Web-Services em .NET

Tiago Rosado

a20105059@alunos.isec.pt

Resumo

Este seminário tem como objetivo dar a conhecer os Web Services como uma solução de integração e comunicação entre sistemas diferentes. O tema vai ser apresentado de modo a clarificar o modo de funcionamento e identificar a tecnologia por detrás de um Web Service e como construir um com base na plataforma .NET.

1. Introdução

O conceito básico de um Web Service é a disponibilização na internet de uma interface para as funcionalidades de uma dada aplicação, dando-as a conhecer para que possam ser utilizadas por qualquer que seja a aplicação cliente.

Web Service é uma solução que permite a integração de sistemas e a comunicação entre aplicações diferentes, independentemente das linguagens em que se encontrem implementadas.

Web Services

- São componentes aplicacionais
- Comunicam através de protocolos abertos
- São autossuficientes e auto descritivos
- Podem ser descobertos utilizando UDDI
- Podem ser utilizados por outras aplicações
- Baseados em XML

De uma forma muito simplista, a arquitetura de um Web Service é descrita com base na seguinte ilustração.

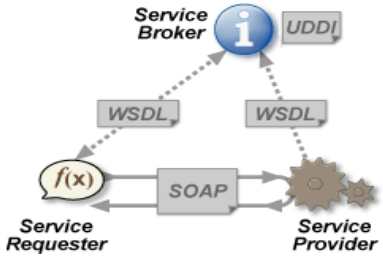

Ilustração 1 - Arquitetura de um Web Service

Conforme mencionado, pretende-se analisar os Web Services. Dar a conhecer a sua origem, o que são e como funcionam.

Faz parte do âmbito deste relatório a apresentação dos Web Services segundo a plataforma .NET com um exemplo de como implementar e utilizar um serviço criado nesta plataforma.

Este texto encontra-se organizado da seguinte forma:

1. Introdução
2. Enquadramento Histórico
3. Web Services
4. Arquitetura dos Web Services
5. Tecnologias Envolvidas nos Web Services
   - 5.1. XML
   - 5.2. SOAP
   - 5.3. WSDL
   - 5.4. UDDI
6. Segurança em Web Services
7. Criar Web Services no Visual Studio 2010
   - 7.1. Criar um Web Service
   - 5.2. Criar um Cliente para o Web Service

A secção 2 descreve o percurso histórico até chegarmos aos web services como são conhecidos hoje. Na secção 3 é apresentado o conceito de web services. A secção 4 indica a arquitetura dos web services. Na secção 5 são descritas as tecnologias que os compõem, como sendo o XML o SOAP, WSDL e UDDI. A secção 6 aborda a questão da segurança em web services. Na secção 7 é apresentado um exemplo de como criar um web service e um cliente que utilize os seus serviços, com recurso à plataforma .NET com recurso ao Visual Studio.

2. Enquadramento Histórico

Os Web Services não são novidade, pelo contrário, representam uma evolução dos princípios que têm orientado a internet durante anos. Surgiram da necessidade de aceder remotamente a serviços alojados

numa outra máquina / local, que no conceito de programação é conhecido como programação distribuída e daí as aplicações distribuídas.

Com sistemas tão diferentes e aumentando os serviços disponibilizados, tornou-se necessário a padronização das comunicações entre as diferentes plataformas (Mainframe, Mac, Windows, Linux) e as linguagens de programação (PHP, Java, VB, C#, etc).

Com o evoluir das plataformas começou a crescer a necessidade de as operar não só localmente mas à distância, como mandar executar comandos, rotinas e procedimentos, surgindo assim a ideia do RPC (acrónimo de *Remote Procedure Call*) que data de 1976 e permite que um programa aceda a funcionalidades de outro, implementando o modelo cliente-servidor.

No modelo RPC uma thread é responsável pela gestão de dois processos, cliente e servidor. O processo cliente envia uma mensagem para o processo servidor e aguarda por uma resposta. A mensagem com o pedido do cliente contem os parâmetros do procedimento e a mensagem de resposta do servidor contem o resultado da execução do procedimento. Uma vez recebida a resposta, o processo do cliente é desbloqueado e prossegue a sua execução.

Do lado do servidor há um processo que se encontra em execução a aguardar pedidos. Quando recebe um, são analisados os parâmetros, processada a informação, e enviada a resposta, voltando ao estado de espera.

A imagem abaixo é ilustrativa do funcionamento de uma chamada a uma função remota (RPC).

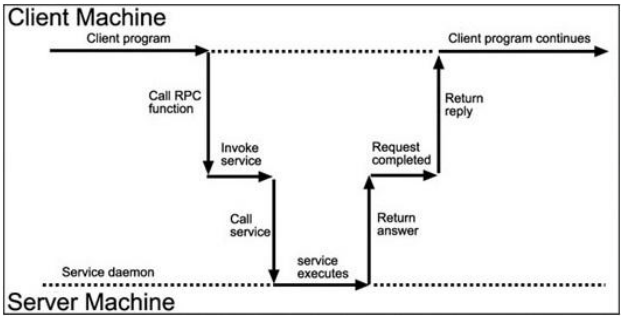

Ilustração 2 - Funcionamento de uma chamada RPC

Uma das primeiras utilizações comerciais do padrão RPC foi feita pela Xerox em 1981, "Courier" The remote procedure call protocol utilizado no sistema de redes da Xerox, e a primeira implementação popular para Unix foi o Sun RPC, atualmente ONC RPC que foi desenvolvido pela Sun Microsystems como parte do projeto Network File System, um sistema de arquivo distribuído com intuito de partilhar dados para uma rede.

O protocolo RPC pode ser implementado sobre diferentes protocolos de transporte, visto que é indiferente o modo como a mensagem é transmitida entre processos. Como o RPC não implementa nenhum mecanismo que garanta o envio e receção de mensagens, torna-se necessário por parte do programador ter atenção sobre qual o protocolo de transporte assenta a comunicação.

Se utilizar o TCP, orientado à ligação, este já implementa mecanismos que garantem a boa receção da informação, no entanto, para o protocolo UDP, esses mecanismos como sendo mecanismos de *timeout*, retransmissão e deteção de erros, são da responsabilidade do programador.

A imagem que se segue ilustra o fluxo de mensagens entre duas máquinas numa rede.

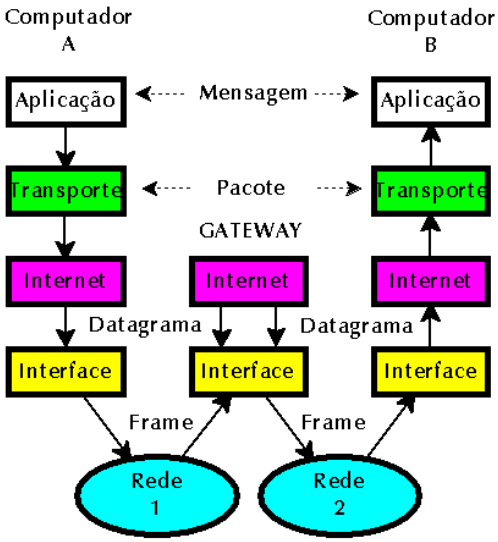

Ilustração 3 - Fluxo de mensagens entre maquinas numa rede

Para permitir a comunicação entre clientes e servidores heterogéneos, foram criados vários sistemas padronizados com base no protocolo RPC e ORBs (acrónimo de *Object Request Broker*) que são os interfaces para os diversos serviços disponibilizados na rede pelos diferentes servidores. Os pedidos dos clientes e as respostas dadas pelos servidores são trocados entre ORBs e utilizam para o efeito, maioritariamente, uma linguagem de descrição de interface, o IDL (acrónimo de *Interface Description Language*).

A descrição dada pela IDL é independente de qualquer linguagem de programação e, por isso, possibilita a comunicação entre componentes implementados em linguagens diferentes.

A competição entre os fabricantes de ORBs é grande, relevando para segundo plano a interoperabilidade entre eles.

Ao longo dos anos foram surgindo as aplicações com implementação do protocolo RPC e na sua maioria usando a linguagem IDL

1991 – CORBA (acrónimo de *Common Object Request Broker Architecture*) é a arquitetura padrão criada pela Object Management Group em resposta à revolução da programação orientada a objetos. DCOM (acrónimo de *Distributed Component Object Model*), tecnologia proprietária da Microsoft.

1997 – RMI (acrónimo de Remote Method Invocation) da Sun, distribuído no JDK 1.1, usando só objetos Java. Por parte da Microsoft houve evolução do DCOM no COM+.

1999 – A Sun distribui o Java 2 Platform Entreprise Edition que integra o RMI e o IIOP (acrónimo de *Internet InterORB Protocol*) que facilita a interligação entre sistemas Java e CORBA. O SOAP (acrónimo de *Simple Object Acess Protocol*) apareceu pela primeira vez.

