VOLTAR

Arquitetura Orientada a Serviços (SOA)

Apresentar os principais conceitos relacionados à tecnologia SOA.

NESTE TÓPICO

A questão central que envolve o desenvolvimento dos sistemas colaborativos, que é a mesma dos demais sistemas informatizados, é garantir a distribuição segura das informações que são inseridas no sistema, realizar o processamento seguro dessas informações, manter a integridade delas e garantir sua disponibilidade para usos futuros, e em muitas situações, garantir o acesso a um histórico das atualizações que essas informações sofreram ao longo do tempo.

Para tipo de sistema, há um conjunto específico de necessidades que precisam ser atendidas, associadas a limitações técnicas que precisam ser contornadas. A forma como essas necessidades são atendidas, ou a forma como as limitações técnicas são solucionadas, depende, e muito, da experiência das pessoas envolvidas com o desenvolvimento desse sistema, e de como eles vão usar as soluções que eles tem em mãos. Isso significa que a forma como um sistema será aplicado para coletar, processar e distribuir a informação entre os seus usuários, dependerá das soluções usadas para desenvolvê-lo, e da forma como ele será usado.

Ao resolver um problema para uma situação que antes era desconhecida, os desenvolvedores aumentam a sua experiência, e aumentam o seu leque de soluções que foram bem-sucedidas em determinadas situações. Assim, ao desenvolver um novo sistema, e ao se deparar com necessidades ou limitações que já foram resolvidas anteriormente, o desenvolvedor pode adaptar uma solução conhecida para um novo problema. E assim, ele

mantém um ciclo de atualização contínuo de soluções e das situações em que elas foram usadas. Principalmente se forem se essas soluções puderem ser disponibilizadas sob a forma de componentes de software.

Vários produtores de ferramentas de desenvolvimento de software, perceberam que eles poderiam agregar essas soluções em suas ferramentas, e assim atrair os desenvolvedores para usá-las. Diante dessa facilidade, as ferramentas de desenvolvimento de software ganharam bastante destaque nos processos de desenvolvimento, pois a grande maioria delas já orienta os desenvolvedores a usar componentes de software e de hardware, uma vez que elas já são projetadas para operar desse modo, pois a maioria dos modelos de arquitetura orienta o uso de componentes de software.

Componente de software é um conceito que ganhou força mais recentemente, e como vários outros conceitos usados na área da tecnologia da informação, dependendo da área e da forma como eles são abordados, acabam tendo significados diferentes. Antes do conceito de componente de software, se costumava usar os termos biblioteca ou módulo, que possuem um significado bastante similar ao de componente de software. Mas, dependendo da tecnologia usada pelo desenvolvedor, esses termos ainda são usados com bastante frequência.

Vamos definir o **conceito de componente de software** como sendo **uma parte do software que é responsável por executar um único serviço para o sistema**, como por exemplo, validar um número de CPF, verificar os dados de um endereço, a partir de um número de CEP, ou gerar um número de identificação para um usuário de um sistema, quando gravar seus dados numa base de dados.

Esse conceito aplicado aos componentes de software é mais comum ao modelo de Arquitetura Orientado a Serviço (SOA – sigla para Service Oriented Architecture). Esse modelo orienta o desenvolvedor a basear seu projeto em componentes de software que executam serviços específicos. Normalmente, eles se comunicam com outros componentes, seja para realizar um serviço para outros componentes, seja para solicitar que outros componentes façam algum serviço.

Comunicação Entre Componentes de Software

Um componente de software se comunica com outro componente de software através da troca de mensagens. Nesse processo, quando um componente de software envia uma mensagem, ele inicia uma comunicação para solicitar que outro componente de software execute algum tipo de processamento. O componente de software que recebe essa mensagem, verifica o que ele precisa fazer com os dados presentes nessa mensagem, e executa o que lhe foi solicitado. Ao término do processamento, o componente de software envia outra mensagem, nesse caso, uma resposta para o componente que fez a solicitação do processamento. Dependendo do processamento sendo executado, o componente de software pode precisar iniciar uma comunicação com outro componente, para que ele processe algum dado, seguindo um fluxo de comunicação parecido ao que foi explicado anteriormente.

