Revista de Informática Aplicada, Volume 12, Número 2, 2016

06

WEB SERVICES: Integração De Sistemas Orientado a Serviços com uma Proposta de Aplicação na EAD[1]

Athos Eulálio[2]

Diego Cordeiro[2]

Rodrigo de Souza[3]

Resumo: A tarefa de integração de sistemas de informação é um grande desafio em Ciência da Computação devido à sua importância no atual cenário de ubiquidade de aplicações digitais, bem como a complexidade subjacente à implementação de canais de comunicação eficientes entre sistemas híbridos. No contexto da educação superior, a massificação da Educação a Distância constituiu nos últimos anos uma instância desse problema, em razão das dificuldades administrativas e pedagógicas oriundas do emprego simultâneo de sistemas com propósitos e origens diferentes, a saber, os Sistemas de Gestão Acadêmica e os Ambientes Virtuais de Aprendizagem. Entre as várias abordagens de integração de sistemas, o uso de serviços web tem alcançado proeminência nos últimos anos. Este artigo apresenta as principais características de integração através de web services, e discute uma proposta de solução de integração de sistemas no contexto da Educação a Distância.

Palavras-chave: Integração. Sistemas de informações. *Web services*.

Abstract: The task of integration of information systems is a major challenge in Computer Science due to its importance in the current scenario of ubiquity of digital applications, as well as the complexity underlying the implementation of efficient communication channels between hybrid systems. In the context of higher education, the massification of Distance Education in recent years represent an instance of this problem, due to administrative and pedagogical difficulties arising from the simultaneous use of systems with different purposes and origins, namely Academic Management Systems and Virtual Learning Environments. Among the several system integration approaches, the use of web services has achieved prominence in recent years. This article presents the main features of integration through web services, and discusses a proposal for a system integration solution in the context of Distance Education.

Keywords: Integration. Information systems. Web services.

[1] O presente artigo é uma extensão de trabalho publicado na conferência: XII - Congresso Brasileiro de Ensino Superior a Distância (ESUD 2017), I - Congresso Internacional de Ensino Superior a Distância Salvador/BA, 30.11.2015 – 03.12.2015 UNIREDE.

[2] Instituto Federal de Educação, Ciência e Tecnologia do Piauí, e-mails: (athos.denis, diego.cordeiro)@ifpi.edu.br

[3] Universidade Federal Rural de Pernambuco, e-mails: rodrigo.npmsouza@ufrpe.br

1 Introdução

Integração de sistemas de informação, independente de plataformas de software e hardware, é um desafio central em sistemas distribuídos na Internet, como na infraestrutura subjacente a plataformas de Educação a Distância. Ademais, a proliferação massiva de dispositivos móveis nesse contexto reforça a necessidade de padronizações, provendo meios de integração e troca de dados em um cenário fortemente híbrido. Os *web services* apresentam-se como uma das principais soluções para esses desafios de integração.

Web service é um meio utilizado para integração de sistemas e de comunicação entre aplicações heterogêneas. Em outros termos, essa tecnologia permite que novas aplicações possam interagir com aquelas existentes em um ambiente de trabalho e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Para Kalin (2010) um *web service* é, então, uma aplicação distribuída, cujos componentes podem ser aplicados e executados em dispositivos distintos.

Os componentes dos *web services* possibilitam às aplicações enviar e receber dados, por exemplo, em formato *XML*[1]. Por exemplo, a aplicação *GoogleMaps*®, mantida pela *Google*, disponibiliza um *web service* que permite localizar cidades, endereços e bairros. Caso um desenvolvedor deseje localizar endereços de seus clientes em suas aplicações, por exemplo, poderá consumir essas informações diretamente do *web service* do *Google*, independente da linguagem de programação utilizada pela aplicação: Java, C#, Perl, *Ruby*, *Microsoft .NET*, *Visual Studio*, *PHP*, *Python*, etc. O *web service* em questão fornece as informações necessárias no formato *XML* e recebe a resposta também no mesmo formato por meio do protocolo *HTTP*[2]. A Figura 1 sintetiza como ocorre o funcionamento de um web service.

Figura 1 – Funcionamento de um Web service.

Fonte: Elaborado pelo autor

[1] XML é a sigla para *Extensible Markup Language*, que significa em português Linguagem Extensível de Marcação. É uma recomendação para gerar linguagens de marcação para necessidades especiais, tais como textos, banco de dados ou desenhos vetoriais. A linguagem XML é classificada como extensível porque permite definir os elementos de marcação.

[2] HTTP: Hyper Text Transfer Protocol (Protocolo de transferência de Hipertexto).

