Web Service de Consulta de CEP e Rastreamento de Encomendas dos Correios
Exercício Computacional I - Sistemas Distribuídos

Rafael Gonçalves de Oliveira Viana[1]
9 de outubro de 2017

[1]Sistemas de Informação – Universidade Federal do Mato Grosso do Sul (UFMS)
Caixa Postal 79400-000 – Coxim – MS – Brazil

rafael.viana@aluno.ufms.br

Resumo. *Este relatório introduz a arquitetura de um Web Service assim como seus componentes, o mesmo relata como foi implementado um Web Server que possui uma página web e serviços para busca de cidade e bairro através do CEP e rastreamento de encomendas dos correios.*

1. Introdução

Com a disseminação de dispositivos com conexão a Internet na sociedade atual, houvesse a necessidade de criar/gerenciar serviços web, estes necessarios para atender as mais diversas funções como exemplo, salas de bate-papo, e-commerces e serviços meteorológicos entre outros.

Este relatório apresenta brevemente a arquitetura de um *Web Service* e como foi implementado, serviços de consulta de cidade e barrio através do CEP e rastreamento de encomendas dos correios, juntamente com uma página web(rafaelbuscas.ddns.net) e um aplicativo Android que oferece uma interface para os serviços disponíveis pelo *Web Server* Rafael Buscas.

O código deste relatório está em sua totalidade no endereço http://github.com/rafaelgov95/SD/Projeto-Correios-CEP, e o mesmo pode ser reutilizado sobre a licença MIT.

2. Web Service

Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes, como pode ser objservado na figura 1, um *Data Server*, disponibiliza serviços para os mais diversos tipos de dispositivos, em diferentes formatos de objetos.

Com está tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. [1].

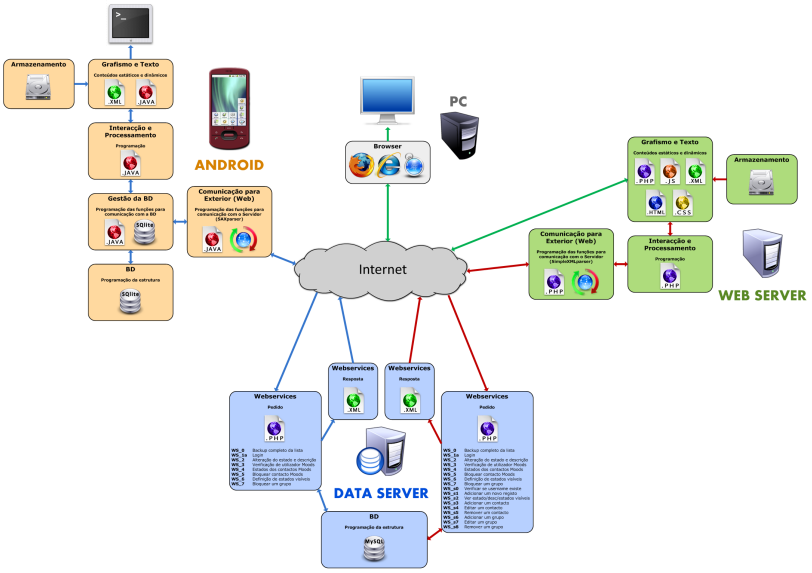

Figura 1. Sistema implantando, fornecendo serviços para PCs, dispositivos móveis e outros web servers.

Cada dia existe mais tipos de protocolos porém existem 4 tipo que se destacam: Serviço de Transporte - ”FTP”, Mensgens ”XML”, Descrição de Serviço ”WSDL”e Descoberta de Serviço ”UDDI”.

Podemos observar na figura 2, a pilha de soluções com a pilha de tecnologia correspondente para cada camada.

1. **Descoberta do serviço:** Responsável por centralizar os serviços em um registro comum e fornecer funcionalidades fáceis de publicação/pesquisa. Atualmente, a descoberta do serviço é tratada através de Descrição Universal, Descoberta e Integração (UDDI).

2. **Descrição do Serviço:** Responsável por descrever a interface pública para um serviço web específico. Atualmente, a descrição do serviço é tratada através do Web Service Description Language (WSDL).