Assim, as mensagens usadas na comunicação entre os componentes de software são usadas para enviar ou solicitações para execução de serviços, ou para transportar os resultados de processamento. Para que as mensagens sejam tratadas corretamente, é necessário controlar esse tráfego de mensagens, e para isso, é possível adotar algumas estratégias.

Figura 1 - Modelo de comunicação unicast

Fonte:

Uma estratégia bastante usada é a comunicação ponto a ponto. Nesse modelo de comunicação, os dados que serão transportados pelas mensagens serão sempre inseridos no sistema através de um nó (que pode ser um computador, um dispositivo móvel ou um servidor). Esse nó que foi usado para inserir os dados, faz uma cópia dessa mensagem, e a envia para um nó, ou conjunto de nós, que deverão estar conectados na rede para receber a mensagem.

Esse modelo é chamado de ponto a ponto, pois as mensagens que estão trafegando pelo sistema são processadas apenas pelos nós que originaram a comunicação e pelos nós que foram marcados para receberão a mensagem. Se ela passar por outros nós, os nós apenas passarão a mensagem adiante. Com este modelo, o processamento das mensagens é mais simples, e elas podem ser trocadas rapidamente entre os vários nós conectados ao sistema. Entretanto, a cada novo aplicativo cliente que venha a ser adicionado ao sistema, o uso da rede de dados aumentará, e poderá ocorrer uma sobrecarga no uso da rede, devido à grande quantidade de mensagens sendo transmitidas. Isso poderá causar problemas como:

- atraso nas transmissões;
- perda de sincronismo nas mensagens; e,
- possivelmente, a escalabilidade do sistema pode se tornar inviável.

Para evitar esses problemas, uma alternativa é trocar as mensagens de acordo com o modelo de arquitetura cliente-servidor. Nesse caso, a mensagem inserida no sistema é enviada para um servidor que centralizará o processamento dela. O servidor pode enviar essa mensagem aos demais clientes da rede, ou permitir que vários clientes tenham acessem essa mensagem. Entre os clientes e um servidor, ou do servidor para outros servidores, e de um servidor para seus clientes, a troca das mensagens pode ser no modelo ponto a ponto.

A vantagem do modelo cliente-servidor é que os clientes que deverão receber as mensagens inseridas pelo servidor, não precisam estar conectados ao sistema. Isso diminui o uso da rede, pois há menos mensagens trafegando na rede. Um dos problemas deste modelo é quando à escalabilidade do sistema, pois devido à centralização do processamento de informações em servidores espalhados pela rede, é possível que vários servidores possam processar a mesma mensagem, causando redundância.

Estes dois modos de comunicação são exemplos de comunicação *unicast*, na qual um nó determina o nó, ou nós, que deverá receber a mensagem. Neste caso, é comum implementar uma verificação do recebimento da mensagem em cada um dos nós, além de manter um controle que assegure que as mensagens foram enviadas e recebidas apenas pelos destinatários aos quais haviam sido endereçadas. Como consequência, haverá grande quantidade de processamento nos vários nós do sistema. Esse processamento pode ser reduzido se foram usados modelos de comunicação conhecidos como *broadcast* ou *multicast*.