2001 – A IBM e a Microsoft propõem as pilhas de protocolos dos Web Services à W3C (acrónimo de Worl Wide Web Consortium)

- Wire stack
- Description stack
- Discovery stack

É deste momento em diante que começa a mudança do paradigma da computação com objetos distribuídos para o paradigma da computação distribuída orientada a serviços.

3. Web Services

A Microsoft criou o termo "Web Services" em junho de 2000 quando lançou a Framework .Net e os web services como sendo o componente chave para uma nova abordagem de programação para a internet. À medida que foram investigando, tornou-se claro que a tecnologia poderia revolucionar e ser o próximo passo na programação distribuída, englobando um conjunto de padrões que permitem dois computadores comunicar e trocar informação através da internet.

Os Web Services são uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. De modo muito simples, é uma aplicação que utiliza o XML para trocar informação com outras aplicações através da internet, utilizando simples protocolos de comunicação.

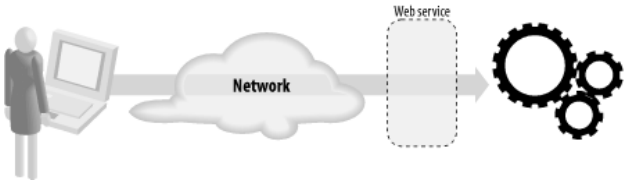

Ilustração 4 - Acesso a serviços de uma dada aplicação através da internet

Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os Web Services são componentes que permitem às aplicações enviar e receber dados em formato XML (acrónimo de eXtensible Markup Language). Cada aplicação pode ter a sua própria maneira de comunicar, que é traduzida para uma linguagem universal, o formato XML.

Principais características dos Web Services

- São componentes aplicacionais
- Independentes da linguagem
- Independentes do protocolo
- Comunicam através de protocolos abertos
- São autocontidos
- São auto descritivos
- Podem ser descobertos utilizando UDDI
- Podem ser utilizados por outras aplicações
- São baseados em XML

Principais Tecnologias dos Web Services

- XML
  Que fornece uma representação dos dados, independentemente da plataforma.
- SOAP
  Que descreve o protocolo de comunicação e troca de dados.
- WSDL
  Que descreve o serviço.
- UDDI
  Que fornece uma maneira de encontrar os serviços na internet.

4. Arquitetura dos Web Services

A imagem abaixo é representativa da interação existente entre os vários elementos que constituem a arquitetura de um Web Service.

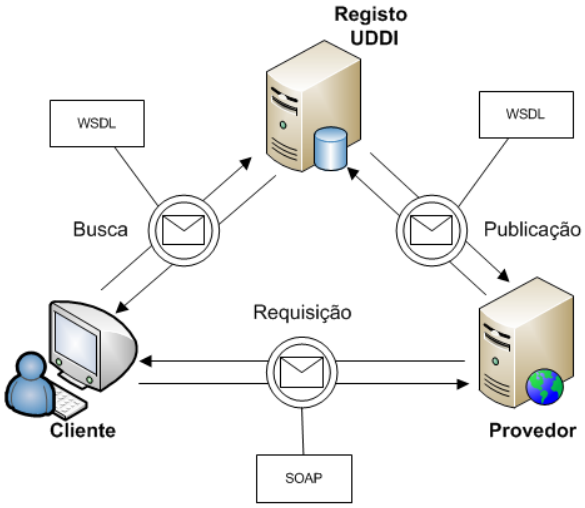

Ilustração 5 – Interação entre os elementos da arquitetura

A implementação de um web service deve estar de acordo com o padrão SOA (acrónimo de *Service-oriented Architecture*), conforme exemplo da figura que se segue.

A arquitetura, pode ser dividida nos seguintes elementos

- Service Consumer
  Procura pelo serviço desejado na base de dados do repositório, o service broker, e utiliza o contrato para estabelecer a ligação com o fornecedor do serviço, o service provider.
- Service Provider
  Disponibiliza o serviço na internet e publica o contrato que descreve a sua interface.
- Service Broker
  Fornece para o cliente as direções necessárias sobre como encontrar o contrato que descreve a interface do serviço.

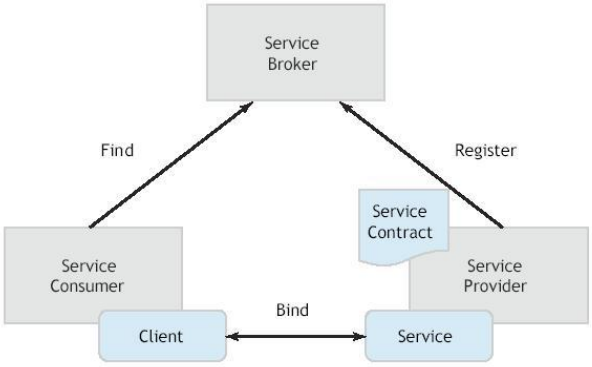

Ilustração 6 - Arquitetura SOA

Funcionalmente, a arquitetura pode-se dividir em três componentes conforme imagem abaixo.

- Transporte – SOAP + XML.
  Para a representação e estruturação dos dados nas mensagens recebidas e enviadas é utilizado o XML. As chamadas às operações, incluindo os parâmetros de entrada/saída, são codificadas no protocolo SOAP (acrónimo de *Simple Object Access Protocol*, baseado em XML).
- Descrição – WSDL.
  É uma linguagem baseada em XML utilizada para descrever os web services, funcionando como um contrato de serviço. Para além da sua descrição, especifica como lhe aceder e quais as operações ou métodos que disponibiliza.
- Procura – UDDI
  (acrónimo de Universal Description, Discovery and Integration). É um serviço onde as empresas podem registar e procurar por serviços.

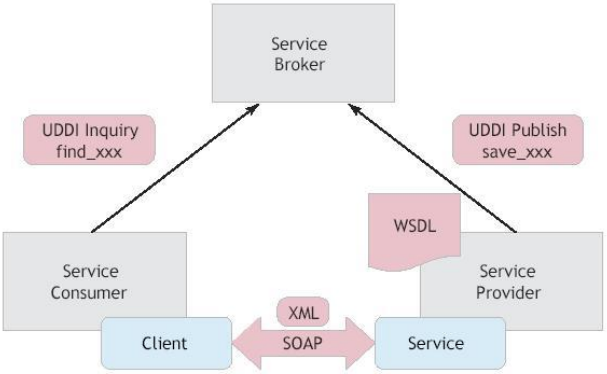

Ilustração 7 - Arquitetura SOA

De seguida podemos ver como a relação existente entre as especificações utilizadas no transporte na descrição e na procura de Web Services.

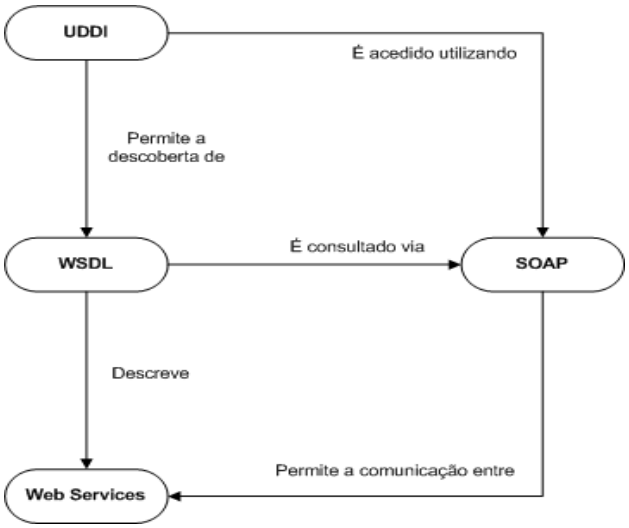

Ilustração 8 - Relação entre as especificações

Foi mostrado funcionalmente qual a estrutura de uma plataforma Web Services, com os seus componentes e tecnologias envolvidas.

Com a representação abaixo temos a perceção da pilha da arquitetura dos Web Services, a estrutura ate aqui apresentada, desta feita, sob a forma de camada de protocolos.

