***SOA***

É um conceito abstrato para desenvolvimento de software que visa definir a integração entre os diversos serviços que compõem uma aplicação sem, entretanto, entrar nos méritos de sua implementação. Em resumo, uma arquitetura orientada a serviços é uma arquitetura de sistemas distribuídos tipicamente caracterizada pelas seguintes propriedades [w3c\_wsarch]:

* **Visão lógica**: o serviço é definido em termos funcionais, tipicamente implementando operações de negócios. A visão lógica do serviço o abstrai dos programas, bases de dados, processos de negócios, etc., necessários à sua implementação;
* **Orientado a mensagens**: o serviço é formalmente definido em termos de mensagens trocadas entre unidades de software provedoras e consumidoras do serviço. A estrutura interna das unidades de software, incluindo sua linguagem de programação, sua estrutura de dados, etc. são deliberadamente abstraída na SOA. Este baixo acoplamento entre aplicações provedoras e consumidoras de um serviço traz benefícios significativos para sistemas legados: a transparência da implementação interna das aplicações de uma arquitetura SOA permite que qualquer componente de software ou aplicação seja envolvido por uma interface de mensagens e adaptado a uma definição de serviços formal.
* **Orientado a descrição**: um serviço é descrito por uma meta-linguagem processável por máquinas. A descrição suporta a natureza pública da SOA: apenas os detalhes do serviço relevantes a seu uso são incluídos na descrição;
* **Granularidade**: serviços tendem a utilizar um pequeno número de operações com mensagens relativamente grandes e complexas;
* **Orientado a redes**: serviços tendem a ser disponibilizados e utilizados através de uma rede, embora este não seja um requisito obrigatório;
* **Independência de plataforma**: mensagens são enviadas num formato padrão, independente da plataforma, através de interfaces bem definidas.

Arquitetura Orientada a Serviços (**SOA**) não é uma tecnologia, não é uma metodologia, não é um serviço, mas é um **conceito** de arquitetura corporativo que promove a integração entre o negócio e a TI por meio de conjunto de interfaces de serviços acoplados. Tanto a orquestração quanto a coreografia, no paradigma SOA, arranjam serviços diferentes para serem executados em uma ordem preestabelecida. A diferença é que a orquestração possui um orquestrador único que define a ordem de realização dos serviços, fazendo com que os mesmos não necessitem conhecer uns aos outros. Na coreografia, é necessário que os serviços saibam da existência uns dos outros, para, então, serem executados em uma ordem preestabelecida.

O **XML** é um formato para a criação de documentos com dados organizados de forma hierárquica, como se vê, frequentemente, em documentos de texto formatados, imagens vetoriais ou bancos de dados. <https://pt.wikipedia.org/wiki/XML>.

Segundo Fábio Perez Marzullo, em seu livro SOA Na Prática, inicialmente, quatro **papéis são essenciais** para a composição do **ciclo de vida de soluções SOA**:

* **Consultor de negócios** -  mapeia processos de negócio da organização.
* **Arquiteto SOA** - quem modela, constrói, instala e dá manutenção ao SOA.
* **Provedor** - quem disponibiliza serviço.
* **Cliente/consumidor** -  quem invoca serviço.

A **Arquitetura Orientada a Serviços** (Service-Oriented Architeture) ou **SOA**, é uma arquitetura de software que se baseia na computação distribuída e cujo princípio é a **entrega de funcionalidades** (serviços) ao invés de sistemas completos. O principal elemento para permitir que aplicações de diferentes sistemas se comuniquem dentro da companhia é através dos **web services**.

Esse é o princípio da arquitetura SOA, lidar com **ambientes heterogêneos** e integrar as **regras de negócio** com o setor tecnológico da companhia, distribuindo seus **serviços** através de um sistema **web service** que pode ser utilizado por diferentes sistemas operacionais e softwares.

**Modelo Triangular x SOA**

* **Provedor de serviços**: cria e desenvolve o serviço Web que serve para expor alguma funcionalidade de negócio da sua organização para a invocação por outros usuários externos. Responsável pela PUBLICAÇÃO do serviço. Também é o responsável por fornecer toda a infraestrutura de acesso e responder a requisições internas e externas.
* **Consumidor de serviços ou cliente**: qualquer usuário que deseja utilizar algum serviço Web. Responsável pela PESQUISA do serviço.
* **Catálogo de Serviços (UDDI)/Registro dos serviços**: um diretório central onde o provedor de serviços possa cadastrar e descrever seus serviços, e onde o consumidor possa pesquisar/procurar pelo serviço desejado. Responsável pela LIGAÇÃO com o serviço.

