Universidade Federal do Rio de Janeiro
Programa de Pós-Graduação em Informática

MAURÍCIO NUNES DA COSTA BOMFIM

INTEGRAÇÃO AUTOMÁTICA DE APLICAÇÕES EXTERNAS EM UM AMBIENTE DE APRENDIZAGEM APOIADO NA WEB 2.0

Rio de Janeiro
2009

MAURÍCIO NUNES DA COSTA BOMFIM

INTEGRAÇÃO AUTOMÁTICA DE APLICAÇÕES EXTERNAS EM UM AMBIENTE DE APRENDIZAGEM APOIADO NA WEB 2.0

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática (PPGI), Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

Orientador: Fábio Ferrentini Sampaio

Rio de Janeiro
2009

B695 Bomfim, Maurício Nunes da Costa.
Integração automática de aplicações externas em um ambiente de aprendizagem apoiado na Web 2.0 / Maurício Nunes da Costa Bomfim. -- Rio de Janeiro, IM/NCE/UFRJ, 2009.
223f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, Programa de Pós-graduação em Informática, 2009.

Orientador: Fábio Ferrentini Sampaio.

1. Integração de Aplicações – Teses. 2. Ambientes de Aprendizagem – Teses. I. Fábio Ferrentini Sampaio (Orient.). III. Universidade Federal do Rio de Janeiro. Instituto de Matemática. Núcleo de Computação Eletrônica. IV. Título.

CDD. .

MAURÍCIO NUNES DA COSTA BOMFIM

INTEGRAÇÃO AUTOMÁTICA DE APLICAÇÕES EXTERNAS EM UM AMBIENTE DE APRENDIZAGEM APOIADO NA WEB 2.0

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática (PPGI), Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

Aprovada em: 3 de março de 2009.

Prof. Fábio Ferrentini Sampaio, Ph.D., NCE e PPGI/UFRJ (Orientador)

Prof ª. Claudia Lage Rebello da Motta, D.Sc., NCE e PPGI/UFRJ

Prof. Carlo Emmanoel Tolla de Oliveira, Ph.D., NCE e PPGI/UFRJ

Prof ª. Flávia Maria Santoro, D.Sc., UNIRIO

Para Sonia e Tomás

Agradecimentos

Gostaria de agradecer, em primeiro lugar, ao grande amigo e orientador Prof. Fábio Ferrentini Sampaio pelo incentivo, dedicação e competência demonstrados durante a realização deste trabalho.

A todos os demais professores deste Programa de Pós-Graduação com quem tive a oportunidade de conviver durante estes anos, pelos inúmeros ensinamentos que me foram transmitidos, proporcionando o meu desenvolvimento acadêmico. Entre eles, um agradecimento especial à Profª Claudia Lage Rebello da Motta e ao Prof. Marcos Elia da Fonseca pelas valiosas contribuições dadas a esta dissertação.

Ao Prof. Carlo Emmanoel Tolla de Oliveira que em 2006 plantou no grupo GINAPE a idéia de explorar o potencial da Web2.0 como ferramenta para auxiliar o ensino e a aprendizagem, e participou ativamente deste trabalho com sugestões, sem as quais esta dissertação não teria tomado o caminho que tomou. Ao pessoal do LABASE pelo apoio para a hospedagem do AvaNCE nos servidores do laboratório.

À Profª Flavia Maria Santoro por participar da banca examinadora desta dissertação, também contribuindo para o seu enriquecimento.

Aos colegas do Núcleo de Computação Eletrônica, João Sergio dos Santos Assis pela constante ajuda, não só na implementação do AvaNCE, mas também com inúmeras sugestões durante todo o processo de desenvolvimento deste trabalho, Juarez de Faria Castro pela criação do logo do AvaNCE, e Selma Regina Mendes Martins pela ajuda na normalização desta dissertação.

Resumo

BOMFIM, Maurício Nunes da Costa. **Integração automática de aplicações externas em um ambiente de aprendizagem apoiado na Web 2.0.** 2009. 223f. Dissertação (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

A incorporação das idéias da Web 2.0 – que envolvem um maior grau de interatividade e colaboração na utilização da Internet – nos Ambientes Virtuais de Aprendizagem parece ser um caminho a ser seguido, dando aos aprendizes a oportunidade de desempenhar um papel mais ativo no processo de aprendizagem. Entre as principais características de um ambiente deste tipo pode-se mencionar a liberdade dada ao usuário para agregação de conteúdos e funcionalidades de diferentes ferramentas e serviços oferecidos pela Web, assim como a possibilidade de exportar seus próprios serviços para outras aplicações. Entretanto, não existem padrões suficientes para que o processo de integração de aplicações possa ser realizado de maneira totalmente genérica. Neste cenário, a adoção do estilo de arquitetura REST poderia ser considerada como um facilitador, mas, como constatado pelo estudo exploratório realizado no escopo deste trabalho, as aplicações existentes aderem apenas parcialmente ao REST, dificultando a definição de uma abordagem genérica que pudesse ter utilização prática. Este trabalho propõe como solução para este problema, um modelo de integração, baseado numa camada intermediária, que permite a incorporação automática de aplicações Web 2.0, a partir da descrição formal de suas APIs. Entre outras aplicações, este modelo pode ser utilizado num ambiente de aprendizagem, onde o usuário seja capaz de integrar aplicações escolhidas livremente de maneira simples e fácil. Como prova de conceito que demonstre a viabilidade da aplicação desta solução, foi desenvolvido um protótipo do ambiente proposto incluindo suas principais funcionalidades, onde algumas aplicações puderam ser integradas.

Abstract

BOMFIM, Maurício Nunes da Costa. **Integração automática de aplicações externas em um ambiente de aprendizagem apoiado na Web 2.0.** 2009. 223f. Dissertação (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

The incorporation of Web 2.0´s ideas - which involve a greater degree of interactivity and collaboration in the use of the internet - in Virtual Learning Environments seems to be a path to be followed, giving students the opportunity to play a more active role in the learning process. Among the main features of such environments, it can be mentioned the freedom given to the user so he can aggregate content and features of different services offered by the Web, as well as the possibility of exporting their own services to other applications. However, the existing standards do not permit the application's integration in a generic way. In this scenario, the adoption of REST architectural style could be regarded as a facilitator, but as found by the exploratory study conducted in the scope of this work, the present applications adhere only partially to REST, making it difficult to define a generic approach that could have any practical use. This work proposes an integration model, based on an intermediate layer, as a solution to this problem. This approach allows an automatic incorporation of Web 2.0´s applications using their API´s formal descriptions. Among other possibilities, this model can be used in a learning environment where the user can integrate new applications in a very easy way. As a proof of concept to demonstrating the feasibility of this solution, a prototype of the proposed environment was developed, where some applications were integrated.

Lista de Figuras

Lista de Quadros

Lista de Siglas

APA	Ambiente Pessoal de Aprendizagem
API	Application Programming Interface
AVA	Ambiente Virtual de Aprendizagem
CC	Creative Commons
CORBA	Common Object Request Broker
CSS	Cascading Style Sheets
DCOM	Distributed Component Object Model
DOM	Document Object Model
ECMA	European Computers Manufacture Association
GINAPE	Grupo de Informática Aplicada à Educação
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IM	Instituto de Matemática
J2EE	Java2 Enterprise Edition
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
MVC	Model-View-Controller
NCE	Núcleo de Computação Eletrônica
PLE	Personal Learning Environment
PPGI	Programa de Pós-Graduação em Informática
RDF	Resource Description Framework
REST	REpresentational State Transfer
RPC	Remote Procedure Call
RSS	Rich Site Summary; RDF Site Summary; Really Simple Sindication
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TIC	Tecnologia da Informação e Comunicação
UDDI	Universal Description Discovery Integration
UFRJ	Universidade Federal do Rio de Janeiro
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WADL	Web Application Description Language

WSDL	Web Services Description Language
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XRI	eXtensible Resource Identifier
XSLT	eXtensible Stylesheet Language Transformations

Sumário

Capítulo 1
Introdução

Este capítulo apresenta sucintamente a pesquisa documentada nesta dissertação, ressaltando suas motivações e justificativas, o problema abordado, seus objetivos e a organização deste texto.

1.1. Motivação e Justificativas

Favorecida pelos avanços tecnológicos dos últimos anos, pela disseminação dos computadores pessoais e pela popularização da Internet, a utilização de Ambientes Virtuais de Aprendizagem (AVAs) como ferramentas de apoio ao ensino e aprendizado tem sido uma prática cada vez mais freqüente. Estes ambientes, apesar de se apoiarem em propostas pedagógicas inovadoras, acabam por implementar, na prática, modelos convencionais de sala de aula presencial. Podemos dizer que, neste cenário, os AVAs são ferramentas institucionais onde o objetivo principal é auxiliar o oferecimento de um curso, através de um conjunto de ferramentas integradas que, entre outras funcionalidades, possibilitem ao professor a distribuição de materiais didáticos para os alunos e a sua comunicação com a turma.

Vivemos um momento em que é muito rápida a evolução da tecnologia e do conhecimento, fazendo com que o aprendizado informal e continuado seja cada vez mais valorizado. Paralelamente a isso estamos diante de uma nova geração de alunos, nascida na era digital para a qual seria muito natural a utilização de novas formas de aprender baseadas no uso do computador através de aplicações como: mensagens instantâneas, redes sociais, Blogs, Wikis e compartilhamento de arquivos (ROLLET et al, 2007; ANDERSON, 2007).

Esta nova geração de aplicativos da Web, assim como as tecnologias e conceitos que permitem um maior grau de interatividade e colaboração na utilização da Internet, conhecida como Web 2.0 (O'REILLY, 2005), têm influenciado, um número cada vez maior de serviços disponíveis na Web. Tais conceitos, se incorporados aos ambientes de aprendizagem, poderiam permitir ganhos no processo de aprendizado, uma vez que vão ao encontro das idéias construtivistas e sócio-interacionistas, onde o conhecimento é obtido através de uma construção contínua, como fruto de interações entre os objetos do meio e o sujeito (VYGOTSKY, 1998).

Este novo cenário sugere que, para um ambiente de aprendizagem ser capaz de atender a estas necessidades, é preciso uma mudança de paradigma. Os ambientes devem passar a ser mais centrados no aluno, permitindo a agregação de conteúdos de diferentes ferramentas e serviços escolhidos pelo usuário, sejam eles provedores de educação formal ou provenientes de fontes informais de conhecimento.

Na tentativa de desenvolver um modelo mais adequado, que permita o apoio à aprendizagem informal e continuada, e que seja baseado nestas idéias, autores como Wilson et al (2006), Milligan et al (2006), Harmelen (2006), Feldstein e Masson (2006) e Fumero et al (2006) propõem

a utilização de Ambientes Pessoais de Aprendizagem (APAs) onde o aluno seja capaz de controlar seu próprio aprendizado. Entre as principais características de um APA, podemos mencionar: a agregação de conteúdos e funcionalidades de diferentes ferramentas e serviços da Web escolhidos pelo próprio usuário; a existência de mecanismos para incentivar o compartilhamento e a publicação; e a possibilidade da sistematização do conhecimento adquirido através da criação de portfólios. Segundo Downes (2005) algumas características desta nova abordagem são: estímulo à comunicação, colaboração e compartilhamento; personalização e possibilidade de uso de ferramentas externas e exploração da inteligência coletiva.

Este panorama nos conduz a algumas premissas nas quais esta pesquisa se baseou e que são enumeradas a seguir:

- Do ponto de vista do ensino, a utilização de conceitos da Web 2.0 permite a criação de ambientes mais ativos que os AVAs tradicionais, adequando-se melhor às idéias das teorias construtivista e sócio-interacionista;

- Estes ambientes devem permitir que os usuários possam agregar funcionalidades de aplicações externas;

- A convivência entre os AVAs e os APAs será facilitada se os AVAs passarem a exportar seus serviços através de APIs abertas de forma que os mesmos possam ser incorporados pelos APAs.

1.2. Problema

Como conseqüência de sua atuação no desenvolvimento de ambientes de apoio ao ensino, tais como as plataformas Pii – Plataforma Interativa para Internet (ELIA e SAMPAIO, 2001) – e TW – TeamWorks (MOTTA e BORGES, 1999) – o Grupo de Informática Aplicada à Educação (UFRJ/PPGI/GINAPE) vem, desde o ano de 2006, trabalhando no sentido de produzir um ambiente que incorpore os conceitos e tecnologias da Web 2.0, e que possa ser utilizado como um ambiente pessoal pela comunidade de alunos, docentes e pesquisadores da UFRJ.

Entre os requisitos principais do ambiente a ser desenvolvido, podemos citar:

- Poder atender também às necessidades da aprendizagem informal e continuada, através de sua utilização como um APA;

- Ser expansível a partir da integração automática de novos serviços, de acordo com as necessidades específicas de cada usuário;

- Poder ser utilizado como um AVA, capaz de exportar seus próprios serviços através de uma API aberta, permitindo sua integração em outros ambientes.

Um problema crucial para o qual precisamos buscar uma solução durante o desenvolvimento deste projeto foi: como viabilizar a criação de um ambiente de integração de aplicações que fosse de fácil utilização por usuários não técnicos, e que permitisse a incorporação de qualquer aplicação escolhida livremente?

O principal objetivo deste trabalho, portanto, foi propor uma solução para este problema, na forma de um modelo para integração de aplicações externas em um ambiente de aprendizagem apoiado na Web 2.0. Sendo assim, existem duas questões principais que foram investigadas no contexto dessa dissertação, uma geral e outra específica. A questão geral indaga se é possível desenvolver um ambiente de aprendizagem baseado neste novo paradigma e que seja capaz de atender aos requisitos mencionados acima. Em caso afirmativo, podemos questionar, mais especificamente – qual abordagem utilizar para facilitar o processo de integração de forma que os usuários possam escolher livremente quaisquer aplicações, sem a necessidade de escrever código de programação.

Considerando que a incorporação de aplicações existentes através de suas APIs e o desenvolvimento de Mashups são atividades complexas e que necessitam de conhecimentos de programação, algumas abordagens freqüentemente utilizadas que visam simplificar este processo para o usuário final são os editores de Mashups (como o Yahoo Pipes[1]) e o uso de plataformas de agregação de Widgets (como o Netvibes[2]). Estas soluções, entretanto, não são totalmente genéricas, uma vez que apresentam apenas algumas fontes de dados predefinidas. Para incorporar aplicações diferentes daquelas oferecidas, é necessário programar algum componente, tarefa que exige conhecimentos técnicos que normalmente o usuário final não possui.

A emergência de alguns padrões como RSS, Atom e XForms, além da aderência aos princípios definidos pelo estilo de arquitetura REST podem facilitar a incorporação de aplicações, mas ainda existem diversas APIs que não são completamente alinhadas a estas tendências, implementando parcialmente os preceitos REST. Sendo assim, este trabalho apresenta uma solução alternativa

[1] http://pipes.yahoo.com
[2] http://www.netvibes.com

para a incorporação automática de aplicações, baseada na utilização de descrições formais de APIs REST, sejam elas total ou parcialmente aderentes a este estilo de arquitetura.

A hipótese dessa dissertação é que a definição de uma abordagem genérica de integração automática de aplicações Web 2.0 a partir de suas APIs pode facilitar a criação de um ambiente de aprendizagem onde os alunos sejam capazes de escolher suas próprias ferramentas, favorecendo com isso, o aprendizado com outras pessoas, a gerência das atividades nas quais participam, e a integração dos seus aprendizados em diferentes contextos.

Para possibilitar a realização de um prova de conceito que demonstre a viabilidade da abordagem proposta, fez-se necessário o desenvolvimento de um protótipo de ambiente, denominado AvaNCE, cuja especificação é apresentada neste trabalho, e no qual foi integrada a implementação da solução proposta, bem como um conjunto de funcionalidades suficiente para proceder estes testes.

1.3. Metodologia

Esse trabalho foi realizado seguindo os procedimentos metodológicos que orientam a realização de uma pesquisa científica, nas seguintes etapas:

A primeira etapa compreendeu a realização de uma revisão bibliográfica em artigos científicos, teses e dissertações sobre as tecnologias da Web 2.0 e suas possibilidades educacionais. Esse procedimento teve como objetivo aprofundar os conhecimentos teóricos referentes ao assunto, no sentido de obter a fundamentação teórica necessária para a pesquisa proposta, permitindo a definição do problema e a formulação de uma hipótese de solução.

A segunda etapa compreendeu a realização de um estudo exploratório com vistas a classificar algumas APIs da Web 2.0, o que indicou um caminho a ser seguido no desenvolvimento de uma proposta de solução para o problema de pesquisa. Essa etapa foi seguida do desenvolvimento de um protótipo, onde a proposta de solução pode ser integrada.

A etapa final foi a realização de uma prova de conceito, a fim de demonstrar a viabilidade da solução proposta através da integração de algumas aplicações existentes.

A Figura 1.1 apresenta um diagrama que ilustra o processo de desenvolvimento da pesquisa apresentada nesta dissertação.

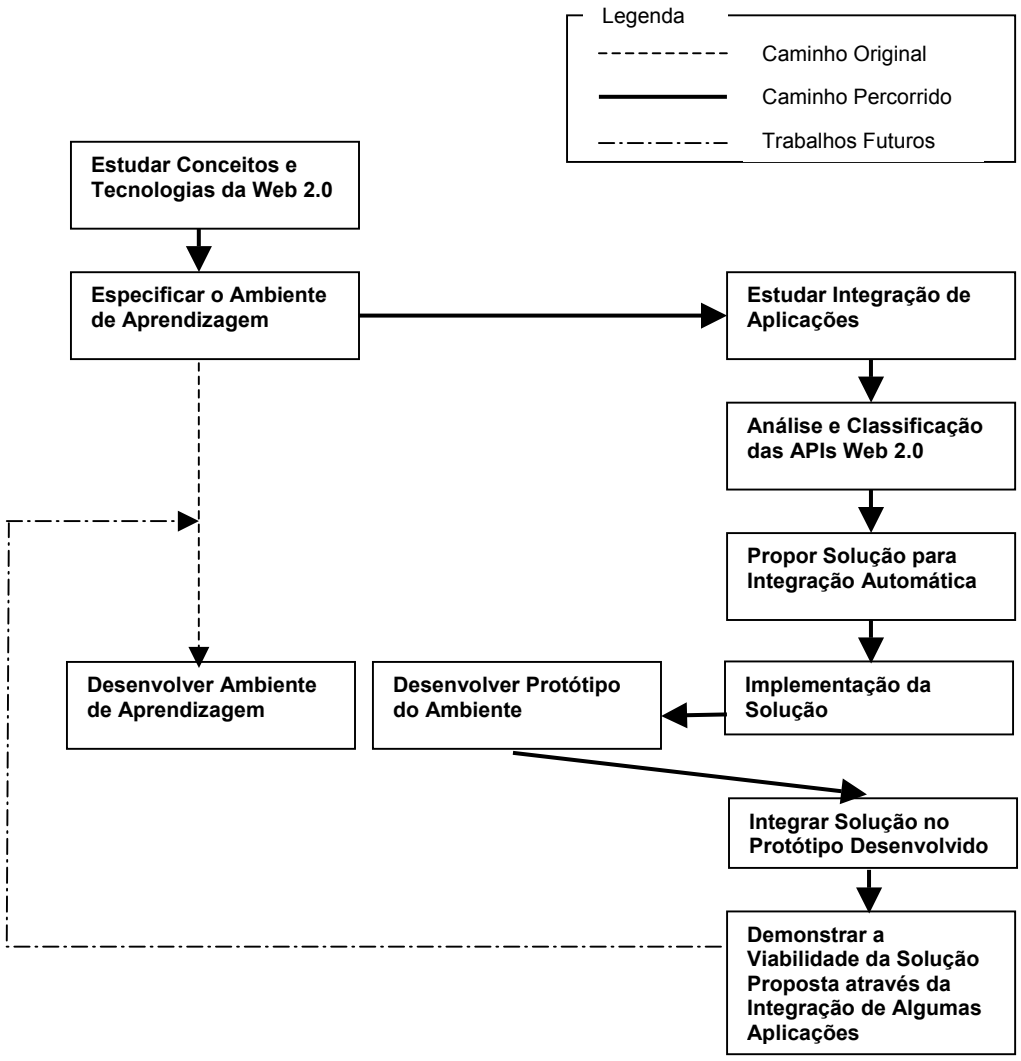

Figura 1.1. Processo de desenvolvimento da pesquisa apresentada nesta dissertação

1.4. Organização da Dissertação

O Capítulo 2 apresenta os princípios básicos da Web 2.0, suas principais aplicações educacionais, assim como uma reflexão das teorias de aprendizagem que podem apoiá-las. Segue-se uma discussão sobre os Ambientes Pessoais de Aprendizagem, outro conceito relacionado com o uso educacional da Web 2.0.

O Capítulo 3 apresenta uma visão geral das tecnologias da Web 2.0. Para isto elas foram agrupadas nas seguintes categorias de acordo com sua utilização neste trabalho: desenvolvimento de aplicações ricas, alternativas para componentização, utilização de padrões e a autenticação de usuários.

O Capítulo 4 faz um estudo e classificação das APIs da Web 2.0 de acordo com o grau de adesão de cada uma delas à arquitetura REST e com os seus mecanismos de autenticação, buscando identificar padrões de comportamento que permitissem a proposição de uma solução para

integração automática. Suas conclusões serviram como base para a definição do modelo de integração apresentado a seguir.

O Capítulo 5 apresenta um modelo de integração de aplicações onde é possível incorporar aplicações de forma automática a partir da descrição no formato WADL de suas APIs. Este modelo permite a criação de um ambiente de aprendizagem baseado na Web 2.0.

O Capítulo 6 apresenta o desenvolvimento do AvaNCE, um protótipo do ambiente de aprendizagem proposto no Capítulo 5. Este ambiente é capaz de integrar aplicações externas automaticamente e de exportar as suas próprias aplicações através do uso de formatos padronizados, favorecendo assim a incorporação de seus serviços em outros ambientes.

O Capítulo 7 apresenta uma prova de conceito que demonstre a viabilidade da solução proposta. Assim, para exemplificar o funcionamento do mecanismo de integração proposto nesta dissertação e implementado no ambiente AvaNCE, foram descritas as APIs de algumas aplicações Web 2.0 típicas de forma a permitir que as mesmas sejam incorporadas ao ambiente. Além disso, foi realizado um experimento de integração de serviços do AvaNCE em um ambiente externo.

O Capítulo 8 apresenta o resumo da pesquisa descrita nessa dissertação, enfatizando as principais contribuições e sugestões para prosseguimento deste trabalho.

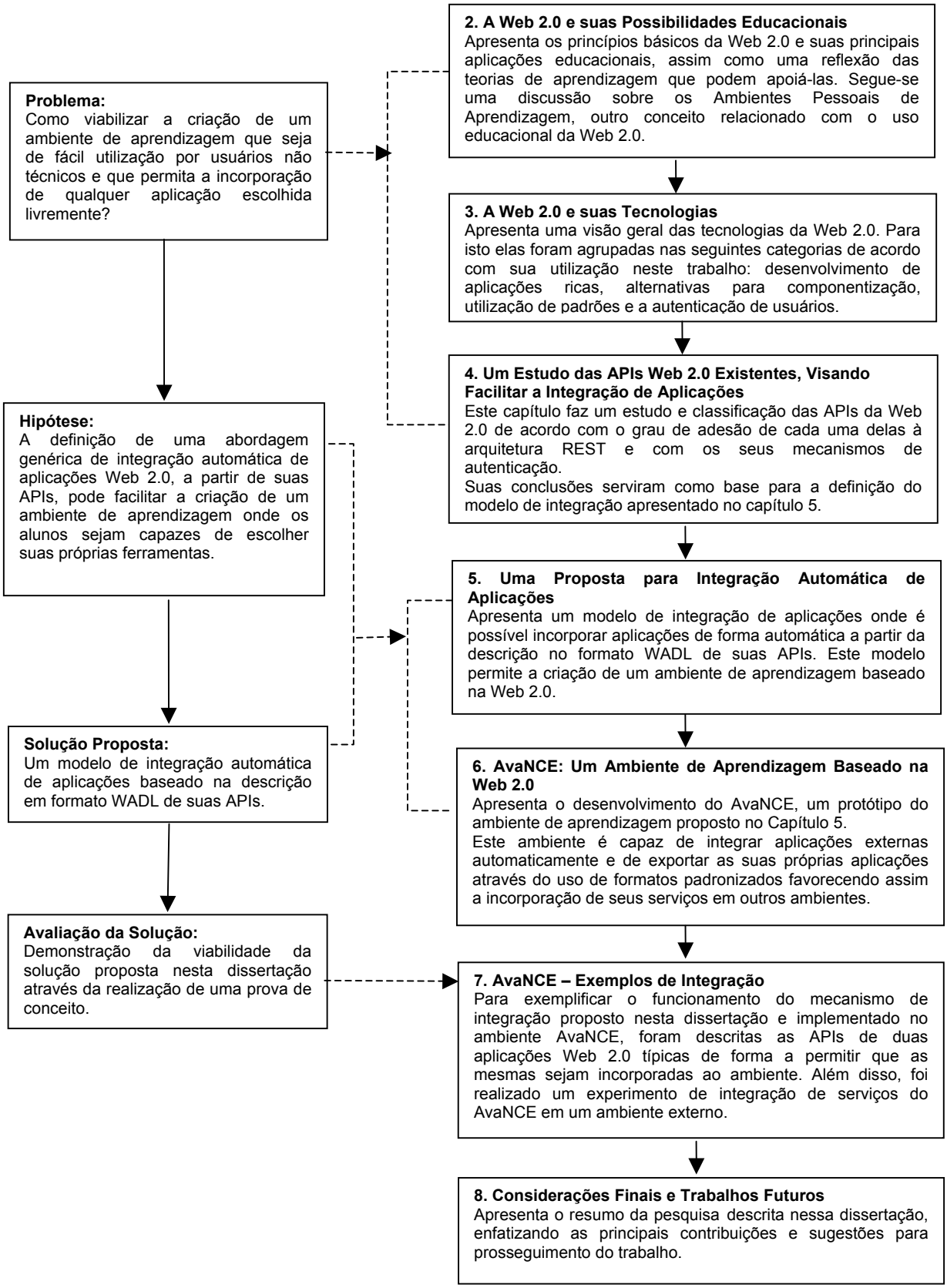

Figura 1.2. Apresentação da dissertação em capítulos

Capítulo 2
A Web 2.0 e suas Possibilidades Educacionais

O objetivo deste capítulo é apresentar as principais aplicações educacionais da Web 2.0. Para isso, serão abordados os seus princípios básicos, assim como uma reflexão sobre as teorias de aprendizagem que podem apoiar as suas utilizações educacionais. Segue-se uma discussão sobre alguns exemplos de aplicações típicas da Web 2.0 que, potencialmente, podem ser utilizadas para promover o aprendizado, culminando com os Ambientes Pessoais de Aprendizagem, outro conceito relacionado com o uso educacional da Web 2.0.

Muito se tem dito sobre uma nova geração de serviços e aplicativos da Web e dos recursos, tecnologias e conceitos que permitem um maior grau de interatividade e colaboração na utilização da Internet. Para designar este fenômeno, Tim O'Reilly cunhou o termo Web 2.0 que popularizou-se rapidamente a partir da publicação do artigo intitulado "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software" (O'REILLY, 2005).

Embora possa parecer que este termo esteja associado a uma grande mudança tecnológica, ele na verdade se refere apenas a uma nova forma com que a Web vem sendo encarada por seus usuários e desenvolvedores. Não existe uma nova versão, afinal a maior parte das tecnologias envolvidas neste processo já existe há alguns anos, elas só passaram a ser utilizadas de uma forma diferente. O que existe é um conjunto de princípios e práticas comuns em algumas aplicações e serviços na Internet, bastante populares hoje em dia[3]. É a este triângulo composto de princípios, tecnologias e aplicações, que comumente se chama de Web 2.0 (BOMFIM e SAMPAIO, 2008).

Estamos vivendo uma transição para uma Web onde os usuários deixam de ser apenas receptores de informação, para serem participantes mais ativos no processo de produção de conteúdos. Assim, a Web 2.0 pode ser considerada como um fenômeno social, caracterizado pela descentralização, pelo compartilhamento e pelo reuso, na qual são valorizados o conteúdo colaborativo e a inteligência coletiva. O conteúdo deve ser produzido e consumido por qualquer um, de forma simples e direta.

2.1. Princípios da Web 2.0

Segundo O'Reilly, a Web 2.0 é baseada em alguns princípios que podem ser enunciados como: "a Web como plataforma", "tirando partido da inteligência coletiva", "os dados de uma aplicação são o seu verdadeiro valor", "modelos leves de programação", "usuários agregam valor" e "experiência rica do usuário". A seguir serão discutidos cada um deles.

2.1.1. A Web como plataforma

A Web é uma fonte de conteúdo e funcionalidades, uma plataforma servindo aplicações a usuários finais. Em lugar de comercializar sistemas, o desenvolvedor publica-os na Web para que sejam utilizados livremente.

---

[3] Entre estes serviços, é possível citar: o Google (http://www.google.com), a Wikipedia (http://wikipedia.org), e a Amazon (http://www.amazon.com).

O software agora é apresentado como um serviço e não um produto, acarretando mudanças fundamentais no modelo de negócio das companhias e o fim do seu ciclo de versões. Como não existe uma versão final a ser distribuída, as ferramentas estão sempre em produção, sendo continuamente testadas por um grande número de usuários. Em oposição ao que acontece com softwares tradicionais, com instaladores e dependentes de um sistema operacional, aplicativos Web podem ser atualizados de forma automática e independente da ação do usuário final. Assim, a atividade de suporte ao serviço passa a ser tão importante quanto o desenvolvimento.

As aplicações Web não se limitam mais à plataforma PC. O princípio da Web como Plataforma estende a idéia de aplicações formadas por serviços que são providos por múltiplos computadores. Além do PC como dispositivo para acesso à Internet, é possível utilizar qualquer equipamento que esteja conectado, como computadores de bolso (PDAs), celulares e ipods.

Esta é uma das áreas da Web 2.0 onde se espera que ocorram as maiores mudanças, à medida que aumente a variedade de dispositivos conectados.

2.1.2. Tirando partido da inteligência coletiva

A inteligência coletiva é um termo utilizado pelo filósofo Pierre Lévy para designar um princípio onde as inteligências individuais são somadas e compartilhadas por toda a sociedade, potencializadas com o advento de novas tecnologias de comunicação, como a Internet (LÉVY, 1998).

O conteúdo da Web 2.0 depende das pessoas que estão conectadas a ela, sendo alimentado dos textos, fotos e vídeos que são publicados pelos próprios usuários. Além do incentivo à colaboração, através da facilidade para publicar conteúdos, as aplicações podem tirar partido de seus usuários, aproveitando-se de informações sobre as escolhas realizadas por eles. É possível então identificar padrões de comportamento que possam ser sugeridos aos demais usuários que tenham o mesmo perfil. É como se os sistemas tivessem a capacidade de melhorar à medida que são mais utilizados.

Neste modelo, podemos pensar a existência de uma parceria implícita entre desenvolvedor e usuário onde este se utiliza dos serviços oferecidos sem pagar pelos mesmos diretamente, mas em contrapartida, agrega valor à própria aplicação através de sua utilização.