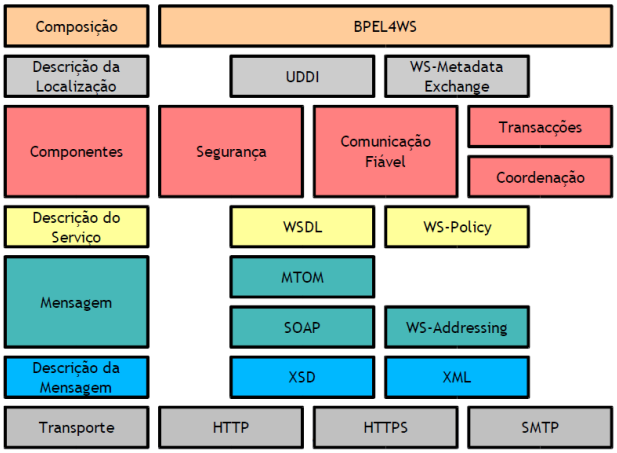

Ilustração 9 - Pilha da arquitetura de um Web Service

5. Tecnologias Envolvidas nos Web Services

5.1. XML

Os sistemas baseados em computadores possuem dados de diferentes tipos. Uma das mais desafiadoras tarefas dos programadores é a troca de informações entre sistemas incompatíveis através da internet. A criação de um documento XML para representar esses dados reduz muito os custos de desenvolvimento e a complexidade, através de um documento que todos os sistemas podem ler. Os dados originais de cada sistema têm de ser mapeados no formato canónico mas depois podem ser usados por todos que sigam o formato.

Como XML apresenta uma maneira eficiente de representar dados através de texto, pode-se utilizar essa tecnologia de forma bastante eficaz para trocas de informação.

O XML deriva de uma linguagem mais antiga para a definição do formato de impressão de documentos, o SGML (acrónimo de Standard Generalized Markup Language) e base no desenvolvimento do HTML (acrónimo de Hyper Text Markup Language).

Essencialmente o XML permite, através de etiquetas ou tags, associar a descrição do formato aos dados de um documento, onde toda a descrição é textual, resolvendo muitos dos problemas de portabilidade.

O XML é uma proposta do W3C e é visto como o substituto dos formatos anteriores de representação de dados como o EDI (acrónimo de Electonic Data Interchange).

Características do XML

- Foi feito para descrever dados independentemente das plataformas.
- Os elementos não são pré-definidos, são criados de acordo com as necessidades do programador.
- É Auto documentado. O próprio formato descreve a sua estrutura e nomes de campos, assim como valores.
- Utiliza um documento de definição de tipos (DTD) para descrever as regras que definem os elementos ou atualmente com XML schema.
- Tecnologia não proprietária

Construção de um XML

- Obrigatório identificar na primeira linha do documento a versão e a codificação.
- Existe um único elemento raiz
- O elemento raiz pode ter vários elementos filhos e os filhos igualmente.
- Todos os elementos têm que fechar.
- Os elementos são fechados pela ordem inversa de abertura.
- Os nomes dos elementos são case-sensitive.

O excerto de código que se segue exemplifica o que poderia ser um documento XML para a listagem de empregados de uma empresa.

```
<?xml version="1.0" encoding ="UTF-8"?>
<employeeList>
    <employee type="contrato">
        <employee_id>100546878</employee_id>
        <name>
            <first_name>Tiago</first_name>
            <last_name>Rosado</last_name>
        </name>
        <extn>239567</extn>
        <dept>PTSI\DOSI</dept>
        <email>o-meu-email@telecom.pt</email>
    </employee>
</employeeList>
```

Ilustração 10 - Exemplo código XML para listagem de empregados

5.2. SOAP

Protocolo de comunicação distribuído que permite o envio de qualquer tipo de informação entre aplicações através dos protocolos que já são um padrão na Web como o http, smtp, ftp. É Um padrão de representação

bastante leve que define o protocolo de pedido resposta, estrutura das mensagens e interação entre cliente e servidor. Baseia-se em XML para descrever o conteúdo da mensagem e a maneira como ela deve ser processada.

Com o protocolo SOAP são definidas um conjunto de regras e especificações, nomeadamente:

- Modelo de encapsulamento
  SOAP Envelope
- Mecanismo de serialização
  SOAP Encoding Rules que são um conjunto de regras para codificação do tipo de dados.
- Chamada de Procedimento Remoto
  SOAP RPC
- Protocolo de transporte
  http Binding

A imagem seguinte representa a estrutura de uma mensagem SOAP. As mensagens são encapsuladas num envelope que pode conter um header (opcional) e o corpo da mensagem, onde pode ser incluída informação como sendo erros que tenham ocorrido aquando uma resposta.

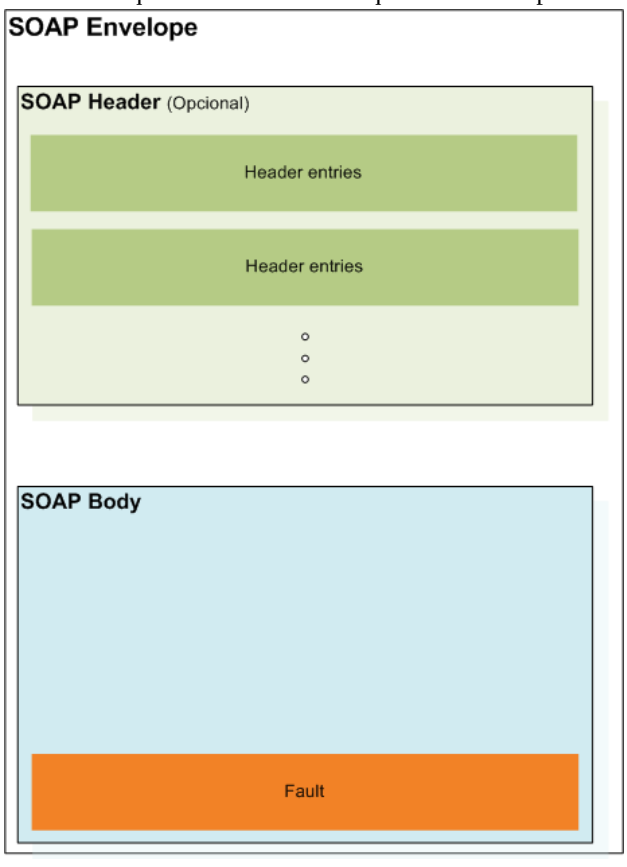

Ilustração 11 - Estrutura de uma mensagem SOAP

Imagem ilustrativa de qual a estrutura de uma mensagem SOAP e os seus componente, nomeadamente:

- Envelope
  Elemento que identifica o documento XML como sendo uma mensagem SOAP.
- Header
  Elemento que contem a informação do cabeçalho.
- Body
  Elemento com os pedidos e as informações de resposta.
- Fault
  Elemento que contem erros e informações de estado.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Ilustração 12 - Modelo de uma mensagem SOAP

Regras de Sintax

- Deve ser escrita em XML
- Deve usar o SOAP Envelope namespace
- Deve usar o namespace de codificação SOAP
- Não pode conter referencia a DTD
- Não pode conter instruções de processamento XML

Exemplo de um pedido SOAP. Neste caso esta a solicitar o valor de mercado das ações da IBM.

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Ilustração 13 - Exemplo de um pedido SOAP

Em resposta, conforme exemplo abaixo, é indicada qual a cotação em causa.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Ilustração 14 - Exemplo de uma resposta SOAP

5.3. WSDL

É uma linguagem baseada em XML que descreve um Web Service e funciona como um contrato de serviço. Para alem da descrição do serviço, descreve como acede-lo e quais as operações e métodos que são disponibilizados. É uma linguagem análoga a um interface Java ou a linguagem de definição de interface CORBA (IDL).

A versão atual do WSDL, a 2.0, tem a estrutura abaixo representada para a definição dos serviços prestados por um web service.

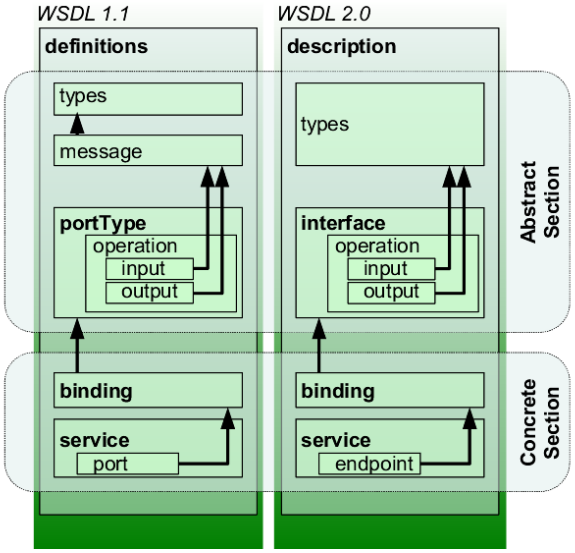

Ilustração 15 - Representação do conceito de documento WSDL na versão 1.1 e versão 2.0.

Esta estrutura é representativa do que é necessário especificar num documento WSDL para que este descreva um web service de modo que qualquer cliente se possa integrar e utilizar o serviço.

A definição permite descrever:

- Qual o serviço.
- Que mensagens devem ser enviadas e qual a sua estrutura.
- Como utilizar os vários protocolos de transporte.
- Onde o serviço esta localizado, qual o endereço para que possa ser acedido.