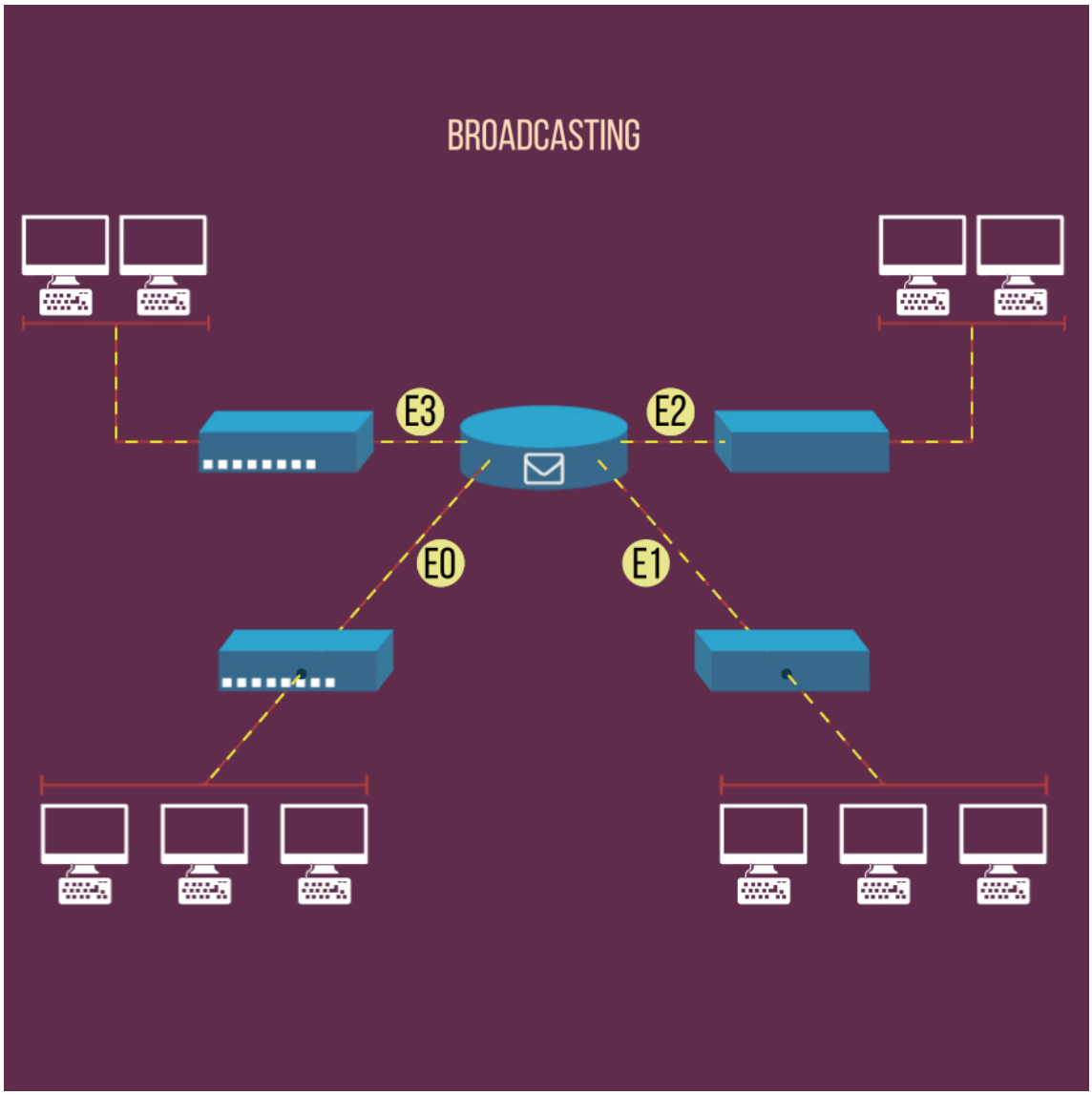


Figura 2 - Modelo de comunicação broadcast

A comunicação ***broadcast*** funciona de modo análogo às transmissões de rádio e TV. Ou seja, um nó envia uma mensagem pela rede, ela trafega na rede, disponível a todo e qualquer nó conectado na rede. Da mesma forma que os programas de rádio ou TV somente são assistidos por quem deseja vê-los, o mesmo ocorre com a mensagem enviada sob a forma de broadcast. Somente os nós que desejam receber determinada mensagem a processam.

