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

Rafael Gonçalves de Oliveira Viana[1]
10 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 um serviço para busca de CEP e rastreamento de encomendas dos Correios, utilizando Angular no front-end e NodeJS no back-end.*

1. Introdução

Com a disseminação de dispositivos com conexão a Internet na sociedade atual, houvesse a necessidade de criar/gerenciar serviços web *Web Services*, estes são necessarios para atender as mais diversas funções como o rastreio de encomendas e consultas de CEPs entre outros serviços. Este relatório demonstra a arquitetura de um *Web Service* e como foi implementado um *Web Service* que possui serviços de consulta de CEP e rastreamento de encomendas dos correios, juntamente com uma página web oferecendo uma interface para os mesmos. O código deste relatório esta 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

Webservice 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].

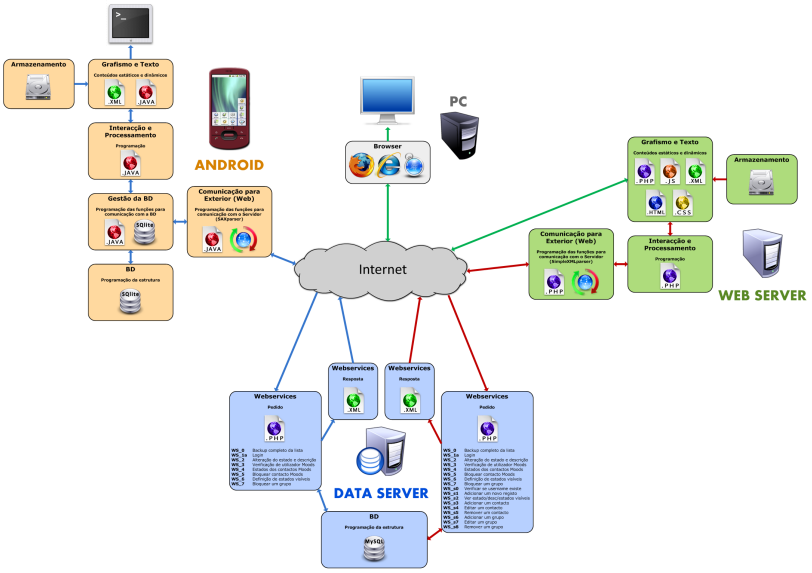

Figura 1. Sistema implantando, fornecendo serviços para PCs, dispositivos móveis e outros web servers.

3. Qual Arquitetura de Web Service utilizar

Para poder identificar qual arquitetura utilizar de *Web Service*, deve-se examinar os papéis individuais de cada ator no contexto, e também a pilha emergente de protocolo que o mesmo pretende utilizar. 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 consultar a referências [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.

Outro modo de identificar qual arquitetura utilizar, é examinar a pilha emergente de protocolo que pretende utilizar. 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).

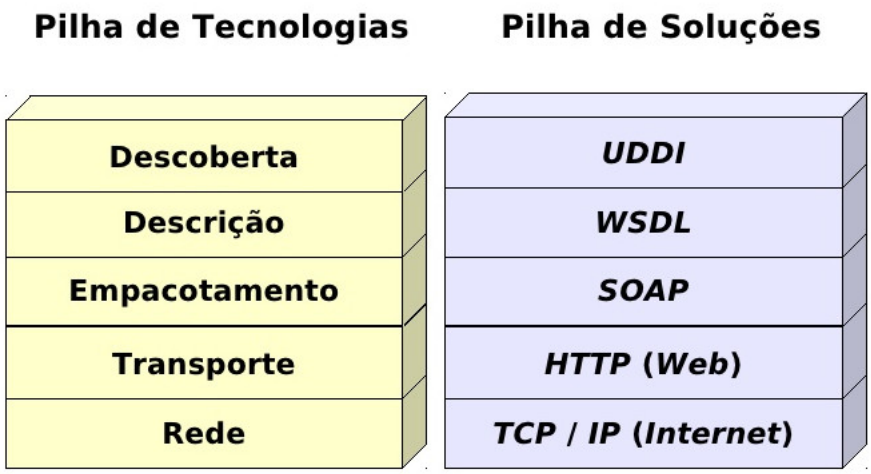

Figura 2. Pilha de protocolo de transporte e suas tecnologias.

3.1. Principais Serviço de Transporte de um Web Service

Os protocolos da camada inferior ficam responsáveis pelo transporte de serviços. Essa camada é responsável por transportar como exemplo mensagens XML e JSON entre dois computadores.

1. **Hyper Text Transfer Protocol (HTTP):** É a opção mais popular para o transporte de serviços. O HTTP é simples, estável e amplamente implantado. Além disso, a maioria dos firewalls permitem o tráfego HTTP. Isso permite que mensagens REST ou SOAP se mostrem como mensagens HTTP. Isso é bom se você quiser integrar aplicativos remotos, mas eleva uma série de preocupações de segurança.
2. **Bloqueia o protocolo de troca extensível (BEEP)** Esta é uma alternativa promissora para o HTTP. O BEEP é uma nova estrutura da Task Force de Engenharia da Internet (IETF) para a construção de novos protocolos.