A mudança entre as versões do WSDL 1.1 e 2.0 ao nível dos elementos que constituem o ficheiro, pode ser verificada na tabela abaixo.

WSDL 1.1	WSDL2.0
Definitions	Description
Types	Types
Message	Não aplicável.
PortType	Interface
Binding	Binding
Service	Service

Tabela 1 - Elementos do ficheiro WSDL para as versões 1.1 e 2.0

De seguida é descrito com mais pormenor os elementos da tabela acima no que concerne a constituição de um ficheiro WSDL da versão 2.0.

- Description

  A imagem abaixo é ilustrativa do elemento num documento WSDL. É a raiz do documento e normalmente começa por identificar os namespaces que serão utilizados no documento.

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns=           "http://www.w3.org/ns/wsdl"
    targetNamespace= "http://oMeuServico.com/MyService"
    xmlns:tns=       "http://oMeuServico.com/MyService"
    xmlns:stns =     "http://oMeuServico.com/MyService/schema"
    xmlns:wsoap=     "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap=      "http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx=     "http://www.w3.org/ns/wsdl-extensions" >
</description>
```

Ilustração 16 - Excerto de um documento WSDL (Description)

- Types

  A imagem abaixo é ilustrativa do elemento num documento WSDL. Descreve os tipos de dados utilizados pelo serviço. Maioritariamente terá um tipo de entrada, um tipo de saída e um tipo em caso de falha. Isto para cada uma das operações mencionadas no web service. Os tipos de dados são normalmente especificados usando XML Schema.

```
<types>
  <xs:schema
      xmlns:xs=       "http://www.w3.org/2001/XMLSchema"
      targetNamespace= "http://oMeuServico.com/MyService/schema"
      xmlns:tns=      "http://oMeuServico.com/MyService/schema"
  >
      <xs:element name="latestTutorialRequest"
                  type="typeLatestTutorialRequest"/>
      <xs:complexType name="typeLatestTutorialRequest">
          <xs:sequence>
              <xs:element  name="date"   type="xs:date"/>
          </xs:sequence>
      </xs:complexType>
      <xs:element name="latestTutorialResponse" type="xs:string"/>
      <xs:element name="invalidDateError" type="xs:string"/>
  </xs:schema>
</types>
```

Ilustração 17 - Excerto de um documento WSDL (Types)

- Interface

  A imagem abaixo é ilustrativa do elemento num documento WSDL. É o núcleo do web service pois é onde são definidas as operações a ser executadas pelo serviço e as mensagens envolvidas para o efeito. Cada operação representa uma interação entre o cliente e o serviço, e em comparação com linguagens de programação, são semelhantes a métodos ou procedimentos.

```
  <interface  name = "latestTutorialInterface" >
    <fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
    <operation name="latestTutorialOperation"
               pattern="http://www.w3.org/ns/wsdl/in-out"
               style="http://www.w3.org/ns/wsdl/style/iri"
               wsdlx:safe = "true">
    <input    messageLabel="In"  element="stns:latestTutorialRequest" />
    <output   messageLabel="Out" element="stns:latestTutorialResponse" />
    <outfault messageLabel="Out" ref    ="tns:invalidDateFault" />
  </operation>
</interface>
```

Ilustração 18 - Excerto de um documento WSDL (Interfaces)

- Binding

  A imagem abaixo é ilustrativa do elemento num documento WSDL. Este elemento contém as informações sobre o formato e detalhes do protocolo utilizado para cada operação.

```
<binding name="latestTutorialSOAPBinding"
       interface="tns:latestTutorialInterface"
       type="http://www.w3.org/ns/wsdl/soap"
       wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
  <fault ref="tns:invalidDateFault" wsoap:code="soap:Sender"/>
  <operation ref="tns:latestTutorialOperation"
             wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
</binding>
```

Ilustração 19 - Excerto de um documento WSDL (Binding)

- Service

  A imagem abaixo é ilustrativa do elemento num documento WSDL. O elemento service descreve o endpoint do web service, ou seja, qual o endereço pelo qual este pode ser acedido.

```
<service
      name     ="latestTutorialService"
      interface="tns:latestTutorialInterface">

    <endpoint name ="latestTutorialEndpoint"
          binding ="tns:latestTutorialSOAPBinding"
          address ="http://oMeuServico.com/FazQQcoisa"/>

</service>
```

Ilustração 20 - Excerto de um documento WSDL (Service)

5.4. UDDI

UDDI (acrónimo de *Universal Description, Discovery and Integration*) é um serviço de diretório onde os sistemas podem registar e procurar por serviços. Esta tecnologia é uma framework para disponibilizar, utilizar e pesquisar por serviços na internet. É um interface de serviço descrito em WSDL e é parte integrante da plataforma .NET.

Foi lançado no mercado em Setembro de 2000 pela Microsoft, IBM e Ariba, representando atualmente não só uma proposta de padronização, como um consórcio de apoiantes do projeto. Foi lançado na versão 1.0 e já se encontra na versão 3.0.

Os registos UDDI, conceptualmente, podem ser divididos em quatro categorias:

- Technical models
- Businesses
- Businesses services
- Service bindings

A figura abaixo ilustra o modo como a base de dados UDDI é preenchida e de que modo os utilizadores encontram e utilizam a informação.

- Empresas, organismos de normalização e programadores definem os modelos dos registos com descrições de diferentes tipos de serviços, comummente chamados de modelos técnicos ou tModels.
- Empresas preenchem os registos com descrições dos serviços que prestam.
- O serviço de registo UDDI atribui um identificador único para cada serviço bem como para cada empresa registada.
- Mercados, motores de busca, e aplicações de negócio consultam o registo para descobrir serviços noutras empresas.
- Empresas utilizam essa informação para facilitar a integração de serviços através da internet.

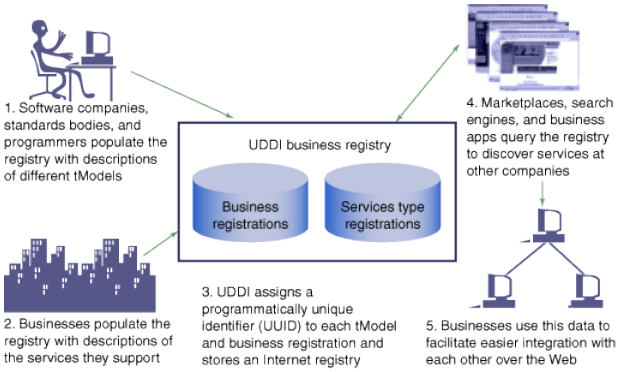

Ilustração 21 - Modo de funcionamento UDDI

A imagem seguinte é representativa do modelo de dados de um documento UDDI.

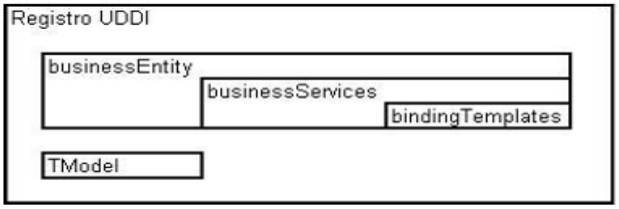

Ilustração 22 - Estrutura de um Registo UDDI

A imagem acima representa o modo como os elementos da estrutura de dados de um documento UDDI estão interligados.

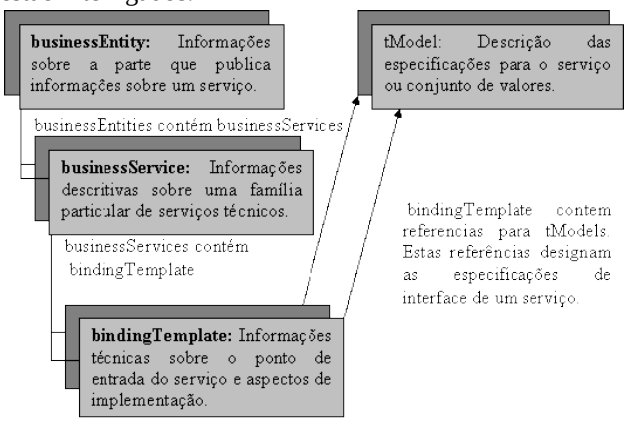

Ilustração 23 - Modelo de dados UDDI

- Business Entity
  Páginas Brancas. Uma empresa prestadora de serviços. Contem informações sobre nomes, endereços, números de telefone, entre outros, acerca dos fornecedores de serviço
  O documento em XML, concetualmente, deverá obedecer á estrutura abaixo representada pela imagem.

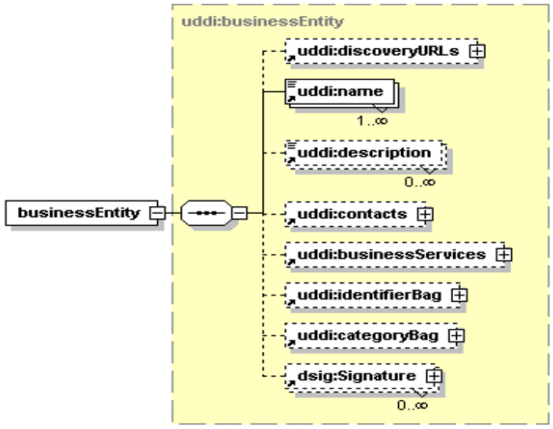

Ilustração 24 - Diagrama da estrutura do businessEntity

A imagem abaixo representa parte de um documento UDDI escrito em XML onde está definido o businessEntity.