Três papéis são identificados de acordo com os comportamentos e responsabilidades próprias de um serviço.

* Provedor de Serviço: aquele que oferece o serviço;
* Consumidor de Serviço: aquele que utiliza o serviço;
* O Registro de Serviço: mecanismo que permite ao provedor cadastrar seus serviços e ao consumidor encontrá-los.

Princípios SOA:

* Padronização do contrato de serviço
* Baixo acoplamento
* Abstração do serviço
* Autonomia do serviço
* Visibilidade do serviço
* Independência do controle de estado do serviço
* Reusabilidade
* Capacidade de composição do serviço

**Orquestração:** composição de processos de negócio (por exemplo, por meio de Web  
Services na qual existe a figura de um processo central (processo mestre) que controla e coordena os demais processos. Neste tipo de composição,cada processo participante não tem conhecimento de que faz parte de uma composição de processos, com exceção do processo mestre.

**Coreografia**: composição de processos de negócio (por exemplo, por meio de Web Services na qual não existe a figura de um processo mestre que controla e coordena os demais processos. Neste tipo de composição, cada processo envolvido tem o conhecimento de que faz parte de uma composição de processos e que precisa interagir com outros processos de maneira ordenada para que a composição resultante tenha sucesso.

"Dentro de uma solução orientada a serviços, as **unidades de lógica (serviços) encapsulam funcionalidades não específicas a nenhum aplicativo ou processo de negócio**. Esses serviços são classificados como ativos de TI **agnósticos** e reusáveis. Serviços agnósticos fornecem um intervalo de funcionalidades genéricas."

Maturidade SOA:

* **Nível 1: Silo –**Cada setor da empresa desenvolve suas próprias aplicações de forma independente, sem integração de dados, processos, padrões ou tecnologia.
* **Nível 2: Integrado –**A tecnologia é usada para integrar os silos e os dados, porém é necessária uma conversão de dados e protocolos, uma vez que os sistemas e processos não obedecem a padrões.
* **Nível 3: Componentizado –**Os sistemas são decompostos em componentes, porém a integração entre eles possui um alto acoplamento. Os componentes são reutilizáveis  através de EAI - *Enterprise Application Integration*, mas o ambiente ainda apresenta muitas replicações e redundâncias.
* **Nível 4: Serviço –**Aplicações são construídas através da composição de Serviços. Os Serviços são identificados e construídos com base nos Princípios de *Design* de Serviços, garantindo o baixo acoplamento e a interoperabilidade, independentemente da tecnologia utilizada.
* **Nível 5: Serviços Compostos –**Neste nível a empresa constrói suas aplicações através da interação de vários serviços, usando uma linguagem de modelagem de processos de negócio, como BPEL. Com isto a empresa ganha agilidade na resposta a novas solicitações das áreas de negócio.
* **Nível 6: Serviços Virtualizados –**Os Serviços passam a ser disponibilizados por meio de fachadas (*façades*) de modo que seu acesso seja feito de maneira indireta, permitindo que os Serviços possuam um baixo acoplamento à infraestrutura utilizada. Isto cria mais possibilidades de composição de Serviços.
* **Nível 7: Serviços Reconfigurados Dinamicamente –**Serviços podem ser localizados e compostos dinamicamente, sem necessidade de intervenção de desenvolvedores. Embora possa parecer muito futurista, já existe tecnologia para que isto possa acontecer.

**As principais características dos serviços Web são:**  
  
**Baseado em XML**: usado para representar os dados. Como transporte de dados, XML (eXtensible Markup Language) [4] elimina qualquer dependência com rede e sistema operacional.

**Fracamente acoplado**: a interface de um serviço Web pode mudar durante o tempo sem comprometer a habilidade do cliente de interagir com o serviço.  
  
**Granularidade grossa**: provê uma maneira natural de definir serviços de granularidade grossa que acessam a quantidade correta de lógica de negócio.  
  
**Chamadas síncronas e assíncronas**: um cliente pode invocá-lo de forma síncrona e assíncrona. Possibilitar chamadas assíncronas é a chave para permitir sistemas fracamente acoplados.