Outro conceito associado à inteligência coletiva é a folksonomia[4] que, em oposição à taxonomia, é uma maneira de categorizar informações colaborativamente através do uso de palavras-chave definidas livremente, conhecidas como tags. Os precursores na sua utilização foram os sítios del.icio.us[5], flickR[6] e youTube[7], três sucessos da Web 2.0.

2.1.3. Os dados de uma aplicação são o seu verdadeiro valor

Os aplicativos da Web são, cada vez mais, apoiados por grandes bancos de dados especializados. Existem bancos de dados utilizados pelos mecanismos de busca que indexam as referências para as páginas da Internet, bancos de imagens capturadas por satélites e mapas, bancos de produtos e consumidores das grandes lojas virtuais, só para citar alguns. O maior valor destas empresas está nos dados que ela mantém e não no software propriamente dito.

2.1.4. Modelos leves de programação

A Web 2.0 prega a construção de sistemas fracamente acoplados[8] onde as funcionalidades são construídas e agregadas. Entre as principais tecnologias utilizadas para atingir este modelo estão a sindicação e a definição de APIs abertas baseadas em protocolos leves como REST e JSON (ver Capítulo 3).

A sindicação possibilita a utilização de conteúdos de sites por terceiros, em outros ambientes. Baseado em formatos padronizados como RSS ou Atom, as informações são publicadas sem se preocupar com o que será feito com elas em seu destino. Cabe à aplicação requisitante decodificar o arquivo recebido, apresentando-o ao usuário de alguma forma.

O projeto reutilizável através de APIs abertas permite que os aplicativos Web 2.0 incorporem facilmente novos serviços e cedam funcionalidades para serem agregadas por terceiros. As APIs da Web 2.0 normalmente fornecem dados em formato XML via HTTP através de requisições aderentes à arquitetura REST. Este fato permite que estes aplicativos sejam construídos a partir de uma rede cooperativa de serviços de dados, oferecendo interfaces para serviços Web e sindicação de conteúdo e reutilizando os serviços de dados de outros.

---

[4] Do inglês folksonomy, um neologismo criado por Thomas Vander Wal através da combinação das palavras folk e taxonomy.
[5] http://del.icio.us
[6] http://www.flickr.com
[7] http://www.youtube.com
[8] Sistemas fracamente acoplados são aqueles onde módulos interagem entre si através de interfaces estáveis sem que haja a necessidade de um conhecer a implementação interna do outro.

2.1.5. Usuários agregam valor

O projeto de uma "arquitetura de participação" no desenvolvimento do software é fundamental para não restringir o envolvimento dos usuários. Alguns aspectos a serem considerados aqui são:

A proteção da propriedade intelectual restringe o reuso e a experimentação. Portanto, as licenças para os usos dos serviços devem ser as menos restritivas possíveis para incentivar não só a reutilização dos serviços oferecidos, como a própria adoção do público, implicando na agregação de inteligência coletiva. O conjunto de licenças padronizadas para gestão aberta livre e compartilhada de conteúdos e informação Creative Commons (CC)[9] é o modelo normalmente utilizado pelos serviços da Web 2.0 (LIMA e SANTINI, 2006).

A Internet é constituída de um pequeno número de grandes sítios provedores de conteúdo e informação, se comparados a infinidade de pequenos sítios existentes. O poder coletivo destes pequenos sítios que constituem a maior parte do conteúdo da Internet, frente ao pequeno número de grandes usuários é um fenômeno conhecido como cauda longa. A Web 2.0 incentiva o auto-serviço do cliente e a automatização dos processos, de forma a contemplar igualmente aos pequenos e aos grandes usuários.

Apesar de muito se falar no incentivo à participação dos usuários, a realidade é que apenas uma pequena percentagem destes se dará ao trabalho de colaborar adicionando valor às aplicações. Uma estratégia a ser adotada é o fato de, apesar disso, dados poderem ser agregados como efeito colateral à sua utilização. Por exemplo, se a aplicação armazena as pesquisas realizadas por seus clientes numa base de produtos, estas informações podem ser usadas posteriormente num Sistema de Recomendação (RESNICK e VARIAN, 1997) para novos clientes.

2.1.6. Experiência rica do usuário

As tecnologias utilizadas pela Web 2.0 permitem a criação de aplicações Web com interfaces muito mais ricas que as aplicações Web tradicionais. O Ajax é uma combinação de algumas tecnologias, capaz de buscar informações assincronamente no servidor exibindo-as na tela do usuário, sem que esta precise ser totalmente recarregada pelo navegador, representando uma mudança no paradigma de funcionamento das aplicações Web através de requisições e respostas.

---

[9] Mais informações sobre o modelo de licenças Creative Commons podem ser obtidas em http://creativecommons.org.

2.2. Teorias que Apóiam o Uso da Web 2.0 na Educação

Durante o século XX surgiram importantes teorias psicológicas que relacionam a aprendizagem com o desenvolvimento cognitivo, notadamente a Epistemologia Genética de Piaget (1978) e o Sócio-interacionismo de Vygotsky (1998). Estas teorias pressupõem primordialmente o fato de que os indivíduos são agentes ativos que constroem o conhecimento, através da compreensão de novas informações e da incorporação das mesmas a um conhecimento preexistente. A interação entre os indivíduos e o ambiente externo é também um fator que potencializa o aprendizado.

2.2.1. O Construtivismo

O Construtivismo é uma das correntes teóricas que se propõe a explicar como o conhecimento humano se desenvolve partindo do princípio de que o seu desenvolvimento é determinado pelas interações entre o indivíduo e o meio. Assim, diferentemente dos Inatistas que consideram que o conhecimento é pré-formado, ou dos Empiristas, para os quais o conhecimento tem origem e evolui a partir da experiência que o sujeito vai acumulando, segundo as idéias construtivistas o indivíduo responde aos estímulos externos que agem sobre ele para construir e organizar o seu próprio conhecimento, de forma cada vez mais elaborada (GOULART, 1998).

Para Campos et al (2003), o aluno é o sujeito ativo no processo de aprendizagem, por meio da experimentação, da pesquisa em grupo, do estímulo à dúvida e ao desenvolvimento do raciocínio. Os conceitos são formados no contato com o mundo e com outras pessoas. O professor assume o papel de provocador e estimulador de novas experiências e deve ser capaz de propor estratégias ou caminhos para buscar respostas. Uma comparação entre as abordagens tradicional e construtivista de aprendizagem é apresentada no Quadro 2.1.

Quadro 2.1 - Abordagens tradicional e construtivista da aprendizagem (REZENDE, 2002)

ABORDAGEM TRADICIONAL	ABORDAGEM CONSTRUTIVISTA
Enfoque no professor	Enfoque no aluno
Enfoque no conteúdo	Enfoque na construção individual de significados
A mente do aluno funciona como uma "tabula rasa"	A aprendizagem é uma construção do aluno sobre conhecimentos prévios
O aluno é receptor passivo de conhecimento	Ênfase no controle do aluno sobre sua aprendizagem
Memorização de conhecimento	Habilidades e conhecimento são desenvolvidos no contexto onde serão utilizados

Piaget (1978) explica essa interação do indivíduo com o ambiente externo através dos conceitos de assimilação e acomodação. Enquanto a assimilação é a incorporação de um novo objeto ou idéia ao que já é conhecido, a acomodação é a transformação que o indivíduo sofre para poder lidar com o novo conhecimento para o qual ele ainda não possui esquemas de assimilação. O construtivismo, portanto, parte da idéia de que o homem, na verdade, depende de sua interação com o meio num processo que resulta na construção e reconstrução de suas estruturas cognitivas.

2.2.2. O Sócio-interacionismo

De acordo com Vygotsky (1998) a aprendizagem e o desenvolvimento são processos distintos, mas que interagem mutuamente. O aprendizado é influenciado pelas interações sociais cuja confrontação de pontos de vista favorece o desenvolvimento do indivíduo. A teoria de Vygotsky dá uma importância especial às ferramentas sociais como a linguagem assim como às interações e aos relacionamentos entre os indivíduos.

Enquanto Piaget focaliza o indivíduo como unidade de análise, Vygotsky enfoca a interação social. Sua unidade de análise não é nem o indivíduo, nem o contexto, mas a interação entre eles (MOREIRA, 1999).

Um dos conceitos mais importantes presentes na obra de Vygotsky é o de Zona de Desenvolvimento Proximal, que é definida como a diferença entre o que o indivíduo consegue realizar sozinho (nível de desenvolvimento real) e aquilo que é capaz de aprender e fazer com a ajuda de outra pessoa (nível de desenvolvimento potencial). A aprendizagem desperta processos internos de desenvolvimento que só podem ocorrer quando o indivíduo interage com outras pessoas, uma vez que a Zona de Desenvolvimento Proximal é potencializada pela interação social.

Para Campos et al (2003), a aplicação da abordagem de Vygotsky na prática educacional requer que o professor reconheça a idéia de zona de desenvolvimento proximal e estimule o trabalho colaborativo, de forma a potencializar o desenvolvimento cognitivo dos alunos.

2.2.3. Reflexão sobre as teorias sob a luz da Web 2.0

Estamos diante de uma nova geração de estudantes, crescida na era digital, que possui habilidades fundamentalmente diferentes das gerações anteriores (PRENSKI, 2001; OBLINGER, 2003; BROWN, 2007). Tais estudantes, naturalmente, se sentem muito mais motivados a aprender quando utilizam recursos com os quais estão envolvidos no dia-a-dia. Hábitos como, estar

permanentemente conectado com amigos e colegas, a preferência por trabalhos em equipe e por aprender "fazendo" ao invés de "ouvindo", são fatores que levam a crer que a teoria construtivista – onde o aluno é um sujeito ativo no processo de aprendizagem – se constitua numa abordagem pedagógica bastante adequada para estes alunos.

É verdade que a utilização da abordagem construtivista independe da tecnologia, pois é possível utilizá-la tanto na sala de aula tradicional quanto no espaço virtual de um ambiente de aprendizagem. Entretanto a Web 2.0 com suas idéias de participação, colaboração e personalização favorecem de maneira mais natural a adoção destas práticas, por exemplo, através do uso de serviços e aplicativos tais como blogs, wikis e demais softwares sociais (ver Seção 2.3). O construtivismo baseia-se no fato de que o aprendizado ocorre com maior facilidade quando os alunos são ativos - pensando, escrevendo, experimentando, criando e inventando - exatamente como pregam os princípios da Web 2.0. Segundo Brown (2007), enquanto na Web 1.0 o usuário consome informação de uma forma mais passiva, na Web 2.0, com suas folksonomias, com a produção de conteúdo descentralizada e com a possibilidade de reutilizar aplicações desenvolvidas por terceiros, a colaboração torna-se uma característica fundamental.

Relacionando os principais aspectos da Web 2.0 apresentados na Seção 2.1 com as teorias construtivista e sócio-interacionista (Quadro 2.2), podemos imaginar que: a Web como plataforma possibilita a educação ubíqua onde habilidades e conhecimento são desenvolvidos no contexto onde serão utilizados; ao tirar partido da inteligência coletiva e permitir que usuários agreguem valor, fomenta-se a participação e colaboração; permitir uma experiência rica ao usuário através da utilização de modelos leves de programação possibilita a personalização, característica fundamental quando se deseja dar ao aluno o controle sobre sua aprendizagem.

Estas conclusões podem apontar para caminhos que provavelmente serão seguidos pelos Ambientes Virtuais de Aprendizagem num futuro próximo. Neste sentido, uma tendência atual é a utilização de ambientes que privilegiem a educação informal e continuada, tais como os Ambientes Pessoais de Aprendizagem que serão abordados na Seção 2.4. Entretanto, segundo Brown (2007), isto não significa que os conceitos utilizados nos AVAs atuais devam ser substituídos. A grande popularização dos ambientes disponíveis atualmente indica claramente que eles podem contribuir muito com o processo de aprendizado. Sendo assim, a incorporação das idéias da Web 2.0 aos AVAs pode ser feita através da criação de uma nova camada que aproveite as tecnologias disponíveis na Web 2.0, acrescentando novas possibilidades educacionais que valorizem cada vez mais as idéias construtivistas.

Quadro 2.2 – Principais aspectos da Web 2.0 e das teorias construtivista e sócio-interacionista

<table>
<tr><th>Principais aspectos da Web 2.0</th><th>Possibilitam...</th><th>Atendendo aos seguintes aspectos das teorias construtivista e sócio-interacionista.</th></tr>
<tr><td>A Web como plataforma:
Independência de dispositivo;
Software é serviço, Fim do ciclo de versões.</td><td>Educação ubíqua</td><td>Habilidades e conhecimento são desenvolvidos no contexto onde serão utilizados</td></tr>
<tr><td>Usuários agregam valor:
Arquitetura de Participação;
Licenças menos restritivas;
Poder coletivo dos pequenos usuários;
Efeitos de rede.</td><td rowspan="2">Participação e Colaboração</td><td rowspan="2">O desenvolvimento do conhecimento humano é determinado pelas interações entre o indivíduo e o meio.
O aprendizado é influenciado pelas interações sociais cuja confrontação de pontos de vista favorece o desenvolvimento do indivíduo.</td></tr>
<tr><td>Tirando partido da inteligência coletiva:
Produção descentralizada de conteúdo;
Folksonomias.</td></tr>
<tr><td>Os dados de uma aplicação são o seu verdadeiro valor.</td><td></td><td></td></tr>
<tr><td>Modelos leves de programação:
Sistemas fracamente acoplados;
APIs abertas.</td><td rowspan="2">Personalização</td><td rowspan="2">Ênfase no controle do aluno sobre sua aprendizagem
Enfoque na construção individual de significados</td></tr>
<tr><td>Experiência rica do usuário:
Interfaces ricas.</td></tr>
</table>

2.3. Aplicativos Web 2.0 e suas Utilizações Educacionais

Existe um grupo de serviços e aplicativos na Web como, blogs, wikis, favoritos compartilhados, e compartilhamento de arquivos multimídia que são intimamente relacionados aos conceitos da Web 2.0. Estas aplicações, freqüentemente chamadas coletivamente de software social, são exemplos claros de valorização da participação dos usuários na produção de conteúdos na Internet (ALEXANDER, 2006).

2.3.1. Blogs

Termo criado em 1997 por Jorn Barger, autor do primeiro blog de que se tem notícia. É uma abreviação de weblog, o termo original. Um blog é uma página na Web que consiste de uma seqüência de fragmentos de texto ou publicações (posts) ordenados cronologicamente com os mais recentes aparecendo primeiro. Os blogs permitem uma troca de opiniões entre o escritor e o seu público extremamente rica, uma vez que os visitantes podem comentar cada publicação.

Os principais fatores responsáveis pela rápida disseminação do uso de blogs na Internet são, sem dúvida, a disponibilidade dos serviços de hospedagem de blogs gratuitos e a grande facilidade de sua utilização para publicar informações. Enquanto criar uma página Web exige alguns conhecimentos de software e programação, nem sempre possuídos pelo usuário comum, criar e manter um blog é muito mais simples e rápido.

A literatura relata diversas experiências bem sucedidas do uso de Blogs na educação (DOWNES, 2004). O principal ganho que pode ser percebido é a maior motivação dos alunos pela própria disciplina. A utilização de blogs para apoio ao aprendizado permite um envolvimento maior com o curso uma vez que é possível entrar em contato com o professor ou com outros colegas de forma mais fácil e rápida, fazer questionamentos, partilhar idéias, publicar trabalhos ou textos. Além disso, participar num blog pode ser um estímulo à reflexão e produção escrita, desde que exista uma orientação e acompanhamento nesse sentido.

As utilizações educacionais de blogs são diversas e podem ser agrupadas de acordo com quem o publica e para qual público-alvo (Quadro 2.3).

Gomes (2005) divide as possíveis utilizações pedagógicas dos blogs em dois grupos: enquanto recurso pedagógico e enquanto estratégia pedagógica. Enquanto recurso pedagógico, os blogs podem ser:

- Um espaço de acesso à informação especializada indicada ou sugerida pelo professor disponível através de blogs com conteúdo educacional que tratem de assuntos com possíveis enquadramentos curriculares ou extracurriculares.

- Um espaço de disponibilização de informação por parte do professor. Este espaço pode ser utilizado para fornecer informações administrativas sobre o curso como calendários e critérios de aprovação, mas principalmente para oferecer materiais de apoio como textos, comentários pessoais, ligações com sites relevantes. O professor deve incentivar uma

prática de consulta continuada, fazendo com que o aluno esteja sempre envolvido com o curso.

Enquanto estratégia pedagógica os blogs podem assumir a forma de:

- Um portfólio digital utilizado como forma de organizar e apoiar as aprendizagens e/ou a possibilidade de se constituir como instrumento de avaliação. A construção de um portfólio deve ser um processo gradual de reflexão e maturação pessoal e não um simples repositório de documentos e artefatos.

- Um espaço de intercâmbio e colaboração entre escolas. Os blogs são uma alternativa mais permanente, visível e colaborativa que o simples uso de correio eletrônico.

- Um espaço de debate com os alunos desempenhando papéis (*role playing*), onde cada aluno (ou grupo) procure apresentar os seus argumentos do ponto de vista da personagem ou entidade que foi chamado a representar.

- Um espaço de integração onde todos apresentam suas perspectivas, experiências e realidades culturais pode ser uma forma de promover a compreensão mútua e facilitar a integração dos alunos pertencentes a minorias étnicas ou culturais.

Entretanto, como na introdução de qualquer nova tecnologia para apoiar a aprendizagem, algumas questões devem ser consideradas. Em primeiro lugar, os professores devem estar motivados e precisam ser capacitados a utilizar esta ferramenta, uma vez que eles também devem ser participantes ativos, alimentando freqüentemente o blog para que o mesmo não fique desatualizado (BALTAZAR e AGUADED, 2005). Além disso, devem ser garantidas condições mínimas de acesso à Internet para todos os alunos. E finalmente, um Blog é um meio totalmente livre para a publicação das idéias dos alunos. Se isso por um lado tem um grande fator positivo que é o estímulo à expressão livre e a participação, por outro pode expor ao risco de alguma publicação indesejada por parte de um aluno. A possibilidade de moderação por parte do professor deve ser considerada nestes casos. Uma visão crítica onde são apontados alguns problemas e questionamentos do uso educacional de Blogs é apresentada por Downes (2004).

Quadro 2.3 – Utilizações educacionais de blogs.

		Quem Escreve		
		Professor	**Aluno**	**Mundo**
Para Quem se Destina	**Professor**	Diário do Professor: informações sobre aulas; programa; matéria dada; bibliografia. Informações específicas da disciplina: compartilhamento de suas práticas, experiências e conhecimentos com outros professores.	Sistematização do aprendizado individual; Resumo de leituras ou de assuntos abordados em aula; Outros trabalhos individuais; Trabalhos em grupo; Organização de seminários; Debates com a turma.	Blogs Educativos
	Aluno	Página do Curso: calendário, informações, leituras sugeridas, exercícios.	Trabalhos em grupo; Organização de seminários; Debates com a turma.	Blogs Educativos
	Mundo	Blogs Educativos	Portfólio pessoal: para avaliação por outra instituição; para estabelecer uma reputação profissional.	---

2.3.2. Wikis

Um Wiki é uma ferramenta colaborativa que facilita a produção do trabalho em grupo. Mais especificamente, é um software baseado na Web que permite a criação de sítios onde é dada a permissão de alterar seu próprio conteúdo, aos seus visitantes. A edição de uma página wiki pode ser realizada através do próprio navegador, bastando selecionar um link apropriado. Isso faz do Wiki uma plataforma simples e fácil de usar para o trabalho colaborativo de produção de textos (EBERSBACH, GLASER e HEIGL, 2006, p10).

Dependendo da finalidade, o acesso para atualização pode ser restringido a um grupo específico de usuários. Como a atualização dos Wikis não é centralizada, é necessário um mecanismo de segurança com controle de versões que permita a recuperação de conteúdos perdidos acidentalmente. Com esta finalidade, é normalmente armazenado um histórico de modificações, para que versões anteriores possam ser recuperadas posteriormente.

Os Wikis são mais flexíveis do que os blogs, pois permitem a criação de uma hierarquia de assuntos relacionados através de hiperlinks em lugar de uma estrutura rígida e cronológica. A recuperação das informações também é facilitada, pois estas ferramentas geralmente incluem algum mecanismo de busca para o rápido acesso ao seu conteúdo. Como ferramenta de apoio ao ensino, podem ser utilizados como repositórios do conhecimento construído coletivamente pelos alunos, em situações em que estas flexibilidades sejam desejáveis.

Normalmente é criado um Wiki para uma turma ou para cada grupo de alunos responsável por desenvolver um projeto, que será usado como um repositório central do conhecimento adquirido e expressado pelos próprios alunos. Isso faz com que estes desenvolvam um senso de propriedade sobre o conteúdo do curso, e passem a ser contribuintes do aprendizado dos demais, tornando-se elementos ativos no processo educacional.

Lund e Smordal (2006) discutem o papel desempenhado pelo professor – de fomentar a cognição coletiva – em alguns projetos educacionais onde a principal ferramenta utilizada foi um Wiki. Uma das principais conclusões deste estudo é que deve haver um espaço para o professor conduzir os alunos na execução das atividades propostas. Neste sentido, uma página para discussão entre os colaboradores até que estes cheguem a um consenso sobre o conteúdo a ser publicado, já implementada por algumas ferramentas disponíveis, pode ser utilizado. É nesse espaço que a atuação do professor deve ser incisiva, questionando, estimulando, provocando a colocação de opiniões ou sugerindo novas fontes de consulta. Outra funcionalidade também existente em alguns Wikis, o controle de versão, pode ser um grande aliado do professor, permitindo sua percepção sobre o aprendizado e a evolução do conhecimento entre os alunos, desde que os autores de cada modificação possam ser identificados.

Schwartz et al (2004) estabelecem um conjunto de critérios e funcionalidades implementadas pelos principais produtos disponíveis, com a intenção de auxiliar a seleção da ferramenta mais adequada para o uso educacional.

Existem problemas em algumas implementações gratuitas de Wikis que, apesar de não inviabilizarem sua utilização pedagógica, podem dificultar algumas aplicações. Wang e Turner (2004) mencionam quatro propriedades dos sistemas tradicionais de Wiki que não são desejáveis neste contexto: todo o conteúdo pode ser alterado por qualquer um; todo o conteúdo é público e pode ser visto por qualquer um; não existe um mecanismo que garanta a integridade da informação em caso de mais de um usuário tentar alterar uma página simultaneamente; e os Wikis estão permanentemente em evolução enquanto que um curso tem um término bem definido, a

partir do qual as alterações não devem mais ser permitidas. Além disso, o fato da edição do conteúdo normalmente ser feita através de comandos de marcação pode dificultar o seu uso por alguns alunos. Assim, dependendo do público-alvo, pode ser desejável selecionar um Wiki que apresente uma interface mais amigável, do tipo WYSIWYG[10].

2.3.3. Favoritos compartilhados e classificação dos conteúdos

Outro grupo de aplicativos relacionados à Web 2.0 são os serviços de gerência de favoritos compartilhados ou *social bookmarking* (HAMMOND, 2005; LUND, 2005). Eles oferecem a seus usuários, a possibilidade de armazenar e compartilhar seus links favoritos. Como as informações são armazenadas remotamente num servidor central, é possível acessá-las de qualquer computador. Além disso, o compartilhamento das informações permite, não só tornar públicas as suas referências, como é possível buscar referências relevantes, nos favoritos de outras pessoas que tenham interesses semelhantes.

Cada link é armazenado através de sua URL, juntamente com um título, uma descrição e algumas palavras-chave ou *tags*. Todas essas informações são definidas livremente pelo usuário que incluiu a referência.

Uma das primeiras aplicações a explorar a inteligência coletiva, utilizando o conceito de classificação de conteúdos através de tags (folksonomia) foi o del.icio.us[11] (Figura 2.1), um site de favoritos compartilhados. Uma tag é uma palavra-chave ou um descritor, que não faz parte de um sistema de classificação formal. As tags são definidas e associadas livremente pelos usuários aos objetos da Web de forma a descrevê-los. A utilização de tags foi tão bem aceita no contexto da Web 2.0 que sua utilização logo transcendeu a marcação de favoritos sendo utilizada para classificar inúmeros recursos na Web como, imagens, filmes, publicações em blogs ou fóruns de discussão.

A utilização de tags é muito mais flexível do que armazenar os objetos em pastas, pois um objeto pode pertencer a várias categorias e não há necessidade de estabelecer uma hierarquia entre elas. Além disso, o fato das categorias serem também compartilhadas permite que o sistema no momento da classificação sugira tags relacionadas que já foram utilizadas por outros usuários.

---

[10] What You See Is What You Get – Forma de edição capaz de exibir o conteúdo na forma com que ele será apresentado ao usuário.
[11] Disponível em http://del.icio.us

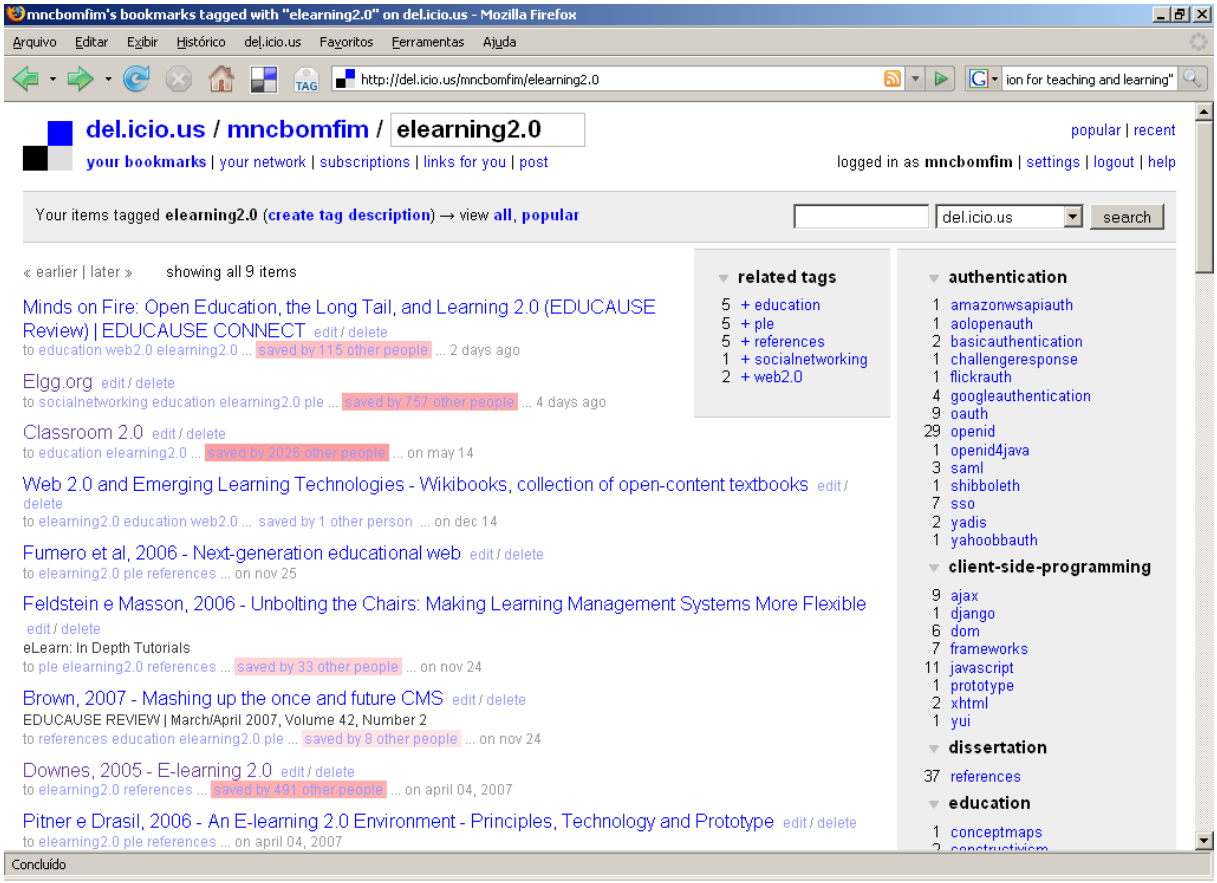

Figura 2.1 - del.icio.us - Sistema de favoritos compartilhados.

Nuvens de tags (*tag clouds*) são representações gráficas das tags utilizadas por um grupo de usuários de um serviço. As palavras são normalmente apresentadas em ordem alfabética, sendo que algumas delas são destacadas, através do tamanho da letra ou da cor com que são representadas. Este destaque representa o número de ocorrências (freqüência) com que são utilizadas sendo que, quanto maior o destaque da palavra, mais popular ela é. Assim, esta representação permite a todos os usuários uma visão estatística do uso das tags, facilitando o seu compartilhamento. A Figura 2.2 apresenta um exemplo de nuvem de tags utilizado pelo del.icio.us.

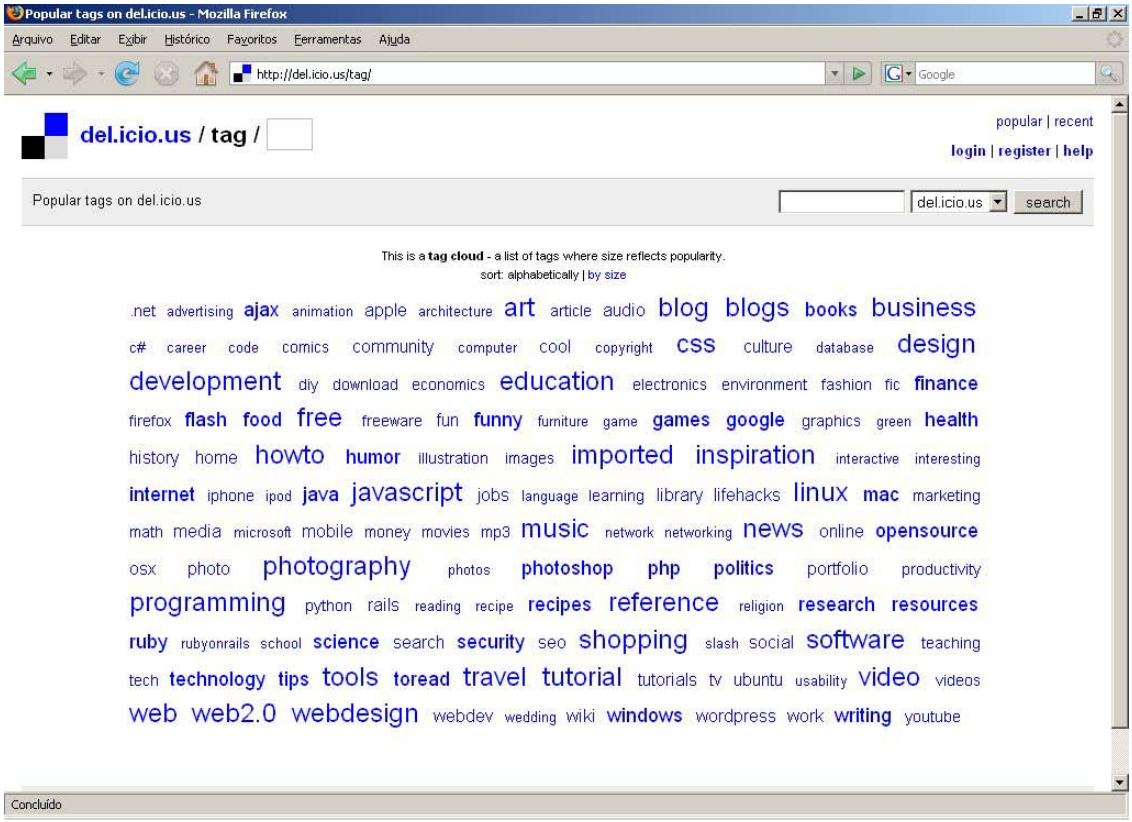

Figura 2.2 - Nuvem de tags do del.icio.us

2.3.4. Compartilhamento de arquivos multimídia

Paralelamente ao surgimento da Web 2.0, ocorreu também uma popularização dos serviços de armazenamento e compartilhamento de conteúdos multimídia como imagens/fotos, vídeos ou áudios. O FlickR e o YouTube são exemplos de serviços de compartilhamento bastante populares (Figura 2.3).

