**DOCUMENTO PATRÓN DE ARQUITECTURA BRÓKER**

**DEFINICIÓN BRÓKER**

-Es un patrón de arquitectura que se utiliza para estructurar sistemas de software distribuido con componentes desacoplados que interactúan por vocaciones de servicios remotos.

-El componente bróker es el responsable de coordinar la información; tanto de enviar como de reenviar las peticiones, así como de transmitir los resultados y excepciones.

**EJEMPLO**

1. Supongamos que estamos desarrollando un sistema de información para la ciudad diseñado para correr en un área amplia de trabajo.

2. Algunas computadoras en la red contienen uno o más servicios de información acerca de eventos, restaurantes, hoteles, monumentos históricos y transportación pública.

3. Se tienen computadoras terminales conectadas a la red.

4. Los turistas a través de la ciudad pueden tener información en la cual están interesados desde las terminales usando un explorador web.

5. El software soporta la obtención información en línea de los servidores apropiados y la muestra en la pantalla.

6. la información se obtendrá a través de la red y no se mantiene en las terminales.

7. Esperamos que el sistema cambie o crezca continuamente así que los servicios individuales deben estar acoplados uno del otro.

8. Además el software de la terminal debe ser capaz de usar los servicios si tener que conocer su localización. Esto nos permitirá mover, replicar o migrar servicios.

**CONTEXTO**

-Su medio ambiente es un sistema distribuido posiblemente heterogéneo con diferentes componentes cooperativos independientes.

-Muchos sistemas se construyen a partir de un conjunto de servicios distribuidos a través de varios servidores.

-La implementación es complejo porque:

1. Cómo los sistemas van a funcionar

2. La forma en que se conectan entre si

3. Cómo van a intercambiar información

4. La disponibilidad de los servicios de componentes

**PROBLEMAS QUE ATACA EL PATRÓN**

1.Construir un sistema de software con un conjunto de componentes desacoplados e interoperativos, en lugar de una aplicación monolítica.

2.Esto resulta en mayor flexibilidad, mantenimiento y posibilidad de cambio.

3.Al hacer la repartición de funcionalidad en componentes independientes el sistema llegara a ser potencialmente distribuible y escalable.

4.Sin embargo, cuando componentes distribuidos se comunican unos con otros, se necesitan algunas formas especiales de comunicación entre procesos.

5.Si los componentes manejan la comunicación ellos mismos el sistema resultante enfrenta dependencias y limitaciones.

6.El sistema será dependiente del mecanismo de comunicación usado, y los clientes deben conocer la localización de los servidores y en muchos casos la solución es limitada a solo un lenguaje de programación.

7.Los servicios para remover, agregar, intercambiar y localizar componentes también son requeridos. Las aplicaciones que usan estos servicios no deben depender de detalles específicos del sistema para garantizar la portabilidad, incluso en una red heterogénea.

8.Desde el punto de vista del desarrollador no debe de haber problema entre el desarrollador de software para sistemas centralizados y desarrollar para sistemas distribuidos. Una aplicación que usa un objeto debe solo ver la interfaz ofrecida, por lo que no debe necesitar saber nada acerca de la implementación de los detalles del objeto o acerca de su localización física.

**SOLUCIÓN**

1.Introduzca un componente bróker para llevar a cabo un mejor desacoplamiento entres los clientes y servidores.

2.Los servidores se registran ellos mismos con el bróker y hacen sus servidores disponibles para los clientes a través de interfaces-métodos.

3.Los clientes usan la funcionalidad de los servidores enviado peticiones vía bróker.

4.La tarea del bróker incluye localizar el servidor apropiado, transmitir las peticiones al servidor y transmitir los resultados y excepciones de regreso al cliente.

5.Usando el patrón bróker una aplicación puede acceder a servicios distribuidos, simplemente enviando llamadas de mensaje al objeto apropiado, en lugar de enfocarse a la comunicación de procesos de bajo nivel.

6.Además la arquitectura del bróker es flexible, es decir, permite el cambio dinámico: adicción, borrado y relocalización de objetos.

7.El patrón bróker reduca la complejidad envuelta en el desarrollo de aplicaciones destruidas, porque hace la distribución transparente para el desarrollador.

8.Esto lo realiza introduciendo un modelo de objetos en el cual los servicios distribuidos son encapsulados por los objetos. Así los sistemas bróker ofrecen un camino para la integración de las tecnologías básicas: distribución y tecnología de objetos.

9.También extiende el modelo de objetos de aplicaciones simples a aplicaciones distribuidas, consistentes de componentes desacoplados que pueden correr en máquinas heterogéneas y que pueden ser escritas en diferente lenguaje de programación.

10.El patrón bróker se utiliza para balancear las siguientes fuerzas:

11.Los componentes deben ser capaces de acezar a los servicios provistos por otros a través de invocaciones de servicios remotos transparentes en ubicación.

12. Se necesita intercambiar, añadir y quitar componentes en tiempo de ejecución.

13. La arquitectura debe esconder los detalles específicos de implementación del sistema de los usuarios de componentes y servicios.

**ELEMENTOS**

1. Servidor

2. Clientes

3. Brokers

4. Puentes “Bridge”

5. Proxy-cliente

6. Proxy-servidor

**SERVIDOR**

**-**Un servidor implementa objetos que exponen su funcionalidad a través de interfaces que consisten en operaciones y atributos.