3. **Mensagens XML:** Responsável por codificar mensagens em um formato XML comum para que as mensagens possam ser entendidas em cada uma das extremidades. Atualmente, esta camada inclui XML-RPC e SOAP.

4. **Serviço de transporte:** Responsável pelo transporte de mensagens entre aplicativos. Atualmente, esta camada inclui o protocolo de transporte de hipertexto (HTTP), protocolo de transferência de correio simples (SMTP), protocolo de transferência de arquivos (FTP) e protocolos mais recentes, como o protocolo de intercâmbio extensível de blocos (BEEP).

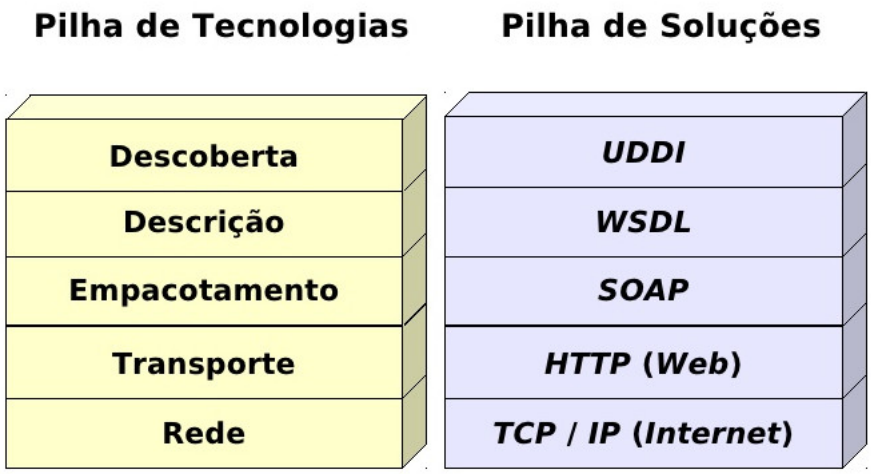

Figura 2. Pilha de protocolo de transporte e suas tecnologias.

2.1. Tipos de Web Services

O mesmo pode ser publicado na intranet ou na Internet, provendo três possiveis formatos de serviços: Provedor de Serviço, Solicitante de Serviço e o Registro de Serviço. Para mais detalhes desta sessão incentivo a consultar a referência [2].

1. **Provedor de serviço:** Fornece serviços web. O provedor de serviços implementa o serviço e disponibiliza-o na Internet ou intranet.
2. **Solicitante de Serviço:** Este é um consumidor do serviço web. O solicitante utiliza um serviço da Web existente abrindo uma conexão de rede e enviando uma solicitação XML.
3. **Registro de serviço:** Este é um diretório de serviços logicamente centralizado. O registro fornece um lugar central onde os desenvolvedores podem publicar novos serviços ou encontrar os existentes. Ele serve como centro de compensação centralizado para empresas e seus serviços.

2.2. Componentes de um *Web Service*

Ao longo dos últimos anos, diversas tecnologias emergiram como padrões mundiais que constituem o núcleo da tecnologia de serviços da Web de hoje. Algumas destas tecnologias e suas caracteristicas são demostradas abaixo.

2.2.1. WSDL - *Web Services Description Language*

O WSDL é um idioma baseado em XML, para descrever os serviços da Web e como acessá-los, nas sessões 3.2 e 3.3.1, será apresentado dois WSDL dos correios, com diversos serviços.

1. É um protocolo baseado em XML, para troca de informações em ambientes descentralizados e distribuídos.
2. É o formato padrão para descrever um serviço web.

3. Descreve como acessar um serviço da Web e quais as operações que ele executará.
4. Descrever como se relacionar com serviços baseados em XML.
5. É o idioma que UDDI utiliza.

O WSDL tem um grande papel na arquiteutura de um *Web Service*, ele divulga e expõem os serviços presentes.

2.2.2. SOAP - *Simple Object Access Protocol*

O SOAP é um protocolo baseado em XML para trocar informações entre computadores.