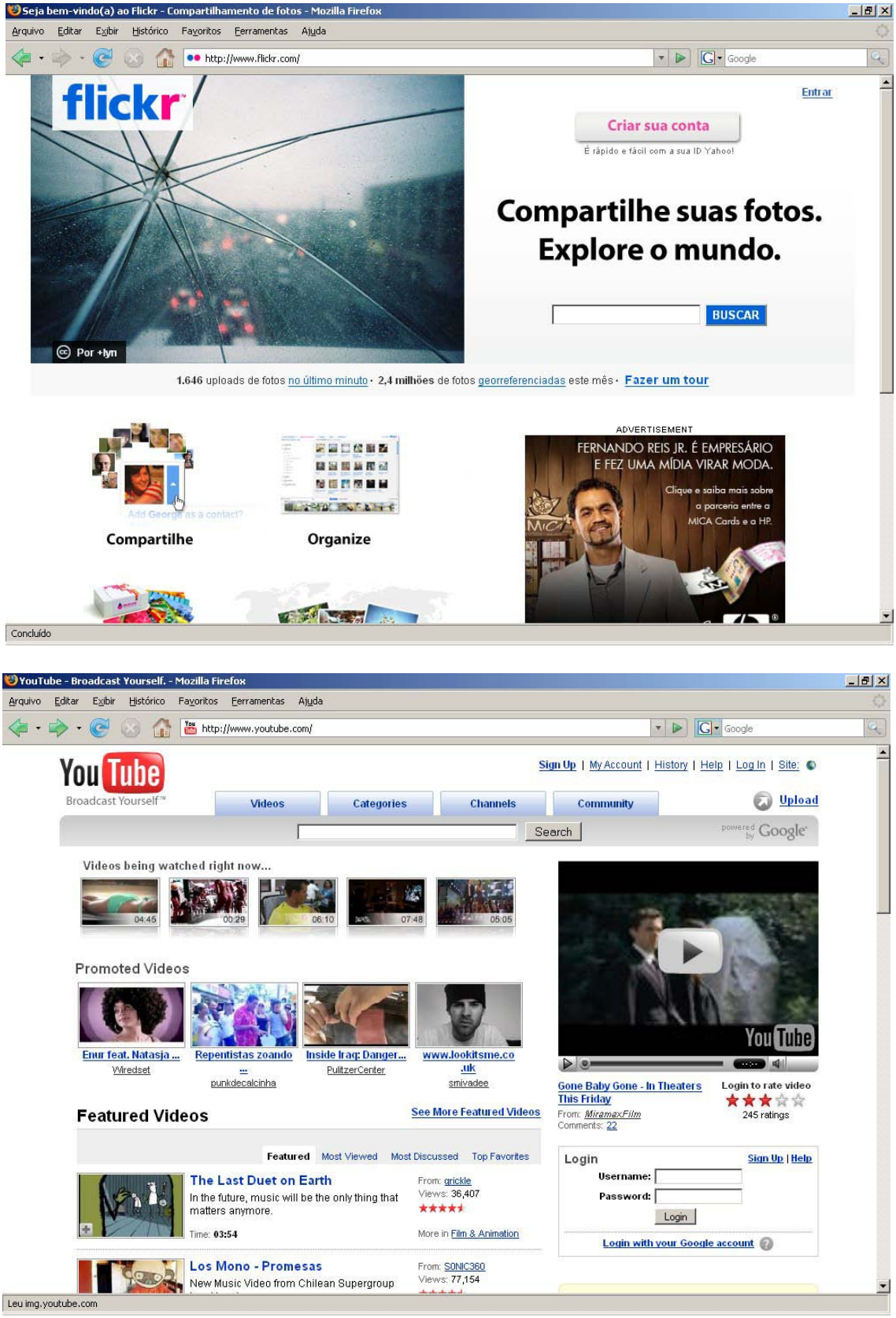

Figura 2.3 - FlickR e YouTube - Repositórios de conteúdo multimídia mais populares.

Esta popularização está associada aos avanços tecnológicos ocorridos nos últimos anos como o aumento da velocidade de conexão dos usuários à Internet, e o barateamento das câmeras

fotográficas e filmadoras digitais. Ao mesmo tempo refletem perfeitamente a filosofia da Web 2.0 onde o usuário participa e produz conteúdo.

A publicação de gravações de áudio é chamada de podcasting. Estes arquivos, usualmente em formato mp3 (McCANDLESS, 1999), podem ser tocados tanto no próprio computador pessoal, como em dispositivos portáteis. Normalmente a publicação de podcasts é realizada através de sindicação de forma que os usuários possam se inscrever em canais, sendo avisados assim que um novo conteúdo estiver disponível.

O uso de multimídia como forma de enriquecer o processo educacional é um assunto que já foi muito estudado e que transcende a discussão das aplicações da Web 2.0. A diferença é que agora a criação e a publicação destes recursos estão ao alcance de qualquer um, criando assim, mais um canal para que os alunos possam se expressar.

2.4. Ambientes Pessoais de Aprendizagem

A proposta da Web 2.0 é freqüentemente associada a dois aspectos: um social e outro tecnológico. O aspecto social refere-se a um grupo de aplicações que permitem um maior grau de interatividade e colaboração, onde todos os usuários são capazes de produzir conteúdo. Sua incorporação nos ambientes de aprendizagem possibilita a adoção de práticas pedagógicas que visam o aumento da colaboração e da participação dos alunos. Já o aspecto tecnológico permite a definição de um ambiente de aprendizagem componível onde o professor e o aluno não ficam restritos a um conjunto de funcionalidades preexistentes, sendo possível incorporar novas aplicações a partir de serviços disponíveis na Internet.

2.4.1. E-learning 2.0

As tendências da Web 2.0 têm influenciado diversas aplicações existentes na Web. No que diz respeito aos Ambientes Virtuais de Aprendizagem (AVAs), alguns autores (DOWNES, 2005; PITNER e DRASIL, 2006) publicaram recentemente, trabalhos em que são apresentadas possibilidades de uso dos conceitos da Web 2.0, a fim de facilitar o aprendizado.

Segundo Downes (2005) as formas de utilização da Internet como uma ferramenta de apoio ao aprendizado, estão evoluindo juntamente com a Web de maneira significativa, o que permite a adoção de um novo nome para identificar este estágio. O E-Learning 2.0, como ele próprio definiu, é baseado não apenas nos avanços tecnológicos, mas principalmente nos princípios sociais da Web 2.0.

2.4.1.1. Características de um ambiente de E-learning 2.0

A seguir, são identificadas algumas características que devem estar presentes num ambiente deste tipo (BOMFIM, ASSIS e SAMPAIO, 2007):

- Difusão de conteúdos. A sindicação permite que um sítio ofereça continuamente através de formatos padronizados (RSS ou Atom), as alterações mais recentes em seus conteúdos, fazendo com que a informação vá ao usuário antes que este a procure. As aplicações da Web 2.0, como blogs, wikis, sítios de compartilhamento de multimídia e de favoritos, em geral, implementam automaticamente a sindicação. Conseqüentemente, esta prática pode facilitar a comunicação entre os aprendizes, ou entre estes e o professor, na medida em que estas aplicações sejam utilizadas (BULL,2006).

- Adaptabilidade e personalização. Uma das bases da Web 2.0 é o incentivo à produção independente de diversos tipos de conteúdo e a fácil integração destes num só ambiente. O uso desta técnica permite a personalização do ambiente de aprendizagem de forma que este possa ser adaptado às diferentes necessidades dos estudantes que o utilizarão.

- Uso de ferramentas externas. A Internet dispõe de inúmeras ferramentas que podem ser utilizadas no contexto educacional. O ambiente não deve obrigar seus usuários a utilizar suas próprias soluções.

- Aderência a padrões. Novas ferramentas têm sido desenvolvidas a cada momento. A adesão da plataforma aos padrões existentes permite a incorporação e o aproveitamento de novas ferramentas de aprendizado ao ambiente de ensino.

- Exploração da inteligência coletiva. Os próprios usuários devem ser capazes de categorizar e identificar informações através do uso de palavras chaves cuja definição é realizada a partir do linguajar natural da comunidade que a utiliza. Entre suas principais utilidades estão: a ajuda para encontrar novamente algo que já foi encontrado antes, e o fato de promover efeitos sociais decorrentes do agrupamento informal de pessoas que passam a utilizar uma determinada palavra-chave.

Recentemente, algumas soluções tentando incorporar estas características têm sido propostas. O L2 (PITNER e DRASIL, 2006) é um exemplo de protótipo que tenta facilitar a integração de outros recursos XML disponíveis na Internet e permitindo sua personalização através de anotações.

2.4.2. Ambientes Pessoais de Aprendizagem

Os cada vez mais freqüentes avanços tecnológicos, a globalização, a necessidade de constante atualização profissional, e a grande disponibilidade de informação a que estamos expostos no dia-a-dia são fatores que tornam a aprendizagem informal e continuada, extremamente valorizada no mundo atual.

O aprendizado é um fenômeno social. A maior parte do que aprendemos, o fazemos a partir do contato com outras pessoas. Segundo Cross (2006), nas empresas, a participação em cursos e treinamentos formais é responsável pelo aprendizado de apenas 10 a 20% do que aprendemos no ambiente de trabalho, sendo que o restante é aprendido informalmente através de processos como: observação, convivência com colegas de trabalho ou tentativa e erro. Desta forma, o ser humano está sempre aprendendo. Este é o conceito de educação continuada[12] (ASPIN e CHAPMAN, 2001). Seja motivado por interesses pessoais, sociais ou profissionais, estamos freqüentemente envolvidos com atividades que resultam direta ou indiretamente num aumento de nossos conhecimentos, habilidades e competências em alguma área particular.

A possibilidade de conectar-se à Internet através de diferentes dispositivos e a popularização das conexões através de redes sem fio e banda-larga, tornam a Internet cada vez mais presente, oferecendo novas oportunidades para o uso de TICs em educação. Assim, estas tecnologias podem ser ferramentas potencialmente muito poderosas para o apoio a estes processos de aprendizado.

Segundo Wilson et al (2006), o modelo dominante de utilização das TICs no apoio ao ensino está baseado na utilização dos recursos oferecidos pelos AVAs atuais. Apesar de estas ferramentas serem alvo de constantes aperfeiçoamentos em função do desenvolvimento de novas tecnologias, elas continuam seguindo um modelo que tenta replicar a educação convencional na sala de aula: o professor é o reprodutor do conhecimento e os alunos são consumidores passivos. Este modelo, embora tenha sido utilizado com relativo sucesso ao longo dos últimos anos, não parece ser o mais adequado quando se trata de apoiar o aprendizado individual, informal e continuado. Os requisitos a serem satisfeitos pela implementação de um ambiente computacional que apóie a aprendizagem continuada, diferem consideravelmente das necessidades de um curso ou treinamento formal (KOPER e TATTERSALL, 2004).

---

[12] Do inglês: Lifelong Learning

Para Wilson et al (2006) existe um modelo alternativo, que privilegia a educação continuada e informal, cujas idéias encontram-se ainda em discussão, que vêm atraindo um interesse cada vez maior na área de educação a distância. O Quadro 2.4 sistematiza esta opinião com uma comparação entre as características destes dois modelos de utilização das tecnologias na educação.

Quadro 2.4 – Comparação entre os modelos, dominante e alternativo (WILSON et al, 2006).

Modelo Dominante	**Modelo Alternativo**
Foco na utilização de ferramentas dentro do contexto de um curso – O aprendizado segue o padrão institucional onde o conhecimento é dividido em unidades discretas como cursos, disciplinas ou turmas. Normalmente não é possível compartilhar conteúdo entre cursos diferentes apoiados pelo mesmo ambiente.	Foco na coordenação de conexões entre os usuários e os serviços – Ao invés de integrar ferramentas num único contexto, o sistema deve coordenar as conexões entre os usuários e os diversos serviços disponíveis na Internet.O aprendizado é orientado às competências. Permite a integração de experiências nas atividades educacionais, assim como de trabalho ou de lazer.
Distinção clara entre os papéis desempenhados por professores e alunos – Se por um lado os alunos são chamados a serem criativos e a participar, por outro, eles esbarram nos limites impostos pela própria filosofia das ferramentas.	Não existe distinção de papéis. Ao utilizar um serviço, qualquer usuário pode ser produtor ou consumidor do conhecimento.Usuários devem poder organizar seus recursos livremente, gerenciando seus contextos e adotando novas ferramentas que se adaptem melhor às suas necessidades.
O modelo organizacional, centrado no curso, e os limites impostos aos alunos, no que diz respeito à organização dos seus espaços, fazem com que a experiência dos usuários seja muito homogênea. Todos vivenciam o aprendizado da mesma forma, através dos mesmos conteúdos e das mesmas ferramentas.	O contexto é individualizado. Cada usuário tem a possibilidade de reorganizar as informações dentro do contexto estudado, de acordo com os seus interesses particulares.
Aderência a um conjunto de padrões e especificações de e-learning, que auxiliem a integração de objetos de aprendizagem em diferentes ambientes. Entretanto, alguns padrões como o RSS, não têm sido utilizados por estas ferramentas, provavelmente devido às suas naturezas fechadas que desencorajam o livre compartilhamento de conteúdos.	Uso de padrões abertos da Internet e APIs leves. O ambiente do aluno não deve ser baseado apenas em serviços oferecidos pelas instituições. Deve ser possível integrar serviços externos. Para isso os serviços oferecem APIs proprietárias ou aderem a padrões abertos.
Seguindo o modelo institucional, os AVAs, em geral, restringem o acesso aos conteúdos e discussões de um curso, somente aos seus alunos, enquanto eles forem alunos. Assim, os materiais publicados têm seus direitos autorais protegidos.	Ênfase nos conteúdos abertos e no reaproveitamento, no compartilhamento de conhecimento e não na proteção de seus direitos autorais. Uso de licenças Creative Commons que permitam consultar, modificar e republicar livremente seus conteúdos.
A instalação e gerência do AVA são realizadas por uma instituição, que o torna disponível para seus alunos, durante o período em que estes estejam envolvidos com as atividades do curso.	Ferramentas pessoais, com o objetivo de coordenar serviços e informações do interesse de cada usuário, que não fica restrito aos serviços oferecidos por uma instituição específica, utilizando-se dos serviços da Internet de uma maneira geral.

Estas considerações conceituais levaram à idéia dos Ambientes Pessoais de Aprendizagem (APA)[13]. Embora não exista ainda um consenso do que seja a definição de um APA, pode-se dizer que são mecanismos que permitem que indivíduos acessem, e organizem de forma sistematizada, artefatos utilizados em seus processos de aprendizado. Alguns conceitos importantes incorporados pelos APAs são: a integração do aprendizado formal com o informal numa única experiência; o uso de redes sociais para comunicação além das fronteiras das instituições (YOSHIZAKI, 2006); e o uso de protocolos de rede (Peer-to-Peer, serviços Web, sindicação) de forma a conectar recursos e sistemas dentro de um espaço pessoal do usuário.

Um APA não é uma aplicação e sim uma composição de ferramentas – baseadas em software social – que são utilizadas no dia-a-dia do usuário e que podem ser utilizadas com o intuito de promover o aprendizado. Aplicações típicas da Web 2.0 como blogs, wikis, compartilhamento de arquivos multimídia e compartilhamento de favoritos, podem ser incorporadas num ambiente deste tipo (ATTWELL, 2007). Segundo Harmelen (2006), além do acesso às aplicações, os APAs devem prover uma interface padronizada para os ambientes utilizados por diferentes instituições de ensino, com a adoção de portfólios como forma de sistematizar o conhecimento adquirido em diferentes contextos criando um registro permanente do aprendizado e permitindo que os trabalhos realizados pelo aluno sejam mantidos entre estas instituições. Desta forma, um APA é um sistema para apoio ao aprendizado individual capaz de prover acesso a diferentes serviços, entre eles os AVAs.

Um aspecto fundamental dos APAs é permitir a organização das informações e do conhecimento adquirido. Para isso é adequada a utilização de mecanismos para classificação através de palavras-chave (tags), e a criação de listas de recursos (como as playlists do iTunes). Estes mecanismos facilitam a localização das informações e o compartilhamento entre os usuários, uma vez que são mais flexíveis que o simples armazenamento numa hierarquia de pastas (WILSON et al., 2006).

Sintetizando a relação entre as idéias da Web 2.0 e os APAs, a Figura 2.4 apresenta um mapa conceitual ilustrando a forma com que estes princípios podem facilitar a aplicação das teorias construtivista e sócio-interacionista nestes ambientes.

[13] Do inglês Personal Learning Environment (PLE)

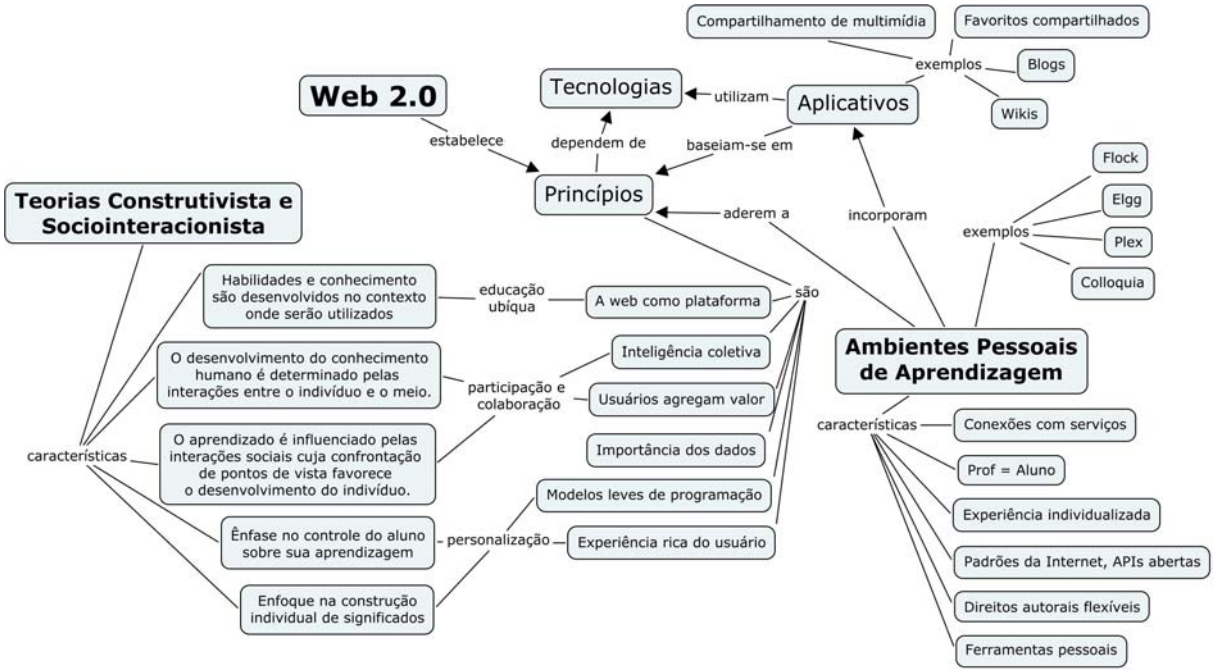

Figura 2.4 – Mapa conceitual da Web 2.0 e dos Ambientes Pessoais de Aprendizagem

2.4.2.1. Alguns exemplos de APAs

A partir das questões apresentadas anteriormente, alguns projetos acadêmicos tiveram como objetivo o desenvolvimento de ferramentas computacionais que pudessem ser utilizadas como Ambientes Pessoais de Aprendizagem. São, na sua maioria, protótipos que foram construídos com o intuito de explorar a viabilidade da aplicação dos conceitos aqui descritos. Mencionaremos então, as principais iniciativas neste sentido.

Colloquia[14]

O Colloquia foi desenvolvido entre 2001 e 2002 numa iniciativa anterior à Web 2.0 de criar um groupware educacional, cujos princípios são semelhantes aos dos APAs. É considerado o primeiro sistema peer-to-peer de apoio ao aprendizado. Segundo Liber (2000), o foco das ferramentas educacionais (AVAs) tem sido mais na publicação de novos conteúdos e na replicação do ambiente da sala de aula, do que no aspecto mais importante do processo de aprendizado que é a interação entre professores e alunos. Assim, o Colloquia é uma ferramenta pessoal que tenta se afastar deste paradigma, na direção de um modelo onde o aprendizado é baseado na interação entre os participantes.

[14] http://www.colloquia.net

The Manchester Framework (Manchester PLE)

O Manchester Framework foi projeto acadêmico realizado pela Universidade de Manchester descontinuado em 2005. O software desenvolvido através deste projeto consiste de um arcabouço cliente-servidor para agregação de serviços que pode ser utilizado tanto como AVA, quanto como APA.

O módulo Cliente é uma aplicação desktop utilizável como um ambiente pessoal e pode ser conectado ao módulo Servidor (um servlet Java) que funciona como um AVA. Esta implementação permite que um cliente seja conectado a vários servidores (AVAs) institucionais.

Para a comunicação cliente-servidor foi desenvolvimento um protocolo baseado no HTTP, chamado de VPTP (VLE-PLE Transfer Protocol).

PLEX[15]

O Plex teve sua primeira versão beta disponível em 2006, desenvolvida pela equipe do projeto PLE da Universidade de Bolton com o apoio do JISC (Joint Information Systems Committee) e do CETIS (Centre for Educational Technology & Interoperability Standards) da Ingleterra. Implementa três conceitos básicos: recursos, pessoas e atividades, onde pessoas possuem recursos e se envolvem em atividades. O aluno determina as oportunidades de aprendizado baseado em seus objetivos, transformando-as depois em atividades. Este modelo incentiva a colaboração através do compartilhamento de recursos e atividades. Foram criadas duas versões do PLEX. A primeira é uma aplicação desenvolvida sobre o Eclipse, na forma de um plugin, enquanto a segunda foi desenvolvida na forma de um portal Web.

Flock[16]

O Flock é um navegador baseado no código-fonte do Mozilla Firefox que se propõe a simplificar a contribuição e a participação na Web. Seu principal diferencial em relação aos demais navegadores é a integração nativa de alguns serviços da Web 2.0. Algumas destas características são: acesso direto aos principais serviços de blog, favoritos podem ser armazenados localmente ou remotamente através do del.icio.us, plugin para o serviço Flickr que permite visualizar diretamente as imagens armazenadas, facilidades para gerência e visualização de RSS. Estas características permitem considerá-lo um ambiente possível de ser utilizado como um APA.

---

[15] http://www.reload.ac.uk/plex/
[16] http://www.flock.com

Elgg[17]

O Elgg, desenvolvido em 2004 por Dave Tosh e Ben Werdmuller, é um software-livre que incorpora uma rede de relacionamentos permitindo a agregação de conteúdos através de blogs e portfólios. Desde sua concepção possui características presentes nos PLEs como: uma rede social baseada em FOAF (*Friend Of A Friend*), utilização de sindicação e alto grau de personalização.

O software pode ser obtido para ser instalado num servidor próprio, ou é possível utilizá-lo gratuitamente através da comunidade EduSpaces[18].

PebblePAD[19]

O PebblePAD é um produto comercial desenvolvido em 2004 pela Pebble Learning Ltd, a partir de um projeto em colaboração com a Universidade de Wolverhampton. Originalmente um sistema de e-Portfólio, hoje um "Personal Learning System" projetado para apoiar o processo de aprendizado.

2.5. Considerações Finais

Hoje em dia, os princípios da Web 2.0 estão cada vez mais presentes na Internet. Este fato é evidenciado pela crescente popularização de aplicativos que utilizam a Web como plataforma e valorizam a inteligência coletiva, entre outras características. Os usuários deixam de ser apenas receptores de informação, para serem participantes mais ativos no processo de produção de conteúdos. Embora seja freqüente o surgimento de novas tecnologias relacionadas, ainda há muito a ser pesquisado. Alguns exemplos de temas emergentes são: a integração da Web 2.0 com a Web semântica através de microformatos e ontologias; arquiteturas para o desenvolvimento de software utilizando a Web como plataforma; aplicações de Redes Sociais e questões relacionadas à privacidade, segurança e confiabilidade das informações.

Esta nova forma de encarar a Web abre caminhos para inúmeras aplicações educacionais. Apesar de ferramentas e serviços como Blogs, Wikis, Compartilhamento de favoritos e de arquivos multimídia, Redes Sociais e Podcasts, estarem sendo cada vez mais utilizadas para aumentar a colaboração e a participação entre os aprendizes, não existe ainda uma abordagem padronizada para agregar estes componentes num ambiente educacional. Assim, temos visto muitas iniciativas

[17] http://elgg.net
[18] http://eduspaces.net
[19] http://www.pebblelearning.co.uk/

isoladas neste sentido. Os Ambientes Pessoais de Aprendizagem – mecanismos que permitem que indivíduos acessem, e organizem de forma sistematizada, os artefatos utilizados em seus processos de aprendizagem – parecem ser uma abordagem adequada para esta questão. Também neste aspecto, ainda há muito campo para pesquisas. Entretanto não existe ainda uma ferramenta estabelecida que siga estes princípios que possa ser adotada.

Capítulo 3
A Web 2.0 e suas Tecnologias

O objetivo deste Capítulo é apresentar as tecnologias da Web 2.0. Para isto elas foram agrupadas nas seguintes categorias: desenvolvimento de aplicações ricas, alternativas para componentização, utilização de padrões e a autenticação de usuários.

Uma tecnologia fundamental para a Web 2.0 é o AJAX (*Asynchronous Javascript and XML*), baseada na realização de requisições assíncronas entre o cliente e o servidor. Tal característica permite a criação de aplicações com interfaces muito mais ricas que as utilizadas na Web tradicional. O AJAX é executado no cliente, normalmente fazendo requisições ao servidor através de um estilo de arquitetura típico chamado REST - Representational State Transfer (FIELDING, 2000). Ao receber então a resposta a estas requisições, o AJAX é capaz de interagir com a página através de seu DOM (*Document Object Model*).

O que torna as aplicações Web 2.0 tão facilmente integráveis é o fato delas possuírem APIs (*Aplication Programming Interfaces*) abertas que podem ser utilizadas por outras aplicações desenvolvidas por terceiros, possibilitando mesclar funcionalidades de aplicações diferentes. Assim, é possível criar novas aplicações, conhecidas como mashups, a partir de chamadas às APIs de outras aplicações já existentes incorporando assim, seus dados e funcionalidades.

Outra característica das aplicações Web 2.0 é a possibilidade delas enviarem informações aos usuários através de formatos padronizados, como acontece no mecanismo de sindicação. Os formatos RSS e Atom são padrões baseados em XML, definidos com esta finalidade.

Finalmente, alguns novos mecanismos de autenticação de usuários têm sido desenvolvidos, decorrentes desta multiplicação dos serviços disponíveis na Internet com a Web 2.0 e que também são abordados neste capítulo.

3.1. Desenvolvimento de Aplicações Ricas

As aplicações Web apresentam inúmeras vantagens sobre as tradicionais aplicações desktop cliente-servidor. A facilidade de distribuição e atualização de versões, devido à ubiqüidade da Internet é atualmente um fator decisivo na hora de optar por um caminho a ser seguido no projeto e implementação de sistemas. Entretanto, o modelo de funcionamento da Web, baseado em requisições e respostas, e o pequeno conjunto de elementos disponíveis nos formulários HTML, sempre foram considerados limitações para o desenvolvimento de interfaces mais ricas baseadas nesta plataforma. Do ponto de vista do usuário, é significativa a perda de interatividade destas aplicações, comparadas com aplicações desktop tradicionais. Historicamente, diversas soluções para minimizar estes problemas foram tentadas como, o uso de *Flash*, o *Microsoft Remote Scripting*, o *JavaScript Remote Scripting* (JRSS) ou simplesmente o uso da tag IFRAME para ocultar chamadas assíncronas ao servidor. Nenhuma delas, entretanto, foi considerada ideal pelos desenvolvedores (ASLESON e SCHUTTA, 2006, p.12).

3.1.1. AJAX (Asynchronous Javascript and XML)

O Ajax surge então como uma alternativa concreta para possibilitar a criação de interfaces mais ricas. A partir da combinação de tecnologias existentes, com Ajax é possível buscar informações assincronamente no servidor e alterar a tela já apresentada para o usuário, sem que esta precise ser totalmente recarregada pelo navegador do usuário. É uma mudança no paradigma de funcionamento das aplicações Web tradicionais. O Ajax, portanto, não é um produto nem uma nova linguagem, é apenas uma nova abordagem para o desenvolvimento de aplicações Web.

O termo AJAX foi criado por Jesse James Garret da empresa Adaptive Path em um artigo publicado em fevereiro de 2005 (GARRET, 2005). Neste artigo ele discute este novo modelo de programação de aplicações para a Web.

As tecnologias envolvidas no Ajax são as seguintes:

- A apresentação é realizada utilizando HTML/XHTML e Folhas de Estilo em Cascata (CSS);

- As requisições assíncronas ao servidor são realizadas através do objeto XMLHttpRequest. Implementado inicialmente pela Microsoft em seu navegador Internet Explorer como um

objeto ActiveX, o XMLHttpRequest é hoje disponível também como objeto nativo nos navegadores Mozilla Firefox e Safari.

- A troca de informações assíncronas entre cliente e servidor é realizada através dos padrões XML (BRAY et al, 2006) e XSLT (CLARK, 1999);

- A interação com o documento e a exibição dinâmica das informações provenientes do servidor é feita através do Modelo de Objetos de Documento (DOM - Document Object Model) - uma especificação da W3C (STENBACK e HENINGER, 2004), independente de plataforma e linguagem, onde se pode trabalhar com cada um dos elementos do documento separadamente, permitindo assim alterar e editar a sua estrutura, criando páginas dinâmicas.

- A lógica de programação é implementada através da linguagem ECMA script, uma especificação proposta pela ECMA (European Computer Manufacturers Association) em 1999 através do padrão ECMA-262 (ECMA, 1999), com o intuito de normalizar as implementações da linguagem Javascript do Netscape Navigator e do Jscript do Internet Explorer. A especificação do ECMAScript padroniza apenas o núcleo da linguagem e alguns objetos nativos, sem considerar a implementação do DOM que é padronizado pela W3C. Hoje, os principais navegadores como Mozilla Firefox, Internet Explorer, Safari e Opera são compatíveis com o ECMA-262, embora eventualmente possam implementar algumas extensões não descritas na especificação.

Para viabilizar estas requisições assíncronas, é introduzido um mecanismo intermediário entre o cliente e o servidor, conhecido como motor Ajax. As Figuras 3.1 e 3.2 apresentam diagramas de seqüência que ilustram os modelos de requisição síncronas e assíncronas. Enquanto no primeiro caso as requisições são comandadas normalmente por cliques em hiperlinks ou envio de formulários, e recebem como resposta páginas HTML completas, no segundo caso é uma chamada em Javascript que, através do objeto XmlHttpRequest, provoca a requisição, cujo resultado, normalmente codificado em XML, é recebido no cliente pelo Javascript e representado através do DOM, na página visualizada pelo usuário.