-Las interfaces están disponibles a través de un lenguaje de definición de interfaz (IDL) a un estándar binario.

-Hay dos tipos de servidores:

1. Los servidores que ofrecen servicios comunes a muchos dominios de aplicación.

2. Los servidores que implementan una funcionalidad específica para un dominio de aplicación particular.

**CLIENTES**

-Son aplicaciones que acceden a los servicios de, al menos, un servidor.

-Para invocar servicios remotos, los clientes envían solicitud al bróker. Después de que la operación se haya ejecutado los clientes reciben respuestas o excepciones del bróker. La interacción entre clientes y servidores se basa en un modelo dinámico lo cual significa que los servidores también pueden actuar como clientes.

-Los clientes no necesitan conocer la ubicación de los servidores que acceden; esto es importante porque permite la agregación de nuevos servicios, y el movimiento de los servicios existentes a otras ubicaciones, aun mientras el sistema este ejecutándose.

**BRÓKER**

**-**Es un mensajero responsable de la transmisión de solicitudes de clientes o servidores, así como la transmisión de respuestas.

-Localiza al receptor de una solicitud basándose en un sistema de identificadores únicos.

**PROXY-CLIENTE**

-Representan una capa adicional entre los clientes y el bróker, para proveer transparencia en el sentido que un objeto remoto aparece como local ante el cliente, es decir, esconden los detalles de implementación.

**PROXY-SERVIDOR**

-Son responsables de recibir solicitudes, desempaquetar los mensajes de entrada, el unmarshaling de los parámetros, llamar al servicio apropiado y el marshaling de resultados y excepciones antes de enviárselas al cliente.

-MARSHALING: Transformar la representación en memoria de un objeto a un formato apropiado para almacenaje o transmisión.

**PUENTES**

-Son componentes opcionales utilizados para esconder los detalles de implementación cuando dos brokers interoperan.

-Supóngase que un sistema bróker se ejecuta en una red heterogénea. Si se transmiten solicitudes sobre la red, se deben comunicar brokers diferentes independientemente de las redes y de los sistemas operativos utilizados.

**RELACIONES Y RESTRICCIONES**

-La relación de unión asocia clientes (y, posiblemente proxies de lado del cliente) y servidores (y, posiblemente proxies del lado del servidor) con brokers.

-El cliente solo puede conectarse a un bróker (posiblemente a través de un proxie del cliente). El servidor solo se puede unir a un bróker (posiblemente a través de un proxie servidor.

**VARIACIONES DEL PATRÓN**

1.SISTEMAS DEL BRÓKER DE COMUNICACIÓN DIRECTA:

-En esta variante los clientes se pueden comunicar directamente con los servidores.

-El bróker indica a los clientes los canales de comunicación.

-En este sistema, los proxies se encargan de las responsabilidades para manejar la mayoría de las actividades de comunicación.

2. SISTEMAS BRÓKER DE PASO A MENSAJES:

-Esta variante es apropiada para sistemas que se enfocan en la transmisión de datos.

-Los servidores utilizan un tipo de mensaje para determinar lo que deben hacer en ves de ofrecer servicios que los clientes pueden invocar.

-En este contexto, un mensaje es una secuencia de datos en un conjunto con información adicional que especifica el tipo de mensaje, su estructura y otros atributos relevantes.

3.SISTEMAS BRÓKER ADAPTADORES:

-Para aumentar la flexibilidad, se puede esconder el interfaz del bróker a los servidores con una capa adicional llamada capa adaptadora que es responsable de registrar e interactuar con los servidores.

4. SISTEMAS BRÓKER NEGOCIANTES:

-Usualmente un cliente envía una solicitud a un servidor identificado en forma única, pero en algunas circunstancias los servicios, pero no los servidores son el destino de la solicitud del cliente.

-En esta variante el bróker debe conocer que servidores pueden proveer un servicio especificado y enviar la solicitud al servidor apropiado.

-Sin embargo, las proxies del lado del cliente usan identificadores de servicio en vez de identificadores de servidor para accesar a la funcionalidad de los servidores. La misma solicitud puede enviarse a varios servidores que implementan el mismo servicio.

5.SISTEMAS BRÓKER CALBACK:

-En vez de implementar un modelo de comunicación activa donde los clientes producen solicitudes y los servidores la consumen, se puede utilizar un modelo reactivo que se maneja por eventos sin hacer distinción entre clientes y servidores.

-Cuando un evento llega el bróker inventa un evento calback del componente registrado para reaccionar a ese evento, cuya ejecución, a su vez puede generar nuevos eventos causando que el bróker dispare nuevas invocaciones de métodos calback.

**USOS:**

**-**CORBA: El patrón bróker fue usado para especificar la common object requets bróker architecture definida por el object mangement group. Corba es una tecnología orientada a objetos para objectos distribuidos en sistemas heterogéneos. Orbix de IONA technologies usa la variante de comunicación directa.

-SOM/DSOM DE IBM: Es un sistema bróker compatible con corba que implementa la interoperabilidad combinado el IDL de corba con un protocolo binario.

-OLE DE MICROSOFT: Define un formato estándar binario para exponer y acceder a las interfaces del servidor.

-RMI DE SUN: Tecnología para la invocación de métodos remotos para la plataforma java.

-ATM-P DE SIEMENS: Es la implementación de la variante de paso de mensajes en sistemas de telecomunicaciones basados en asynchronous transfer mode (ATM).