```
<businessEntity
    businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40"
    operator="http://www.ibm.com"
    authorizedName="John Doe">

    <name>Acme Company</name>

    <description>
        We create cool Web services
    </description>

    <contacts>
        <contact useType="general info">
            <description>General Information</description>
            <personName>John Doe</personName>
            <phone>(123) 123-1234</phone>
            <email>jdoe@acme.com</email>
        </contact>
    </contacts>

    <businessServices>
        ...
        ...
        ...
    </businessServices>

    <identifierBag>
        <keyedReference
            tModelKey="UUID:8609C81E-EE1F-4D5A-B202-3EB13AD"
            name="D-U-N-S"
            value="123456789" />
    </identifierBag>

    <categoryBag>
        <keyedReference
            tModelKey="UUID:C0B9FE13-179F-413D-8A5B-5004DB8"
            name="NAICS"
            value="111336" />
    </categoryBag>
</businessEntity>
```

Ilustração 25 - Exemplo XML do businessEntity

- Business Service
  Páginas Amarelas. Serviços prestados por uma dada empresa. Contem listagens comerciais baseadas nos tipos desses negócios, de maneira organizada por categoria ou regiões demográficas. O documento em XML, concetualmente, deverá obedecer á estrutura abaixo representada pela imagem.

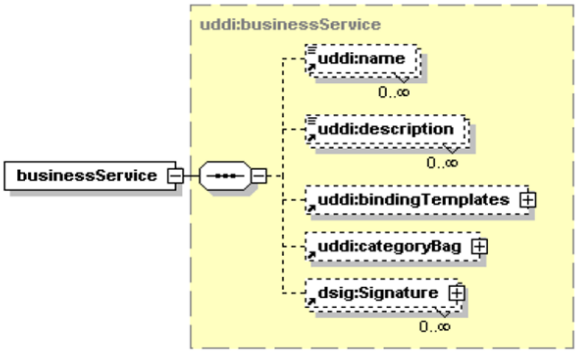

Ilustração 26 - Diagrama da estrutura do businessService

A imagem abaixo representa parte de um documento UDDI escrito em XML onde está definido o businessService.

```
<businessService
    serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
    businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">

    <name>Hello World Web Service</name>

    <description>A friendly Web service</description>

    <bindingTemplates>
        ...
        ...
        ...
    </bindingTemplates>

    <categoryBag />
</businessService>
```

Ilustração 27 - Exemplo XML do businessService

- Binding Template
  Páginas Verdes. Disponibiliza a informação necessária para aceder a um determinado serviço. O documento em XML, concetualmente, deverá obedecer á estrutura abaixo representada pela imagem.

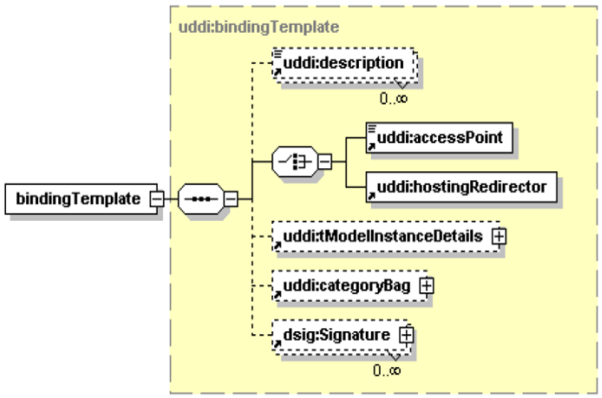

Ilustração 28 - Diagrama da estrutura do bindingTemplate

A imagem abaixo representa parte de um documento UDDI escrito em XML onde está definido o bindingTemplate.

```
<bindingTemplate
    serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
    bindingKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">

    <description>Hello World SOAP Binding</description>

    <accessPoint URLType="http">
        http://localhost:8080
    </accessPoint>

    <tModelInstanceDetails>
        <tModelInstanceInfo
            tModelKey="uuid:EB1B645F-CF2F-491f-811A-4868705F5904">
            <instanceDetails>
                <overviewDoc>
                    <description>
                        references the description of the
                        WSDL service definition
                    </description>
                    <overviewURL>
                        http://localhost/helloworld.wsdl
                    </overviewURL>
                </overviewDoc>
            </instanceDetails>
        </tModelInstanceInfo>
    </tModelInstanceDetails>
</bindingTemplate>
```

Ilustração 29 - Exemplo XML do bindingTemplate

- Technical Model (tModel)
  É uma maneira de descrever os vários negócios, serviço e estruturas de modelos armazenados no registro UDDI. Qualquer conceito abstrato relacionado com a caraterização de um serviço pode ser registado no UDDI como um tModel. O documento em XML, concetualmente, deverá obedecer á estrutura abaixo representada pela imagem.

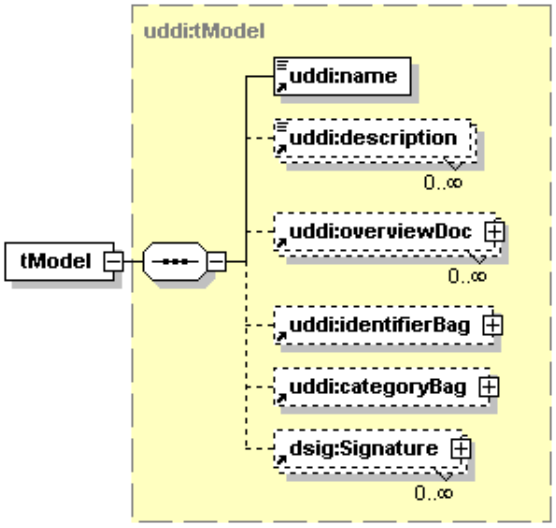

Ilustração 30 - Diagrama da estrutura do Technical Model (tModel)

A imagem acima representa parte de um documento UDDI escrito em XML onde está definido o tModel. Neste caso em concreto representa a interface do HelloWorldInterface.

```
<tModel tModelKey="uuid:xyz987..."
        operator="http://www.ibm.com"
        authorizedName="John Doe">

    <name>HelloWorldInterface Port Type</name>

    <description>
        An interface for a friendly Web service
    </description>

    <overviewDoc>
        <overviewURL>
            http://localhost/helloworld.wsdl
        </overviewURL>
    </overviewDoc>
</tModel>
```

Ilustração 31 - Exemplo XML do Technical Model (tModel)

Os elementos constituintes de um registo UDDI acabaram de ser explicados e resumindo temos.

- businessEntity
- businessService
- bindingTemplate
- tModel

Acontece que muitas empresas e serviços, não são completa e devidamente representados por um único registo UDDI, ou businessEntity. Nestes casos, como por exemplo empresas com varias filiais, fornecedores diferentes dependendo da zona ou pais, torna-se necessário mais de um registo para clarificar o funcionamento e o serviço que a empresa presta, vista como um todo. A associação de vários registos da mesma entidade faz-se com recurso à estrutura publisherAssertion.

O documento em XML, concetualmente, deverá obedecer á estrutura abaixo representada pela imagem.

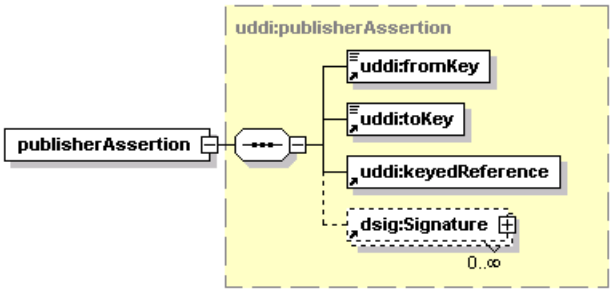

Ilustração 32 - Diagrama da estrutura do publisherAssertion

A imagem abaixo representa parte de um documento UDDI escrito em XML onde está definido o publisherAssertion que neste caso indica qual a relação entre duas empresas.