O mesmo não está vinculado a nenhum protocolo de transporte específico. Na verdade, você pode usar SOAP via HTTP, SMTP, FTP ou BEEP. Algumas caracteriscas estão presentes no SOAP elas são:

1. Será desenvolvido como um padrão W3C.
2. Protocolo de comunicação.
3. Comunicação entre aplicativos.
4. Formato para enviar mensagens.
5. Projetado para se comunicar via Internet.
6. Independente da plataforma.
7. Independente da linguagem.
8. Permite que você percorra os firewalls.

Requisições SOAP serão utilizadas nas sessões 3.2 e 3.3.1, onde foi utilizado o node-soap uma biblioteca de SOAP para NodeJs.

2.2.3. REST - *Representation State Transfer*

É uma arquitetura criada para ser mais simples de se usar que o SOAP. Pode ser usado em vários formatos de texto, como CSV (Comma-separated Values), RSS (Really Simple Syndication), JSON e YAML. Porém, só pode ser utilizado com o protocolo HTTP/HTTPS, por exemplo utilizando os métodos GET, POST, PUT e DELETE.

1. Melhor curva de aprendizado.
2. Mensagens menores e mais eficientes como o formato JSON comparado com XML.
3. Os dados podem ser colocados em cache, retornando sempre a mesma resposta para a mesma requisição.
4. Mais rápido pois precisa de menos processamento que o SOAP.

Como os correios não disponibiliza serviços por REST, foi desenvolvido um serviço alternativo que será detalhado na sessão 3.3.2.

3. *Web Service* - Rafael Buscas

Nesta sessão relato como foi o desenvolvimento e a implantação de um *Web Service* em um *Web Server*, que promove serviços de busca de CEPs e rastreamento de encomendas dos correios, utilizando uma página web e um aplicativo android como interface para consultas.

3.1. *Web Server*

O *Web Server* foi implementado utilizando o NodeJS com o framework ExpressJs, onde se cria um ambiente de desenvolvimento para criação de serviços web.

> O Node.js usa um modelo de E/S não bloqueante, que o torna leve e eficiente. O ecossistema de pacotes Node.js, npm , é o maior ecossistema de bibliotecas de código aberto do mundo [3].

> O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel [4].

3.2. *Web Service* de Busca de CEP

A busca de cidade e bairro pelo CEP está sendo realiza, a partir de um serviço disponível, no WSDL dos correios com a url https://apps.correios.com.br/SigepMasterJPA/AtendeClienteService/AtendeCliente?wsdl.

Conforme pode-se observar na figura 3, existem diversos serviços neste WSDL, porém só será utilizado o serviço de consulta de cep, com o nome *consultaCEP*.

```
<xs:element name="buscaTarifaValeResponse" type="tns:buscaTarifaValeResponse"/>
<xs:element name="calculaTarifaServico" type="tns:calculaTarifaServico"/>
<xs:element name="calculaTarifaServicoResponse" type="tns:calculaTarifaServicoResponse"/>
<xs:element name="cancelarObjeto" type="tns:cancelarObjeto"/>
<xs:element name="cancelarObjetoResponse" type="tns:cancelarObjetoResponse"/>
<xs:element name="cancelarPedidoScol" type="tns:cancelarPedidoScol"/>
<xs:element name="cancelarPedidoScolResponse" type="tns:cancelarPedidoScolResponse"/>
<xs:element name="consultaCEP" type="tns:consultaCEP"/>
<xs:element name="consultaCEPResponse" type="tns:consultaCEPResponse"/>
<xs:element name="consultaSRO" type="tns:consultaSRO"/>
<xs:element name="consultaSROResponse" type="tns:consultaSROResponse"/>
<xs:element name="consultarPedidosInformacao" type="tns:consultarPedidosInformacao"/>
```

Figura 3. Um pedaço dos serviços do WSDL dos correios, com o serviço consultaCEP em azul escuro.

Utilizando o node-soap (já comentado na sessão 2.2.2), uma requisição HTTP-GET de consulta de cep encaminhada para *Web Serve* Rafael Buscas, na url http://rafaelbuscas.ddns.net/api/correios/json/cep/Codigo, a mesma é recebida por um serviço web, que dispara uma requisição de consulta de cep por SOAP para o serviço de consulta dos correios, onde após retorno da mesma as informações são repassadas para a requisição inicial.