Qual o relacionamento entre eles (*web services* X SOA)? Embora eles apareçam juntos e o uso deles seja feito de forma conjunta (visto que a forma mais comum de se implementar um SOA é por meio de WS), pode-se ter um webservice isolado e vc pode ter uma arquitetura orientada a serviços (SOA) sem webservice;

Os Webservices frequentemente são usados para se implementar uma arquitetura orientada a serviços (SOA), mas não é obrigatório. Ter webservices não implica SOA;

Da mesma forma, ter SOA não implica o uso de Webservices. Ou seja, embora a utilização deles estejam diretamente relacionadas, eles podem ser implementados de forma independente.

**SOAP - Simple  Object  Access  Protocol**

É um protocolo baseado em XML que permite às aplicações trocarem informações estruturadas nas implementações de Web Services. Os dados são Transferidos no formato XML, encapsulados pelo protocolo SOAP. A sintaxe de uma mensagem SOAP é bem simples e contem algumas regras, tais como a utilização de XML, o uso dos namespace específicos da mensagem SOAP, seu namespace deve utilizar a codificação definida pela w3c, não deve conter referencia DTD e nem ter instruções de processamento XML.

* O elemento **ENVELOPE**é o elemento raiz de uma mensagem SOAP e corresponde à descrição da mensagem e do que deve ser processado.
* O elemento **HEADER**é opcional e tem como função estender as funcionalidades das mensagens SOAP.
* O elemento **BODY**é obrigatório e contém a informação propriamente dita.

O elemento **fault** SOAP tem os seguintes sub elementos:

* **faultcode**: um código para a identificação da falha.
* **faultstring**: Uma explicação legível humana da falha.
* **faultactor**: Informações sobre quem causou a falha acontecer.
* **detail**: Mantém aplicação específica informações de erro relacionado com o elemento do corpo.

**Regras de sintaxe XML:**  
  
**•**Deve ser codificada em XML.  
**•**Deve usar o SOAP Encoding namespace. (Não contém um valor default, deve-se, portanto, configurá-lo).Exemplo: soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
**•**Deve usar o SOAP Envelope namespace. Tal namespace DEVE ser o seguinte: "xmlns:soap="http://www.w3.org/2003/05/soap-envelope/". Se for utilizado um namespace diferente, será gerado um erro e a mensagem será descartada.  
**•**Não deve conter instruções de processamento XML.   
**• Não** pode conter referência a uma DTD.

**WSDL - WEB SERVICES DESCRIPTION LANGUAGE**

Define como as interfaces dos Web services podem ser representadas. É uma linguagem baseada em XML utilizada para descrever e localizar Web Services.

Elementos Básicos WSDL:

* <types>: aqui deverão ser descritos os tipos de dados suportados pelo serviço em questão.
* <message>: aqui devem ser especificados os padrões de entrada e saída de dados dos web services.
* <portType>: aqui devem ser descritos os agrupamentos lógicos das operações. São as operações executadas pelo web service.
* <binding>: aqui devem ser apresentados os protocolos de comunicação que os web services utilizam.
* <operation>: região que permite a especificação das assinaturas dos métodos disponibilizados.
* <definitions>: elemento padrão de todos os documentos WSDL. Permite efetuar descrições sobre schemas e namespaces.

A especificação WSDL define três aspectos de um web service : o que faz o serviço, como ele se comunica e onde o encontrar:

1. O tópico "o que" de um documento WSDL, denominado **interface**, especifica quais operações o serviço suporta e define o formato das mensagens que são enviadas e recebidas pelo serviço.
2. O "como" de um documento WSDL, denominado **ligação** ,mapeia a interface abstrata para um conjunto concreto de protocolos. A ligação especifica os detalhes técnicos de como se comunicar com o webservice.
3. A parte "onde" de um documento WSDL descreve o**local da implementação** de um web service específico (seu ponto final)."

**Descrições de Interface do serviço do WSDL**

* <**definitions**>: elemento padrão de todos os documentos WSDL (raiz). Permite efetuar descrições sobre schemas e namespace.
* <**types**>: aqui deverão ser descritos os tipos de dados suportados pelo serviço em questão.
* <**message**>: aqui devem ser especificados os padrões de entrada e saída de dados dos web services.
* <**portType**>: aqui devem ser descritos os agrupamentos lógicos das operações. São as operações executadas pelo web service.
* <**binding**>: aqui devem ser apresentados os protocolos de comunicação que os web services utilizam.
* <**operation**>: região que permite a especificação das assinaturas dos métodos disponibilizados.
* **<service>** - coleção de portas (endpoints) suportadas pelo serviço.