```
<publisherAssertion>

    <!-- Specify Tokyo Traders' businessKey as fromKey-->
    <fromKey>
        uddi:tokyotraders.example:business
    </fromKey>

    <!-- Specify Chiba Traders businessKey as toKey-->
    <toKey>
        uddi:chibatraders.example:business
    </toKey>

    <!--Specify a subsidiary rel using uddi-org:rel -->
    <keyedReference
        keyName="subsidiary"
        keyValue="parent-child"
        tModelKey="uddi:uddi.org:relationships"/>
</publisherAssertion>
```

Ilustração 33 - Exemplo XML do publisherAssertion

A imagem resume a arquitetura dos web services e os seus recursos, focando neste caso o registo UDDI.

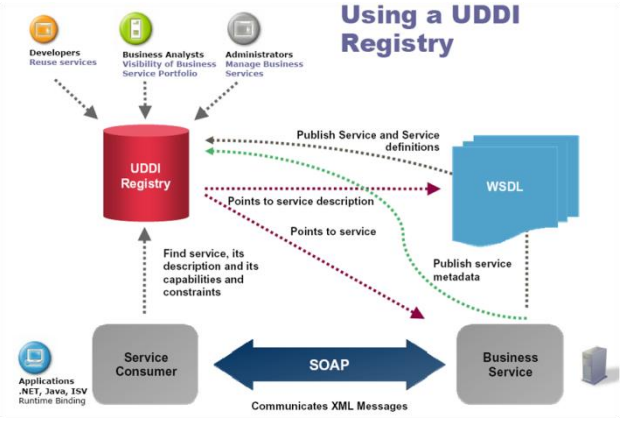

Ilustração 34 - Registo UDDI

Os servidores que mantêm todo o sistema de registos UDDI podem ser classificados segundo três categorias, nomeadamente:

- Node
  Servidor UDDI que suporta um número mínimo de funcionalidades definidas nas especificações.
- Registry
  Servidor composto por um ou mais nodes. Um registry realiza o conjunto completo de funcionalidades conforme definido nas especificações.
- Affiliated Registries
  Registries individuais que implementam uma política baseada em partilha de informação entre eles.

Com a nova versão UDDI 3.0 os registos podem ser de três tipos de acesso diferentes, nomeadamente:

- Privado
  É um registo interno, protegido por uma firewall, isolado da rede pública. O acesso as tarefas administrativas e aos dados é de forma segura.
- Semiprivado
  É um registo desenvolvido dentro de um ambiente controlado. O acesso para o exterior é partilhado apenas com entidades confiáveis.
- Público
  As funções administrativas continuam a ser efetuadas de forma segura, mas o acesso aos dados dos registos é essencialmente aberto e público.

A imagem acima representa a interação entre os vários servidores UDDI, dentro e fora do mesmo nível de acesso.

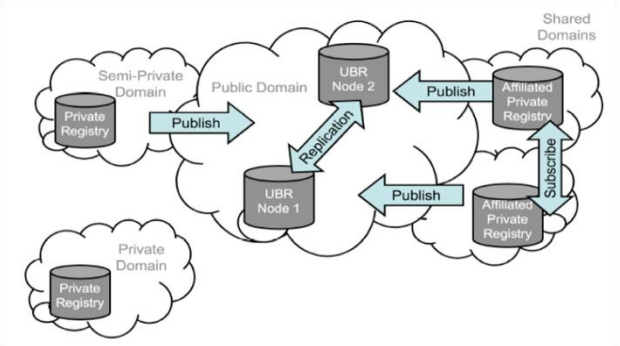

Ilustração 35 - Relação entre servidores UDDI

6. Segurança em Web Services

As empresas temiam a divulgação de serviços na internet devido á possibilidade de exporem os seus dados confidenciais a terceiros. A introdução dos web services veio trazer segurança no sentido em que cria uma barreira entre a base de dados com a informação e a efetiva disponibilização dos serviços que a empresa possa fornecer com base nessa informação, no entanto o fato de todas as suas operações estarem baseadas na Internet torna os Web Services muito sensíveis quanto à segurança.

Agentes não-autorizados podem intercetar e modificar dados no tráfego da rede, interromper o funcionamento de serviços, utilizar recursos sem permissão. Por esse motivo, conceitos, mecanismos e padrões de implementação de segurança devem ser aplicados, para que a arquitetura de Web Services seja considerada confiável no que tange à segurança da informação.

Quando falamos de segurança, há vários pontos fulcrais que é necessário ter em consideração, nomeadamente:

- Autenticação
  Validação das credenciais do cliente junto a uma entidade certificadora. Saber quem enviou a informação.
- Autorização
  Após autenticação, saber se um cliente tem acesso ou não a um determinado conteúdo. Saber o que o cliente pode fazer.
- Confidencialidade
  Garantia de que a informação guardada ou transmitida não são vistas ou alteradas por terceiros. Saber quem pode ler a mensagem.
- Integridade
  Certificação de que a informação transmitida não é alterada acidental ou maliciosamente no caminho entre cliente e servidor. Saber se alguém alterou a informação transmitida.

Para que a comunicação seja considerada segura, devem ser implementadas medidas em vários níveis como sendo as camadas de rede, transporte e aplicacional. Os mecanismos de segurança que podem ser implementados estão descritos de seguida.

Segurança na camada de rede

- Firewall
- VPN

Segurança na camada de transporte

- Certificados digitais
- PKI
- SSL

Segurança na camada de aplicação

- Nativa dos Sistemas Operativos
- IP e DNS
- HTTP
  - Basic Authentication
  - Basic Authentication com SSL
  - Digest Authentication
  - Integrated Windows Authentication
  - Client Certification Authentication
- ASP .NET
  - Forms Authentication
  - Passport Authentication
  - Controlo de acesso (webconfig)
- XML
  - XML Signature
  - XML Encryption
  - SAML
  - XKMS
  - WS-Security

No sentido de responder à necessidade de segurança mais abrangente, a IBM, Microsoft e a VeriSign criaram uma família de especificações, a WS-* que aborda a segurança, aproveitando os padrões e especificações já existentes.

A imagem abaixo representa a família de especificações de segurança para web services, conhecido como WS-*.

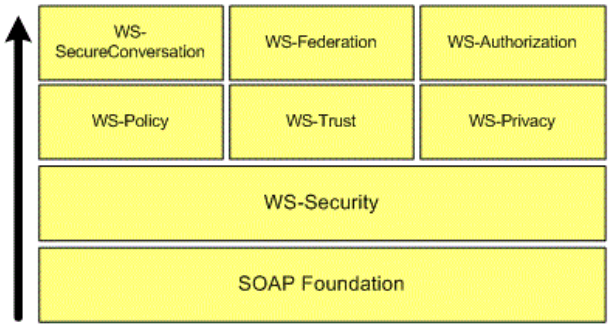

Ilustração 36 - Modelo de segurança para Web Services

É sabido que o SOAP é o método para transmitir mensagens entre dois pontos através de http e que através do http podemos autenticar o cliente, assinar a mensagem e encriptar o conteúdo. A segurança oferecida pelo protocolo http é de ponto a ponto e quando se coloca a questão da necessidade de vários saltos até se atingir o objetivo, a identidade, integridade e segurança da mensagem pode ser colocadas em causa.

WS-Security trata de como manter um contexto seguro ao longo de um caminho multiponto.

WS-Security é uma Framework, que incorpora mecanismos de segurança existentes, nas mensagens SOAP de modo a que estas sejam transmitidas de forma mais segura.

A arquitetura pode ser vista na imagem que se segue.

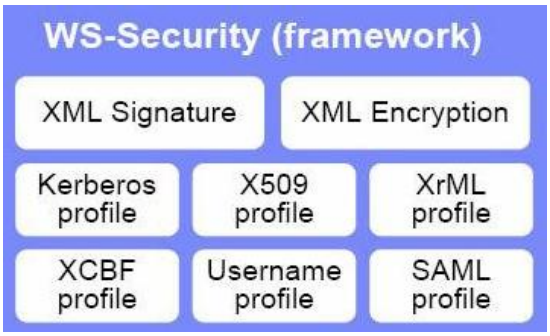

Ilustração 37 - WS-Security framework

WS-Security não especifica o formato de assinatura ou criptografia, em vez disso, especifica como se pode incorporar as informações de segurança estabelecidas por outras especificações dentro de uma mensagem SOAP.

A implementação das diretivas WS-Security começa no elemento Header de uma mensagem, sendo nele que se definem as extensões do SOAP que implementam:

- Autenticação do cliente
  Identificação do cliente e de que forma ele prova a sua identidade por forma a dar as devidas autorizações de acesso a informação ou serviços. Para tal é utilizado o Security Token, podendo ser definido com recurso a:
  - Nome de utilizador e palavra-chave
  - Tickets Kerberos
  - Certificados X.509
- Integridade da mensagem
  Assegura que a mensagem não é adulterada durante a transmissão. Para tal é utilizada encriptação com recurso a XML signature. É utilizada para verificar a origem da resposta de um web service ou para verificar que a mensagem não foi alterada desde que foi assinada.
  É gerada uma chave com base em transformações efetuadas ao XML do documento, segundo algoritmos específicos para o efeito, como MD5 ou SHA1.

- Confidencialidade da mensagem
  Assegura que a mensagem não é lida durante a transmissão. Para tal é utilizada encriptação com recurso a XML encryption. A encriptação da mensagem é feita com recurso a chaves simétricas, partilhadas pelo emissor e recetor, ou incluídas no corpo da mensagem. Qualquer bloco de código XML de uma mensagem SOAP pode ser encriptado.

Os elementos de segurança definidos pelo WS-Security e implementados nas mensagens SOAP, conforme visto, podem ser identificados na imagem que se segue.

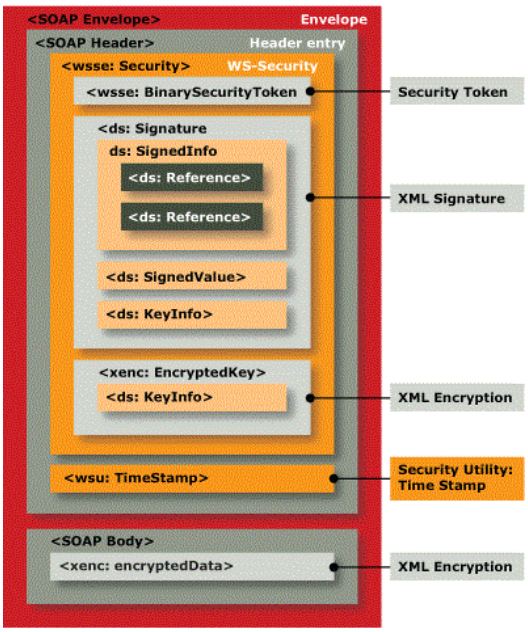

Ilustração 38 - Mensagem SOAP com implementação de segurança

Para o exemplo que se segue é mostrado como cada uma das partes do documento se interligam, mostrando alguns dos conceitos de segurança implementados pelo WS-Security, mais concretamente, o tipo de autenticação utilizada e como assinar uma mensagem.

No elemento Security Token está identificado o tipo de segurança binaria com recurso aos certificados X.509. De seguida, para o elemento XML Signature podemos ver uma referência na propriedade SignedInfo que identifica o que é para ser assinado ou encriptado ou ambos, e nesta situação é o MyBody.

A imagem que se segue é um excerto de código XML de uma mensagem com autenticação e assinatura, implementados sobre as diretivas da framework WS-Security.