As linhas de código que recebe a requisição incial, conecta-se ao WSDL dos correios e reenvia a consulta, pode ser objservada na figura 4.

```
app.get('/api/correios/json/cep/:cep', function (req, res) {
    var url = "https://apps.correios.com.br/SigepMasterJPA/AtendeClienteService/AtendeCliente?wsdl";
    var options = {
        ignoredNamespaces: {
            namespaces: ['targetNamespace', 'typedNamespace'],
            override: true
        }
    };
    var cep = req.params.cep.replace("/\D/", '');
    if (cep.length == 8) {
        soap.createClient(url, options, function (err, client) {
            if (err) {
                res.json(false);
            } else {
                client.consultaCEP({ cep: req.params.cep }, function (errCli, result) {
                    var resp = [];
                    resp.push(errCli ? false : result['return'])
                    res.json(resp);
                });
            }
        });
    } else {
        res.json(false);
    }
});

```

Figura 4. Código de rota, para consulta de cep presente no Web Server Rafael Buscas.

Para mais informações sobre consulta de cep e outros serviços disponíveis por esse WSDL consultar a referência [5].

3.3. Web Service para rastreamento de encomendas dos Correios

O rastreamento por encomendas dos correios, desenvolvido nesta sessão, dispõe de duas tecnologias diferentes de empacotamento.

A primeira utiliza a tecnologia SOAP e será apresentada na sessão 3.3.1 , a segunda funcional porém alternativa utiliza uma requisição REST HTTP-POST, para obter uma requisição da página oficial dos correios, após receber a resposta em um objeto HTML, é realizado um parsear na resposta para extrair os dados da encomenda, está forma de rastreamento será apresentada na sessão 3.3.2.

3.3.1. Serviço de Rastreamento Encomenda via SOAP

O WSDL dos correios que fornece o serviço de rastreamento de encomenda conhecido como BuscaEventos, demostrado na figura 5, está disponível na url http://webservice.correios.com.br/service/rastro/Rastro.wsdl.

```
<message name="buscaEventos">
  <part name="parameters" element="tns:buscaEventos"></part>
</message>
<message name="buscaEventosLista">
  <part name="parameters" element="tns:buscaEventosLista"></part>
</message>
<message name="buscaEventosResponse">
  <part name="parameters" element="tns:buscaEventosResponse"></part>
</message>
<message name="buscaEventosListaResponse">
  <part name="parameters" element="tns:buscaEventosListaResponse"></part>
</message>
```

Figura 5. Um pedaço dos serviços do WSDL de rastreamento de encomendas dos correios, com o serviço BuscaEventos em azul escuro.

Na figura 6 utilizando o NodeJS ea biblioteca node-soap, uma solicitação para o *Web Servece* Rafael Buscas com a url http://rafaelbuscas.ddns.net/api/correios/json/objeto/CodigoDoRastreamento, cria uma requisição SOAP, que essa é enviada para o *Web Service* de rastreamento de encomendas, fornecido pelos correios, que retorna uma mesagem JSON que posteriormente e repassada para requisição incial.

Para mais informações referente a API de serviços dos correios, incentivo consultar a referência [6].

```
1  app.get("/api/correios/json/objeto/:code", function (req, res) {
2      var url = "http://webservice.correios.com.br/service/rastro/Rastro.wsdl";
3      var objeto = req.params.code;
4      soap.createClient(url, function (err, client) {
5          if (err) {
6              res.send("ERRO")
7          } else {
8              client.buscaEventos({
9                  usuario: "ECT", senha: "SRO", tipo: "L",
10                 resultado: "T", lingua: 101, objetos: objeto
11             }, function (errCli, result) {
12                 var resp = [];
13                 resp.push(errCli ? false : result['return'])
14                 res.json(resp);
15             });
16         }
17     });
18 });
```

Figura 6. Código de rota, para consulta de encomendas presente no Web Server Rafael Buscas.