Nesse contexto, é importante destacar a filosofia *SOA*, que em Português significa Arquitetura Orientada a Serviços (do Inglês *Service-Oriented Architecture*). Trata-se de uma especificação para a troca de informação entre sistemas, ou seja, uma especificação de formato de dados para envio de estruturas de dados entre serviços, com um padrão para permitir a interoperabilidade entre eles. *SOA* não é uma tecnologia, não é uma metodologia, não é um serviço, mas é um conceito de arquitetura que provê a integração entre aplicações que venham a ser construídas através da reutilização de serviços, e não da criação de um novo código fonte, fazendo com que as organizações baseadas em *Web* venham a oferecer comodidade para seus clientes/fornecedores por meio de um conjunto de rotinas para comunicação com diferentes plataformas. Uma das maneiras possíveis de implementar *SOA* é através do uso de *web services*, onde o *web service* desempenha a função de prover serviços.

Em vista dessas funcionalidades, conclui-se que os *web services* proporcionam uma ferramenta indispensável para cursos de Educação a Distância, onde, em um cenário ideal de funcionamento, sistemas de gerenciamento acadêmicos, ambientes virtuais de aprendizagem (AVA) e dispositivos móveis interagem constantemente em uma escala de centenas ou milhares de usuários. Neste texto, apresentamos os principais conceitos e tecnologias empregadas para a implementação de *web services*, e discutimos uma possibilidade de uso no problema, enfrentado em muitas universidades brasileiras, de prover um intercâmbio automático de dados entre Sistemas de Gerenciamento Acadêmicos (SGA) e ambientes virtuais de aprendizagem.

Nossa solução utiliza a abordagem *web services* baseados em *REST* trafegando dados por *JSON* (conceitos que serão brevemente discutidos na sequência). Segundo Liu et al (2008), com essa solução torna-se possível a construção de *web services* por meio *URI's* genéricas, sendo estas responsáveis por representar requisições de dados por meio de agregação, generalização, composição e associação. Da mesma forma a resposta de uma requisição ao cliente por meio de *JSON* possibilita grande portabilidade dos serviços oferecidos pelo *web service*, permitindo implementações genéricas em várias plataformas, considerando ainda que o mesmo é suportado por uma gama extensa de linguagens.

2 Formato de dados

Conforme comentado na seção anterior, o formato de dados *XML* apresenta-se como forma tradicional de solicitar e receber dados por meio de *web services*. Contudo há outros formatos que podem representar

dados de forma estruturada e aninhada; o *JSON*[3] é um deles. A presente seção apresenta os dois formatos.

2.1 O formato XML

O formato *XML* é recomendado pela *W3C*[4] para criar documento com dados organizados de forma hierárquica. Está entre suas principais características o fato de fornecer uma sintaxe básica que pode ser utilizada para compartilhar informações entre diferentes arquiteturas e aplicações.

Quando aliado com outros padrões, o uso de *XML* torna possível definir o conteúdo de um documento separadamente de seu formato, tornando simples a reutilização de código em outras aplicações, para diferentes propósitos. Por outro lado, pode representar um *overhead* de espaço, em razão do uso sistemático de marcadores. Citando Mitchell (2013):

> Qualquer pessoa que já tenha passado um bom tempo na Internet entenderá o estilo "cheio de sinais de maior e de menor" do *XML* e será capaz de lê-lo. O XML é um formato bem prolixo; a pontuação adicional e o escopo para atributos, os dados de caracteres e as tags aninhadas podem gerar dados um pouco mais extensos do que gerados por outros formatos.