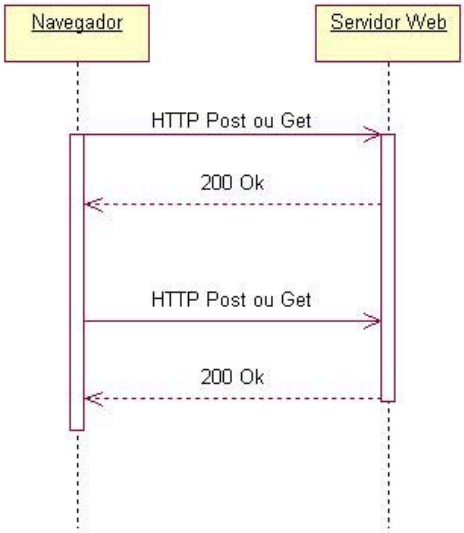

Figura 3.1 - Requisição Síncrona Tradicional

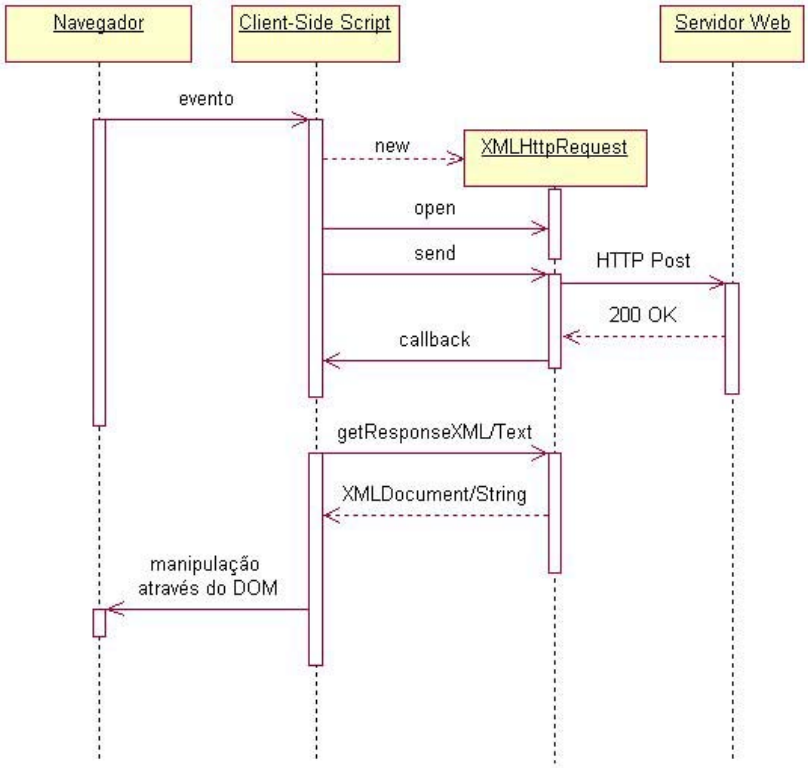

Figura 3.2 - Requisição Assíncrona com AJAX

3.1.1.1. Frameworks Ajax

De forma a facilitar a programação com Ajax, existem diversos frameworks que incorporam funções básicas recorrentes neste tipo de aplicação. Existem dois níveis de atuação destes frameworks, auxiliando o trabalho de programação realizado no cliente ou no servidor.

Para facilitar a programação no cliente, são oferecidas funções em javascript, não só para realizar as requisições ao servidor, assim como para auxiliar a construção das interfaces com o usuário. Assim, é possível criar widgets, criar menus e outros efeitos visuais, encapsulando no framework o cuidado com a compatibilização entre os diversos navegadores existentes. Entre estes frameworks, especializados no lado cliente, podem ser citados: o prototype[20], o YUI[21], e o script.aculo.us[22].

No servidor, é necessário desenvolver as funções que serão responsáveis por responder a estas chamadas. Para isso, existem frameworks desenvolvidos em liguagens de programação normalmente utilizadas no servidor, como Java, C++, ou PHP. Estes são capazem de processar as requisições feitas pelos clientes, interagindo, se necessário, com um banco de dados para buscar as informações que devam ser retornadas aos clientes. O Google Web Toolkit[23] (GWT) é um exemplo de framework em java para a criação de aplicações Ajax.

Alguns frameworks são bastante completos, chegando a prover bibliotecas para a criação de todas as etapas do processo de desenvolvimento de aplicações Web.

3.1.1.2. Aplicações e problemas

Como toda nova tecnologia que passa a ser muito comentada, existe uma tendência dos desenvolvedores começarem a utilizá-la, apenas para seguir um modismo, sem que haja uma necessidade ou um ganho real a ser obtido a partir da sua utilização. O uso do Ajax deve ser considerado quando representar uma facilidade para a interação do usuário, conseqüentemente melhorando o desempenho da aplicação Web.

Algumas situações em que o uso de Ajax pode ser útil são:

- Auxiliar a interação com o usuário. Normalmente a interação do usuário em aplicações Web é dirigida por formulários. São comuns aplicações em que são necessários vários cliques (e conseqüentemente a exibição de várias páginas HTML) para completar uma única operação.

- Auxiliar a navegação numa hierarquia muito profunda, exibindo e escondendo sub-árvores da hierarquia;

[20] http://www.prototypejs.org/
[21] http://developer.yahoo.com/yui/
[22] http://script.aculo.us/
[23] http://code.google.com/webtoolkit/

- Evitar recarregar a tela toda em aplicações de comunicação instantânea entre usuários;
- Implementar operações rápidas que provoquem alguma alteração na visualização da tela atual pelo usuário como: ligar ou desligar um filtro ou alterar um critério de ordenação;
- Apresentar sugestões de preenchimento à medida que o usuário digita um campo.

Existem, entretanto, alguns problemas a serem considerados quando do uso de Ajax. No que diz respeito à usabilidade, alterar dinamicamente uma parte de uma página já carregada é um comportamento, com o qual os usuários da Web não estão acostumados. Isto pode fazer com que alterações passem despercebidas pelo usuário. Outra característica do uso de Ajax a ser citada, é o fato das requisições que são realizadas assincronamente não serem armazenadas no histórico de navegação normalmente mantido pelos navegadores. Desta forma, não é possível utilizar os botões de "voltar" e "avançar" durante a navegação na aplicação. É necessário que os desenvolvedores estejam atentos a estes fatos, evitando projetar sistemas cuja operação confunda o usuário.

Mais uma questão a ser levada em consideração ao utilizar Ajax, é a dificuldade das ferramentas de busca em encontrar conteúdos publicados através de requisições assíncronas, uma vez que os robôs que fazem a indexação das páginas da Web não executarem javascript.

3.2. Alternativas para Componentização

Conectar sistemas remotamente é um problema para o qual, diversas abordagens de computação distribuída foram utilizadas ao longo dos anos. O CORBA (*Common Object Request Broker Architecture*) e o DCOM (*Distributed Component Object Model*) são dois exemplos de tecnologias com esta finalidade, baseadas em chamadas de procedimentos remotos[24]. Entretanto, estas soluções, comumente utilizados em redes locais, tiveram um sucesso muito limitado quando transpostas para ambientes de grandes dimensões como a Internet (NUNES e DAVID, 2005). A utilização de RPC na Internet apresenta riscos de segurança, fazendo com que *Firewalls* e *Proxys* normalmente bloqueiem este tipo de tráfego.

[24] Do inglês Remote Procedure Calls ou RPC (NELSON, 1981).

3.2.1. Serviços Web (*Web Services*)

Uma solução para o problema da componentização é a utilização de Serviços Web (BOOTH et al, 2004) que segundo o W3C:

> *são sistemas de software projetados para prover interação entre máquinas numa rede de computadores. Possuem interfaces descritas num formato processável por máquinas (WSDL), que são utilizadas para que outros sistemas possam interagir com o Serviço Web utilizando mensagens SOAP. Estas mensagens são transportadas utilizando o protocolo HTTP e o padrão XML, além de outros padrões relacionados.*

O termo Serviços Web, portanto, refere-se às aplicações que empregam uma combinação específica de tecnologias, capaz de fazê-las acessíveis a outros sistemas, executados em máquinas remotas (GOTTSCHALK e GRAHAM, 2002). As principais tecnologias envolvidas são:

- SOAP (*Simple Object Access Protocol*) - É um protocolo para comunicação entre aplicações. Baseado em XML, é simples, extensível, independente de plataforma e de linguagem (MITRA e LAFON, 2007).

- WSDL (*Web Services Description Language*) - É um documento XML usado para descrever as interfaces dos Serviços Web. A versão 2.0 já é um padrão da W3C (BOOTH e LIU, 2007).

- UDDI (*Universal Description Discovery Integration*) - É um diretório capaz de armazenar informações sobre Serviços Web, especialmente as interfaces descritas em WSDL. Comunica-se através de SOAP.

Desta forma, é possível responder as seguintes perguntas:

- Como eu encontro o Serviço Web de que eu preciso?

- Uma vez que o encontrei, como o utilizo?

- Qual o formato das mensagens que devo enviar?

Assim, o provedor do serviço descreve suas interfaces com WSDL e registra-o no UDDI. O usuário, então, utiliza-se destes mecanismos para encontrar o serviço e conhecer sua descrição, de forma que possa conectar-se ao provedor do serviço e executá-lo. (Figura 3.3).

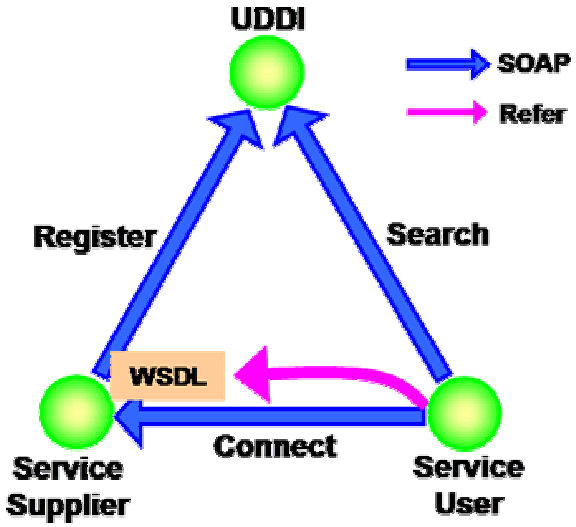

Figura 3.3 - Esquema do funcionamento dos Serviços Web (extraído de http://www.nec.co.jp/middle/WebOTX-e/function/webservice.html)

Alguns estilos de arquitetura normalmente utilizados para implementar Serviços Web são o RPC, o SOA e o REST.

- RPC (*Remote Procedure Calls* ou Chamadas de Procedimentos Remotos) é uma abordagem onde os serviços são baseados em operações que são executadas pela chamada de funções remotas.

- SOA (*Service Oriented Architecture* ou Arquitetura Orientada a Serviços) é uma arquitetura baseada em serviços implementados por componentes fracamente acoplados e reutilizáveis, com o objetivo de prover independência de plataforma, tecnologia e linguagem.

- REST (*Representational State Transfer*) é um estilo de arquitetura cliente-servidor para a criação de serviços Web que não mantenham estado, que baseia a comunicação em operações padronizadas (interface uniforme) sobre um conjunto de recursos existentes. Como o REST vem cada vez mais se tornando uma tendência no desenvolvimento de serviços e aplicações na Web 2.0, este será abordado de forma mais detalhada na seção seguinte.

3.2.2. REST (*REpresentational State Transfer*)

A principal tecnologia que apoia este modelo de componentização de Web 2.0 é o REST (FIELDING, 2000; FIELDING e TAYLOR, 2002), um estilo de arquitetura de software para

sistemas hipermídia distribuídos, definido como um conjunto de critérios e padrões de projeto utilizados com o intuito principal de obter maior escalabilidade dos sistemas de software.

REST se aplica muito bem ao funcionamento do protocolo HTTP e da Web, sendo que, na prática, este termo é freqüentemente utilizado referindo-se a uma arquitetura específica aplicada a algumas tecnologias da Web, particularmente: HTTP, URIs e XML. Entretanto, é importante esclarecer que é possível desenvolver um sistema REST sem utilizar nenhuma destas tecnologias, da mesma forma como também é possível utilizar estas três tecnologias, sem que os preceitos REST sejam atendidos, produzindo assim sistemas não aderentes ao estilo REST. No escopo deste trabalho, como estamos interessados em discutir estas idéias aplicadas ao desenvolvimento de aplicações para a Web, freqüentemente restringiremos o significado do termo REST à sua utilização através do protocolo HTTP.

A Web é uma aplicação distribuída cuja escalabilidade provou-se bem sucedida, apesar das grandes proporções de seu crescimento verificadas no decorrer dos anos. Este fato decorre de algumas características do protocolo HTTP como: o fato de ser um protocolo cliente-servidor sem estado, a existência de um mecanismo de endereçamento global através de URIs, o uso de formatos de mensagens padrão extensíveis, entre outras. A motivação para o estilo de arquitetura REST é tentar desenvolver sistemas utilizando as características do protocolo HTTP, que viabilizaram este sucesso da Web.

3.2.2.1. Principais conceitos e princípios

O conceito de recurso é a principal abstração do estilo de arquitetura REST. Enquanto no modelo RPC, a comunicação se apóia nas operações que são expostas através do protocolo, no modelo REST, são os recursos que devem ser expostos.

Um recurso, portanto, é qualquer informação relevante para o sistema, que precise ser referenciada através do protocolo. Imaginando um sistema de apoio a um ambiente de ensino, alguns exemplos de recursos seriam: alunos, turmas, cursos e disciplinas.

Para identificação dos recursos, é necessário um espaço unificado de nomes, e para isso, a sintaxe universal utilizada é a URI. Cada recurso deve ter uma URI associada a ele, assim como cada URI deve estar associada a um único recurso.

Representação de um Recurso é o conjunto de dados que define as informações associadas ao estado de um recurso. Utiliza para isso um formato padronizado que pode ser, por exemplo, um

arquivo XML, uma página Web, uma string JSON, ou simplesmente um arquivo texto com seqüência de valores separados por vírgulas. Ao requisitar um recurso, o que se obtém é uma representação do mesmo. Da mesma forma, ao incluir ou alterar um recurso envia-se ao servidor uma representação.

Para determinar as operações realizadas sobre os recursos, aplicações REST devem utilizar-se de interfaces uniformes, ou seja, um conjunto restrito de operações com significados bem definidos, diferentemente das soluções RPC onde existe um vocabulário próprio de verbos cujo significado é conhecido somente pelas aplicações que o utilizam (por exemplo: loginUser, doSearch, movePhoto, etc). O protocolo HTTP implementa através dos métodos GET, POST, PUT e DELETE as operações básicas de consulta, inclusão, alteração e exclusão. Sendo assim, é natural que a utilização de REST sobre HTTP utilize-se destes métodos sempre que referenciar um recurso. Uma URI, quando requisitada através de um verbo HTTP, implica na execução de um método, cujas informações sobre o escopo de sua execução devem ser mantidas na própria URI. A grande vantagem do uso de interfaces uniformes decorre do fato de a Internet ser uma rede composta não apenas de clientes e servidores, mas possuir elementos intermediários como firewalls, roteadores e proxys, responsáveis respectivamente por aumentar a segurança, prover escalabilidade e melhorar o desempenho da rede de uma maneira geral. O uso de uma semântica padronizada permite sua interpretação por estes elementos otimizando seu funcionamento.

REST é um protocolo cliente/servidor sem estado: cada requisição HTTP deve conter todas as informações necessárias para a sua execução, pois ela deve ser completamente independente das demais. Se o servidor, por alguma razão possui estado, este também deve ser encarado como um recurso, de forma que tenha sua própria URI. Qualquer informação de estado deve permanecer no cliente, sendo transmitida ao servidor a cada requisição que a necessite.

As representações dos recursos devem ser interligadas, de forma a permitir a navegabilidade entre eles, alterando assim o estado da aplicação a cada novo recurso requisitado. Este conceito foi chamado por Fielding (2000) de "hipermídia como um mecanismo de estado da aplicação" e significa que as representações devem ser projetadas como documentos hipermídia possuindo ligações com outros recursos. Ao seguir uma dessas ligações efetua-se uma transição de estado da aplicação. Como resultado, é possível navegar a partir de um recurso REST para os demais, seguindo as ligações existentes nas representações de recursos obtidas.

3.2.2.2. Exemplo de utilização de REST

Para ilustrar a utilização de um serviço desenvolvido segundo a arquitetura REST, vamos imaginar um ambiente de ensino que represente os conceitos de curso e disciplina, onde um curso é composto de várias disciplinas. Neste caso, as listas de cursos e disciplinas existentes, assim como cada curso ou disciplina individualmente serão considerados como recursos e o protocolo REST para acesso aos serviços desta aplicação define uma URL para cada um deles.

Da mesma forma, devem ser definidas representações para cada um destes recursos. As Figuras 3.4 e 3.5 apresentam, respectivamente duas possibilidades de representação para uma lista de disciplinas e para uma lista de cursos, utilizando arquivos em formato RSS.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
   <title>Curso de Formação de Programadores</title>
   <link>/Avance/rest/curso/12345</link>
   <description></description>
   <item>
      <title>Criação de Páginas Web com HTML</title>
      <link>/Avance/rest/curso/12345/7890</link>
      <description>-</description>
      <author>Mauricio Bomfim</author>
      <pubDate>Qua, 28 Mar 2007 16:20:36 ACT</pubDate>
   </item>
   <item>
      <title>Linguagem Javascript</title>
      <link>/Avance/rest/curso/12345/7880</link>
      <description>-</description>
      <author>João Sergio</author>
      <pubDate>Qua, 28 Mar 2007 16:21:37 ACT</pubDate>
   </item>
</channel>
</rss>
```

Figura 3.4 - Lista de disciplinas de um curso

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
   <title>Listagem de Cursos</title>
   <link>/Avance/rest/curso</link>
   <description>Cursos existentes</description>
   <item>
      <title>Formação de Programadores</title>
      <link>/Avance/rest/curso/12345</link>
      <description>-</description>
      <author>Mauricio Bomfim</author>
      <pubDate>Qua, 28 Mar 2007 16:20:36 ACT</pubDate>
   </item>
</channel>
</rss>
```

Figura 3.5 - Lista de cursos existentes

O Quadro 3.1 lista um conjunto de possíveis operações nesta situação. Cada uma delas é definida por uma URI e um método HTTP.

Quadro 3.1 - Lista de possíveis operações sobre os recursos

Método	**URI do Recurso**	**Descrição da operação realizada**	**Representação**	**Códigos Retornados[25]**
GET	/curso	Obtém lista de cursos	Lista de cursos	200, 401, 500
POST	/curso	Cria um novo curso	Curso	201, 400, 401, 500
PUT	/curso/12345	Altera o curso 12345	Curso	200, 400, 401, 500
DELETE	/curso/12345	Remove o curso 12345	---	200, 400, 401, 500
GET	/curso/12345	Obtém disciplinas do curso 12345	Lista de disciplinas	200, 401, 500
POST	/curso/12345	Cria uma disciplina do curso 12345	Disciplina	201, 400, 401, 500
GET	/curso/12345/7890	Obtém informações da disciplina 7890 que pertence ao curso 12345	Disciplina	200, 401, 500
PUT	/curso/12345/7890	Altera a disciplina 7890 que pertence ao curso 12345	Disciplina	200, 400, 401, 500
DELETE	/curso/12345/7890	Remove a disciplina 7890 que pertence ao curso 12345	---	200, 400, 401, 500

3.2.3. Integração de Aplicações Através de Mashups

Uma de principais características da Web 2.0 é o fato de suas aplicações oferecerem APIs abertas que exponham seus dados e funcionalidades, possibilitando que outros desenvolvedores incorporem-nos em suas próprias aplicações. Estas APIs encapsulam a implementação interna de seus serviços, divulgando apenas a especificação de uma interface onde são definidas as requisições HTTP, seus respectivos parâmetros e seus formatos de saída.

[25] Códigos de retorno definidos no protocolo HTTP:
200 OK
201 Created
400 Bad Request
401 Unauthorized
500 Internal Server Error

Uma conseqüência da proliferação destas APIs é a criação de novas aplicações, conhecidas como mashups, capazes de mesclar serviços já existentes com a proposta de prover soluções para problemas mais específicos. Alguns exemplos de aplicações deste tipo são:

Localização em mapas: Estas aplicações combinam as imagens cartográficas produzidas através da API de algum serviço de apresentação de mapas (Google Maps ou Yahoo Maps, por exemplo) com informações provenientes de alguma outra base de dados (informações meteorológicas, localização de estabelecimentos comerciais ou de atrações turísticas, por exemplo);

Fotos e vídeos: A grande popularização dos serviços de hospedagem de fotos e vídeos permite a criação de inúmeras aplicações que utilizem estes conteúdos como base. Um exemplo é o serviço de edição de imagens Picnik, que é capaz de obter a imagem original em diferentes repositórios como FlickR ou Picassa, processá-la e, posteriormente, salvar o resultado no local designado pelo usuário.

Pesquisa de produtos e compras: Consulta a preços de produtos, mesclando informações provenientes de diferentes origens. Amazon e Ebay oferecem APIs abertas que incentivam esta prática, facilitando o acesso aos seus dados por parte de aplicações de terceiros.

Filtragem de notícias através de combinação de feeds: As informações oferecidas nos formatos padrão RSS e Atom podem ser facilmente processadas e combinadas, de acordo com critérios definidos pelo usuário. O Yahoo Pipes é um exemplo de interface gráfica que implementa uma solução deste tipo.

3.3. Utilização de Padrões

A Web é uma aplicação distribuída que foi desenvolvida baseada em padrões, especialmente, o protocolo HTTP, o sistema de endereçamento baseado em URIs, e a linguagem HTML. Devido a esta natureza descentralizada, desde o seu aparecimento no início dos anos 90, existiu a preocupação com a definição de padrões que orientassem o desenvolvimento de seu conteúdo. Uma importante iniciativa nesse sentido foi a criação em 1994 do W3C (*World Wide Web Consortium*), uma organização destinada a desenvolver e padronizar tecnologias envolvidas com o desenvolvimento de sítios e aplicações para a Web. Entre as muitas tecnologias padronizadas pelo W3C durante este tempo, podemos citar: o HTML; o XML e seus formatos associados como XML Schema, XPath, XPointer, XSL e XSLT; o XHTML; o CSS; o DOM; e a arquitetura de desenvolvimento de Serviços Web através de SOAP e WSDL.

Outras organizações também têm se proposto a padronizar tecnologias para a Web. É o caso do ECMA (*European Computer Manufacturers Association*) com a linguagem ECMAScript (ou javascript), do IETF (*Internet Engineering Task Force*) com o formato e com o protocolo Atom, da ISO (*International Organization for Standardization*) com o SGML (*Standard Generalized Markup Language*), apenas para citar algumas.

Uma das principais conseqüências do uso de padrões na Web é garantir a estabilidade dos produtos desenvolvidos, compatibilizando-os tanto com aplicações futuras quanto com tecnologias passadas. A padronização permite também, facilitar o desenvolvimento de sistemas acessíveis através de diferentes dispositivos ou que funcionam independentemente da versão do navegador utilizado, além de garantir a acessibilidade de todos às informações – inclusive dos portadores de necessidades especiais.

A arquitetura da Web 2.0 e o favorecimento à componentização destas aplicações propiciaram uma explosão no número de componentes capazes de prover inúmeras funcionalidades para novas aplicações. A integração destas aplicações é apoiada na padronização de interfaces e dos formatos de representação das informações trocadas entre estes componentes.

Um padrão que notadamente influenciou o desenvolvimento na Web nos últimos anos foi o XML. Com o propósito principal de facilitar o compartilhamento de dados estruturados entre diferentes sistemas de informação e projetado para ser extensível – diferentemente do HTML – o XML tornou-se um formato extremamente popular, no qual se basearam diversos outros padrões como o RDF, o SOAP, o WSDL, o RSS, o Atom, e o XForms, entre outros.

Entretanto, com a Web 2.0 cuja filosofia privilegia o uso de alternativas mais leves, o uso de XML tem sido recentemente questionado. A utilização do formato JSON pode ser apontada como uma tendência devido à facilidade com que pode ser interpretado pelos clientes Ajax nas aplicações Web 2.0.

3.3.1. Sindicação: RSS e Atom

A sindicação (*Web syndication*) é uma tecnologia que permite que sítios na Web ofereçam conteúdo num formato aberto padronizado, capaz de ser interpretado por outras aplicações ou sítios. Grandes portais de conteúdo como jornais, redes de televisão e blogs oferecem esta facilidade a seus usuários. As informações são oferecidas através de canais (*feeds*) associados a uma URL e cujo conteúdo é um arquivo XML codificado num dos formatos padrão existentes (RSS ou Atom). Para utilizar efetivamente este serviço o usuário precisa de um programa, também

chamado de agregador ou leitor de feeds, que seja capaz de verificar periodicamente as atualizações em cada canal assinado pelo usuário, exibindo estas informações à medida que são por ele detectadas. Existem ainda sítios como o Google Reader[26] ou o Yahoo Pipes[27] que oferecem serviços de agregação de RSS para usuários cadastrados. Além disso, os navegadores atuais já são capazes de interpretar nativamente estes formatos. De forma a facilitar o reconhecimento por parte do usuário, utiliza-se o ícone padrão desenvolvido pela Mozilla Foundation para o Firefox e adotado posteriormente pela Microsoft para o Internet Explorer (Figura 3.6).

Figura 3.6 - Imagem padrão utilizada para representar conteúdo oferecido por sindicação. Fonte: http://www.feedicons.com/

Os principais formatos existentes para realizar sindicação são o RSS, uma família de formatos não necessariamente compatíveis entre si, e o Atom, uma iniciativa com o intuito de definir um padrão único (JOHNSON, 2006).

O *RDF Site Summary*, primeiro formato a ser conhecido como RSS, surgiu em 1999, época em que a Netscape passou a oferecer um novo serviço em seu portal de conteúdo My Netscape. Esta nova tecnologia permitia distribuir automaticamente o conteúdo publicado no portal aos usuários assinantes, sem que estes precisassem se conectar ao site explicitamente.

O RSS era um formato para descrição de informações baseado na linguagem *Resource Description Framework* (RDF), voltada à descrição de recursos na Web. Esta primeira especificação do RSS foi chamada de RSS 0.9. Foi quando então a comunidade de desenvolvedores começou a se dividir. Enquanto alguns achavam que o RDF deveria ser mais bem utilizado, outros eram partidários da simplificação do formato RSS, excluindo assim a necessidade de utilizar o RDF e seu formalismo considerado desnecessário.

A Netscape então seguiu o segundo caminho, lançando ainda em 1999, a especificação RSS 0.91, onde todas as referências ao RDF foram removidas. O formato agora era baseado apenas em

[26] Disponível em http://www.google.com/reader
[27] Disponível em http://pipes.yahoo.com/pipes/

XML e passou a ser conhecido como *Rich Site Summary*. Outra característica era a definição de uma DTD para permitir a sua validação pelos interpretadores de XML

Como não havia nenhum organismo responsável pela padronização do formato, utilizações diferentes se sucederam, como a simplificação proposta por Dave Winer da Userland Software. Ele propôs sua própria versão da especificação RSS 0.91, onde eram feitas mais algumas simplificações, abrindo mão inclusive do DTD para validação.

O RSS 0.91 era bastante útil, mas na visão do grupo que desde o início defendeu o uso de RDF, precisava de melhorias. Foi quando este grupo propôs em 2000, a versão RSS 1.0, voltando a utilizar RDF. Entre os recursos incorporados, está a possibilidade de utilizar módulos externos através de namespaces. Esta nova versão desagradou aos partidários da versão 0.91 que continuou evoluindo independentemente para o RSS 0.92, o RSS 0.93 e posteriormente para o RSS 0.94. Foi quando em 2002 foi lançada a versão RSS 2.0 que seguia a linha das versões 0.9x, sem utilizar RDF e incorporando facilidades introduzidas na versão 1.0, como o uso de módulos externos. Nesta versão, o significado da sigla RSS foi alterado para *Really Simple Syndication*. A Figura 3.7 ilustra o formato de um arquivo RSS na versão 2.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
   <title>Disciplinas</title>
   <link>/Avance/rest/E4659FD54E398B561AD784313350B7B5/curso/11</link>
   <description>Disciplinas do curso 11</description>
   <language>pt</language>
   <generator>jRSSGenerator by Henrique A. Viecili</generator>
   <docs>http://blogs.law.harvard.edu/tech/rss</docs>
   <item>
      <title>Criação de Páginas Web com HTML</title>
      <link>/Avance/rest/E4659FD54E398B561AD784313350B7B5/curso/11/2</link>
      <description>-</description>
      <author>Mauricio Bomfim</author>
      <pubDate>Qua, 28 Mar 2007 16:20:36 ACT</pubDate>
   </item>
   <item>
      <title>Linguagem Javascript</title>
      <link>/Avance/rest/E4659FD54E398B561AD784313350B7B5/curso/11/3</link>
      <description>-</description>

      <author>João Sergio</author>
      <pubDate>Qua, 28 Mar 2007 16:21:37 ACT</pubDate>
   </item>
</channel>
</rss>
```

Figura 3.7 - RSS versão 2.0

Em 2003, um grupo de desenvolvedores se reuniu para propor um novo padrão que pudesse substituir o RSS, acabando assim com o problema das inconsistências entre as inúmeras versões existentes. Além disso, pensava-se também em padronizar uma API que permitisse a publicação de recursos, tal como utilizado nos servidores de Blogs. Em 2004 este grupo juntou-se à *Internet Engineering Task Force* (IETF), que assim como o W3C é um organismo responsável por desenvolver e promover padrões para a Internet. Este novo formato ficou conhecido como Atom e é mais poderoso que o RSS, exatamente porque ele é composto de dois padrões: o *Atom Syndication Format*, uma linguagem baseada em XML para a descrição de informações enviadas por sindicação, e o *Atom Publishing Protocol* (APP), um protocolo baseado em HTTP, para criação e atualização de recursos na Web. Em 2005 o *Atom Syndication Format* foi oficialmente reconhecido como um padrão, através da RFC-4287 (IETF, 2005). Um exemplo deste formato é apresentado na Figura 3.8.

O *Atom Publishing Protocol* é um protocolo leve para Web Services baseado em REST, padronizado através da RFC-5023 (IETF, 2007). Em lugar de definir Web Services através de chamadas remotas (*remote procedure calls*) como no SOAP, suas interfaces são definidas num nível mais baixo, baseadas em HTTP e XML. Sendo assim, o protocolo Atom utiliza os próprios verbos definidos no protocolo HTTP (GET, POST, PUT e DELETE) para operar sobre recursos disponíveis na Web.

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
   <title>Disciplinas</title>
   <link href="/Avance/rest/E4659FD54E398B561AD784313350B7B5/curso/11"/>
   <updated>2003-12-13T18:30:02Z</updated>
   <author>
      <name>Mauricio Bomfim</name>
   </author>
   <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
   <entry>
      <title>Criação de Páginas Web com HTML</title>
      <link
href='/Avance/rest/E4659FD54E398B561AD784313350B7B5/curso/11/2'/>
      <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
      <updated>2003-12-13T18:30:02Z</updated>
      <summary>Esta disciplina se propõe a ensinar a linguagem
HTML.</summary>
   </entry>
</feed>
```

Figura 3.8 - Formato Atom

3.3.2. XForms

O HTTP é um protocolo cliente-servidor que permite que informação seja enviada nos dois sentidos. Quando visitamos um sítio na Web e recebemos uma página no nosso navegador, a informação flui do servidor para o cliente. Entretanto, quando preenchemos dados pessoais para realizar a compra de um produto, fornecemos palavras para serem pesquisadas numa ferramenta de busca, ou digitamos uma identificação bancária para acesso aos dados de uma conta corrente, estamos enviando informação no sentido contrário, do cliente para o servidor.