Esse serviço disponibilizado pelos correios apesar de ser gratuito, somente retorna a última atualização do objeto.

3.3.2. Serviço de Rastreamento de Encomendas alternativo utilizando Crawling na Página dos Correios

Existe um serviço disponível na página dos correios para rastreamento de encomendas, onde é possivel realizar um crawling na página, para obter um objetivo HTML, contendo as informações da encomenda buscada.

O serviço está disponível na url ”http://www2.correios.com.br/sistemas/rastreamento/”, a imagem 7 demostra como pode-se utilizar a ferramenta *Postman*, para realizar uma requisição REST HTTP-POST, direto para página de rastreamento, com objetivo de extrair informações da mesma.

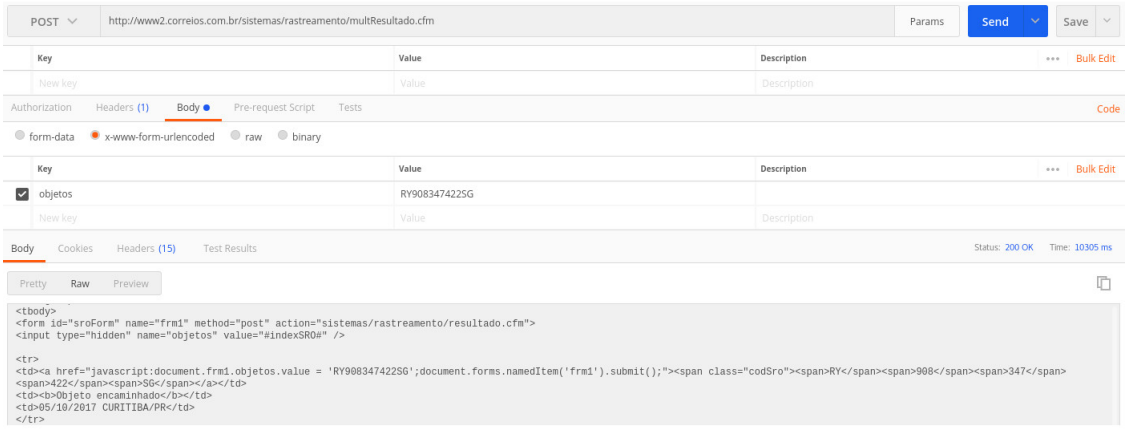

Figura 7. Postman realizando um requisição REST HTTP-POST.

Para consumir o serviço da página dos correios, primeiramente, foi desenvolvivdo um *Web Service* com NodeJs e o framework ExpressJs, junto com a biblioteca request-promise para requisição JavaScript e a biblioteca cheerio para realizar o parse das tags ”tr” e ”td” (onde estão as informações da encomenda, demostrado na figura 7, no campo body do response) do HTML.

Na figura 8 consta um método que realiza uma requisição REST - HTTP-POST no *Web Server* da página web dos correios.

```
1  request: function (req) {
2      var correios = {
3          uri: "http://www2.correios.com.br/sistemas/rastreamento/multResultado.cfm",
4          form: {
5              objetos: req.params.code
6          },
7          method: 'POST',
8          headers: {}
9      };
10     response = requestPromise(correios);
11     return response
12 }
```

Figura 8. Criando requisição em Java Script.

Após a requisição da figura 8, teremos uma variável contendo todo o HTML da página web, que foi obtida pela requsição.

Utilizando a função da figura 9, realizamos um parse no HTML, com o objetivo de extrair as informações referente a encomenda.

O parse percorre a tabela e extrai, algumas informações (código, localidade, data, situação) criando um objeto JSON, com as informações de rastreamento obtidas, que será consumido pela página web RafaelBuscas.