Isso ajuda a evitar a redundância de processamento, mas para funcionar, a rede de comunicação usada pelo sistema deverá permitir esse tipo de envio de mensagens, e desde que todos os clientes estejam conectados na rede no momento em que as mensagens estão sendo enviadas, pois depois de um determinado período de tempo, as mensagens enviadas serão eliminadas automaticamente pelo sistema.

Nos aplicativos atuais, há uma demanda excessiva para usar a rede de comunicação, pois boa parte dos aplicativos é executada em algum servidor conectado à rede, que é acessado através de um dispositivo móvel ou através de browsers. Os exemplos mais populares são o Facebook, Whats App,

Instagram, entre outros. Mas há outros aplicativos, como os sistemas bancários, que possuem outros dispositivos que se conectam ao sistema, e dependem do processamento que é realizado pelos servidores.

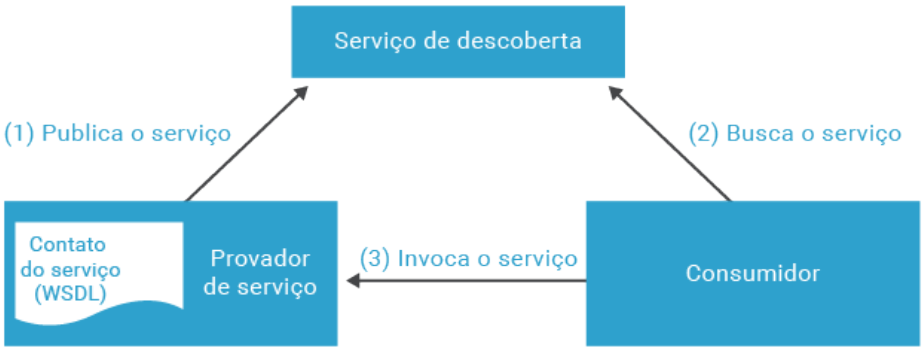
Aqui estamos falando principalmente da rede de comunicação, mas também é necessário implementar soluções relacionadas a segurança, gerenciamento de dados, balanceamento da capacidade de processamento, etc... Em geral, todos eles são problemas relacionados à infraestrutura do aplicativo, e são requisitos de infra-estrutura que também costumam usar a rede de comunicação para poderem funcionar. Assim, é comum combinar vários modelos de comunicação, dependendo da tarefa que precisa ser executada.

O problema no uso combinado desses modelos, as desvantagens e as vantagens deles se somam. Para diminuir o impacto negativo das desvantagens, e aumentar os aspectos positivos das vantagens, foi pensado em um modelo em que os nós do aplicativo possam desempenhar ao mesmo tempo os papéis de cliente e servidor. Como resultado, o modelo de arquitetura orientada a serviços se tornou bastante popular.

Arquitetura Orientada a Serviços (SOA)

O modelo de Arquitetura Orientada a Serviços, conhecida como SOA (do inglês Service Oriented Architecture), prevê um modelo em que o aplicativo é implementado através da composição de serviços. Nesse modelo, o termo serviço tem um significado parecido com o de componente de software, ou seja, um serviço é entendido como sendo um componente de software que executa uma funcionalidade, ele é independente dos outros componentes de software e é autocontido, ou seja, ele deve ser capaz de terminar o que está fazendo, independentemente do resultado do processamento ser negativo ou positivo.

A forma de análise para determinar os serviços do software é praticamente o mesmo usado para projetar os componentes de software. O projetista identifica quais são os serviços que o software precisa para funcionar, e como eles deverão se comunicar com outros componentes, seja através de metadados, ou através de mensagens.



Modelo de Arquitetura SOA

Fonte: PIMENTEL, Mariano; FUCKS, Hugo. Sistemas Colaborativos. Editora Campus, 2011.