A linguagem HTML permite que o usuário envie informação ao servidor através do uso de formulários, que são o principal mecanismo para promover a interatividade nas aplicações Web (RAGGETT, LE HORS e JACOBS, 1999). Um formulário é implementado em HTML como uma seção de um documento que contém, entre outras coisas, alguns elementos especiais chamados controles. Exemplos de controles são: campos para digitação de texto, botões, caixas de seleção, entre outros. O usuário pode interagir com o formulário digitando informações e selecionando opções antes de enviá-lo ao servidor para ser processado. Linguagens de script processadas no cliente também podem programaticamente interagir com formulários, respondendo a eventos, e realizando ações como criíticas ou imposição de restrições de preenchimento.

Entretanto, a implementação de formulários HTML apresenta algumas limitações (BRUCHEZ, 2006):

- É dependente de dispositivo, não sendo apresentada adequadamente em navegadores fora da plataforma PC, como computadores de bolso ou telefones celulares;
- Pouca variedade de controles disponíveis, comparado aos controles existentes nas aplicações desktop;
- A impossibilidade de definir critérios de validação faz com que dependam de linguagens de script para isso, o que torna mais difícil sua implementação e manutenção;
- A mescla entre informações de apresentação com o propósito do próprio formulário num único arquivo dificulta sua legibilidade;
- Possui características limitadas no que diz respeito à acessibilidade.

A constatação destas limitações levou à criação de um grupo de trabalho do W3C para estudar este problema. Foi produzida então, a recomendação XForms (W3C INTERACTION DOMAIN; BOYER, 2007), que é uma nova linguagem, baseada em XML, para a criação de formulários na Web, que resolve as limitações mencionadas acima, apresentadas pelos formulários HTML.

3.3.2.1. Características

XForms utiliza o padrão de arquitetura de software MVC (*Model-View-Controller*), implicando numa separação clara entre a descrição dos dados tratados pelo formulário (modelo), a interface com o usuário (vista) e as regras de negócio que definem as validações e críticas nos dados fornecidos (controle). Como a interface é descrita de maneira abstrata, cabe ao interpretador definir a forma com que cada elemento será apresentado ao usuário, dependendo da plataforma utilizada. Assim, um campo para entrada de uma data pode ser apresentado como um calendário por um navegador em plataforma PC ou como um seimples campo para digitação de texto num telefone celular.

Por ser implementada como um vocabulário XML, XForms apresenta ainda as seguintes vantagens adicionais:

- É possível incorporar um XForms em documentos hospedeiros que também utilizam a sintaxe XML como, por exemplo, num documento em formato XHTML (W3C, 2000);

- É possível enviar as informações fornecidas pelo usuário em formato XML, ao invés de utilizar pares nome/valor como nos formulários HTML tradicionais, facilitando assim a integração com os serviços Web (veja Seção 3.2.1) que esperam receber requisições neste formato;

- A descrição de regras para validação pode ser realizada em XML (normalmente utilizando XML Schema).

A Figura 3.9 apresenta um exemplo do uso de XForms onde é definido um formulário simples com dois campos para digitação de nome e telefone, além de um botão de envio. O modelo de dados está descito entre as linhas 2 e 10 e a apresentação, na forma de controles e suas respectivas legendas, está entre as linhas 11 e 13. Nas linhas 11 e 12, os elementos da interface referenciam o modelo através de expressões XPath.

```
<?xml version="1.0" encoding="utf-8"?>
1 <xforms>
2 <model>
3 <instance>
4   <pessoa>
5     <nome/>
6     <tel/>
7   </pessoa>
8 </instance>
9 <submission id="form1" action="submit.asp" method="get"/>
10 </model>
11 <input ref="/pessoa/nome"><label>Nome</label></input>
12 <input ref="/pessoa/tel"><label>Telefone</label></input>
13 <submit submission="form1"><label>Enviar</label></submit>
14 </xforms>
```

Figura 3.9 – Exemplo de um documento XForms.

Um exemplo mais completo, é apresentado na Figura 3.10 onde o XForms é incorporado num documento XHTML. Aqui, nas linhas 2 a 5, são definidos namespaces a serem utilizados para referenciar os diferentes vocabulários utilizados. A lógica de negócio que definirá as validações necessárias está representada nas associações dos tipos de dados nas linhas 10 e 11 além da definição de um campo de preenchimento obrigatório na linha 14.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <html
3 xmlns:xf="http://www.w3.org/2002/xforms"
4 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6 <head>
7 <xf:model>
8 <xf:instance>
9 <pessoa xmlns="">
10   <nome xsi:type="xsd:string"/>
11   <idade  xsi:type="xsd:integer"/>
12 </person>
13 </xf:instance>
14 <bind nodeset="pessoa/nome" required="true()"/>
15   <xf:submission id="form1" method="get" action="submit.asp"/>
16 </xf:model>
17 </head>
18
19 <body>
20 <xf:input ref="nome">
21 <xf:label>Nome</xf:label></xf:input>
22 <br />
23 <xf:input ref="idade">
24 <xf:label>Idade</xf:label></xf:input>
25 <br />
26 <br />
27 <xf:submit submission="form1">
28 <xf:label>Enviar</xf:label></xf:submit>
29 </body>
30 </html>
```

Figura 3.10 - Exemplo de uso de XForms incorporado num documento XHTML

3.3.2.2. Interpretação de XForms

Existem várias implementações capazes de interpretar XForms no cliente ou no servidor (W3C XFORMS GROUP WIKI PUBLIC). As soluções que interpretam XForms no cliente são compostas de programas que devem ser instalados pelo usuário, em geral, na forma de plug-ins dos navegadores, adicionando a estes a capacidade de interpretar os comandos XForms – apresentando-os e executando as regras de negócio neles definidas. A principal vantagem desta abordagem é a redução do tráfego na rede uma vez que o processamento é totalmente realizado nos clientes.

A interpretação no servidor realiza-se traduzindo os comandos para formatos compreensíveis pelos navegadores despreparados para receber XForms – normalmente utilizando HTML e javascript – não exigindo instalação de nenhum software adicional. Entretanto o uso das facilidades incorporadas com o novo padrão fica restrito, podendo não funcionar adequadamente nos navegadores responsáveis por sua interpretação.

XForms são considerados os sucessores dos formulários HTML, e futuramente serão o padrão para uso de formulários adotado pela versão XHTML 2.0. Sendo assim é de se esperar que as futuras versões dos principais navegadores já possuam suporte nativo a este formato.

3.3.3. JavaScript Object Notation (JSON)

O *JavaScript Object Notation* (JSON)[28], é um formato leve para intercâmbio de dados baseado na sintaxe definida no padrão ECMAScript/Javascript para definição de estruturas de dados, tais como coleções ou listas ordenadas de valores (CROCKFORD, 2006), que pode ser facilmente lido ou escrito tanto por humanos quanto por máquinas. Além disso, sua interpretação e geração não são restritas ao javascript, podendo ser realizadas através de qualquer outra linguagem.

O exemplo de arquivo JSON apresentado na Figura 3.11 representa uma coleção contendo os valores de título, tema, link e itens; onde itens por sua vez, é um vetor com quatro ocorrências de outra coleção que contêm os elementos: tipo, título, link, descrição, data e autor.

[28] Mais informaçõpes sobre o formato JSON em: http://www.json.org