Assim sendo, uma requisição através da página RafaelBuscas ou utilizando diretamente a url http://rafaelbuscas.ddns.bet/api/correios/json/aobjeto/"codigo-de-rastreamento", primeiramente a requisição é recebida pelo *Web Service* (que está aguardando na rota da url acima), onde o mesmo envia uma requisição HTTP-POST, para a página de rastreamento dos correios com o código de encomenda fornecido, após o parse realizado as informações são encaminhandas em um objeto JSON, como resposta para requisição inicial.

```
1   parser: function (data) {
2       var $ = cheerio.load(data);
3       var objetos = [];
4       var tableObjetos = $('table').find('tr');
5       $(tableObjetos).map(function (key, objeto) {
6           objeto = $(objeto).children('td').map(function (key, field) {
7               return $(field).text();
8           }).toArray();
9           if (objeto[0]) {
10              var rastreio = {
11                  codigo: null,
12                  situacao: null,
13                  local: null,
14                  data: null
15              };
16              rastreio.codigo = objeto[0].trim();
17              if (objeto[2]) {
18                  rastreio.situacao = objeto[1];
19                  rastreio.local = objeto[2].substr(11, objeto[2].length).trim();
20                  rastreio.data = objeto[2].substr(0, 10).trim();
21              } else {
22                  rastreio.situacao = 'Objeto ainda não consta no sistema'
23              }
24
25              objetos["return"] = rastreio;
26          }
27      });
28      return Object.assign({}, objetos["return"]);
29  }
```

Figura 9. Criando um Crawling para parsear as tags do HTML.

4. Página Rafael Buscas e Aplicativo Android Rafael Buscas

Para fins didaticos, os serviços de rastreamento de encomendas e buscas de cep pelo código, relatados estão disponíveis na url http://rafaelbuscas.ddns.net, a mesma possui

uma página web como interface para os serviços, estes que estão hospedados em uma máquina virtual do Google Cloud.

Na página inical no canto inferior, existe um input para o código do CEP ou do Rastreamento (fornecidos pelos correios).

Um componente para seleção de consulta entre Cep ou Rastremamento, com o nome de ”TIPO”, deve ser marcado como CEP, Rastreio REST ou Rastreio SOAP, antes da busca (botão com uma lupa) como demostrado na figura 10.

Figura 10. Página Web Rafael Buscas, demostração de busca de encomenda via SOAP.

Além da página web, um aplicativo para plataforma Android (figura 11), foi desenvolvido pra demostração de portabilidade dos serviços desenvolvidos neste documento, o mesmo possui as mesmas funcionalidades da página web.

Figura 11. Aplicativo Android Rafael Buscas, demostração de busca de encomenda via SOAP.

5. Conclusão

Neste relatório foi apresentado a estrutura básica de um *Web Service*, e como o próprio pode ser utilizado para consumir serviços de outros *Web Services* disponíveis por outras empresas, outra abordagem exposta seria utilizar os mesmos para criação de um novo serviço, em seu próprio *Web Server*, para atender as mais diversas necessidades, como exemplo o rastreiamento de encomendas ou obter a cidade e o bairro através de um CEP, a partir de dispositivos diferentes, porém do mesmo *Web Servece*.

Referências

[1] SOAWebServices., "Como funcionam os WebServices." http://www.soawebservices.com.br/como-funciona.aspx, 2012. [Online; acesso em 05-Outubro-2017].

[2] T. Spoint, "Web Services - Examples." www.tutorialspoint.com/webservices/web_services_examples.htm, 2017. [Online; acesso em 05-Outubro-2017].

[3] NodeJs, "Documentação de referência da API." https://nodejs.org/en/docs/, 2017. [Online; acesso em 02-Outubro-2017].

[4] ExpressJs, "Documentação de referência da API." http://expressjs.com/pt-br/4x/api.html, 2017. [Online; acesso em 05-Outubro-2017].

[5] CORREIOS, "Sistema de Gerenciamento de Postagem dos Correios." http://www.corporativo.correios.com.br/encomendas/sigepweb/doc/Manual_de_Implementacao_do_Web_Service_SIGEP_WEB.pdf, 2017. [Online; acesso em 05-Outubro-2017].

[6] Correios, "Web Service de Rastreamento." https://www.correios.com.br/para-voce/correios-de-a-a-z/pdf/rastreamento-de-objetos/manual_rastreamentoobjetosws.pdf, 2017. [Online; acesso em 05-Outubro-2017].