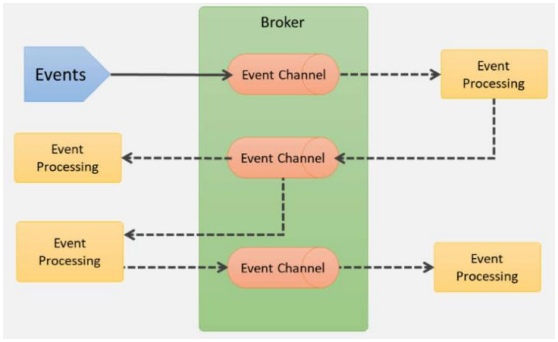

Figura 11. Orquestación con Broker

Esta arquitectura es útil en aplicaciones altamente escalables, en paralelismo, concurrencia y asincronismo y en aplicaciones en tiempo real.

I. Estilo Representational State Transfer (REST)

REST es un conjunto de restricciones que crean un estilo arquitectónico y que se utiliza en aplicaciones distribuidas. Ignora los detalles de implementación de componentes y se enfoca en el rol y las restricciones de su aplicación.

En Rest los datos deber ser transmitidos a donde serán procesados, propone servir los datos en crudo para que sea el consumidor quien se encargue de procesar los datos y mostrarlos. El recurso hace referencia a la información con una dirección única.

Los conectores son componentes que encapsulan las actividades de recurso y transferencia, REST describe los conectores como interfaces abstractas que permiten la comunicación entre componentes.

Los componentes son software que utiliza los conectores para consultar los recursos o servirlos.

Actualmente REST es la base sobre la que están construidas casi todas las API de las principales empresas del mundo. Por lo tanto, puede ser utilizada para crear integraciones con

diferentes dispositivos, incluso con empresas o proveedores externos, interactuar con otra plataformas o aplicaciones sin importar la tecnología y las representaciones de los datos.

III. CONCLUSIÓN

Los estilos de arquitectura deben definirse al momento de analizar los requerimientos y funciones de una aplicación, pues nos permitirán modelar su funcionamiento, el consumo de los datos y la organización de los componentes.

Las aplicaciones monolíticas destacan en su independencia y el performance, el cliente – servidor está enfocada en aplicaciones de alto rendimiento. La arquitectura P2P crece a medida que se agregan nodos a la red. El estilo por capas es uno de los más sencillos de implementar y no agrega mucha carga de mantenimiento. El estilo Microkernel nos permite tener múltiples equipos de desarrollo de las aplicaciones. SOA es muy flexible y permite la interoperabilidad, EDA es el estilo más escalable pues atiende millones de solicitudes y al final REST se centra en la transmisión de datos y sus representaciones.

REFERENCES

[1] Oscar, Blancarte, "*Introducción a la Arquitectura de Software, un enfoque práctico*" 1a ed. Oscar Blancarte Blog. 2020