Considere como exemplo que se deseje representar, através de *XML*, uma lista de marcas de carros: Volkswagen, Chevrolet, Ford, Fiat, Nissan. Uma possível representação em *XML* está ilustrada a seguir:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <lista>
3   <item>Volkswagen</item>
4   <item>Chevrolet</item>
5   <item>Ford</item>
6   <item>Fiat</item>
7   <item>Nissan</item>
8 </lista>
```

Listagem 1 – Representação de dados em XML.

Fonte: Elaborado pelo autor

Outra de suas principais características é a maneira de representar dados de forma bastante detalhada, pois é possível incluir informações sobre os tipos de dados com os quais se está trabalhando, usando para isso mesmo tipos personalizados de dados.

O *XML* é bastante difundido entre desenvolvedores e por diversas plataformas tecnológicas tais como *Oracle*, *.NET* e *Java*, e, portanto, é um formato muito popular para transmissão de dados. Contudo, quanto ao volume de dados a serem transferidos neste formato, dependendo dos dispositivos, da largura de banda e do formato mais extenso aplicado aos dados, pode apresentar problemas significativos conforme descreve Richardson e Ruby (2007):

> O efeito é que o *HTTP* é reduzido a um protocolo de transporte para um peso *XML* enorme que explica o que "realmente" está ocorrendo. O serviço resultante é muito complexo, impossível de depurar e não funcionará, a menos que seus clientes tenham a configuração exatamente igual à sua.

2.2 O formato JSON

Este formato de dados pode ser escrito e lido facilmente por uma gama de linguagens e não apenas *JavaScript*[5]. É caracterizado por ser um formato leve, bastante simples e capaz de representar dados estruturados e aninhados.

Tomando o exemplo anterior utilizado no formato *XML* da lista de marca de carros, uma representação típica em *JSON* é:

{"Lista":
["Volkswagen","Chevrolet","Ford","Fiat","Nissan"]}

Múltiplos valores podem ser agrupados entre colchetes e as chaves e os valores são separados por dois pontos e cada registro é separado por vírgula. Esse exemplo mostra que os dados listados possuem um valor/chave chamado "Lista". É possível representar estes mesmos dados de uma forma bastante simplificada:

["Volkswagen","Chevrolet","Ford","Fiat","Nissan"]

[3] JSON – JavaScript Object Notation (Notação de Objetos JavaScript).

[4] O World Wide Web Consortium (W3C) é a principal organização de padronização da World Wide Web. Consiste em um consórcio internacional com quase 400 membros, agregando empresas, órgãos governamentais e organizações independentes com a finalidade de estabelecer padrões para a criação e a interpretação de conteúdos para a Web.

[5] JavaScript: Linguagem de programação interpretada originalmente implementada para navegadores web para que scripts pudessem ser executados no lado do cliente e interagissem com usuário.

A característica mais positiva do *JSON* está no fato de ser um formato de dados econômico, que não exige muito espaço se comparado ao *XML*, assim como não é complicada sua transferência por meio de conexões lentas, sendo ótimo para uso de aplicações móveis, pois seu custo de decodificação é baixo.

O seu uso é recomendado quando as informações sobre o formato de dados não forem essenciais, uma vez que ele não fornece informações específicas sobre os tipos de dados que estão sendo usados.

Figura 2 – Funcionamento do SOAP.
Fonte: Elaborado pelo autor.

3 Padrões para desenvolvimento de web services

Os conceitos aqui explorados visam auxiliar na definição da escolha da arquitetura de software que será utilizada para desenvolver um *web service*, pois esta escolha definirá todos os componentes que devem ser utilizados na concepção do projeto de integração de sistemas heterogêneos.

Atualmente existem dois padrões para o desenvolvimento de web services: o padrão *SOAP* e o padrão *REST* ou *RESTful*. As características de ambos são discutidas brevemente a seguir, a fim de que possamos justificar as escolhas feitas em nosso modelo de integração para a Educação a Distância.

3.1 Web services SOAP

Originalmente *SOAP* significava *Simple Object Access Protocol* (Protocolo de Acesso a Objetos), mas nos dias atuais essa descrição não é mais válida. Segundo Gomes (2014) *SOAP* é o protocolo padrão para transmissão de dados dentro da arquitetura de web services proposta pelo W3C. O *SOAP* é um protocolo baseado no *XML* e segue o modelo "*REQUEST-RESPONSE*" (solicitação – resposta) do *HTTP*.

Por padrão, as trocas de informações de tecnologias baseadas em *XML SOAP* são descritas por um arquivo WSDL[6]. O WSDL descreve a localização de um serviço em particular, os tipos de dados, valores, métodos e parâmetros utilizados. A figura 2 destaca o funcionamento do protocolo SOAP.

Uma questão a destacar refere-se ao protocolo sobre o qual trafegam as mensagens *SOAP*: embora comumente estas utilizem o *HTTP*, a arquitetura proposta pelo *W3C* especifica que as respostas e solicitações *XML* possam trafegar por outros protocolos como *FTP*, *TCP*, *SMP*, etc.

Conforme anteriormente mencionado no item 2.1, referente ao formato *XML*, a pontuação adicional e o escopo para atributos, os dados de caracteres e as *tags* aninhadas podem gerar dados mais extensos do que gerados por outros formatos; isto ocorre devido ao fato de existirem uma série de normatizações denominados *Web Services Standards* para desenvolver neste padrão. O *Web Services Standards* define especificações de segurança, de mensagens, de interoperabilidade, etc. Cada padrão estabelece um conjunto de normativas em *XML* de como se deve desenvolver para o padrão *SOAP*, ou seja, caso desenvolvedor que esteja utilizando o padrão *XML SOA* para desenvolver sua aplicação web service e necessite implementar as especificações de segurança, deverá consultar o *Web Service Standards* na seção *Security Specifications* para estar em acordo com a forma com a qual deve trabalhar utilizando o padrão *SOAP*.

O desenvolvedor de posse do arquivo *WSDL* poderá assim criar uma aplicação cliente que efetuará uma chamada ao provedor web service, que conseguirá obter resposta conforme o serviço que foi solicitado. As solicitações do cliente serão encaminhadas em formato *XML*, por sua vez o provedor *web service* receberá a solicitação e produzirá uma resposta no mesmo formato. O *WSDL* funciona como um descritor cujo objetivo é especificar o formato de entrada e saída de cada operação realizada de acordo com as especificações do W*eb Service Standards*.

Sobre o formato da solicitação e resposta (*REQUEST – RESPONSE*) do formato *XML* entre o cliente e o provedor *web service*, ambos se apresentam da seguinte forma:

[6] WSDL - Web Service Description Language (Linguagem para descrição de web service). É um arquivo do tipo XML, cuja finalidade é descrever detalhadamente um web service. Essa descrição específica as operações que compõem o web service e define de forma clara como deve ser o formato de entrada e saída de cada operação.