**find-bind-execute paradigm**: define o ciclo que envolve o planejamento, a execução, o monitoramento e a tomada de ação pró ativa para a melhoria da qualidade.

Tecnicamente falando, o processo preconiza que os provedores de serviços registrem informações em um registro central, com suas características, indicadores, e aspectos relevantes às tomadas de decisões. O registro é utilizado pelo cliente para determinar as características dos serviços necessários, e se o mesmo estiver disponível no registro central, como por exemplo, por um catálogo de serviços, o cliente poderá utilizá-lo, sendo este oficializado através de um contrato que enderece este serviço.

**Tipos de serviços usados na SOA:**

* Business services;
* Entity services;
* Functional services;
* Utility services;

**Functional services**

Functional services do not represent business-related tasks or functions. Rather it usually can be represented in a sequence diagram. In other words, it is usually a technology-oriented service and not a business oriented one. Task services can be thought of as controller of composition of services and hence its reusability is usually lower.

**Business services**

Business service can be defined as the logical encapsulation of business functions. It has to be relevant to the business of the organization is running. An easy way to determine whether a service is a business service is to ask whether the service can be created without the consultation of business managers. If not, the service isn’t probably a business service.

**Entity services**

An entity service usually represents business entities (e.g. Employee, Customer, Product, Invoice etc.). Such entity service usually expose CRUD (create, read, update, delete) operations.

**Utility services**

Utility services offers common and reusable services that are usually not business centric. They might include event logging, notifications exception handling etc.

WebServices é uma das formas de implementação (realização) da arquitetura SOA.

**Barramento de Serviços (Enterprise Service BUS - ESB):** modelo conceitual tecnológico que se utiliza de padrões e ferramentas de modelagem e desenvolvimento para unir e conectar serviços, aplicações e recursos de TI da organização.

É um conceito que viabiliza o uso de SOA como infraestrutura de soluções corporativas; regulamenta a forma como os serviços se comunicam e interagem uns com os outro.

**Arquitetura de objetos distribuídos.**

Uma abordagem mais geral para sistemas distribúidos é eliminar a distinção entre cliente e servidor e projetar a arquitetura de sistema como uma arquitetura *de objetos distribuídos. Os componentes do sistema são objetos que oferecem uma interface para um conjunto de serviços que eles fornecem. Outros objetos solicitam esses serviços sem fazer distinção lógica entre um cliente (quem recebe o serviço) e um servidor (quem provê o serviço).*

*Nesse caso pode-se pensar na camada de middleware como um barramento de software.*Ela permite que objetos se comuniquem e sejam acrescentados aos sistemas. Essa arquitetura permite adiar decisões sobre onde ou como os serviços devem ser fornecidos e também permite que novos recursos sejam acrescentados conforme necessário.

Definição perfeita de Web service.

Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. As Web Services são componentes que permitem às aplicações enviar e receber dados. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, um formato intermediário como XML, Json, CSV, etc.

Para as empresas, as Webs Services podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana.

Essencialmente, o Web Service faz com que os recursos da aplicação do software estejam disponíveis sobre a rede de forma normalizada. Outras tecnologias fazem a mesma coisa; por exemplo, os browsers da Internet acessam as páginas Web disponíveis usando por norma as tecnologias da Internet, HTTP e HTML. No entanto, estas tecnologias não são bem sucedidas na comunicação e integração de aplicações. Existe uma grande motivação sobre a tecnologia Web Service, pois possibilita que diferentes aplicações comuniquem-se entre si e utilizem recursos diferentes.

Utilizando a tecnologia Web Service, uma aplicação pode invocar outra para efetuar tarefas simples ou complexas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo Web Service.

Os Web Services são identificados por um URI (Uniform Resource Identifier), descritos e definidos usando XML (Extensible Markup Language). Um dos motivos que tornam os Web Services atractivos é o facto deste modelo ser baseado em tecnologias standards, em particular XML e HTTP (Hypertext Transfer Protocol). Os Web Services são utilizados para disponibilizar serviços interativos na Web, podendo ser acessados por outras aplicações usando, por exemplo, o protocolo SOAP (Simple Object Access Protocol).

http://blog.caelum.com.br/morte-a-sessao-entenda-esse-tal-de-stateless-session-com-tokens/