3.2. 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.

3.2.1. WSDL - *Web Services Description Language*

O WSDL é um idioma baseado em XML para descrever os serviços da Web e como acessá-los.

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.

3.2.2. 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.

3.2.3. 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.

4. Metodologia

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 como interface para consultas.

4.1. Web Server

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

> 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].

4.2. Web Service de Busca de CEP

A busca de ceps, está sendo tercerizada por um *Web Service* que prove servico de consultas de CEPs, nos mais diversos tipos de empacotamento. Esses serviços são fornecidos pela empresa *VIACEP*, o site da mesma é https://viacep.com.br/.

Uma página Web RafaelBuscas foi desenvolvida utilizando o framework Angular, a mesma consome o serviço de buscas de cep, no formato JSON fornecido pelos servidores da Viacep.

A url do Web Service da VIACEP, consumido pela página RafaelBuscas, foi o http://viacep.com.br/ws/CodigoCEP/json/, o mesmo encaminha uma requisição HTTP-GET com o código do CEP válido conforme, o mesmo retorna uma resposta em JSON, com as informações do CEP (Localização, Bairro, Código IBGE, GIA entre outras), conforme a figura 3, para mais informações sobre requisições consultar a referência [5].

Figura 3. Modelo de requisição HTTP-GET e seu retorno.

4.3. Web Service Encomendas dos Correios

O rastreiamento por encomendas dos correios, dispõe de duas tecnologias diferentes de empacotamento, fornecida pelos Correios. A primeira utiliza a tecnologia SOAP, e outra

alternativa é utilizar uma requisição REST HTTP-POST, para obter uma requisição da página oficial dos correios e após receber a resposta em um objeto HTML, e realizado um parsear na resposta.

4.3.1. Serviço de Rastreamento Encomenda via SOAP

O WSDL dos correios que fornece recursos para os serviços de requisição SOAP,o serviço de rastreamento de encomendas dos correios via SOAP é gratuito.

A Url "http://webservice.correios.com.br:80/service/rastro"é o endereço do *Web Service* e não do WDSL, o webservice espera uma mensagem SOAP como na Listing 4 a mesma deve ser assinada com cabeçalhos HTTP no formato "Content-Type", "text/xml".

```
"<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/
    envelope/\" xmlns:res=\"http://resource.webservice.correios.com.
    br/\">
                  <soapenv:Header/>
                    <soapenv:Body>
                       <res:buscaEventos>
                          <usuario>ECT</usuario>
                          <senha>SRO</senha>
                          <tipo>L</tipo>
                         <resultado>T</resultado>
                         <lingua>101</lingua>
                           <objetos>RY907728402SG</objetos>
                       </res:buscaEventos>
                    </soapenv:Body>
                </soapenv:Envelope>
```

Listing 1. Código XML SOAP para requisição de localização de pacote

Utilizando a linguagem Java e a biblioteca *javax.xml.soap*, houve simulação de envio de requisição SOAP com a mensagem Listing 4 para o *Web Service* de rastreamento de encomendas, fornecido pelos correios para mais informações referente a API de serviços dos correios, incentivo consultar a referência [6].

O retorno da requisição é um código XML, contendo a resposta da requisição, como demostrano na Listing 5.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
    envelope/">
    <soapenv:Header>
        <X-OPNET-Transaction-Trace:X-OPNET-Transaction-Trace
            xmlns:X-OPNET-Transaction-Trace="http://opnet.com">pid
            =131564,requestid=6039
            e4d6630ff66f79faef6849a968e0177c404c7e65d86e</X-OPNET-
            Transaction-Trace:X-OPNET-Transaction-Trace>
    </soapenv:Header>
    <soapenv:Body>
        <ns2:buscaEventosResponse xmlns:ns2="http://resource.
            webservice.correios.com.br/">
            <return>
                <versao>2.0</versao>
                <qtd>1</qtd>
```