```
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>
```

Listagem 2 - Solicitação em XML.

Fonte: W3 Schools (2015)

```
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>
```

Listagem 3 - Resposta em XML.

Fonte: W3 Schools (2015)

Com o formato *XML* no padrão *SOAP* todo o código das solicitações e respostas estão bem mapeados, ou seja, este padrão de arquitetura é muito completo e robusto. Contudo, ele tem um custo, conforme já mencionamos, pois enquanto é gerado um pedido do cliente que será enviado ao servidor web service *SOAP*, este responde "empacotando" em um envelope a resposta de acordo com os padrões estabelecidos pelo descritor (*WSDL*). Isso pode tornar muito lento o tempo de resposta e aumentar o consumo de banda se comparado com outro padrão de arquitetura como o *REST,* que descrevemos a seguir.

3.2 Web services *REST* ou *RESTful*

O padrão arquitetural *REST* é acrônimo de *REpresentational State Transfer* (Transferência de Estado Representacional), não é um protocolo mais sim um conjunto de boas práticas na modelagem de web services estabelecidas por Roy Fielding (http://roy.gbiv.com) para sua tese de *PhD* onde este descreve todas as diretrizes do estilo *REST* (a tese de Fielding está disponível para consulta no seguinte endereço eletrônico: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm).

O princípio básico do *REST* é possibilitar o desenvolvimento de web services de boa eficiência, que sejam simples, e que permitam trabalhar com dados via *HTTP*. Ao contrário do padrão arquitetural fixo *XML SOAP*, o *REST* pode ter uma representação de formato por *XML*, *JSON* ou qualquer outra opção; de fato, ao se caracterizar o padrão *REST,* percebe-se que este é bastante similar se comparado em termos de finalidade ao padrão *SOAP*, contudo o elemento diferencial entre ambos está no fato do padrão *REST* não trazer consigo toda carga estabelecida no padrão *SOAP* por meio de seu *Web Services Standards* e WSDL. O mais próximo disso que há em *REST* é o *WADL*[7].

Segundo Richardson e Ruby (2007) como na *WSDL*, um cliente genérico pode carregar um arquivo *WADL* e ter uma permissão imediata para acessar a funcionalidade completa do *web service* correspondente. Contudo *WADL* é mais falado do que utilizado, pois com base no princípio da simplicidade na qual opera o padrão *REST* ele funcionaria conforme descrito na figura abaixo:

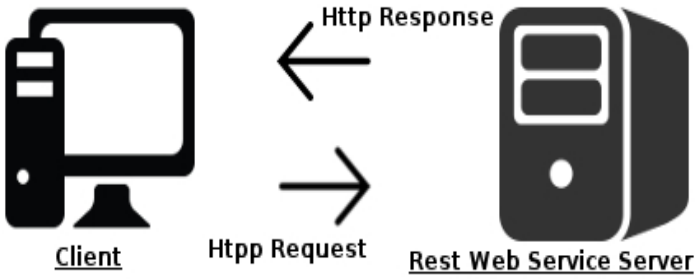


Figura 3 – Funcionamento do REST.

Fonte: http://www.chemaxon.com/app/themes/chemaxon/images/product_pages/jws/rest.jpg

Sem a necessidade de estabelecer descrições existentes no padrão *XML SOAP* por meio do *WSDL*, os *web services REST* se tornam simples de serem desenvolvidos. As características citadas anteriormente definem o modelo de arquitetura *REST* e caso uma aplicação siga todas as funcionalidades e definições deste modelo é chamada de *RESTful*.

Entre as características-chave do *REST* pode-se também destacar o fato de ser um modelo Cliente-Servidor sem estado (*stateless*), na qual cada mensagem *HTTP* contém toda a informação necessária para interpretação do pedido e em que grande parte ou todo o código é executado no Servidor, minimizando

[7] WADL: Web Application Description Language (Linguagem de descrição de aplicação Web) é um vocabulário XML utilizado para descrever os serviços web REST.

problemas na aplicação oriunda da parte do Cliente.

Entre as vantagens do *REST*, podem ser destacadas: 1. O tempo reduzido de aprendizagem para implantar o modelo *REST*, assim como sua baixa complexidade de codificação, que favorecem a redução do tempo necessário entre a construção da aplicação e a sua disponibilização; 2. O consumo de rede e processamento de informações/dados são menores se comparados com o padrão *XML SOAP*; 3. A implementação da arquitetura *REST* pode ocorrer com qualquer linguagem que tenha suporte ao protocolo *HTTP*; 4. A arquitetura *REST* fornece serviços independentemente da plataforma e tecnologia, garantido interoperabilidade. Entre suas desvantagens podem ser destacadas:

- Em se tratando de aplicações críticas no qual o desempenho seja um requisito a arquitetura *REST* pode não atender satisfatoriamente;
- A ausência de um descritor para especificar o formato de entrada e saída de cada operação realizada para auxiliar desenvolvedores a criarem suas aplicações, tal qual como ocorre com o descritor *WSDL* no padrão *SOAP*.

Em síntese o *REST* é um modelo arquitetural prático capaz de atender requisitos de integração sem agregar complexidade, por meio de boas práticas de programação aliado a um modelo arquitetural simplista e robusto que reduz consideravelmente o tempo necessário para construção de aplicações.

4 Integração de sistemas no meio acadêmico na modalidade EAD

No âmbito acadêmico, conforme mencionado na introdução deste trabalho, é destacada a problemática de integração entre os dois principais sistemas que compõem o arcabouço de gerenciamento e execução de cursos a distância, a saber, o sistema de gestão acadêmica (SGA) e ambiente virtual de aprendizagem (AVA). De fato, administradores e docentes, deparam-se comumente com a necessidade de transportar dados (dados de alunos, conceitos, etc.) entre esses sistemas. Os tópicos a seguir discutem uma aplicação para web services que possibilita prover interoperabilidade entre estes dois sistemas.

4.1 Problemática para integrar sistemas por meio de web services

O processo de integrar sistemas de informação no contexto acadêmico comumente ocorre entre os sistemas de gerenciamento acadêmico e o ambiente virtual de aprendizagem.

A importação de dados do sistema de gerenciamento acadêmico para o ambiente virtual de aprendizagem normalmente envolve os seguintes dados:

Quadro 1 - Dados a serem importados para ambiente virtual

Municípios/Polos	Nome da localidade de ensino dos discentes
Cursos	Descrição dos cursos em que estão matriculados os discentes.
Disciplinas	Descrição das disciplinas que serão cursadas por semestre/período
Turmas	Descrição das turmas de cada curso ofertado
Matrículas	Vinculo para cada discente aos cursos, turmas e disciplinas.
Usuários	Dados cadastrais de cada discente.

Fonte: Elaborado pelo autor

Os dados devem ser importados do ambiente virtual de aprendizagem para o gerenciamento acadêmico compondo assim um processo síncrono entre ambos. Toda replicação entre os dois sistemas deve ser automática e transparente.

Considere um cenário no qual os cursos na modalidade de Educação a Distância possuam uma estrutura onde estes podem ser compostos por várias turmas. Estas por sua vez possuirão diversas disciplinas nas quais deveram existir o registro de cada discente matriculado por semestre/período nos diversos municípios/polos em que estarão sendo ofertados cada um destes cursos. Hierarquicamente deverá ser estabelecido no ambiente virtual de aprendizagem uma estrutura que comporte, por exemplo, a seguinte situação:

Município/Polo
Cursos
Turma 1
Disciplina 1
Disciplina 2
Disciplina n...
Turma n...
Disciplina 1
Disciplina 2
Disciplina n...

Para detalhar claramente, cada discente matriculado fará parte de uma turma na qual estará ligado a um conjunto de disciplinas. Em uma situação hipotética supondo 15.000 discentes, cada um destes matriculados em 06 disciplinas por semestre/período, haveriam cerca de 90.0000 registros a serem importados do sistema de gerenciamento acadêmico para o ambiente virtual de aprendizagem.

4.2 Solução proposta

Conforme destacado anteriormente sobre os padrões *SOAP*/*REST* ambos possuem vantagens e desvantagens a serem consideradas para o processo de integração de sistemas.