```
    <wsse:Security>
      <wsse:BinarySecurityToken
                ValueType="...#X509v3"
                EncodingType="...#Base64Binary"
                wsu:Id="X509Token">
              MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm=
                "http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod Algorithm=
                "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#myBody">
            <ds:Transforms>
              <ds:Transform Algorithm=
                    "http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm=
                  "http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>EULddytSo1...</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          BL8jdfToEb11/vXcMZNNjPOV...
        </ds:SignatureValue>
        <ds:KeyInfo>
            <wsse:SecurityTokenReference>
                <wsse:Reference URI="#X509Token"/>
            </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="myBody">
    <tru:StockSymbol xmlns:tru="http://www.fabrikam123.com/payloads">
      QQQ
    </tru:StockSymbol>
  </S11:Body>
</S11:Envelope>
```

Ilustração 39 - Exemplo de uma mensagem segundo ws-security

7. Criar Web Services no Visual Studio 2010

7.1. Criar um Web Service

Para a criação de um Web Service podemos recorrer à plataforma de desenvolvimento, Microsoft Visual Studio, neste caso a versão utilizada foi a 2010.

Vamos criar um projeto novo, seguindo os passos:

- File
- New Project

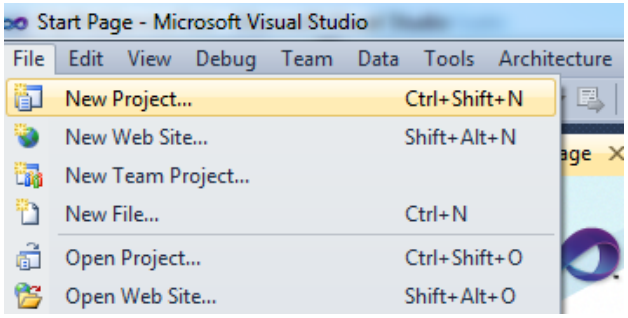

Ilustração 40 - Criar um Web Service (Passo 1)

De seguida é-nos mostrada uma caixa com varias informações, das quais necessitamos de selecionar:

- À esquerda (Installed Templates), escolher a linguagem C# e a subopção Web.
- Em cima, selecionar a Framework 3.5.

- No centro escolher então o projeto pretendido, que neste caso é ASP .NET Web Service Application.
- Escolher a diretoria onde guardar o projeto.
- Escolher o nome que se pretende para o projeto.

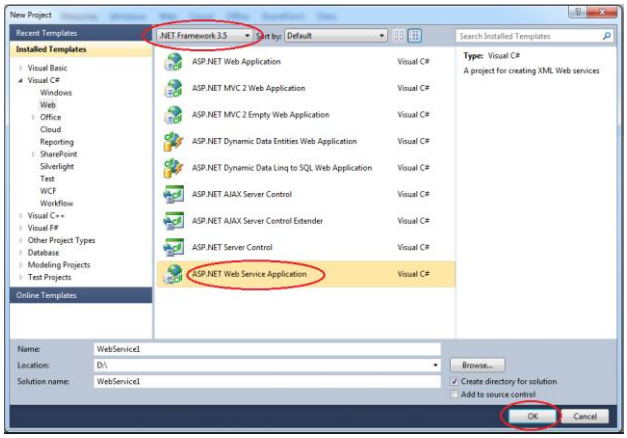

Ilustração 41 - Criar um Web Service (Passo 2)

Seguindo estes passos, o Microsoft Visual Studio 2010 automaticamente gera o código básico para um weeb service, que simplesmente retorna uma frase a dizer "Hello World".

Por forma a termos uma interação com o web service, vamos acrescentar código que recebendo dois números inteiros, faça a sua soma e retorne o total.

No ficheiro de código "*MyWebService.asmx.cs*" adicionar o seguinte excerto de código.

```
WebMethod()]
public int Add(int a, int b)
{
        return a + b;
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace MyWebService
{
    /// <summary>
    /// Summary description for Service1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // [System.Web.Script.Services.ScriptService]
    public class MyWebService : System.Web.Services.WebService
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }

        [WebMethod()]
        public int Add(int a, int b)
        {
            return a + b;
        }
    }
}
```

Ilustração 42 - Exemplo do código MyWebService.asmx.cs

Neste momento, ao executarmos o projeto acabado de criar, já podemos ver que existem dois métodos disponibilizados pelo web service.

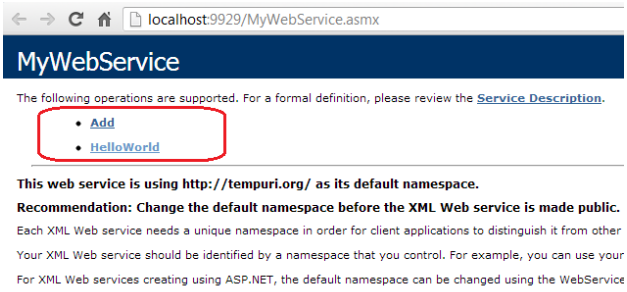

Ilustração 43 - execução do projeto MyWebService

Ao carregar na opção "Add" o próprio web service disponibiliza um interface para testarmos os seus serviços, bem como o código SOAP de troca das mensagens.

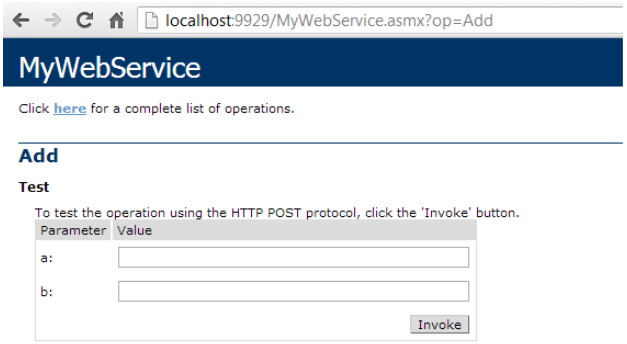

Ilustração 44 - Serviço ADD

7.2. Criar um cliente para o Web Service

A ideia dos web services é termos um cliente que utilize os serviços que estes disponibilizam na internet. Como tal, vamos criar um cliente por forma a aceder a este serviço que soma dois números.

Vamos utilizar novamente a ferramenta Microsoft Visual Studio, mas desta vez para a criação de uma página web, seguindo os passos:

- File
- New Web Site

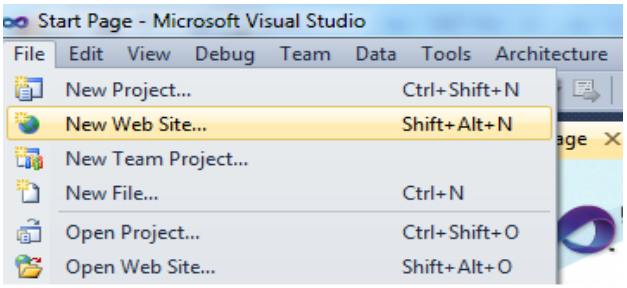

Ilustração 45 - Criar um Web Site (Passo 1)

De seguida é-nos mostrada uma caixa com varias informações, das quais necessitamos de selecionar:

- À esquerda (Installed Templates), escolher a linguagem C#.
- No centro escolher então o projeto pretendido, que neste caso é ASP .NET Empty Web Site.
- Escolher a diretoria onde guardar o projeto.
- Escolher o nome que se pretende para o projeto.

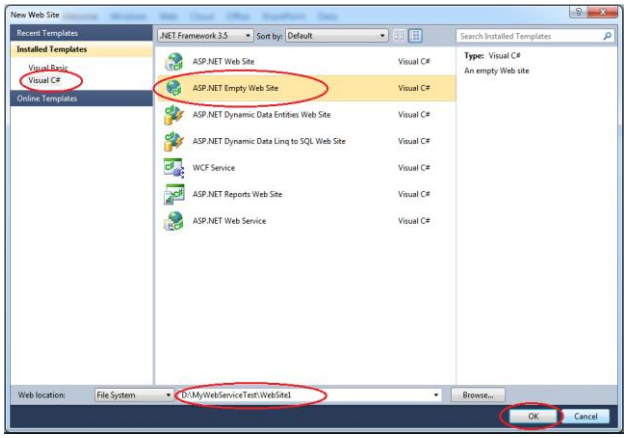

Ilustração 46 - Criar um Web Site (Passo 2)

Com o botão direito do rato selecionamos "Add New Item…" por forma a ser adicionado um ficheiro à nossa página de internet.

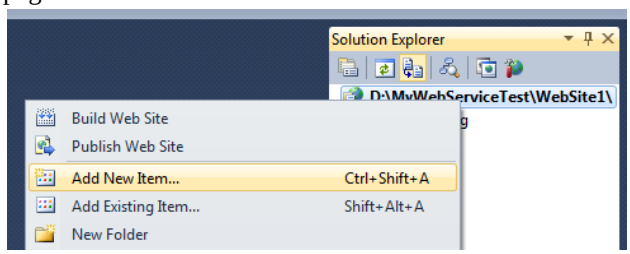

Ilustração 47 - Criar um Web Site (Passo 3)

Escolhemos o ficheiro "Web Form" e em baixo damos o nome pretendio. No nosso caso como não temos mais nenhum, vamos deixar o que esta por defeito, "Default.aspx".

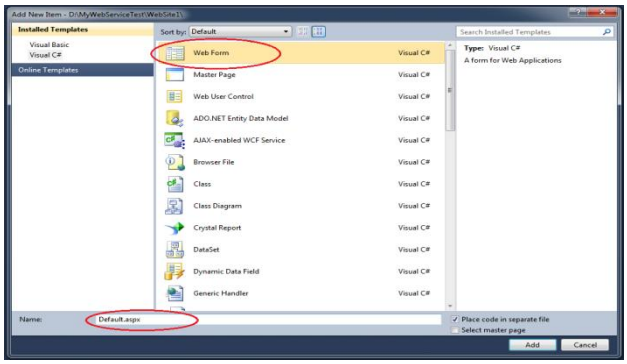

Ilustração 48 - Criar um Web Site (Passo 4)

Na página acabada de criar, vamos adicionar o seguinte excerto de código

```
<form id="Form1"
action="http://localhost:9929/MyWebService.asmx/Add"
method="POST"
runat="server">

        <input name="a"/>
        <input name="b"/>
        <input type="submit" value="Enter"
        onclick="btnEnterClic"/>
</form>
```

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
    Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>

    <form id="Form1"
        action="http://localhost:9929/MyWebService.asmx/Add"
        method="POST"
        runat="server">

        <input name="a"/>

        <input name="b"/>

        <input type="submit" value="Enter" onclick="btnEnterClic"/>

    </form>

</body>
</html>
```

Ilustração 49 - Exemplo do código Default.aspx

Ao executarmos a página web temos um formulário para colocar dois números.

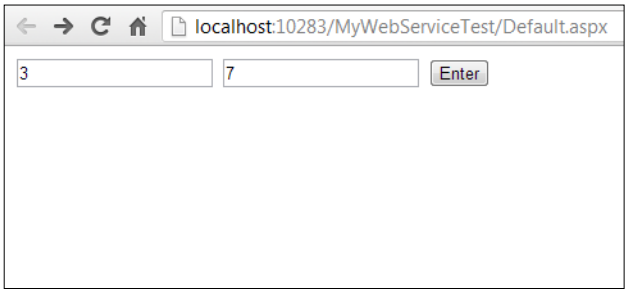

Ilustração 50 - Exemplo da página web

Como resultado do serviço prestado pelo Web Service, temos o resultado do somatório dos dois números.

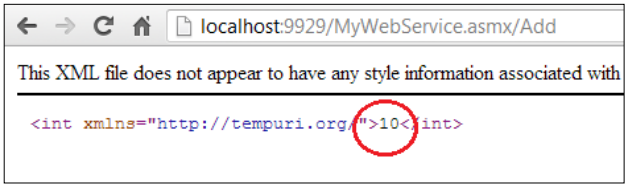

Ilustração 51 - Resposta do web service

Referências

- http://www.braziloutsource.com/wss2.html
- [RPC]http://pt.wikipedia.org/wiki/Chamada_de_procedimento_remoto
- [CORBA]http://pt.wikipedia.org/wiki/CORBA
- [IDL]http://en.wikipedia.org/wiki/Interface_description_language
- [ORB]http://en.wikipedia.org/wiki/Object_request_broker
- http://pt.wikipedia.org/wiki/Web_service
- http://en.wikipedia.org/wiki/Web_service
- http://www.gta.ufrj.br/grad/05_1/webservices/definicao.htm
- [XML]http://www.gta.ufrj.br/grad/05_1/webservices/xml.htm
- [SOAP]http://www.gta.ufrj.br/grad/05_1/webservices/soap.htm
- [WSDL]http://www.gta.ufrj.br/grad/05_1/webservices/wsdl.htm
- [UDDI]http://www.gta.ufrj.br/grad/05_1/webservices/uddi.htm
- [Programming.NET Web Services, O'Reilly] http://oreilly.com/catalog/prognetws/chapter/ch02.html
- http://msdn.microsoft.com/en-us/library/yzbxwf53.aspx
- http://uddi.org/pubs/uddi_v3.htm#_Toc85908023
- http://msdn.microsoft.com/pt-br/library/bb507204.aspx
- http://www.tutorialspoint.com/uddi/index.htm
- http://en.wikipedia.org/wiki/Web_Services_Description_Language
- http://www.w3schools.com/webservices/ws_platform.asp
- http://www.w3schools.com/webservices/ws_example.asp
- http://www.w3schools.com/webservices/ws_use.asp
- http://www.w3schools.com/webservices/ws_summary.asp
- [Segurança]http://www.tutorialspoint.com/webservices/web_services_security.htm
- [Segurança]http://pt.wikipedia.org/wiki/Web_service#Seguran.C3.A7a
- [Segurança]http://msdn.microsoft.com/en-us/library/ms977312.aspx
- [Segurança]http://www.microsoft.com/en-us/download/details.aspx?id=14089
- [Segurança]http://www.ibm.com/developerworks/webservices/library/ws-security.html
- [Segurança]http://www.ibm.com/developerworks/webservices/library/ws-security/
- [Segurança]http://www.codeproject.com/Articles/7062/An-introduction-to-Web-Service-Security-using-WSE
- [Segurança]http://www.codeproject.com/Articles/7275/An-Introduction-To-Web-Service-Security-Part-II
- Erin Cavanaugh, Altova WhitePaper - Web services: Benefits, challenges, and a unique, visual development solution, 2006
- Chad Vawter, Ed Roman, J2EE vs. Microsoft.NET A comparison of building XML-based web services, Junho 2001
- Doug Tidwell, James Snell, Pavel Kulchenko, Programming Web Services with SOAP, 1ª Edição, Dezembro 2001
- Peter Swithinbank, Francesca Gigante, Hedley Proctor, Mahendra Rathore, William Widjaja, WebSphere and .Net Interoperability Using Web Services, 1ª Edição, Junho 2005