```
10                  <objeto>
11                      <numero>RY907728402SG</numero>
12                      <sigla>RY</sigla>
13                      <nome>REM ECON TALAO/CARTAO COM AR DIGITAL</
                            nome>
14                      <categoria>REMESSA ECONOICA TALAO/CARTAO</
                            categoria>
15                      <evento>
16                          <tipo>RO</tipo>
17                          <status>01</status>
18                          <data>02/10/2017</data>
19                          <hora>13:49</hora>
20                          <descricao>Objeto encaminhado </descricao>
21                          <local>CENTRO INTERNACIONAL</local>
22                          <codigo>81010971</codigo>
23                          <cidade>CURITIBA</cidade>
24                          <uf>PR</uf>
25                          <destino>
26                              <local>Fiscalizacao Receita Federal Do
                                    Brasil</local>
27                              <codigo>00002991</codigo>
28                          </destino>
29                      </evento>
30                  </objeto>
31              </return>
32          </ns2:buscaEventosResponse>
33      </soapenv:Body>
34  </soapenv:Envelope>
```

Listing 2. Resposta da Requisição SOAP

Esse serviço disponibilizado pelos correios apesar de ser gratuito, somente retorna a última atualização do objeto como demostrado na Listing 5.

4.3.2. 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 da 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 4 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.

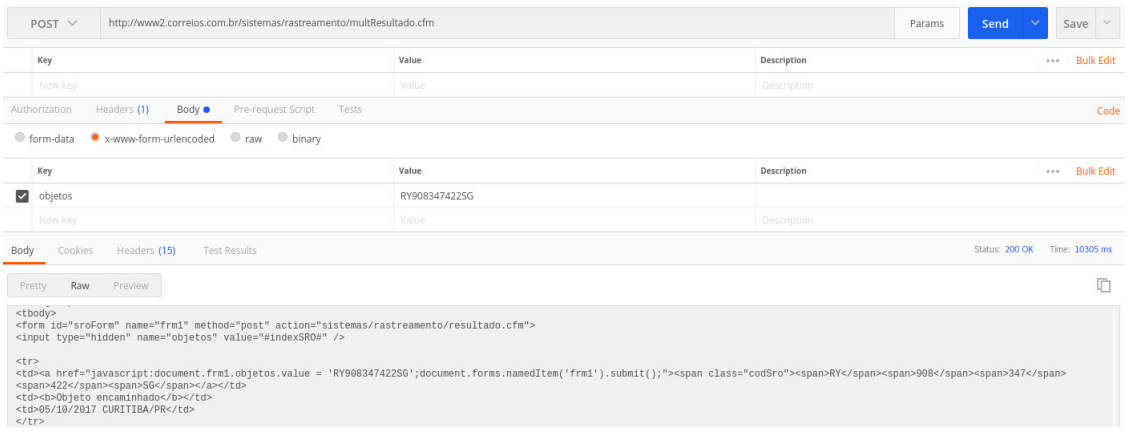

Figura 4. Postman realizando um requsição REST HTTP-POST.

Para consumir o serviço da página dos correios, primeiramente, temos que criar 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 tag do HTML.

Na Listing 3 consta um método que realiza uma requisição REST - HTTP-POST no *Web Server* da página web.

```
request: function (req) {
var correios = {
uri: "http://www2.correios.com.br/sistemas/rastreamento/
    multResultado.cfm",
form: {
objetos: req.params.code
},
method: 'POST',
headers: {}
};
 response = requestPromise(correios);
 console.log(response)
 return response
}
```

Listing 3. Criando Requisição em Java Script

Após a requisição da Listing 3, teremos uma variável, contendo todo o HTML da página web, que foi obtida pela requsição.

Utilizando a função na Listing 4, 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.

Realizando uma requisição através da página RafaelBuscas ou utilizando diretamente a url http://rafaelbuscas.ddns.bet/json/"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
2  parser: function (data) {
3   var $ = cheerio.load(data);
4   var objetos = [];
5   var tableObjetos = $('table').find('tr');
6   $(tableObjetos).map(function(key, objeto){
7    objeto  = $(objeto).children('td').map(function (key, field) {
8     return $(field).text();
9    }).toArray();
10   if (objeto[0]) {
11
12    var rastreio = {
13     codigo  : null,
14     situacao: null,
15     local   : null,
16     data    : null
17    };
18
19    rastreio.codigo     = objeto[0].trim();
20    if (objeto[2]) {
21     rastreio.situacao   = objeto[1];
22     rastreio.local      = objeto[2].substr(11, objeto[2].length).
          trim();
23     rastreio.data       = objeto[2].substr(0,10).trim();
24    } else {
25     rastreio.situacao   = 'Objeto ainda nao consta no sistema'
26    }
27
28    objetos['rast'] = rastreio;
29   }
30  });
31
32  return Object.assign({}, objetos);
33 }
```

Listing 4. Criando um Crawling para parsar as tags

5. Página RafaelBuscas

Para fins didaticos, os serviços criados nesse relatório estão disponíveis na url http://rafaelbuscas.ddns.net, a mesma possui uma página web como interface para os serviços citados nas seções anteriores.

Esta página está hospedada em uma máquina virtual do Google Cloud, a mesma disponível para buscas de cep e rastreamento de encomendas sem custos. Na página inical no canto inferior, exite 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 ou Rastreio, antes da busca (botão com uma lupa) como demostrado na Figura 5.

Figura 5. Página RafaelBuscas, demostração de busca de cep.

6. Conclusão

Esse relatório demostro como pode ser criado uma página web e um *web service* que consome serviços de outros *Web Services* disponíveis por outras impresas, ou utilizar os mesmos para criação de um novo serviço, para atender as mais diversas necessidades.

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] VIACEP, “Códigos de Endereçamento Postal (CEP) do Brasil.” `https://viacep.com.br/`, 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].