Na problemática apresentada no item 4.1 tem-se um volume considerável de registros a serem importados. O quadro abaixo apresenta a diferença entre a utilização de cada um dos padrões no desenvolvimento da solução que visa integrar os sistemas descritos.

Quadro 2 - Resumo de padrões

PADRÃO	SOAP	REST
Transporte de dados	Complexo, pois necessita empacotar em um envelope os registros de acordo com o padrão.	Simplificado, por não possuir um padrão determinado para os dados.
Tipo de dados	*XML*	Qualquer tipo: *XML*, *JSON* ou qualquer outra opção.
Modelo de Arquitetura	**É um protocolo reconhecido pela** *W3C*.	Conjunto de boas práticas na modelagem de web services.
Funcionalidades	Possui diversas previstas no *Web Service Standards*.	Determina o básico dos serviços.

Fonte: Adaptado de Web (2015)

Ao considerar o padrão *SOAP* para integração dos sistemas deve se levar em conta o volume de dados a serem importados, pois como o mesmo está atrelado um rígido e inflexível modelo de arquitetura, pode-se comprometer o desempenho e a performance da estrutura de rede tornando-a muito lenta em decorrência do tempo de resposta para processar os registros a serem importados.

O padrão *REST* por outro lado apresenta condições mais favoráveis para resolução da problemática apresentada devido ao fato de ser mais ágil e leve em virtude de não seguir um modelo de arquitetura rígido, no qual é possível determinar qual melhor tipo de dados para situação descrita anteriormente e por se basear em um conjunto de boas práticas para construção de *web services*.

Liu et al. (2008) representa as trocas de dados entre as entidades e as suas interações por meio de *URI's*. Como citado anteriormente, a importação de dados do sistema de gerenciamento acadêmico para o ambiente virtual de aprendizagem normalmente envolve as seguintes entidades: cursos, disciplinas, turmas, matrículas e usuários. Baseado na comunicação por JSON por meio de mapeamento de *URI's* padronizadas, torna-se possível a construção de um *web service* capaz de realizar as seguintes operações: relação de generalização, relação de agregação, relação de composição e relação de associação.

Na tabela abaixo podemos identificar como essas relações serão estabelecidas, bem como, propor um modelo genérico de *URI's* para criação de um *web services* baseado em *JSON* e *REST*.

Tipo de Relação	URI
Generalização Regra de Mapeamento: "/" Representa generalização	http://academico.com/usuario/discente http://academico.com/usario/professor
Agregação Regra de mapeamento: ":" representa agregação. Se as entidades são interdependentes as mesmas podem ser representadas separadamente via URI	http://academico.com/turma http://academico.com/turma/discente http://academico.com/discente
Composição Regra de Mapeamento: ":" Representa composição.	http://academico.com/academia http://academico.com/academia:departamento
Associação Regra de Mapeamento: "-" Representa associação.	http://academico.com/usuario-curso

Fonte: Adaptado de Liu et al. (2008)

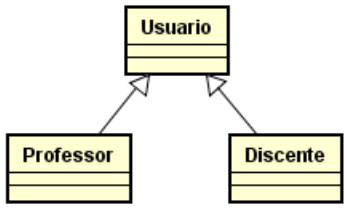
O relacionamento de generalização representa uma hierarquia e cada entidade nesse relacionamento tem um pedaço de informação do escopo. O tipo de

relação agregação representa o relacionamento típico todo/parte, sendo o relacionamento de composição exatamente igual à agregação, exceto que a "parte" é controlada pelo "todo" e não podem existir independentemente e a o tipo de relação associação representa uma entidade associada a outra.

Ensaios iniciais foram realizados a fim de atestar simplicidade da metodologia destacada por Liu et al (2008). Os experimentos descritos a seguir fizeram uso das mesmas bases de dados utilizadas no trabalho intitulado "Integração de sistemas de gestão acadêmica e ambiente virtual de aprendizagem: uma abordagem orientada a banco de dados", dos mesmos autores, apresentado no XIII Congresso Brasileiro de Ensino Superior a Distância e II Congresso Internacional de Educação Superior a Distância.