```
{"título": "Lista De Assuntos",
 "tema": "HTML Avançado",
 "link": "/Forum/rest/D6535204E857F7EC0F8046A0D0CE2879/4/27",
 "itens": [
           { "tipo": "item",
             "titulo": "Recursos Multimídia",
             "link":
"/Forum/rest/D6535204E857F7EC0F8046A0D0CE2879/4/27/15",
             "descrição": "Imagemaps, menus com javascript",
             "data": "Ter, 17 Jul 2007 16:58:30 ACT",
             "autor": "Mauricio Bomfim" },
           { "tipo": "item",
             "titulo": "Folhas de estilo",
             "link":
"/Forum/rest/D6535204E857F7EC0F8046A0D0CE2879/4/27/16",
             "descrição": "Separando o conteúdo da apresentação",
             "data": "Ter, 17 Jul 2007 16:59:13 ACT",
             "autor": "Mauricio Bomfim" },
           { "tipo": "botao",
             "titulo": "Novo(a)",
             "link":
"/Forum/rest/D6535204E857F7EC0F8046A0D0CE2879/4/27/inserir"},
           { "tipo": "botao",
             "titulo": "Tema",
             "link": "/Forum/rest/D6535204E857F7EC0F8046A0D0CE2879/4"}
         ]
}
```

Figura 3.11 - Exemplo de uma representação JSON

JSON é baseado em estruturas de dados universais, implementadas por praticamente todas as linguagens de programação modernas:

- Coleções de pares do tipo nome/valor – representados por uma seqüência de pares string:valor, separados por vírgulas e entre chaves. São implementados como objetos em Javascript.

- Listas ordenada de valores (vetores) – representados por uma seqüência de valores separados por vírgulas entre colchetes. São implementados como arrays em Javascript.

Os tipos de dados representáveis em JSON são: strings, números, valores lógicos (true ou false), e a constante null, além dos tipos estruturados objetos e arrays mencionados anteriormente.

A escolha entre utilizar JSON ou XML é bastante controversa, existindo defensores das duas abordagens[29]. Entretanto, JSON tem se mostrado uma alternativa bastante comum nas aplicações Web 2.0, devido à simplicidade com que é possível interpretá-lo através da linguagem Javascript utilizada pelos clientes Ajax.

[29] Em http://www.json.org existem ponteiros para alguns artigos com estes dois pontos de vista.

3.4. Autenticação de usuários

Autenticação é o processo de identificação da identidade de um usuário que acessa um determinado sistema. Considerando que as aplicações Web são baseadas num protocolo sem estado, a autenticação deve prover uma maneira de determinar qual usuário é responsável por cada requisição realizada. Uma vez que um usuário tenha sido autenticado, é necessário poder definir quais as operações permitidas a ele. Autorização é o nome dado a este processo, de determinar se uma requisição pode ou não ser realizada por um dado usuário.

A autenticação e a autorização de usuários no contexto das aplicações Internet são problemas que têm sido tratados há algum tempo, provendo-se soluções específicas dentro do contexto de cada aplicação. Com a Web 2.0, e a multiplicação dos serviços disponíveis na Internet, esta questão voltou a ser discutida, devido a dois problemas principais. O primeiro, relacionado à autenticação, é a necessidade dos usuários se cadastrarem em cada serviço provocando a manutenção de um grande número de contas e senhas. O outro problema, relacionado à autorização, ocorre quando uma aplicação consumidora depende do acesso através de APIs, aos dados provenientes de outras aplicações já existentes. Não se trata apenas de determinar se um usuário é válido, mas sim de saber se ele autoriza ou não o acesso aos seus dados armazenados nos outros serviços. Dois padrões emergentes utilizados na Web 2.0 com o intuito de resolver estes problemas são o OpenID e o OAuth.

3.4.1. OpenID

A grande diversidade de novos serviços disponíveis atualmente na Internet produziu um problema para os seus usuários que é a necessidade de manter inúmeros cadastros de contas e senhas para acessá-los. Blogs, fóruns de discussão, wikis, repositórios de fotos, redes sociais, são exemplos de sistemas onde o acesso é controlado através da identificação de usuários cadastrados. Assim, seria conveniente para os usuários que estes pudessem se identificar apenas uma vez numa sessão, de forma que todas as aplicações pudessem confiar naquela única autenticação. Este é o princípio básico do SSO (*Single Sign-On*), definido pelo Open Group como um mecanismo onde uma única ação de autenticação do usuário pode autorizar seu acesso a todos os computadores e sistemas onde este esteja registrado, sem a necessidade de diferentes contas e senhas (OPEN GROUP, 2005).

Algumas soluções para este problema são apresentadas por Myllyniemi (2006). Entre estas, o OpenID é um sistema de identificação de usuários para serviços oferecidos na Internet, baseado

numa rede distribuída de servidores de autenticação (também chamados de provedores de identidade) que permite que o usuário mantenha um único cadastro com o qual fará o acesso aos serviços que desejar. Além disso, como o sistema é descentralizado, o usuário pode escolher o provedor de identidade onde deseje manter seu cadastro, de acordo com sua conveniência e confiança. O usuário recebe então um identificador padronizado, normalmente uma URL (BERNERS-LEE, MASINTER e McCAHILL, 1994), URI (BERNERS-LEE, FIELDING e MASINTER, 2005) ou XRI (OASIS, 2007), com o qual ele poderá se identificar, nos diferentes serviços. O OpenId é uma tecnologia aberta e gratuita, que têm se tornado cada vez mais popular, sendo adotado por um número crescente de empresas e serviços na Internet (POWELL e RECORDON, 2007).

O processo de autenticação envolve três atores (o usuário, a aplicação que precisa autenticar o usuário, e o provedor de identidade OpenId) e é realizado da seguinte forma:

1. O usuário identifica-se, fornecendo para a aplicação, a sua URL, obtida através do seu provedor de identidade.

2. Cabe agora à aplicação:

   - Normalizar o identificador fornecido pelo usuário para um formato padrão.

   - Conectar-se à URL fornecida pelo usuário, já normalizada, requisitando uma descrição dos servidores OpenID que possam ser utilizados para autenticar o identificador fornecido pelo usuário. A versão 2.0 do OpenID permite que, dependendo do tipo da identificação do usuário, sejam retornados diferentes formatos como: a resolução de uma XRI, um arquivo XRDS de acordo com o protocolo YADIS (MILLER, 2006), ou um arquivo HTML contendo tags específicas que indiquem o endereço dos servidores a serem procurados.

   - Opcionalmente, a aplicação estabelece uma associação com o servidor OpenID, permitindo uma comunicação segura entre eles, para posteriormente, requisitar a autenticação propriamente dita.

   - Caso a aplicação não deseje manter estado, guardando a associação para ser utilizada em futuras requisições de autenticação, é possível não estabelecer esta associação. Neste caso, a aplicação precisará verificar a autenticação numa requisição adicional posterior.

Todo este processo pode ser mais bem compreendido através do diagrama de seqüência apresentado na Figura 3.12.

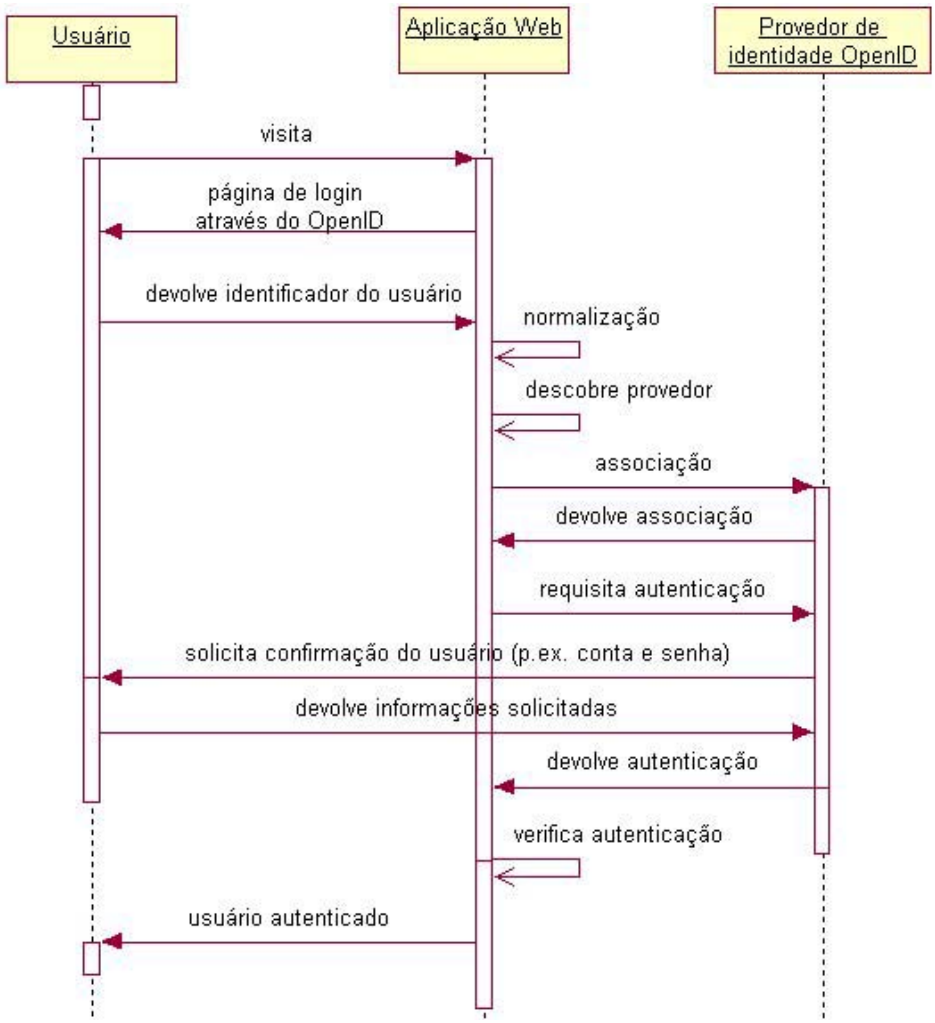

Figura 3.12 - Diagrama de seqüência do processo de autenticação OpenID. Adaptado de Stepka (2007).

É possível citar as seguintes vantagens da utilização do protocolo OpenID como um processo de autenticação único compartilhado entre diferentes aplicações:

- As aplicações não precisam ter acesso a informações particulares dos seus usuários como e-mails e senhas;
- O OpenID é baseado apenas em requisições e respostas HTTP(S), não exigindo nenhuma habilidade especial dos clientes;
- Não depende do uso de cookies nem de nenhum outro mecanismo de gerenciamento de sessões;

- É possível manter no cadastro informações específicas do domínio de determinada aplicação, mesmo que estas não estejam previstas na especificação original.

OpenID, entretanto, não deve ser visto como uma solução para todos os problemas de identificação de usuários, não sendo adequado, por exemplo, para sistemas que exijam um maior grau de segurança como aplicações bancárias ou de comércio eletrônico.

3.4.2. OAuth

O OpenID permite ao usuário a criação de uma única credencial que pode ser utilizada para lhe dar acesso a diferentes sites. Entretanto ele não resolve o problema de autenticação quando uma aplicação consumidora depende do acesso aos dados provenientes de outras aplicações já existentes através de APIs. Em outras palavras, a autenticação através de OpenID informa à aplicação se um usuário é válido mas não diz se ele autoriza o acesso aos seus dados armazenados por outros serviços. É necessário, portanto, um procedimento de autenticação através do qual o usuário autorize operações deste tipo.

O OAuth[30] é uma tentativa de padronização dos processos de autenticação e autorização para APIs na Web. É um protocolo aberto, que permite que sítios, ou aplicações ditas Consumidoras (*Consumers*), possam acessar recursos protegidos de um serviço Web ditos Provedores do Serviço (*Service Providers*) através de sua API, sem que seja necessário que os usuários forneçam suas senhas às aplicações Consumidoras. Assim, o cliente redireciona o usuário para o provedor do serviço para que este solicite diretamente suas credenciais e faça a autenticação. Uma chave é devolvida à aplicação consumidora que permite a realização de novas requisições.

Existem diversas soluções proprietárias que utilizam este tipo de abordagem, entre elas o Google AuthSub[31] , o FlickrAuth[32] , o AOL OpenAuth[33] e o Yahoo BBAuth[34] . A especificação OAuth 1.0 (ATWOOD et al, 2007) foi desenvolvida em 2007 e, por ser muito recente, ainda não existem muitas implementações disponíveis.

Um exemplo do uso deste protocolo, extraído da especificação do OAuth, ilustra uma sessão típica, incluindo os papéis dos três atores neste processo: o usuário, a aplicação consumidora e a

---

[30] http://oauth.net
[31] http://code.google.com/apis/gdata/auth.html
[32] http://www.flickr.com/services/api/auth.spec.html
[33] http://dev.aol.com/openauth
[34] http://developer.yahoo.com/auth/

aplicação provedora do serviço. Seja um sítio que oferece o serviço de impressão de fotos (Serviço A), permitindo que estas sejam obtidas a partir de um outro sítio que ofereça um serviço de compartilhamento de fotos (Serviço B). O caso de uso para realizar esta interação pode ser descrito da seguinte forma:

1) O usuário se autentica no serviço A;

2) O usuário inicia um pedido de impressão;

3) O serviço A pergunta pelas fotos a serem impressas e oferece opções de locais de onde estas fotos podem ser obtidas;

4) O usuário seleciona o site de origem das fotos (serviço B) a partir da lista de opções oferecida;

5) O serviço A redireciona o usuário para o serviço B onde estão as fotos para que este solicite a autenticação do usuário;

6) O serviço B solicita autorização do usuário;

7) O usuário fornece sua identificação ao serviço B e autoriza o acesso do serviço A às suas fotos;

8) O usuário é redirecionado de volta ao serviço A que agora tem acesso às fotos armazenadas no serviço B.

O processo de autenticação é controlado por chaves de acesso chamadas de *Tokens*. A aplicação consumidora ao receber uma requisição de um serviço que envolve informações que devam ser obtidas a partir de outra aplicação, deve executar os seguintes procedimentos (Figura 3.13):

1) Solicitar um *Request Token* à aplicação provedora. Este *token* é uma chave usada pelo consumidor para solicitar uma autenticação.

2) Redirecionar para a aplicação provedora que será responsável por autenticar o usuário e solicitar a sua autorização para a operação que a aplicação consumidora deseja realizar. Devem ser enviados o *Request Token* obtido anteriormente e uma URL de retorno da aplicação.

3) Ao ser chamada de volta através da URL de retorno, a aplicação consumidora deve agora solicitar a troca do *Request Token* por um *Access Token*, que permitirá o acesso aos dados.

4) Uma vez que o *Access Token* tenha sido obtido, os recursos protegidos da aplicação provedora podem ser agora acessados.

Todas as requisições entre as aplicações (consumidora e provedora) devem ser assinadas, garantindo assim a segurança nestas comunicações.

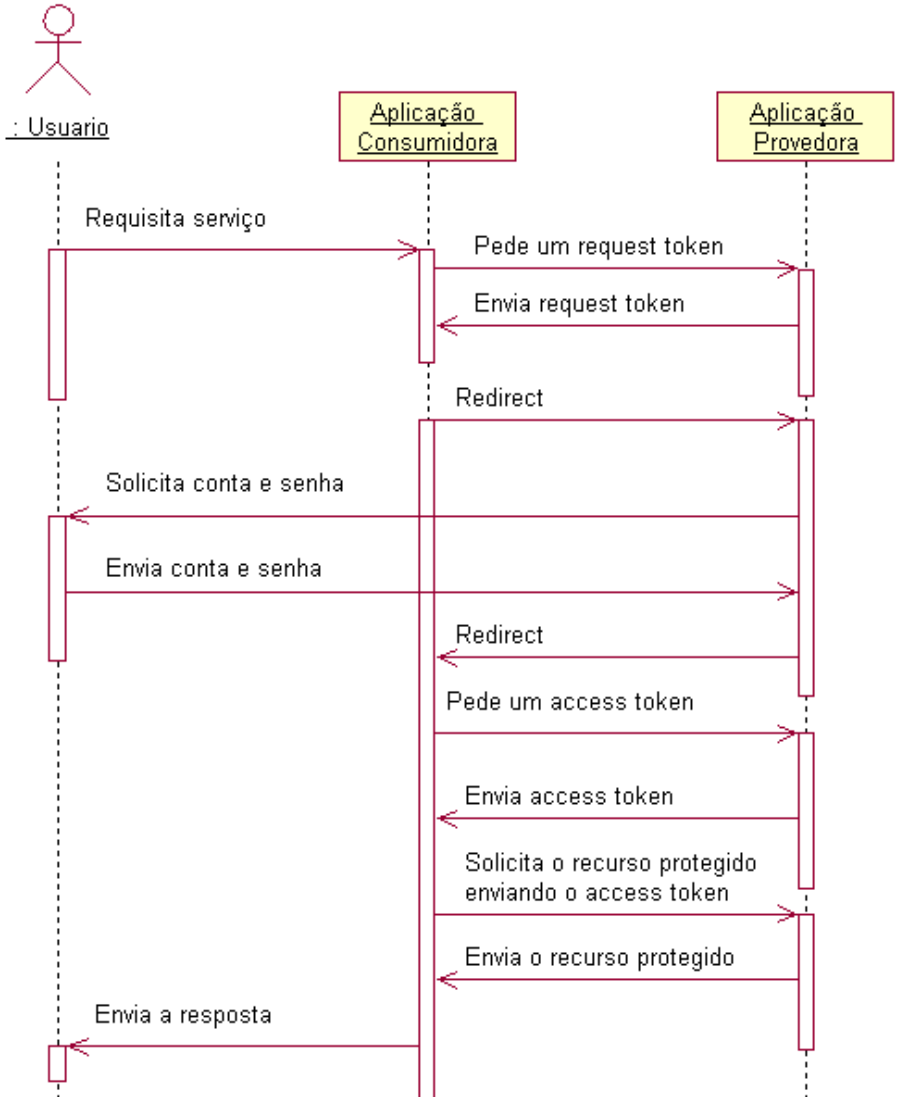

Figura 3.13 - Diagrama de seqüência do processo de autenticação OAuth

3.5. Considerações Finais

Uma das principais razões para o surgimento e a popularização da Web 2.0 foi o desenvolvimento de uma nova geração de tecnologias e padrões para a Web que pudessem apoiar suas idéias centrais. Assim, a utilização da Web como uma plataforma proporcionando uma experiência rica ao usuário, só foi viabilizada com o surgimento do AJAX e de alternativas leves para a componentização através do estilo de arquitetura REST. A utilização de padrões como o RSS e o Atom utilizados para sindicação, o XForms e a sintaxe JSON também podem favorecer esta filosofia de desenvolvimento distribuído.

Para uniformizar os processos de autenticação utilizados por estas aplicações, surgem alguns padrões emergentes como o OpenID e o OAuth.

Capítulo 4
Um Estudo das APIs Web 2.0 Existentes Visando Facilitar a Integração de Aplicações

Este capítulo apresenta alguns aspectos sobre a integração de aplicações na Web 2.0, com o intuito de definir um modelo que permita a criação de um ambiente de aprendizagem baseado nos seguintes requisitos principais: possibilidade de integrar aplicações escolhidas livremente pelo usuário e facilidade de uso. Para isso apresentam-se algumas abordagens normalmente utilizadas para a integração de aplicações, assim como suas principais características e limitações. A partir daí, é proposta uma nova alternativa que será utilizada neste trabalho e que consiste na integração automática de aplicações.

Uma rede sistêmica[35] organiza os principais aspectos relacionados à componentização das aplicações da Web 2.0, criada a partir de um estudo exploratório comparando algumas APIs existentes. Desta forma foi possível a classificação destas APIs de acordo com o grau de adesão de cada uma delas à arquitetura REST e seus mecanismos de autenticação. As conclusões deste estudo serviram como base para a definição do modelo de integração apresentado no Capítulo 5.

[35] Conforme sugerido por Bliss, Monk e Ogborn (1983), Redes Sistêmicas são formas gráficas de se fazer uma representação resumida de um determinado conhecimento. Trata-se de um instrumento de pesquisa que, a exemplo dos Mapas Conceituais, vem sendo cada vez mais utilizado entre os pesquisadores da área educacional para análise de dados qualitativos. Sua origem, segundo esses autores, vem da lingüística sistêmica ou da também chamada sociolingüística.

4.1. Abordagens Utilizadas para Integração de Aplicações

Antes mesmo do advento da Web 2.0, soluções têm sido buscadas para a integração de dados provenientes de diferentes aplicações Web. A grande dificuldade encontrada neste sentido foi o fato das páginas Web serem voltadas primariamente para sua apresentação ao usuário. Sendo assim, utiliza-se um arquivo HTML para representar, não só o conteúdo a ser exibido, mas também a forma com que isso será feito, dificultando a criação de programas capazes de processar o conteúdo destas páginas.

Uma alternativa para este problema é a utilização de técnicas de captura de tela (*screen-scrapping*) realizada por programas conhecidos como extratores (ou *wrappers*) (YEE, 2008, p59). Estes programas são capazes de obter a saída em HTML e, conhecendo sua organização interna, separar as informações relevantes para o seu processamento. Um exemplo de aplicação desta técnica seria a criação de um sistema de consulta de preços que fosse capaz de coletar produtos em diferentes sítios de fornecedores, apresentando-os numa única lista ordenada. O problema deste tipo de solução é que o extrator precisa ser desenvolvido especificamente para processar um formato de saída determinado. Caso o projeto visual do sítio de uma das fontes de informação seja modificado – o que é uma prática comum nos sítios Web – o extrator deverá ser adaptado de forma que possa compreender e processar o novo formato de apresentação.

O novo modelo de arquitetura utilizado no desenvolvimento de aplicações Web 2.0, que se baseia na definição de APIs abertas, permite que o desenvolvimento de mashups seja realizado com muito mais facilidade. Normalmente, a integração de aplicações utiliza uma das abordagens descritas a seguir.

4.1.1. Programação individual de mashups

A primeira abordagem é aquela onde o programador desenvolve o mashup através da criação de um programa com uma finalidade específica. Por exemplo, a combinação de um mapa obtido através da API do Google Maps com uma base de dados montada colaborativamente pela própria comunidade de usuários, onde são armazenadas as localizações de alguns pontos de interesse, como restaurantes, bares e cinemas, como no serviço u.find[36] (Figura 4.1). Neste caso o trabalho de programação é desenvolvido sob medida para resolver um problema específico. A

[36] http://www.ufind.com.br

integração é realizada em tempo de desenvolvimento do mashup, não existindo possibilidade do usuário interferir neste processo.

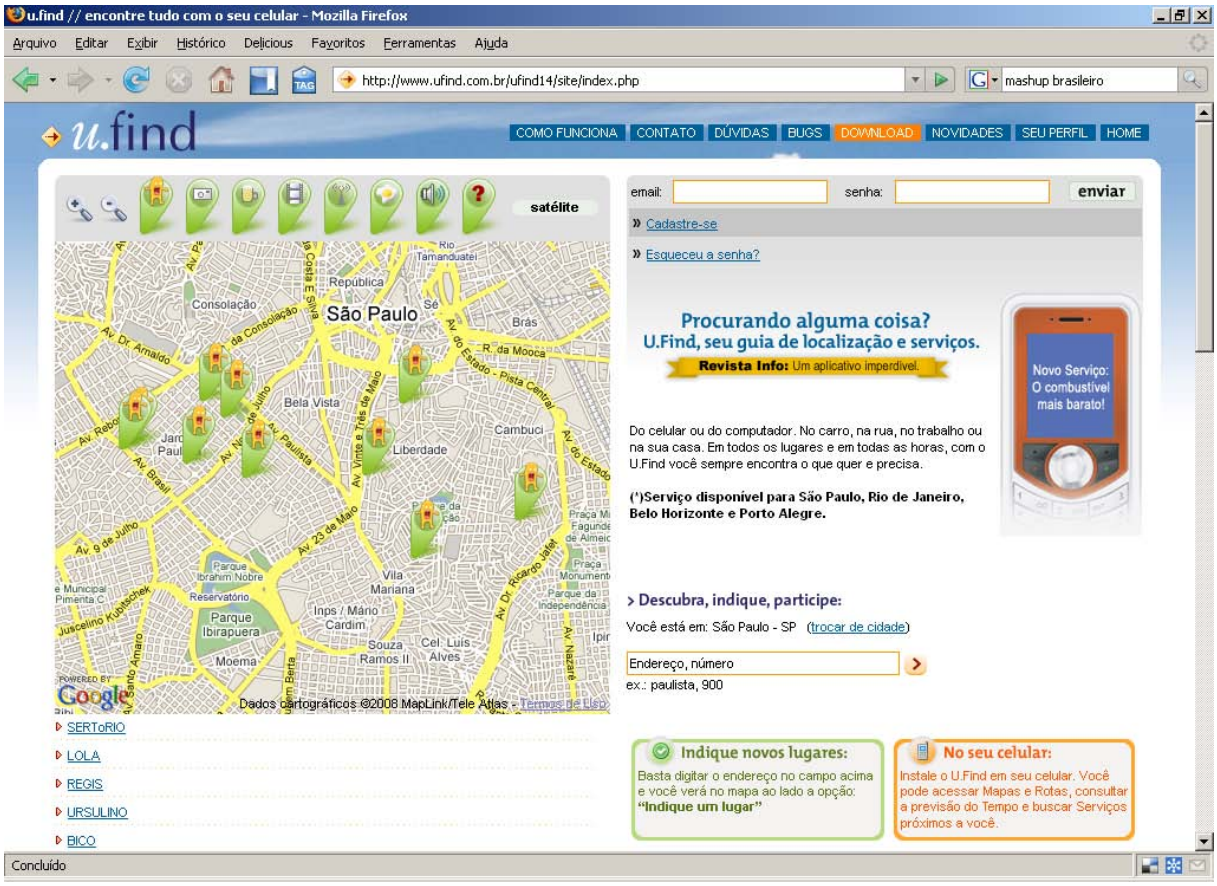

Figura 4.1 – Mashup que permite a localização de pontos de interesse num mapa

4.1.2. Editores de mashups

Alguns ambientes que visam facilitar a criação de mashups como o Dapper[37], o Google Mashup Editor[38], o Microsoft Popfly[39] e o Yahoo Pipes[40] (Figura 4.2) são soluções um pouco mais genéricas, pois podem permitir que o próprio usuário, com algum conhecimento técnico, defina as suas fontes de dados e a maneira com que estes devem ser combinados. Nestes editores, quanto maior o poder e a flexibilidade dados ao usuário para definir suas aplicações, maior o conhecimento técnico exigido para produzi-las.

[37] http://www.dapper.net/
[38] http://www.googlemashups.com
[39] http://www.popfly.com/
[40] http://pipes.yahoo.com

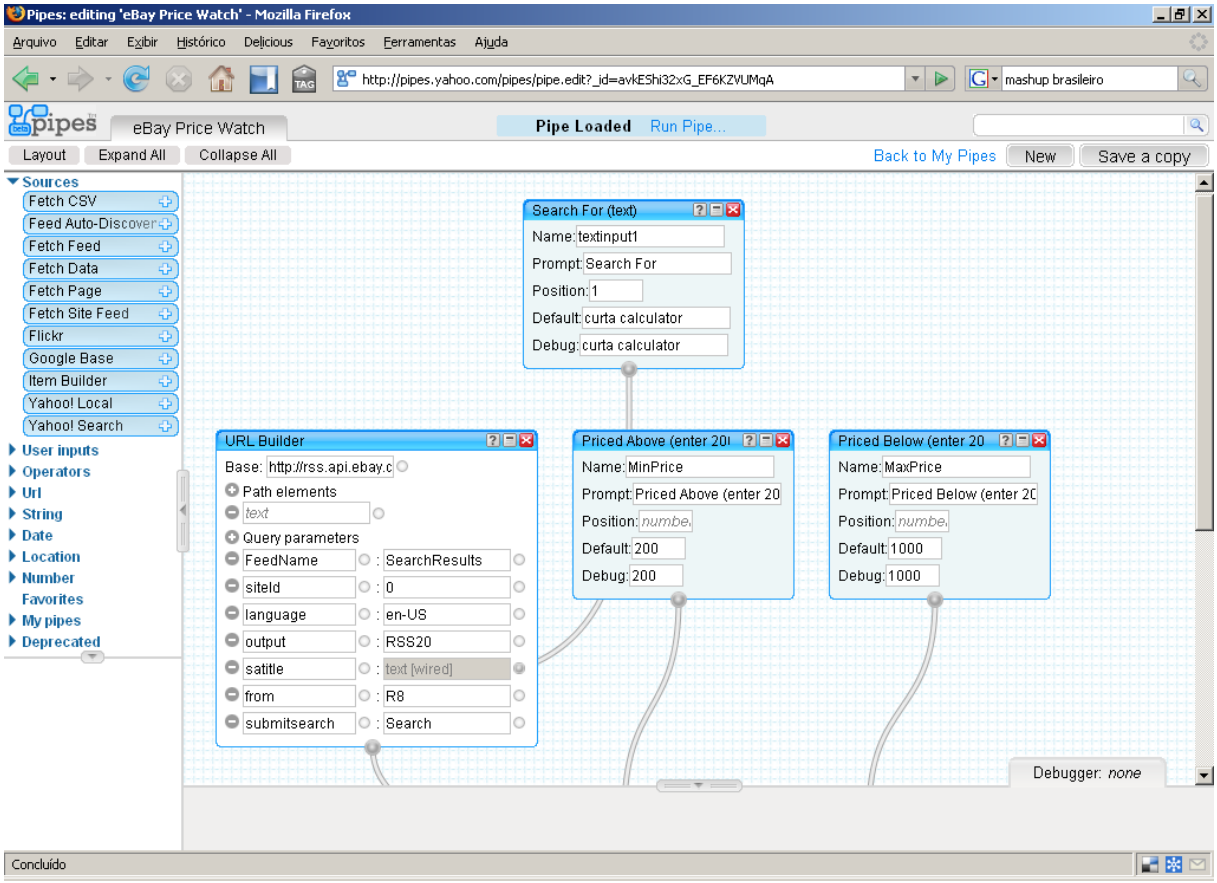

Figura 4.2 – Ambiente onde é possível definir critérios para a combinação de informações

4.1.3. Plataformas para agregação de Web Widgets

Outra abordagem possível é a utilização de Web Widgets para plataformas agregadoras como iGoogle[41], PageFlakes[42], Netvibes[43] ou MyYahoo[44]. Estas plataformas são ambientes personalizados pelo próprio usuário onde podem ser incorporados os aplicativos (ou *Widgets*) selecionados a partir de uma lista fornecida pelo próprio serviço.

Estas Widgets são pequenas porções de código que implementam alguma funcionalidade predeterminada e são desenvolvidas de acordo com um padrão definido pela aplicação hospedeira na qual ela será embutida. Nestes casos, o usuário está limitado à utilização das Widgets existentes, embora estas plataformas geralmente ofereçam APIs para facilitar o processo de desenvolvimento de novas Widgets. O Netvibes, por exemplo, utiliza-se do Netvibes

[41] http://www.igoogle.com
[42] http://www.pageflakes.com
[43] http://www.netvibes.com
[44] http://my.yahoo.com/

Universal Widget API (UWA), um framework gratuito que utiliza padrões Web abertos como XHTML, CSS, Javascript e Ajax, e é compatível com as principais plataformas existentes.

A Figura 4.3 apresenta um exemplo de utilização do Netvibes, onde Widgets podem ser utilizadas como portas de entrada para outras aplicações como Gmail ou Google Maps, ou para a integração de informações provenientes de mecanismos de sindicação.

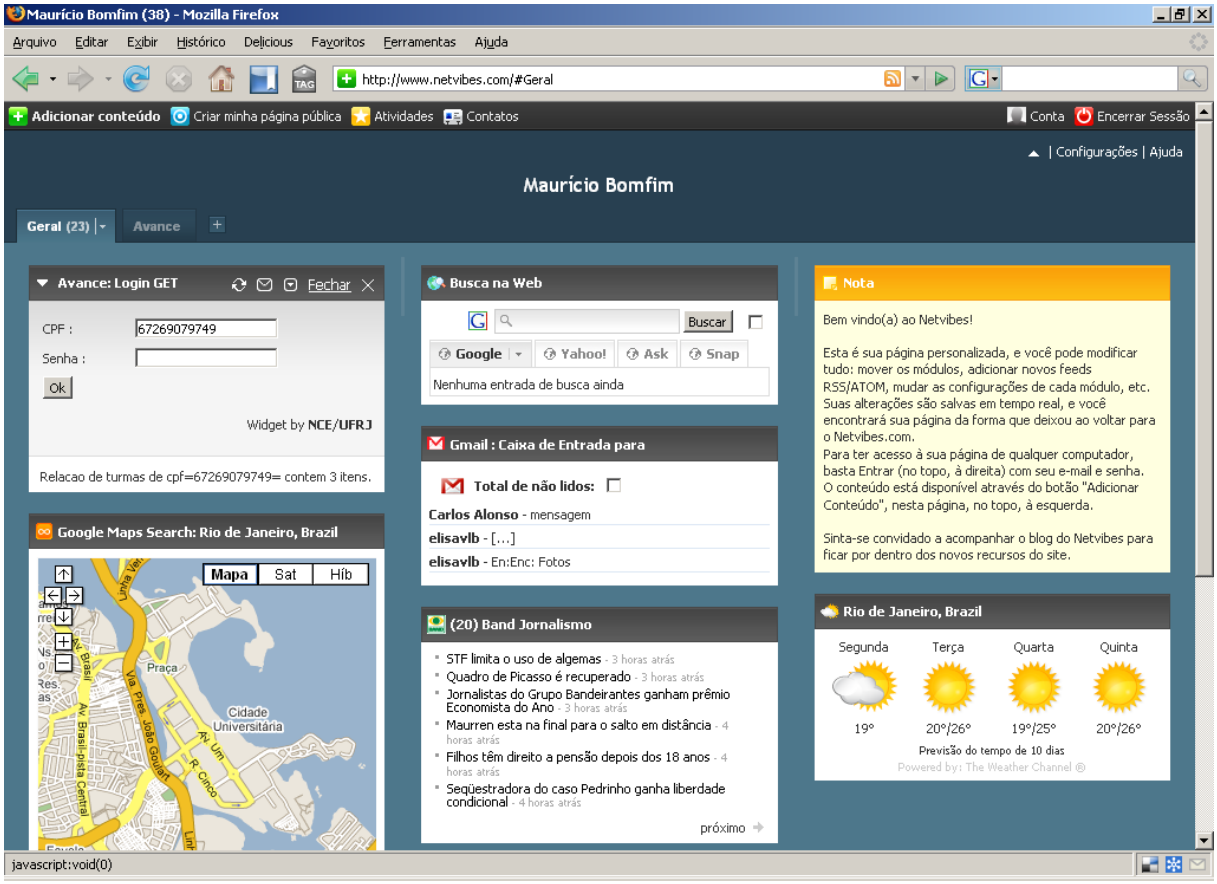

Figura 4.3 – Netvibes: Exemplo de plataforma que permite a agregação de widgets

4.2. Integração Automática de Aplicações

O problema das soluções vistas anteriormente é que, geralmente elas oferecem um conjunto predeterminado de componentes, não permitindo que o usuário escolha livremente a aplicação que deseja utilizar. Para integrar uma nova aplicação a um ambiente já existente, é necessário programar um novo componente, tarefa que não é necessariamente muito simples, além de exigir conhecimentos técnicos que nem sempre o usuário possui.

Com o intuito de oferecer ao usuário a possibilidade de incorporar de maneira simples – e sem a necessidade de programação – qualquer aplicação externa que possua uma API aberta, a

abordagem que foi explorada neste trabalho é a da integração automática. Assim, o ambiente deve permitir que o usuário cadastre uma API através do fornecimento de sua descrição formal para que, a partir daí, o sistema seja capaz de interpretá-la, incorporando suas funcionalidades para o ambiente automaticamente. O quadro 4.1 apresenta uma comparação entre as abordagens mencionadas na Seção 4.1 e a solução proposta, onde o usuário integra aplicações automaticamente.

Quadro 4.1 – Comparação entre as abordagens de criação de mashups levantadas

	Programação individual de Mashups	**Editores de Mashups**	**Plataformas de agregação de Widgets**	**Ambiente para integração automática de aplicações**
O que é?	Desenvolvimento de um programa capaz de mesclar funcionalidades de aplicações diferentes, com uma finalidade específica.	Plataformas genéricas que permitem o desenvolvimento de mashups a partir de algumas fontes de dados disponíveis.	Ambientes personalizados pelo usuário onde podem ser incorporados os aplicativos (ou Widgets).	Ambientes personalizados onde o usuário pode incorporar um novo aplicativo automaticamente.
A quem se destina?	São necessários conhecimentos de programação. Esta tarefa não é realizável por usuários finais.	Embora o uso destes editores não exija conhecimentos de programação, a operação dos mesmos, muitas vezes, não é intuitiva para o usuário final. Quanto maior o poder e a flexibilidade dados ao usuário, maior o conhecimento técnico exigido.	A operação é muito simples. A seleção das Widgets é feita através de uma interface amigável, arrastando e soltando componentes.	A operação é muito simples: basta informar o endereço da API e selecionar dentre os métodos/ recursos disponíveis na mesma, quais são aqueles que o usuário deseja utilizar.
Características e Limitações.	O programador possui recursos ilimitados sobre a aplicação (desde que as APIs utilizadas ofereçam as operações necessárias).	O escopo destas ferramentas está limitado a algumas fontes de dados provenientes de feeds RSS ou de algumas APIs de aplicações populares.	O usuário está limitado às Widgets disponíveis. Criar uma nova Widget, nem sempre é uma tarefa simples.	O usuário deve poder incorporar funcionalidades de qualquer aplicação que possua uma API aberta.

4.3. Análise e Classificação das APIs Web 2.0

Para que seja possível definir uma estratégia que viabilize o processo de integração automática de aplicações, foi realizado um estudo exploratório comparando algumas APIs existentes. Foram analisados: até que ponto as aplicações da Web 2.0 são aderentes ao estilo de arquitetura REST e os modelos de autenticação utilizados por estas aplicações. As principais características levantadas que as diferenciam foram organizadas numa rede sistêmica conforme apresentado na Figura 4.4. Estas características são discutidas nas seções 4.3.1 e 4.3.2, a seguir.

Este estudo levou à conclusão que dois problemas precisam ser resolvidos quando desejamos automatizar a integração de aplicações: a existência de APIs que não são totalmente aderentes aos preceitos REST e a falta de padronização no processo de autenticação.

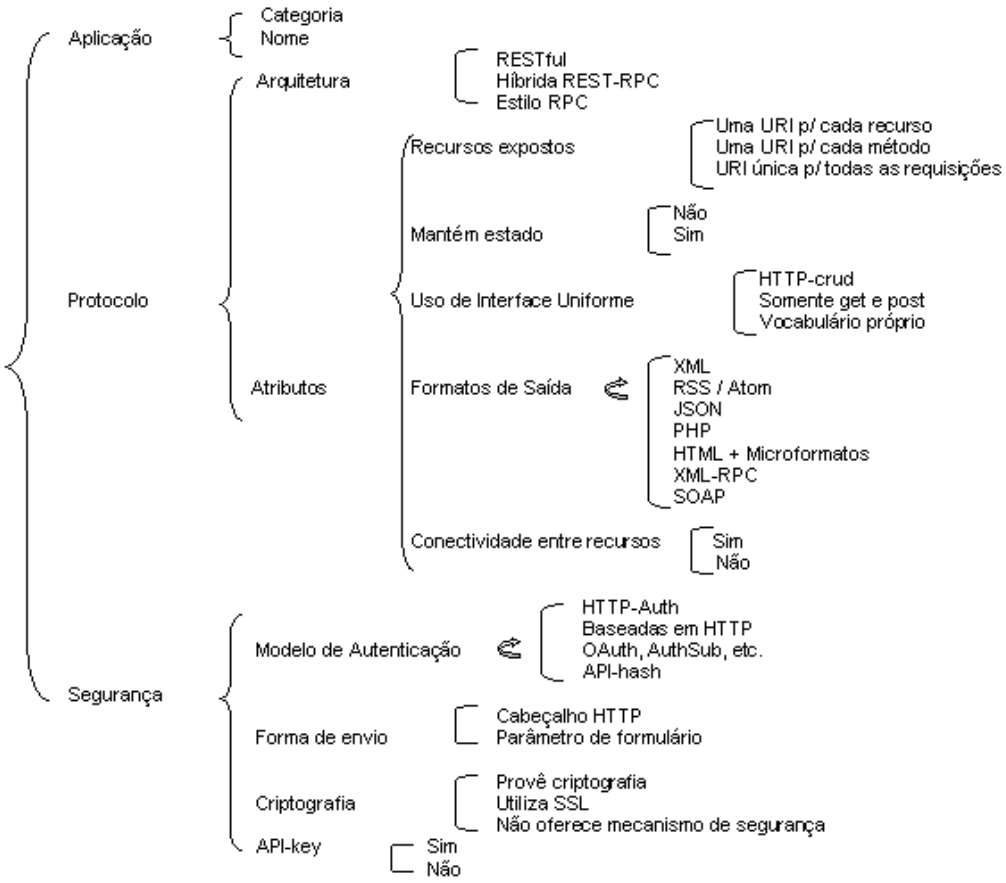

Figura 4.4 – Rede sistêmica das APIs da Web 2.0

4.3.1. Quanto ao estilo de arquitetura

Como visto no Capítulo 3, a principal tecnologia que apóia este modelo de componentização da Web 2.0 é o REST, um estilo de arquitetura de software para sistemas hipermídia distribuídos que se aplica perfeitamente ao funcionamento do protocolo HTTP e conseqüentemente, da Web.

Apesar das aparentes vantagens existentes em seguir a arquitetura REST, muitas APIs não o fazem completamente, levando à criação de sistemas híbridos.

Algumas situações que ocorrem freqüentemente, contrariando os princípios do modelo REST e fazendo com que as APIs deixem de segui-lo totalmente, são:

- A exposição de métodos em vez de recursos;
- A manutenção de estado, com intuito de evitar o envio repetido das credenciais de autenticação do usuário a cada nova requisição de um recurso;
- Uso de interface uniforme incompleta, uma vez que alguns firewalls não permitem o tráfego de operações que utilizam os métodos PUT e DELETE, e além disso, através de formulários HTML só é possível fazer requisições do tipo GET ou POST;
- Ausência de conectividade entre as representações dos recursos, inviabilizando assim a navegação direta entre os recursos da aplicação.

A principal alternativa ao uso de REST na definição de APIs é o RPC ou Remote Procedure Call (NELSON, 1981), um protocolo para chamada remota de procedimentos. A maior diferença conceitual entre REST e RPC é o fato de RPC ser baseado na diversidade de operações do protocolo, enquanto que REST baseia-se na diversidade de recursos. O SOAP é um exemplo de protocolo do tipo RPC, proposto pela W3C, utilizado com freqüência na definição de Serviços Web.

De acordo com a aderência das APIs aos preceitos das arquiteturas REST ou RPC, estas podem ser consideradas como representativas de uma das seguintes classificações propostas por Richardson e Ruby (2007):

1. RESTful – Totalmente aderente à arquitetura REST. Suas principais características são: expõe os recursos da aplicação através de URIs e utiliza uma interface uniforme (veja a Seção 3).
2. Híbrida REST-RPC – Não pode ser considerada totalmente REST nem totalmente RPC, incorporando simultaneamente características destas duas arquiteturas.
3. Estilo RPC – Totalmente aderente à arquitetura RPC. Suas principais características são: a utilização de um único endereço para realização das requisições; a utilização de envelopes

de dados que são enviados entre o cliente e o servidor; e a definição de um novo vocabulário para especificar cada operação a ser realizada através da API.

4.3.1.1. Critérios para a Classificação de Aplicações REST

Baseado na própria definição do estilo de arquitetura REST apresentado na Seção 3.2.2, podemos eleger cinco critérios para comparar as APIs, permitindo enquadrá-las numa das três classificações vistas anteriormente. Assim, para que uma aplicação seja considerada totalmente REST, é necessário que:

- **Seus recursos sejam expostos através de URIs** onde cada recurso deve ter uma URI associada a ele, e cada URI deve estar associada a um único recurso. Informações sobre o escopo de execução de um método (sobre quais informações o método deve operar) são mantidas na própria URI. Parâmetros na *query_string* podem ser utilizados para definir filtros sobre a operação a ser realizada. Soluções híbridas costumam expor métodos em vez de recursos.

- **Seus serviços não mantenham estado**. Cada requisição HTTP deve conter todas as informações necessárias para a sua execução, pois ela deve ser completamente independente das demais. A autenticação é uma situação onde algumas aplicações mantêm estado, contrariando este princípio.

- **Utilize uma interface uniforme**. Como alguns firewalls não permitem o tráfego de operações que utilizam os métodos PUT e DELETE, o uso de GET para operações seguras (consultar) e POST para operações inseguras (criar, atualizar e remover), é uma abordagem comumente utilizada para simular uma interface uniforme. Nestes casos é necessário enviar uma informação adicional às requisições do tipo POST para informar qual a real operação a ser realizada, como o cabeçalho X-HTTP-Method-Override utilizado pelas APIs da Google[45] ou o parâmetro adicional (por exemplo, method=PUT) utilizado pelas aplicações desenvolvidas no ambiente Ruby on Rails [46].

- **Utilize formatos de representação de recursos que não incluam informações do escopo de suas execuções em seus conteúdos,** como é o caso dos formatos XML-RPC e SOAP.

---

[45] http://code.google.com/apis/gdata/index.html
[46] http://www.rubyonrails.com

- **Exista conectividade entre as representações dos recursos** de forma a permitir a navegabilidade entre eles. Como resultado, deve ser possível navegar a partir de um recurso REST para os demais, apenas seguindo as ligações existentes nas representações de recursos obtidas.

4.3.1.2. Análise de algumas APIs

O Quadro 4.2 mostra cada um dos critérios apresentados na seção anterior, e algumas soluções freqüentemente adotadas na definição de APIs. Nesta tabela, a numeração (1), (2) e (3) refere-se respectivamente à classificação REST, Híbrida e RPC, conforme foi descrito na Seção 4.3.1.

Quadro 4.2 – Características das APIs REST

Critérios	**Possibilidades**
1. Recursos expostos através de URIs	(1) As informações para identificação do recurso são incluídas na URI (no *path* ou na *query_string*). Utilização de uma URI para cada recurso. (2) As informações para identificação do recurso são incluídas na URI (no *path* ou na *query_string*). Utilização de uma URI para cada método. (3) As informações para identificação do recurso são incluídas no corpo da mensagem. Utilização de uma única URI para todas as requisições.
2. Serviços REST não mantém estado	(1) Não mantém estado entre requisições (2) ou (3) Mantém estado entre requisições
3. Uso de uma interface uniforme	(1) Uso dos métodos HTTP (GET, POST, PUT, DELETE). (2) Uso dos métodos HTTP (GET para operações seguras e POST para inseguras). (3) Uso de um vocabulário próprio (search, getTags, movePhoto, etc)
4. Representações de recursos (formatos de saída)	(1) XML (1) RSS / Atom (1) JSON (1) PHP (1) HTML + Microformatos (3) XML-RPC (3) SOAP
5. Conectividade entre recursos	(1) Existe conectividade entre os recursos (2) e (3) Não existe conectividade entre os recursos

A fim de buscar uma validação dos critérios de API REST apresentados anteriormente (Quadro 4.2), decidiu-se aplicar os mesmos na classificação de um conjunto de APIs disponíveis na Web selecionadas de acordo com sua popularidade. Tal classificação foi feita de acordo com a seguinte convenção: aquelas com todas as respostas marcadas com (1) são totalmente REST; aquelas com todas as respostas marcadas com (3) são totalmente RPC; qualquer outra combinação indica que a API é híbrida. O resultado desta análise é apresentado no Quadro 4.3.

4.3.2. Quanto à Autenticação de Usuários

Uma característica que diferencia as APIs é a maneira com que seus usuários são autenticados. Embora o processo de autenticação não esteja diretamente relacionado com o grau de aderência de uma API à arquitetura REST, vale à pena estudá-lo, pois qualquer solução proposta para integração automática de aplicações deve levá-lo em consideração.

4.3.2.1. Aspectos Relevantes

Os aspectos mais relevantes a serem considerados para a classificação dos processos de autenticação utilizados são: o uso de criptografia e de chaves de autenticação; a forma de envio das informações de autenticação; o modelo utilizado.

Criptografia - A autenticação exige o envio de dados sigilosos como contas e senhas. Para garantir a segurança das informações que trafegam na rede, as APIs normalmente provêm alguns mecanismos que envolvem criptografia. Existe uma forma padrão de fazer isso que é a utilização do protocolo SSL (RESCORLA, 2001). Em alguns casos, quem implementa os serviços desenvolve seu próprio mecanismo de criptografia.

Chaves de Autenticação - Algumas APIs exigem que as aplicações clientes sejam cadastradas previamente para poderem fazer uso de suas funcionalidades. Neste caso, o desenvolvedor precisa obter uma chave (API Key) através da qual a aplicação cliente se identifica para a API. Este procedimento não exclui a necessidade do usuário também se identificar através de um dos processos de autenticação descritos anteriormente.

Forma de Envio das Informações - Dentro do escopo deste trabalho, identificamos duas formas principais para o envio de informações de autenticação: utilizando um cabeçalho HTTP ou parâmetros de um formulário.

Quadro 4.3 – Análise da conformidade com arquitetura REST de algumas APIs

Aplicação			Protocolo												
			arquitetura			atributos									
						1			2	3			4		5
			RESTful	Híbrida	Estilo RPC	uma URI por recurso	uma URI por método	uma única URI	não mantém estado	get, post, put e delete	get e post	vocabulário próprio	Formatos REST	Formatos RPC	Conectividade
Favoritos compartilhados	del.icio.us[47]			√			√		√			√	√		
	ma.gnolia[48]			√			√		√			√	√		
	simpy[49]			√			√		√			√	√		
Gerenciamento de tarefas	voo2do[50]			√			√					√	√		
	Google Calendar[51]		√			√			√	√			√		√
Repositório de arquivos	Amazon S3[52]	Rest	√			√			√	√			√		
		Soap			√			√	√			√		√	
	box.net[53]	Rest		√			√					√	√		
		xml-post			√			√				√		√	
		Soap			√			√				√		√	
Repositório de imagens	Flickr[54]	Rest		√			√		√			√	√		
		Xml-rpc			√			√	√			√		√	
		soap			√			√	√			√		√	
	Google Picasa[55]		√			√			√	√			√		√
Wiki	PBwiki[56]			√			√					√	√		
Blog	Google Blogger[57]		√			√			√	√			√		√
	Pownce[58]		√			√			√	√			√		√
Repositório de vídeos	Google YouTube[59]		√			√			√	√			√		√
	AOL Truveo[60]			√			√		√			√	√		
	AOL Video Upload[61]			√		√			√	√			√		
Conteúdo Cultural	BBC[62]			√			√		√			√	√		

[47] http://del.icio.us/help/api/
[48] http://wiki.ma.gnolia.com/Ma.gnolia_API
[49] http://www.simpy.com/doc/api/rest
[50] http://voo2do.com/help/api
[51] http://code.google.com/apis/calendar/
[52] http://www.amazon.com/gp/browse.html?node=16427261
[53] http://dev.box.net/
[54] http://www.flickr.com/services/api/
[55] http://code.google.com/apis/picasa/
[56] http://api.pbwiki.com/
[57] http://code.google.com/apis/blogger/
[58] http://pownce.pbwiki.com/API+Documentation
[59] http://www.youtube.com/dev
[60] http://developer.truveo.com/index.php
[61] http://dev.aol.com/video/upload
[62] http://www0.rdthdo.bbc.co.uk/services/

Modelos de Autenticação - Modelos de autenticação são definidos com o intuito de resolver os problemas de autenticação e autorização descritos acima. Identificamos quatro modelos freqüentemente utilizados nas APIs analisadas.

- **Implementados pelo protocolo HTTP**

O protocolo HTTP provê mecanismos padronizados de autenticação que podem ser utilizados com esta finalidade como, o HTTP Basic Authentication e o HTTP Digest (FRANKS et al, 1999). A comunicação das informações de autenticação entre cliente e servidor, é realizada através de cabeçalhos HTTP Authorization e WWW-Authenticate.

O HTTP Basic Authentication funciona da seguinte forma:

O cliente envia uma requisição ao servidor:

GET /recurso.html HTTP/1.1

Host: www.exemplo.com

O servidor responde que este recurso está protegido e que para acessá-lo é necessário fornecer conta e senha:

401 Unauthorized

WWW-Authenticate: Basic realm="My private data"

O cliente deve então solicitar conta e senha ao usuário e repetir a requisição enviando as credenciais (conta e senha codificadas em Base64) do usuário.

GET /recurso.html HTTP/1.1

Host: www.exemplo.com

Authorization: Basic bWF1cmljaW86bWF1cmljaW8=

Como o protocolo HTTP não mantém estado, e o servidor não tem como associar requisições provenientes de um mesmo usuário, os procedimentos de autenticação HTTP exigem que as credenciais do usuário sejam re-enviadas a cada nova requisição.

Como a codificação em Base64 é um procedimento reversível, este processo não garante a segurança da conta e senha enviadas quando Basic Authentication for utilizada. Uma solução possível para este problema é a utilização de SSL/HTTPS, criptografando assim toda a comunicação entre o cliente e o servidor. Outra possibilidade é utilizar o Digest Autenthication, outro modelo de autenticação implementado pelo protocolo HTTP onde os dados enviados do cliente para o servidor já são criptografados, garantindo assim a segurança do processo.

Outro problema decorrente do uso deste tipo de autenticação é a questão da confiança que o usuário tem no cliente, uma vez que ele deve fornecer sua identificação e senha para que o cliente possa repassá-las ao servidor através da API.

- **Baseados em HTTP**

Existem ainda soluções equivalentes que utilizam os mesmos cabeçalhos HTTP (WWW-Authenticate e Authorization) para implementar algoritmos próprios de autenticação. Nestes casos, assim como na autenticação HTTP, o servidor não mantém estado, sendo necessário reenviar o pedido de autenticação a cada nova requisição. Um exemplo deste tipo de autenticação é a utilizada pelo serviço Simple Storage Service da Amazon .

- **Baseados em *authentication token***

Outra solução freqüentemente utilizada em serviços que mantêm estado entre requisições, é a geração de uma chave de identificação de sessão pelo servidor (API-Hash). Esta chave é obtida numa primeira requisição responsável por iniciar uma sessão, devendo então ser devolvida pelo cliente a cada nova requisição ao servidor. A devolução da chave de identificação pode dar-se através de cabeçalhos HTTP ou de um parâmetro de formulário numa requisição POST.

- **Soluções onde o usuário não precisa fornecer sua conta e senha para o cliente**

Nestas soluções, o cliente redireciona o usuário para o provedor do serviço para que este solicite diretamente a senha ao usuário e faça a autenticação. É devolvida ao cliente uma chave que permite a realização de novas requisições. Existem vários exemplos de uso deste tipo de abordagem como: o OAuth[63], o Google AuthSub[64], o FlickrAuth[65], o AOL OpenAuth[66] e o Yahoo BBAuth[67].

---

[63] http://oauth.net
[64] http://code.google.com/apis/gdata/auth.html

OAuth é uma tentativa de padronização dos processos de autenticação e autorização para APIs na Web. É um protocolo aberto para autenticação, que permite que sítios, ou aplicações ditas Consumidoras (Consumers), possam acessar recursos protegidos de um serviço Web ditos Provedores do Serviço (Service Providers) através de sua API, sem que seja necessário que os usuários forneçam suas senhas às aplicações Consumidoras.

4.3.2.2. Análise de algumas APIs

O Quadro 4.4 apresenta as soluções utilizadas com maior freqüência para cada um dos aspectos mencionados na Seção 4.3.2.1.

Quadro 4.4 – Características das APIs com relação a seu modelo de autenticação

Aspectos	**Possibilidades**
Modelo de autenticação	HTTP-Auth Baseadas em HTTP Chave de identificação de sessão (API-Hash) OAuth, Google AuthSub ou equivalente
Forma de envio das informações de autenticação	Utiliza o cabeçalho HTTP Authorization Utiliza parâmetros de um Form
Criptografia	Provê criptografia Utiliza SSL Não oferece mecanismo de segurança
API-key	Utiliza API Key Não utiliza API Key

Comparamos estas soluções com aquelas implementadas pelas APIs analisadas na Seção 4.3.1.2. No caso de aplicações que apresentam mais de uma versão, como estamos interessados em comparar apenas o procedimento de autenticação das APIs ditas REST, excluímos desta análise as versões SOAP e XML-RPC. O Quadro 4.5 mostra o resultado desta comparação.

[65] http://www.flickr.com/services/api/auth.spec.html
[66] http://dev.aol.com/openauth
[67] http://developer.yahoo.com/auth/

Quadro 4.5 – Análise do processo de autenticação de algumas APIs

Aplicação		Segurança													
		Modelo de autenticação								Autorização		Criptografia			
				Tipo OAuth											
		HTTP-Auth	Baseada em HTTP	OAuth	AuthSub	flickr	box.net auth	aol	API-Hash	HTTP-Header	Form-encoded	provê criptografia	utiliza SSL	não provê segurança	API-key
Favoritos compartilhados	**del.icio.us**	√								√			√		
	ma.gnolia			√					√	√		√			
	simpy	√								√				√	
Gerenciamento de tarefas	**voo2do**								√		√			√	
	Google Calendar				√					√		√			√
Repositório de arquivos	**Amazon S3**		√							√		√			
	box.net						√				√	√			√
Repositório de imagens	**Flickr**					√					√	√			√
	Google Picasa				√					√		√			√
Wiki	**PBwiki**														√
Blog	**Google Blogger**				√					√		√			√
	Pownce	√		√						√		√			√
Repositório de Vídeos	**YouTube**				√					√		√			√
	AOL Truveo							√			√	√			√
	AOL Video Upload							√			√	√			√
Conteúdo Cultural	**BBC**	Read Only - Não tem autenticação													

Verificamos que a diversidade de meios de autenticação é muito grande. Mesmo quando os serviços utilizam modelos parecidos, detalhes de implementação são diferentes.

4.4. Adoção de um Formato para Descrição das APIs

A criação de interfaces do tipo REST é uma boa maneira de facilitar o trabalho de integração de ferramentas na Web, já que a partir de um endereço de entrada inicial seria possível atingir a todos os recursos oferecidos pela aplicação, graças à navegabilidade entre os recursos proposta por este tipo de arquitetura[68]. Alguns autores (GREGORIO, 2007; PAGALTZIS, 2007) argumentam que, por esta razão, aplicações REST não precisam ser descritas.

Porém, como mostra o Quadro 4.3, apesar do grande sucesso atual do REST, nem todas as aplicações utilizam de fato suas idéias, não aderindo a todos os seus preceitos. Em vista da atual realidade, qualquer solução para integração de aplicações terá dificuldades para tirar proveito

[68] Veja a Seção 3.3 onde, segundo Fielding (2000), hipermídia é um mecanismo de estado da aplicação.

desta facilidade. Por exemplo, basta que os recursos não sejam convenientemente interligados para que seja inviável a navegação entre as representações dos recursos, tornando necessária a descrição formal da API.

Normalmente, as aplicações Web 2.0 que oferecem APIs do tipo REST para manipulação de seus dados através de HTTP, são documentadas utilizando uma descrição textual das suas chamadas, uma descrição dos seus parâmetros e uma listagem das possíveis respostas e erros produzidos. No máximo, é possível encontrar uma descrição das saídas produzidas através de algum formalismo tipo XML-Schema. Estas documentações são suficientes para que desenvolvedores possam compreendê-las utilizando-as como fonte de consulta para a implementação de novas aplicações.

Entretanto, se desejamos automatizar o processo de integração de aplicações, é necessário desenvolver programas que sejam capazes de reconhecer outras aplicações, o que é dificultado pelo fato de descrições informais não serem facilmente compreendidas por máquinas. Para isso, seria necessária a existência de algum mecanismo de introspecção[69], baseado numa descrição mais formal das APIs, que pudesse ser compreendida pela máquina, permitindo assim automatizar este processo.

Em 2005 o W3C criou uma lista de discussão (W3C, 2005) para estimular um debate sobre a necessidade da descrição de serviços REST, que pudesse eventualmente conduzir a uma proposta de padrão de um novo formato para descrição de aplicações Web. Surgiram então diversas propostas de formalismos capazes de descrever aplicações Web como: WRDL[70] (Web Resource Description Language), NSDL[71] (Norm's Service Description Language), SMEX[72] (Simple Message Exchange Descriptor), Resedel[73] (REstful SErvices DEscription Language), RSWS[74] (Really Simple Web Service Descriptions), WDL[75] (Web Description Language) e WADL[76] (Web Application Description Language) .

[69] Introspecção é a capacidade de o servidor prover metadados (informações acerca de seus próprios serviços) aos clientes, permitindo que estes descubram quais são os métodos e recursos disponíveis, e como proceder para executá-los.
[70] http://www.prescod.net/rest/wrdl/wrdl.html
[71] http://norman.walsh.name/2005/03/12/nsdl
[72] http://www.tbray.org/ongoing/When/200x/2005/05/03/SMEX-D
[73] http://recycledknowledge.blogspot.com/2005/05/resedel.html
[74] http://webservices.xml.com/pub/a/ws/2003/10/14/salz.html
[75] http://www.pacificspirit.com/Authoring/WDL
[76] https://wadl.dev.java.net

Outra solução freqüentemente considerada para este problema é a utilização de WSDL, a mesma linguagem utilizada para descrição de Serviços Web. Neste sentido, a versão WSDL 2.0 incorpora algumas funcionalidades adicionais em relação à sua versão anterior, o WSDL 1.1, que viabilizam tecnicamente a sua utilização para descrever serviços REST (HAAS, 2005; TAKASE et al, 2008). Por exemplo, no WSDL 2.0 é possível descrever serviços acionados através dos métodos HTTP PUT e DELETE, quando na versão 1.1 só era possível utilizar os métodos HTTP GET e POST. Entretanto WSDL 2.0 continua sendo uma linguagem complexa demais para a descrição de serviços REST, sendo raramente utilizada com este propósito.

4.4.1. WADL (Web Application Description Language)

Dentre as alternativas mencionadas anteriormente, aquela que tem sido considerada como a mais promissora é o WADL, uma linguagem de descrição de aplicações baseadas em HTTP, proposta por Hadley (2006) no estilo WSDL, porém especificamente voltada para a descrição de recursos.

Um arquivo WADL representa a hierarquia de recursos oferecidos pela aplicação, onde cada recurso (tag resource) possui métodos (tag method) que por sua vez possuem requisições (tag request) e respostas (tag response). A Figura 4.5 apresenta um exemplo de uma descrição utilizando WADL para o serviço Yahoo News Search. Aqui, as linhas 2-8 iniciam a descrição de uma aplicação, listando os XML namespaces utilizados no restante da descrição. As linhas 9-14 definem as gramáticas utilizadas pelo serviço na forma de arquivos XML Schema. As linhas 16-45 descrevem o recurso Yahoo News Search e seu método correspondente: entre as linhas 19-36 são descritos os parâmetros esperados, enquanto que nas linhas 37-40 estão descritas as possíveis saídas.