O modelo de arquitetura SOA se baseia em três componentes principais:

- Provedor de Serviço: responsável por executar um determinado serviço, e por manter a rede de comunicação informada sobre qual é o tipo de serviço que ele está oferecendo;
- Serviço de Descoberta: responsável por armazenar a localização dos serviços disponíveis na rede, e conectar o serviço solicitado ao serviço que solicitou a execução desse serviço;
- Consumidor: é quem solicita a execução de um serviço.

O SOA se baseia no modelo de comunicação híbrida, onde um nó presenta na rede de comunicação pode ser definido tanto como sendo um cliente quanto sendo um servidor. Adaptando os nomes desses nós ao modelo SOA, é o mesmo que dizer que o nó deve ser projetado para ser tanto um Provedor, quanto um Consumidor de serviços.

Para implementar uma aplicação baseada em SOA, o desenvolvedor do software precisa projetar seus componentes de software para prover um serviço ao aplicativo. Esse componente de software denominado Provedor de Serviços define uma descrição do que ele faz, chamada Descrição de Serviço. Essa descrição contém as informações a respeito do serviço que ele executa, e quais as informações que o Consumidor precisa fornecer, para usar esse serviço. Desse modo, o Consumidor é capaz de decidir se o serviço atende às necessidades do Consumidor, quais os requisitos definidos pelo Provedor de Serviços, e como ele pode se conectar à interface de serviço.

A interface de serviço representa uma forma de contato através de um conjunto de métodos que podem ser invocados pelo Consumidor, e os tipos de dados que serão usados durante a invocação e execução do serviço. O uso de interfaces permite que o Consumidor use um serviço sem ser necessário saber como o Provedor de Serviço funciona internamente. As interfaces também permitem que os serviços sejam usados por uma infinidade de Consumidores, implementando um conceito conhecido como reuso de software.

Essas características fazem com que um aplicativo desenvolvido com base em SOA, seja um sistema distribuído, e que muito provavelmente será usado num ambiente heterogêneo para processamento das informações. A existência do ambiente heterogêneo automaticamente exige que os aplicativos baseados no modelo SOA tenham interoperabilidade, pois o Provedor de Serviço desse aplicativo pode ser acessado por Consumidores de outros aplicativos. Como é bastante possível que esses aplicativos tenham sido desenvolvidos com diferentes tecnologias de desenvolvimento, o Provedor de Serviços precisará estabelecer alguns padrões de protocolos de comunicação, e/ou de metadados.

Todas essas características juntas permitirão que os componentes de software desse sistema apresentem baixo acoplamento, podendo ser localizados e invocados dinamicamente, o que lhes permite a independência em relação aos demais serviços.

Outra característica de SOA é a possibilidade de combinar os serviços já existentes para criar novos serviços. Nesse caso, deve haver um componente de software que coordenará a execução dos serviços. Esse componente de software também controlará os componentes de software que executam os serviços do Provedor de Serviços.

Aplicando o Modelo SOA

Aplicando esse modelo ao desenvolvimento dos aplicativos colaborativos, e seus similares, é através de um modelo em que os usuários possam colaborar entre si através da criação e compartilhamento de serviços. Desse modo, os usuários do sistema podem atuar tanto como Provedores de Serviços, no momento em que ele cria e publica um serviço, quanto como Consumidores de serviços, momento em que o usuário pesquisa por um serviço capaz de atender às suas necessidades, para invocá-lo e usá-lo.

Quiz

Exercício Final

Arquitetura Orientada a Serviços (SOA)

INICIAR

Referências

PIMENTEL, Mariano; FUCKS, Hugo. *Sistemas Colaborativos*. Editora Campus, 2011.

SWENEY, R. *Achieving Servive-Oiented Architecture, Applying an Enterprise Architecture Approach*. Wiley, 2010

Avalie este tópico

ANTERIOR
Arquitetura em 3 Camadas

Índice

PRÓXIMO
Servidores de Aplicação

Ajuda?
(https://ava.un
idCurso=)

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site