Elaboramos um projeto simples de serviços baseados nas ações fundamentais de um *CRUD*[8] em alusão aos métodos *HTTP*. Como resultado de consultas efetuadas em um banco de dados de um sistema de gestão acadêmica genérico contendo as tabelas elencadas no quadro 01 por meio de URI's obteve-se como resultado os seguintes arquivo JSON para os casos que se seguem:

Exemplo 1:

Tipo de Relação	Regras de Mapeamento	URI
Generalização	"/" Representa generalização.	http://academico.com/usuario/discente http://academico.com/usario/professor
		

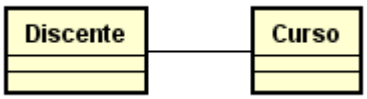
Resultados das URI's:

{"/professor"[{"matricula":355,"nome":"Athos","sobrenome":"Eulalio","email":"athos.denis@ifpi.edu.br"}{"matricula":356,"nome":"Rodrigo","sobrenome":"deSouza","email":"pmsrodrigo@gmail.com"}]}

{"/discente"[{"matricula":1026729,"nome":"anderson","sobrenome":"dias","email":"modifique@seuprovedor.com","cidade":"Cristalandia"},{"matricula":1026733,"nome":"cleiton","sobrenome": "custodio","email":"seraine-custodio@yahoo.com.br","cidade": Corrente"},{"matricula":1026739,"nome":"eduardo","sobrenome":"cunha","email":"eduardomagnata2010@hotmail.com","cidade":"-Corrente"},{"matricula":1026737,nome":"diego","sobrenome":"souza","email":"

diegoguedes@gmail.com","cidade":"Corrente"},{"matricula":1026749,"nome":"helielson","sobrenome":"lima","email":"modifique@seuprovedor.com","cidade":"Parnagua"},{"matricula":1026765,"-nome":"moris","sobrenome":"lima","email":"morislima@hotmail.com","cidade":"Parnagua"},{"matricula":1026756,"nome":"jose","sobrenome":"valle","email":"modifique@seuprovedor.com","cidade":"Corrente"},{"matricula":1026772,"nome":"salvador","sobrenome":"rocha","email":"saroch@bol.com.br","cidade":"Corrente}] }

Exemplo 2:

Tipo de Relação	Regras de Mapeamento	URI
Associação	"-" Representa associação.	http://academico.com/usuario-curso
		

Resultado da URI:

{"**discente-curso**" [{"matricula":

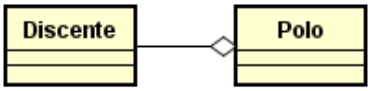
1026729,"curso":"Tecnico de Informatica"{"matricula":1026733, "curso": "Tecnico de Informatica"}, {"matricula": 1026739,

"curso": "Tecnico de Informatica"},{"matricula":1026737, "curso":"Tecnico de Informatica"},{"matricula": 1026749, "curso": "Tecnico de Informatica"}, {"matricula": 1026765, "curso": "Tecnico de Informatica"}, {"matricula": 1026756,"curso": "Tecnico de Informatica"},{ "matricula": 1026772, "curso": "Tecnico de Informatica"}]}

[8] CRUD: acrônimo de Create, Read, Update e Delete na língua Inglesa) para as quatro operações básicas utilizadas em bases de dados relacionais: Correspondentes aos cabeçalhos GET, PUT, POST, DELETE no protocolo HTTP.

Exemplo 3:

Tipo de Relação	Regras de Mapeamento	URI
Agregação	":"Representa agregação.	http://academico.com/discente:polo

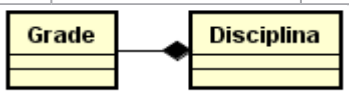


Resultado da URI:

{"**discente:polo**":[{"Matricula": "1026729","Polo": "Corrente- PI"},{"Matricula": "1026733","Polo": "Corrente-PI"},{"Matricula": 1026737","Polo": "Corrente-PI"},{"Matricula": "1026739","Polo": "Corrente-PI"},{"Matricula":"1026749","Polo":"CorrentePI"},{"Matricula":"1026756","Polo": "Corrente-PI"},{"Matricula": "1026765","Polo": "Corrente-PI"},{"Matricula": "1026772","Polo":"Corrente-PI"}]}

Exemplo 4:

Tipo de Relação	Regras de Mapeamento	URI
Composição	":" Representa composição.	http://academico.com/grade:disciplina



Resultado da URI:

{"**grade:disciplina**":[{"Grade": "Modulo IV","Disciplina": "Relações Interpessoais"},{"Grade": "Modulo IV","Disciplina": "Autoria Web"

},{"Grade": "Modulo IV","Disciplina": "Aplicativos Web"},{"Grade": "Modulo IV","Disciplina": "Programação Web"},{"Grade": "Modulo IV",

"Disciplina": "Projeto de Sites"},{"Grade": "Modulo IV","Disciplina": "Ética e Trabalho"}]}

5 Conclusões