```
1 <?xml version="1.0"?>
2 <application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 xsi:schemaLocation="http://research.sun.com/wadl/2006/10 wadl.xsd"
4 xmlns:tns="urn:yahoo:yn"
5 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6 xmlns:yn="urn:yahoo:yn"
7 xmlns:ya="urn:yahoo:api"
8 xmlns="http://research.sun.com/wadl/2006/10">
9 <grammars>
10 <include
11 href="NewsSearchResponse.xsd"/>
12 <include
13 href="Error.xsd"/>
14 </grammars>
15
16 <resources base="http://api.search.yahoo.com/NewsSearchService/V1/">
17 <resource path="newsSearch">
18 <method name="GET" id="search">
19 <request>
20 <param name="appid" type="xsd:string"
21 style="query" required="true"/>
22 <param name="query" type="xsd:string"
23 style="query" required="true"/>
24 <param name="type" style="query" default="all">
25 <option value="all"/>
26 <option value="any"/>
27 <option value="phrase"/>
28 </param>
29 <param name="results" style="query" type="xsd:int" default="10"/>
30 <param name="start" style="query" type="xsd:int" default="1"/>
31 <param name="sort" style="query" default="rank">
32 <option value="rank"/>
33 <option value="date"/>
34 </param>
35 <param name="language" style="query" type="xsd:string"/>
36 </request>
37 <response>
38 <representation mediaType="application/xml" element="yn:ResultSet"/>
39 <fault status="400" mediaType="application/xml" element="ya:Error"/>
40 </response>
41 </method>
42 </resource>
43 </resources>
44
45 </application>
```

Figura 4.5 – Exemplo de descrição com WADL para o serviço Yahoo News Search. Extraído de Hadley (2006)

4.4.1.1. Extensão ao WADL para representar o modelo de autenticação

A proposta do WADL apresenta alguns problemas, tendo em vista a sua utilização para a automatização da incorporação de aplicações. O principal deles é o fato do ambiente de integração precisar, a partir da descrição, saber como deve proceder com relação à autenticação. Como no WADL não há como descrever o modelo de autenticação utilizado pela API, se faz

necessário criar uma extensão que permita fazê-lo. O Apêndice III apresenta uma descrição em XML-Schema do formato WADL com esta extensão propostas por este trabalho.

A extensão proposta constitui-se da incorporação de uma informação sobre a forma de autenticação utilizada pela aplicação descrita, ficando a cargo do ambiente de integração conhecer e implementar alguns formatos de autenticação com os quais ele deseje ser compatível. Entre os formatos mais relevantes, discutidos na Seção 4.3.2.1, podemos mencionar: o HTTP Basic Authentication e o OAuth, assim como alguns formatos proprietários que, devido à grande quantidade de usuários atingidos e de ferramentas oferecidas, também devem ser considerados, como o Google Authsub, o FlickRAuth, o Yahoo BBAuth e o AOL OpenAuth.

A extensão ao WADL para incluir a autenticação consiste da utilização de um novo atributo na tag <param>. Quando um parâmetro for utilizado para enviar informações de autenticação à aplicação, deve ser incluído o atributo "authmode", podendo este receber os valores: "none", "basic", "oauth", "authsub", "flickr", "bbauth" ou outros valores que vierem a ser implementados.

A Figura 4.6 apresenta um trecho da descrição WADL do del.icio.us, onde é especificado o modelo de autenticação utilizado (HTTP Basic Authentication). Neste caso, a tag <param> utilizada no recurso v1 indica que todos os recursos definidos abaixo de v1 (como tags/get ou tags/rename) necessitam de autenticação segundo este modelo para serem acessados.

```
<resources base="https://api.del.icio.us">
  <doc xml:lang="en" title="The del.icio.us API v1">
   Post or retrieve your bookmarks from the social networking website.
   Limit requests to one per second.
  </doc>

  <resource path="v1">
    <param name="Authorization" style="header" required="true"
           authmode="basic">
    <doc xml:lang="en">All del.icio.us API calls must be authenticated
     using Basic HTTP auth.</doc>
   </param>

   <resource path="tags">
     <resource path="get"><method href="#getTags" /></resource>
     <resource path="rename"><method href="#renameTag" /></resource>
   </resource>

   . . .
  </resource>
 </resources>
```

Figura 4.6 – Exemplo de utilização do atributo authmode na API do del.icio.us

4.4.1.2. Definição do formato das respostas retornadas por uma requisição

Embora o formato WADL permita a utilização de XML Schema para a descrição formal das respostas possíveis retornadas por um recurso, esta solução é muito complexa para ser interpretada em tempo real. É possível utilizar uma abordagem simplificada desta descrição através das tags <param> incluídas dentro da tag <representation>, descrevendo assim cada um dos elementos produzidos como resposta e seus respectivos atributos.

A forma com que os recursos devem ser interligados é indicada através da tag <link>. Isso permite que, ainda que a API não ofereça interligação entre os recursos, estes podem ser definidos no arquivo WADL de forma que a aplicação possa simular esta característica do estilo de arquitetura REST. As Figuras 4.7 e 4.8 exemplificam a definição de um método, sua saída XML, e como esta deve ser descrita no formato WADL.

```
<method id="getPosts" name="GET">

  <request>
  . . .
  </request>
  <response>
   <representation mediaType="application/xml" element="posts">
    <param name="dt" style="plain" path="@dt" />
    <param name="tag" style="plain" path="@tag" />

    <param name="user" style="plain" path="@user" />

    <param name="description" style="plain" path="post/@description" />
    <param name="url" style="plain" path="post/@href" >
    <link href="#getPosts" rel="" rev=""> </link>
    </param>
    <param name="extended" style="plain" path="post/@extended" />
    <param name="hash" style="plain" path="post/@hash" />
    <param name="others" style="plain" path="post/@others" />

    <param name="tag" style="plain" path="post/@tag" />
    <param name="dt" style="plain" path="post/@time"  >
    <link href="#getPosts" rel="" rev="" />
    </param>
   </representation>
  </response>
 </method>
```

Figura 4.7 – Exemplo de descrição da saída de um método da API do del.icio.us

```
<?xml version="1.0" encoding="UTF-8"?>
<posts dt="2005-11-28" tag="webdev" user="user">
  <post href="http://www.howtocreate.co.uk/tutorials/texterise.php?dom=1"
      description="JavaScript DOM reference"
      extended="dom reference"
      hash="c0238dc0c44f07daedd9a1fd9bbdeebd"
      meta="92959a96fd69146c5fe7cbde6e5720f2"
      others="55" tag="dom javascript webdev" time="2005-11-28T05:26:09Z"
   />
</posts>
```

Figura 4.8 – Exemplo da saída XML produzida pelo método descrito na Figura 4.7

4.5. Considerações Finais

Este capítulo propõe a integração automática de aplicações como uma maneira de viabilizar a criação de um ambiente de aprendizagem onde seus usuários possam escolher livremente as ferramentas Web 2.0 mais adequadas aos seus objetivos. Para viabilizar a definição de uma arquitetura onde isso seja possível, realizou-se um estudo das APIs Web 2.0 existentes.

Com este objetivo, foi construída uma rede sistêmica organizando os principais aspectos relacionados à componentização das aplicações da Web 2.0, que permitisse facilitar o estudo e a análise das APIs oferecidas por estas aplicações, e apontar direções para o desenvolvimento de ferramentas que permitam a incorporação automática destas aplicações.

Um resultado obtido a partir deste processo de classificação foi a constatação de que uma grande parte destas APIs pode ser considerada híbrida, implementando, simultaneamente, características das arquiteturas REST e RPC. Assim, para viabilizar a incorporação automática de funcionalidades de um aplicativo, é necessário que as suas APIs sejam descritas formalmente. Uma alternativa neste sentido é a utilização de WADL, uma linguagem proposta recentemente para este fim. Para isso, entretanto, foi necessário propor uma extensão ao WADL que permitisse especificar o modelo de autenticação utilizado.

Capítulo 5
Uma Proposta para Integração Automática de Aplicações

Este capítulo define um modelo de integração, baseado nas conclusões do estudo das APIs Web 2.0 apresentado anteriormente, onde é possível incorporar aplicações de forma automática.

Este modelo é apresentado primeiramente de maneira genérica onde uma camada intermediária permite a integração de diferentes clientes com as aplicações Web 2.0, desde que suas APIs sejam descritas no formato WADL. Posteriormente discute-se como um ambiente de aprendizagem pode ser desenvolvido utilizando estas idéias, de forma que o usuário seja capaz de integrar aplicações escolhidas livremente de maneira simples e fácil.

5.1. Modelo de Integração Proposto

O modelo de integração proposto é baseado na arquitetura apresentada na Figura 5.1, onde uma camada intermediária é responsável por interpretar a descrição WADL de APIs externas e oferecer seus serviços para aplicações clientes através de requisições REST.

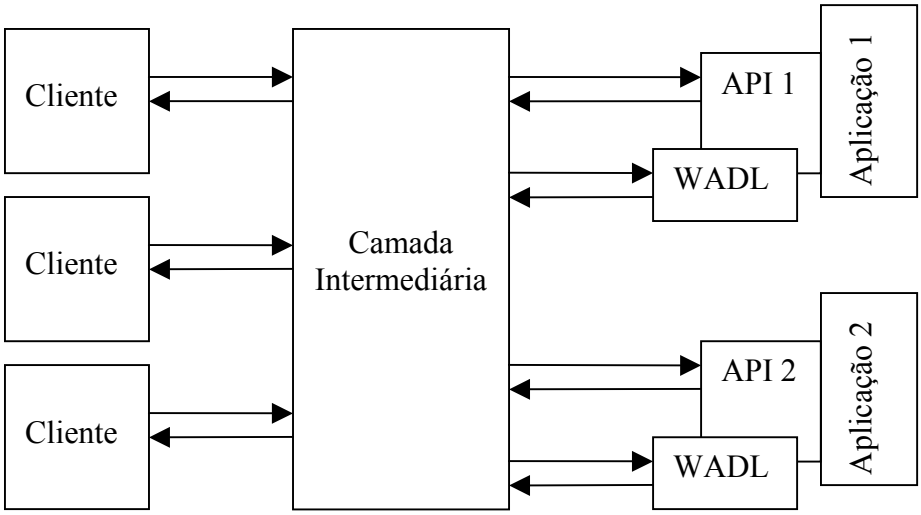

Figura 5.1 – Arquitetura de integração de serviços Web 2.0

A utilização deste modelo pode facilitar a integração de aplicações uma vez que a camada intermediária pode assumir a realização de algumas funções evitando, portanto, que estas sejam desenvolvidas no cliente. As principais atribuições desta camada intermediária são as seguintes:

- Oferecer os serviços básicos do ambiente como, por exemplo, a autenticação de usuários, e a incorporação de aplicações externas através da interpretação das suas descrições WADL.

- Fornecer às aplicações clientes um arquivo de índice dos métodos disponíveis na API assim como, para cada método, fornecer a relação de parâmetros esperados num formato padronizado, de forma a permitir a automatização do processo de interação com a API externa.

- Realizar as chamadas às APIs externas, repassando para estas aplicações, as requisições realizadas pelo cliente.

- Converter o resultado do processamento enviado pela API externa para um formato de apresentação esperado pelo cliente.

Este modelo permite ainda que diferentes clientes configurem a API de acordo com suas necessidades como será visto na Seção 5.1.2, a seguir. Além disso, é possível definir ligações entre os recursos no arquivo WADL mesmo que estas não existam originalmente nas aplicações simulando assim, esta característica do estilo de arquitetura REST.

A interação do cliente com a API se dá através de três processos distintos: a integração da aplicação externa, a configuração da mesma pelo cliente, e a execução dos métodos da API externa.

5.1.1. O processo de integração

O processo de integração de uma nova aplicação consiste da sua inclusão no sistema e da interpretação da sua descrição WADL. A partir deste momento, as chamadas da API podem ser requisitadas através da camada intermediária.

A realização prévia da integração da aplicação permite otimizar a execução da API pois a interpretação do WADL é feita uma única vez, gerando uma estrutura de dados onde são representados seus recursos, métodos, parâmetros e respostas. Além disso, é possível ainda que cada cliente configure a aplicação como será apresentado na Seção 5.1.2, a seguir.

Como a camada intermediária oferece seus serviços através de requisições REST, as operações realizadas no processo de integração devem, por sua vez, ser mapeadas em recursos da API. Sendo assim, a Figura 5.2 apresenta um exemplo de requisições para executar operações possíveis nesse contexto.

```
Incluir um novo cliente
POST /URLBase/cliente

Listar as aplicações cadastradas no sistema
GET /URLBase/aplicacao

Incluir aplicação no sistema
POST /URLBase/aplicacao

Carregar WADL de uma aplicação
POST /URLBase/aplicação/[idAplic]/wadl
```

Figura 5.2 – Chamadas da API REST para realizar a integração de uma API externa

5.1.2. Configuração de aplicações

A configuração de aplicações pode permitir que o cliente defina diversas características que determinem a maneira com que o usuário pretende interagir com a aplicação. Podem ser definidos, por exemplo: que métodos e parâmetros serão efetivamente utilizados, por quais nomes devem ser conhecidos pelo usuário da aplicação, que informações retornadas são relevantes e como estas respostas devem ser formatadas.

Um cliente deve poder configurar várias aplicações de acordo com suas preferências. Uma aplicação, por sua vez, deve poder ser configurada de diferentes maneiras por clientes diferentes. A Figura 5.3 apresenta um esboço de um modelo de classes para representar estes conceitos.

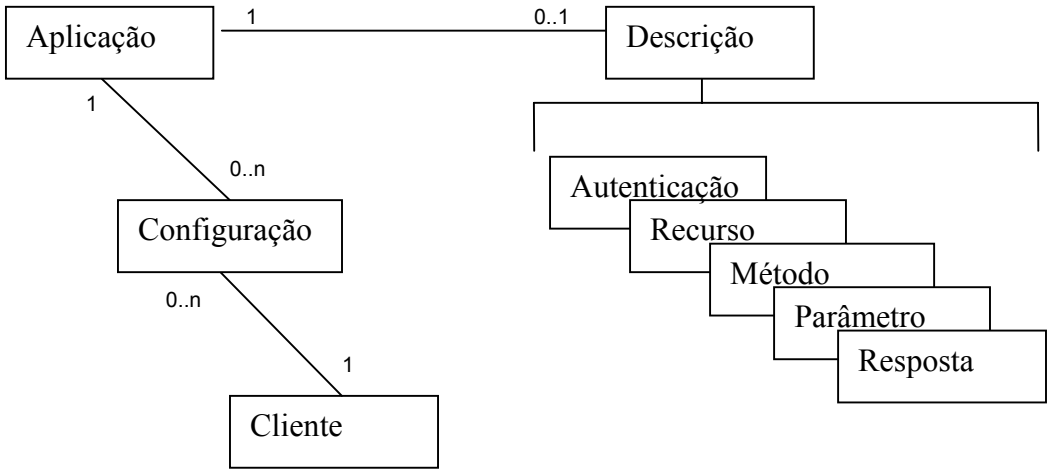

Figura 5.3 – Principais conceitos envolvidos para a integração de serviços

Um exemplo de recurso na API REST que permita a um cliente configurar uma aplicação externa é apresentado na Figura 5.4.

```
Configurar uma aplicação
POST /URLBase/cliente/[idCliente]/aplicacao/[idAplicacao]/configuracao
```

Figura 5.4 – Chamadas da API REST para realizar a configuração de uma API externa

5.1.3. Execução de métodos da API externa

Uma vez que a aplicação externa tenha sido integrada, o cliente pode requisitar seus recursos através da camada intermediária. Para isso, é realizado um mapeamento entre a descrição dos recursos da API obtida no WADL e as URLs que serão utilizadas para suas requisições. Estas requisições podem fornecer, opcionalmente, um formato de retorno (XML, JSON ou XHTML,

por exemplo) esperado pela aplicação cliente. Neste caso, a camada intermediária deve converter a resposta fornecida pela API para este formato.

Para facilitar a automatização destas requisições, a camada intermediária oferece ainda uma lista dos métodos disponíveis e dos parâmetros de cada método, através do formato padronizado XForms, que pode ser interpretado pelo cliente. Para isso, cada requisição original à API integrada é desmembrada em duas: uma primeira obtém os parâmetros e uma segunda executa efetivamente a ação. As três operações básicas oferecidas pela camada intermediária para automatizar a execução de métodos da API externa são apresentadas na Figura 5.5.

```
Listar métodos da aplicação
GET /URLBaseExecutor/[nomeAplic]

Obter os parâmetros de um método de uma aplicação
GET /URLBaseExecutor/cliente/[idCliente]/[nomeAplic]/.../.../form
                            (hierarquia de recursos descrita no wadl)