Este trabalho introduziu algumas das características presentes nos padrões de arquiteturas *SOAP*/*REST* para auxiliar a escolha do desenvolvedor sobre qual padrão adotar no desenvolvimento de suas aplicações, por meio de *web services*, para que estas sejam capazes de atender aos requisitos de integração entre sistemas de informação e interoperabilidade entre eles, assim como apresentou-se algumas funcionalidades de ambas arquiteturas que visam a definição de qual formato de dados (*XML*, *JSON*) seria o mais adequado para o problema de integração em questão.

A opção sobre qual arquitetura optar no momento do desenvolvimento de aplicações tem gerado discussões, pois ambas apresentam características favoráveis e desfavoráveis de acordo com alguns aspectos a serem considerados:

O padrão *SOAP* possui um arcabouço estrutural bem definido e robusto que lhe garante confiabilidade, sendo o mesmo amparado pelo *W3C* como protocolo padrão para o desenvolvimento de *web services*. No entanto, a grande quantidade de serviços que precisa ser gerenciada lhe confere complexidade para desenvolver e sua performance depende do servidor onde o serviço está publicado, como também da rede.

O padrão *REST* é tido como simples para desenvolvimento de aplicações e de boa performance, e a possibilidade de adoção de qualquer formato de dados lhe confere flexibilidade, porém para aplicações críticas e de alta disponibilidade sua performance pode ser prejudicada e a falta de padrão para desenvolvedores dificulta a criação de um modelo padrão para arquitetura.

Em síntese, *SOAP* é uma boa opção para desenvolvimento de aplicações com padrões rígidos e complexos, pois possui um conjunto de parâmetros e especificações previstos em seu *Web Service Standards*. O *REST* por ser mais leve e dinâmico pode ser utilizado para aplicações web service que exigem rapidez para serem desenvolvidas e quanto o requisito de performance não for exigido.

Os experimentos iniciais conduzidos neste trabalho demonstraram a viabilidade para utilização das tecnologias *web services* por meio de *REST* e *JSON* para uma problemática comum a diversas instituições de ensino superior para integrar sistemas de gestão acadêmica e AVA

Importante destacar, o padrão *REST* tem crescido como padrão para desenvolvimento de soluções baseadas em *web services* dado sua simplicidade e flexibilidade.

Referências

[1] GOMES, D. A. **Web services SOAP em Java: Guia prático para o desenvolvimento de web services Java**. 2.ed. São Paulo: Novatec, 2014.

[2] MITCHEL, L. J.; **Web services em PHP. APIs para web moderna.** São Paulo: O'Reilly, 2013.

[3] KALIN, M.; **Java web services: Implementando. Uma introdução rápida, prática e completa**. São Paulo: O'Reilly, 2010.

[4] LIU, Y; WANG, Q; ZHUANG M; ZHU Y; **Re-engineering Legacy Systems with RESTful Web Service.** Annual IEEE International Computer Software and Applications Conference. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4591667>. Acesso em: 10 out. 2016.

[5] RICHARDSON, L; RUBY, S.; **RESTful Serviços Web: web services para o mundo real**. São Paulo: O'reilly. 2007.

[6] **SOA aplicado: Integrando com web services e além.** Disponível em: <http://www.devmedia.com.br/ferramentas-de-integracao-com-soa/29214#ixzz3hzMtfBtp>. Acesso em: 03 ago. 2015.

[7] **ARQUITETURA REST: um estudo de sua implementação em linguagens de programação.** Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_673/TCCRicardoGhisi.pdf>. Acesso em: 03 ago. 2015.

[8] **CONHECENDO o modelo arquitetural REST.** Disponível em: <http://www.devmedia.com.br/conhecendo-o-modelo-arquitetural-rest-engenharia-de-software-magazine-58/28052>. Acesso em: 05 ago. 2015.

[9] W3 Schools. **WS Soap example.** Disponível em: <http://www.w3schools.com/webservices/ws_soap_example.asp>. Acesso em: 05 ago. 2015.

[10] **WEB Services REST versus SOAP**

<http://www.devmedia.com.br/web-services-rest-versus-soap/32451>. Acesso em: 22 ago. 2015.