Executar um método de uma aplicação
GET, POST, PUT ou DELETE
/URLBaseExecutor/cliente/[idCliente]/[nomeAplic]/.../...
                            (hierarquia de recursos descrita no wadl)
```

Figura 5.5 – Chamadas da API REST para execução de métodos de uma API externa

5.2. Uma Proposta de Ambiente de Aprendizagem Baseado neste Modelo de Integração

Este modelo de integração pode ser utilizado na definição de um Ambiente de Aprendizagem onde seus usuários sejam capazes de incorporar aplicações escolhidas livremente. Para isso, partindo do modelo genérico apresentado na Seção 5.1, e especializando-o, seria necessário desenvolver um cliente que oferecesse uma interface adequada a este ambiente e incluísse novas funcionalidades na camada intermediária que pudessem atender a suas principais necessidades.

A apresentação desta proposta, realizada a seguir, compreende: a descrição das características e requisitos básicos do ambiente, uma visão geral da sua arquitetura, o modelo de classes estendido, a descrição de seus casos de uso e do processo de incorporação de uma aplicação. Um protótipo desta proposta, o ambiente AvaNCE, foi implementado dentro do escopo deste trabalho, sendo descrito posteriormente, no Capítulo 6.

5.2.1. Características e requisitos básicos do ambiente

As principais características e requisitos básicos deste ambiente são: a possibilidade de sua utilização para promover o aprendizado, através de um modelo centrado no curso (como nos AVAs) ou no aluno (como nos APAs); a possibilidade de integração automática de aplicações Web 2.0; e a facilidade de exportação de suas aplicações para outros ambientes.

5.2.1.1. Utilização de modelo centrado no curso ou no aluno

Este ambiente pode ser utilizado como um AVA (modelo centrado no curso) ou como um APA (modelo centrado no aluno). Nos AVAs, os alunos estão vinculados a turmas, oferecidas no contexto de cursos e disciplinas. Um aluno não tem acesso ao sistema se ele não estiver relacionado a uma turma. O professor ou tutor é uma espécie de mediador da turma, responsável, entre outras coisas, por propor e conduzir a realização de atividades pelos alunos. Estes conceitos de curso, disciplina e turma estão presentes no ambiente, permitindo que o mesmo seja definido pelo professor para sua utilização por uma turma.

Nos APAs, é o aluno quem controla seu próprio aprendizado. Assim, ele deve ser capaz de definir comunidades onde integre suas experiências, e que permitam a interação com outras pessoas interessadas nos mesmos assuntos. No contexto deste ambiente, uma comunidade é uma especialização de turma onde é o aluno quem configura o ambiente de acordo com seus objetivos e interesses.

5.2.1.2. Integração automática de aplicações

O ambiente oferece ao usuário a possibilidade de incorporar de maneira simples e sem a necessidade de programação, qualquer aplicação externa que possua uma API aberta, desde que esta seja descrita formalmente com WADL. Assim, o usuário (aluno ou professor) pode cadastrar uma API para que, a partir daí, o sistema seja capaz de interpretá-la, incorporando suas funcionalidades para o ambiente automaticamente.

5.2.1.3. Facilidade de exportação de suas aplicações para outros ambientes

Um dos pilares da Web 2.0 é a utilização de modelos leves de programação (ver Seção 2.1.4) através da construção de sistemas fracamente acoplados. Os APAs utilizam-se destas características promovendo não só a integração de aplicações, mas também a possibilidade de exportar suas aplicações para outros ambientes.

A produção de informações em formatos padronizados como RSS e XForms tem como objetivo facilitar essa exportação de aplicações.

5.2.2. Visão geral da arquitetura do ambiente

O ambiente de aprendizagem (Figura 5.6) é uma especialização da arquitetura genérica de integração proposta na Figura 5.1. Ele é composto de uma camada cliente que implementa a interface com o usuário e se comunica assincronamente através de requisições REST com a camada intermediária no servidor, que é constituída de um núcleo e de um conjunto de aplicações internas.

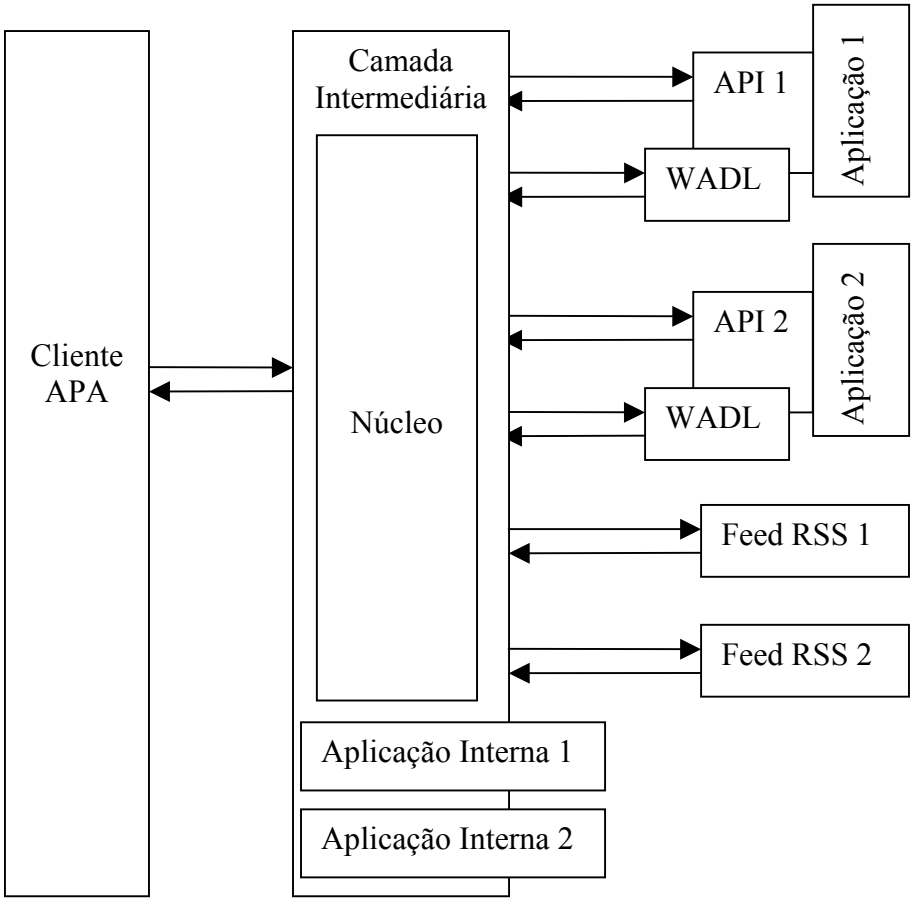

Figura 5.6 – Arquitetura do ambiente de aprendizagem proposto

O núcleo é responsável por oferecer os serviços básicos do ambiente como, por exemplo, a autenticação de usuários e a incorporação de aplicações externas através da interpretação das suas descrições WADL.

Além disso, é o núcleo também quem realiza as chamadas às APIs externas, repassando para estas aplicações, as requisições realizadas pelo cliente. Para isso, ele utiliza o conhecimento destas APIs, obtido a partir da descrição WADL. O resultado do processamento é devolvido para o próprio núcleo, que pode eventualmente convertê-lo para um formato de apresentação esperado pelo cliente.

O núcleo funciona ainda como um *Proxy*, realizando as requisições às URLs externas que, por medida de segurança, não podem ser realizadas diretamente pelo cliente. Desta forma, é possível que o cliente requisite também *feeds* RSS externos.

O ambiente pode ainda oferecer algumas aplicações nativas, como um fórum de discussão, uma ferramenta de chat, ou um servidor de blogs. Estas aplicações são desenvolvidas separadamente do núcleo, possuindo um espaço próprio de URLs REST através das quais seus recursos são expostos. Entretanto, apesar de independentes, estas aplicações podem compartilhar informações com o núcleo, seja acessando o mesmo banco de dados ou reutilizando suas classes.

5.2.3. Modelo de classes estendido

O ambiente de aprendizagem deve estender o modelo apresentado na Figura 4.10, sendo capaz de representar também os conceitos de Usuário, Ambiente, Curso, Disciplina, Turma, além de Configuração, Aplicação e Descrição já apresentados anteriormente.

O principal conceito adicional é o conceito de Usuário que representa qualquer pessoa com acesso ao ambiente. Usuários podem possuir privilégios adicionais, assumindo papéis como, Administrador ou Professor.

Cada usuário pode criar seus ambientes individuais ou comunidades e pode também participar das comunidades de outros usuários. Desta forma, as comunidades funcionam como espaços de trabalho de seus usuários onde eles podem incluir e configurar suas ferramentas preferidas. Além disso, uma comunidade pode também agregar pessoas com interesses comuns, favorecendo a discussão e a troca de conhecimentos.

Cada aplicação incluída numa comunidade é apresentada na forma de um menu que permite acesso às suas funcionalidades através da execução dos métodos de sua API, de acordo com a sua configuração.

O ambiente implementa também os conceitos de curso, disciplina e turma, de forma a permitir utilizá-lo também como um AVA, apoiando a aprendizagem segundo um modelo mais tradicional centrado no curso. Neste caso, uma turma é uma comunidade especial cujo acesso é restrito aos seus participantes e que é configurado pelo seu professor.

A Figura 5.7 apresenta um esboço de um modelo de classes para representar estes conceitos neste ambiente de aprendizagem.

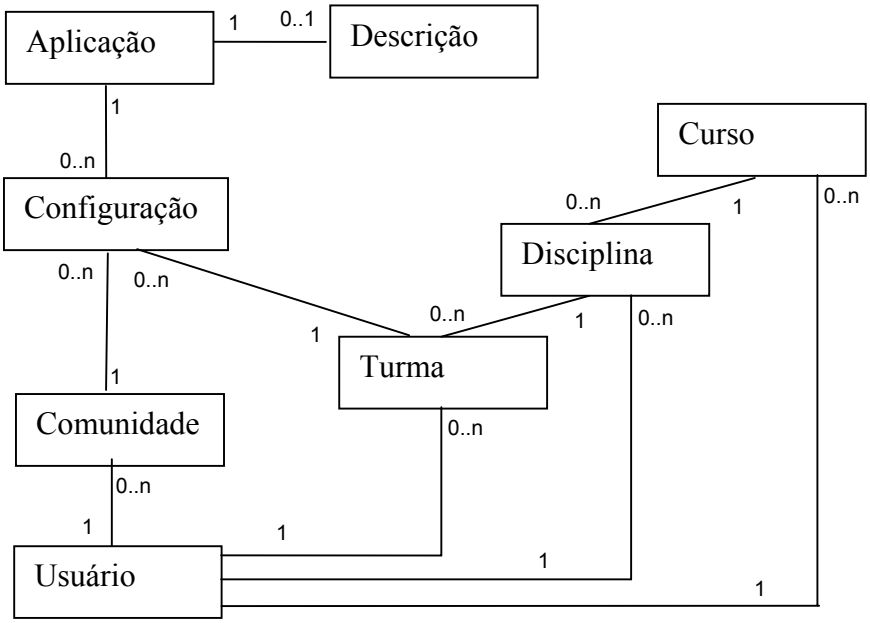

Figura 5.7 – Principais conceitos envolvidos no ambiente de aprendizagem

5.2.4. Casos de uso

Com o objetivo de facilitar a compreensão, os casos de uso aqui apresentados foram divididos em dois grupos. O primeiro diz respeito à utilização do ambiente como AVA, o segundo apresenta as interações possíveis quando este é utilizado como um APA. A relação completa dos casos de uso aqui mencionados pode ser encontrada no Apêndice A deste trabalho.

5.2.4.1. Utilização do ambiente como AVA

Neste caso, os alunos estão vinculados a turmas, oferecidas no contexto de cursos e disciplinas, sendo que eles não têm acesso ao sistema se não estiverem relacionados a uma turma. Uma turma é um espaço configurado pelo professor para utilização por seus alunos. A página de uma turma é um espaço onde o professor pode incluir aplicações, sejam elas internas ou externas, de acordo

com seus objetivos educacionais. Através da utilização destas aplicações, os alunos podem interagir entre si, receber e produzir conteúdo, etc.

Quando o ambiente é utilizado como AVA, os atores envolvidos, em ordem crescente de privilégios, são: Pessoa, Usuário, Professor e Administrador. Cada ator desta lista pode executar um conjunto de casos de uso permitidos a ele, além dos casos de uso permitidos aos demais atores de menor privilégio. Segue-se a definição de cada um destes atores, assim como a lista dos casos de uso executados por cada um deles:

Uma **pessoa** é qualquer um que acesse o ambiente sem se identificar. As únicas funcionalidades que uma pessoa pode executar são: autenticar-se (UC01 e UC02) ou cadastrar-se no ambiente (UC03).

Um **usuário** é uma pessoa cadastrada no ambiente e que passou por um processo de identificação através de fornecimento de conta e senha. Em geral, alunos são usuários do sistema. Usuários podem listar as turmas em que estejam envolvidos (UC06), entrar na página destas turmas (UC07) e executar aplicações associadas a estas turmas (UC08).

Um **professor** é um usuário cadastrado no ambiente, possuindo mais privilégios que um usuário comum, podendo também ser responsável por ministrar turmas. Sendo assim, professores podem: listar os alunos de uma turma (UC20), incluir alunos em turmas (UC19), listar as aplicações externas cadastradas no sistema (UC09), listar as aplicações de uma turma (UC15), incluir uma aplicação em uma turma (UC18) e configurar uma aplicação externa de acordo com sua utilização por uma turma (UC11).

Um **administrador** é um usuário especial cadastrado com o privilégio de administrador do ambiente. Pode realizar funções de administração em todas as turmas. A um administrador é permitido: listar e incluir cursos (UC12 e UC04), listar e incluir disciplinas (UC13 e UC16), listar e incluir turmas (UC14 e UC17), incluir aplicações no ambiente (UC05) e carregar a descrição WADL destas aplicações (UC10).

A Figura 5.8 apresenta o diagrama dos casos de uso oferecidos pelo ambiente quando este é utilizado como um AVA.

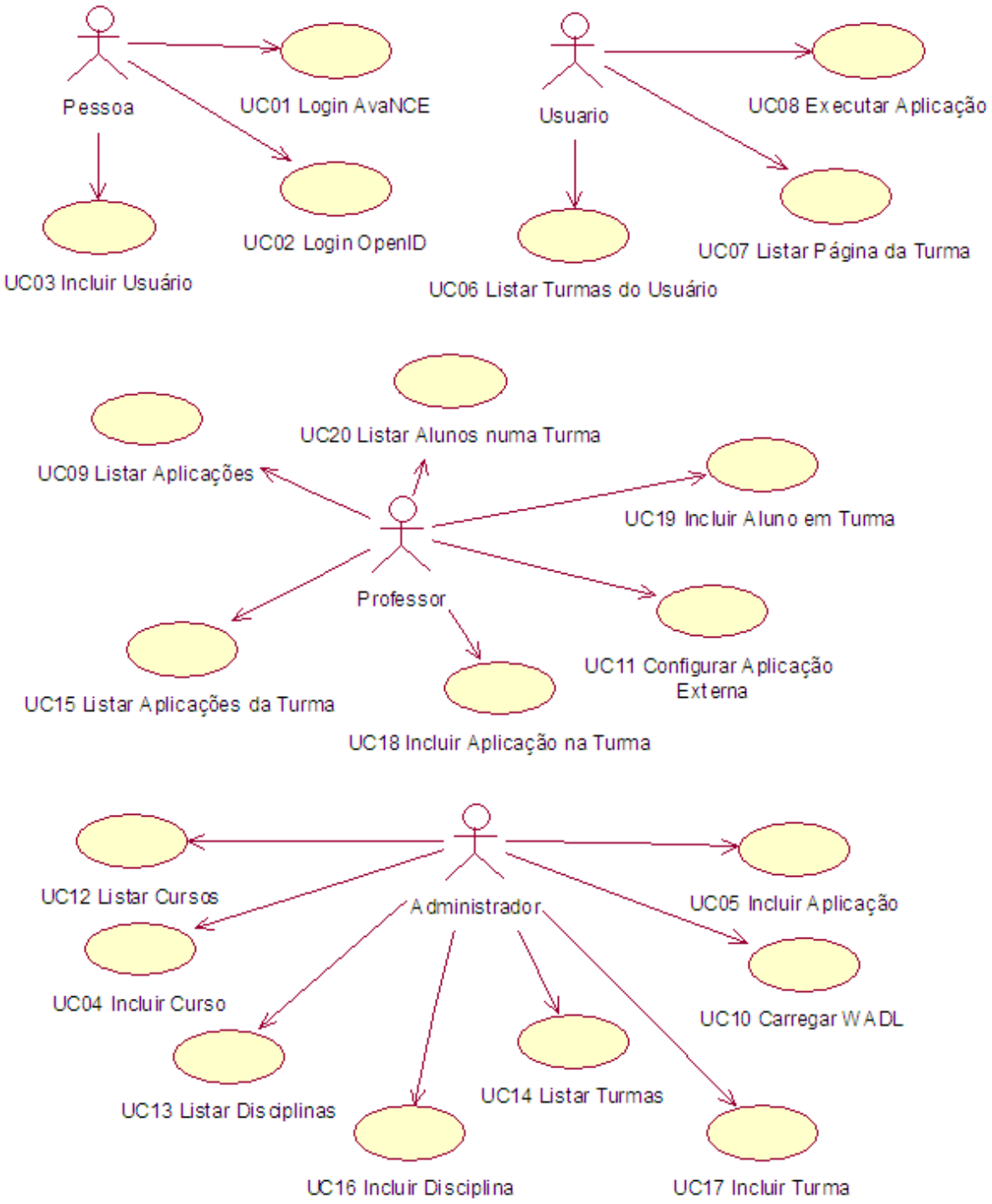

Figura 5.8 - Diagrama de casos de uso para utilização do ambiente como AVA

5.2.4.2. Utilização do ambiente como APA

Neste caso, o usuário deve ser capaz de configurar um ambiente onde integre suas experiências, e que permita a interação com outras pessoas interessadas nos mesmos assuntos.

Quando o ambiente é utilizado como um AVA, o espaço dos alunos é a página da turma. A utilização como APA, entretanto, baseia-se no conceito de comunidades, uma especialização de turma onde é o aluno quem configura o ambiente de acordo com seus objetivos e interesses.

Uma comunidade pode, portanto ser vista como um espaço do usuário onde ele desempenha um papel de aprendiz de um determinado assunto.

Quando o ambiente é utilizado como APA, os atores envolvidos, em ordem crescente de privilégios, são: Pessoa, Usuário e Administrador.

Pessoas podem executar apenas as funcionalidades de autenticação (UC01 e UC02) ou cadastramento no ambiente (UC03).

Usuários podem incluir comunidades (UC21), listar as aplicações cadastradas no ambiente (UC09), incluir aplicações nas suas comunidades (UC22), configurar uma aplicação externa para utilização na sua comunidade (UC11). Além disso, usuários podem participar de comunidades criadas por outros usuários. Sendo assim, é possível também: procurar por comunidades existentes (UC24), incluir-se em comunidades já existentes (UC23), listar aplicações de uma comunidade (UC25), listar comunidades das quais o usuário participa (UC26), listar a página de uma comunidade (UC27), executar uma aplicação de uma comunidade (UC08).

A um administrador é permitido: incluir aplicações no ambiente (UC05) e carregar a descrição WADL destas aplicações (UC10).

A Figura 5.9 apresenta o diagrama dos casos de uso oferecidos pelo ambiente quando este é utilizado como um APA.

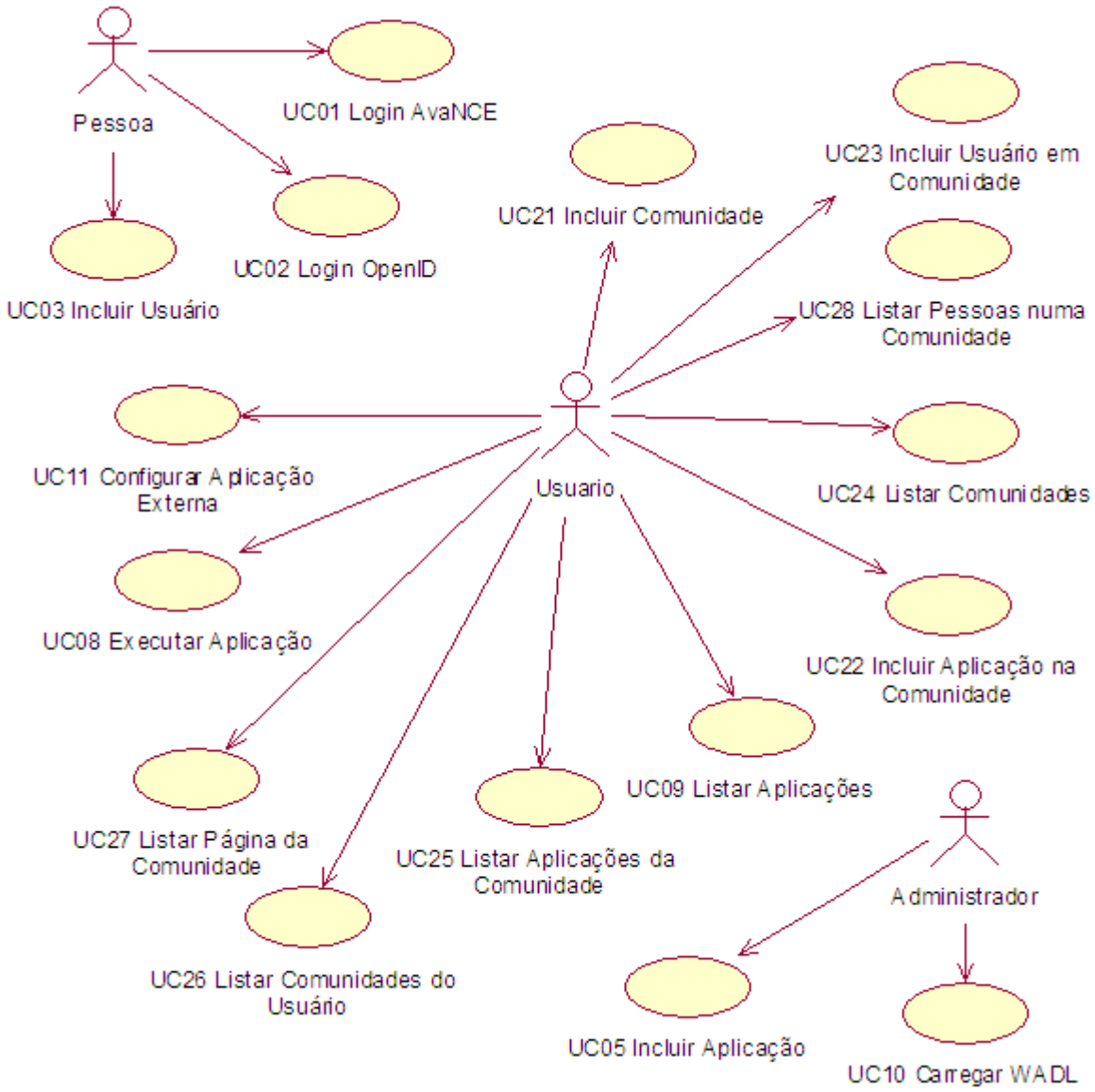

Figura 5.9 - Diagrama de casos de uso para utilização do ambiente como APA

5.2.5. Incorporação de uma aplicação

O processo de incorporação de aplicações consiste dos passos descritos a seguir (Figura 5.10). Em primeiro lugar, o administrador faz a inclusão de uma aplicação no sistema e a carga do respectivo arquivo WADL no banco de dados. Posteriormente, a aplicação fica disponível para ser selecionada pelos usuários do ambiente. Uma vez selecionada por um usuário, a aplicação poderá ser configurada por ele. Neste momento, o usuário poderá estabelecer algumas características que definam a visão que ele deseja ter da aplicação. A partir daí, ele passa a interagir com a aplicação executando os métodos oferecidos no menu do ambiente.

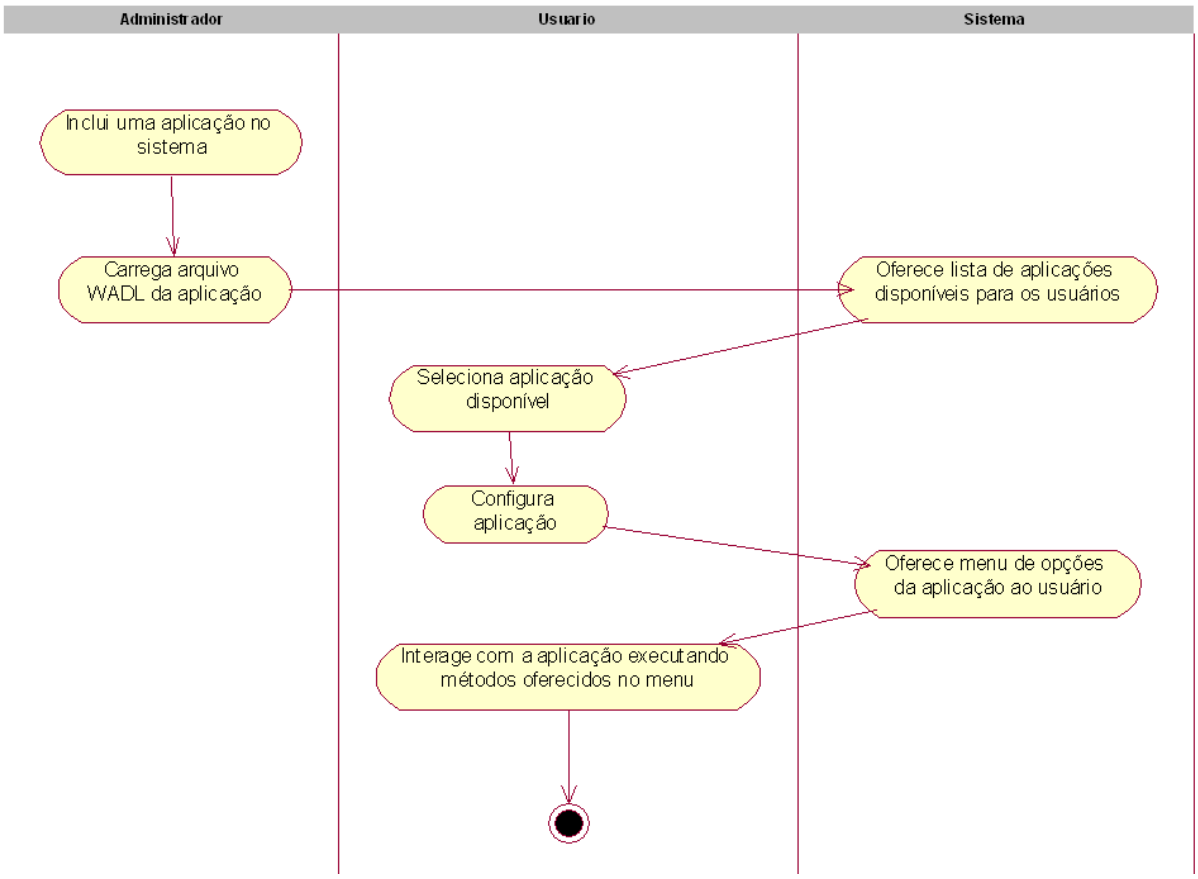

Figura 5.10 – Incorporação de uma aplicação externa

5.2.5.1. Inclusão de uma aplicação no sistema

Um pré-requisito para a integração de um serviço é que a API a ser integrada seja descrita em WADL. Assim, ao incluir uma nova aplicação no sistema, o usuário deve informar, além do nome da aplicação, a URL da descrição de sua API. Como WADL é um formato aberto, espera-se que as próprias aplicações que desejem ser integradas desta forma publiquem suas próprias descrições. Ainda que isto não aconteça inicialmente, é possível que qualquer pessoa escreva e publique as descrições de qualquer aplicação de forma a permitir sua integração.

No momento da inclusão de uma aplicação no sistema, é possível ainda definir parâmetros globais constantes que devam ser passados a todas as chamadas da API. Isto é particularmente útil para APIs que exigem que o desenvolvedor cadastre sua aplicação, enviando um código deste cadastro (freqüentemente chamado de API-KEY) junto a cada chamada realizada. Neste caso, o administrador deve fornecer esta API-KEY como um parâmetro global, de forma que o ambiente possa repassá-lo para a API integrada, a cada requisição que ele venha a realizar.

5.2.5.2. Carga da descrição WADL

Esta descrição passa por um processo de interpretação, cujo produto é uma representação de sua estrutura num banco de dados. Esta solução é mais dinâmica que algumas propostas de utilização de WADL para integração de ferramentas[77], que utilizam a descrição dos recursos para a produção de um código fonte. Desta forma, é possível recarregar o WADL sempre que necessário (por exemplo, se a API for modificada), atualizando assim a estrutura de um serviço.

5.2.5.3. Configuração da aplicação

Depois de incorporada no sistema, uma aplicação fica disponível para ser selecionada pelos usuários. Uma vez que um usuário selecione uma aplicação, este passa a ter acesso a um menu com as funções oferecidas pela aplicação. Com o intuito de adaptar este menu às reais necessidades de cada usuário, a aplicação pode opcionalmente ser configurada por seus usuários para uso individual, ou para uma turma ou grupo de usuários. Esta configuração se baseia, na seleção das funcionalidades que o usuário deseja efetivamente utilizar, além de, para cada método, selecionar os parâmetros de saída que devem ser apresentados e as legendas para os mesmos.

A configuração de uma aplicação está associada a sua utilização por uma turma ou por um usuário individual. Sendo assim é possível que uma mesma aplicação seja configurada de maneiras diferentes para sua utilização por diferentes usuários.

A Figura 5.11 apresenta um exemplo de tela de configuração onde o usuário pode especificar suas preferências com relação a alguns recursos da API do Del.icio.us. Aqui, estão selecionados os métodos “tags/get” e “posts/get”, indicando que o usuário deseja que estes métodos estejam disponíveis para ele utilizando apenas os parâmetros de entrada e saída escolhidos nesta mesma tela. Desta forma, quando o usuário executar o método “posts/get”, por exemplo, o sistema deve pedir o valor do parâmetro de entrada “tag” e produzir uma resposta, somente com as informações “description”, “href” e “tag”. Ao parâmetro de saída “description”, associou-se o nome “Descrição” com o qual o usuário deseja que a respectiva informação seja identificada.

---

[77] Uma destas propostas de utilização de WADL é o REST API Code Generation, um compilador de descrições de APIs REST capaz de gerar código para acesso a estas APIs. Disponível através do Google Code no endereço http://code.google.com/p/rest-api-code-gen/.

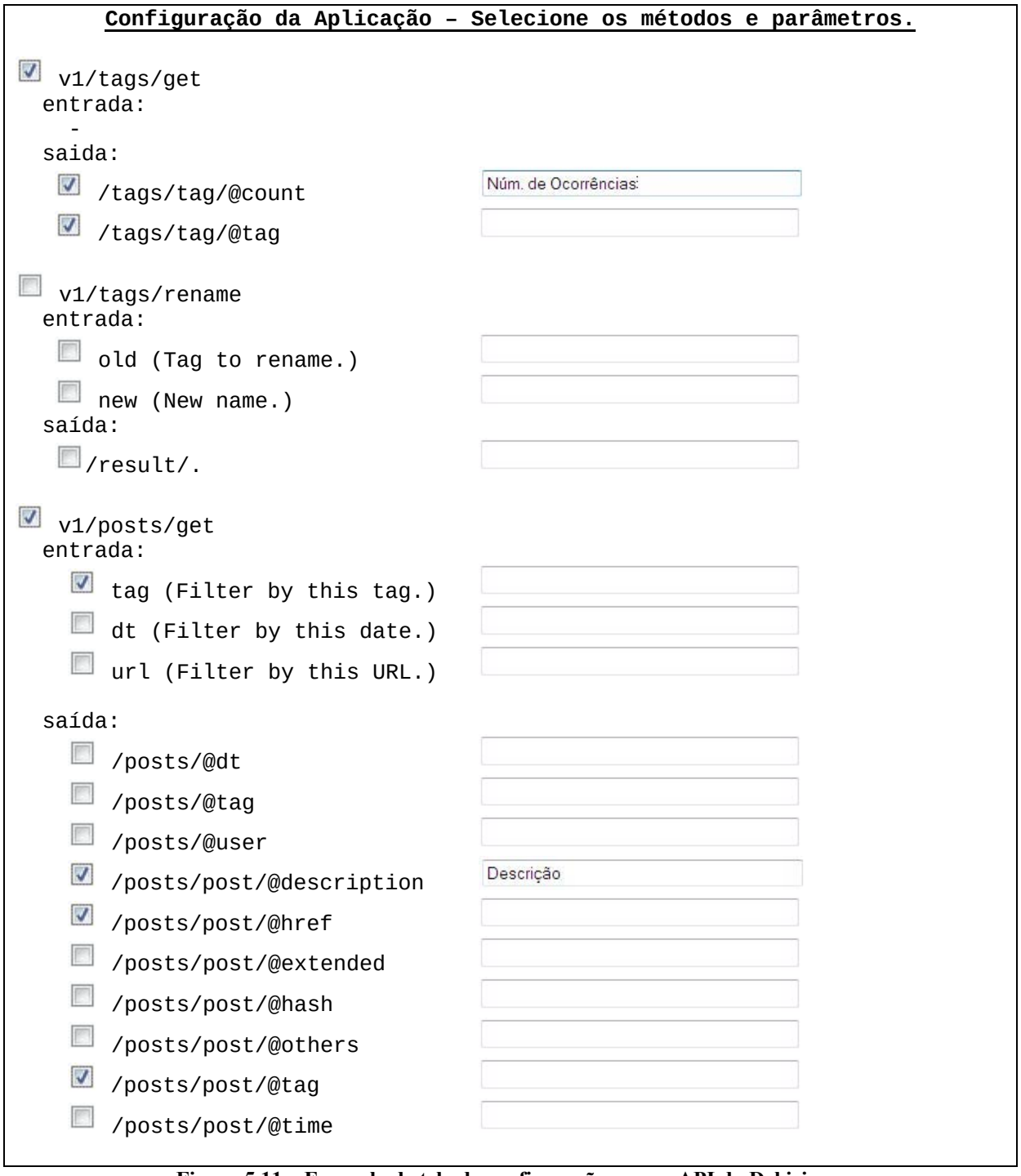

Figura 5.11 – Exemplo de tela de configuração para a API do Del.icio.us

5.2.6. Execução de métodos de uma aplicação externa

Quando o usuário entra no ambiente, ele pode acessar através de um menu, as funções oferecidas por cada aplicação externa incorporada. A execução de um método de uma aplicação externa se realiza como descrito a seguir, e ilustrado no diagrama de atividades apresentado na Figura 5.12.

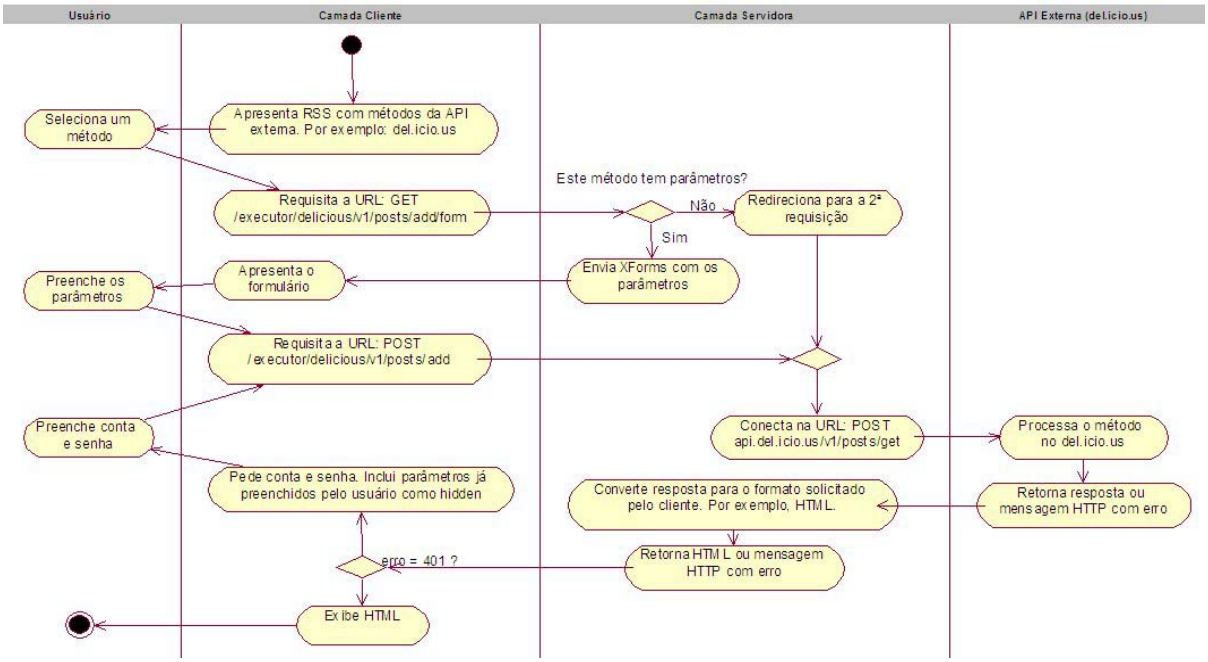

Figura 5.12 – Execução de um método de uma API externa

Cada chamada à aplicação externa é mapeada em duas requisições ao núcleo do ambiente. A primeira utiliza sempre o método HTTP GET e retorna um formulário com os parâmetros a serem passados para esta chamada. A segunda utiliza o mesmo método HTTP definido na chamada original (GET, POST, PUT ou DELETE) e envia os dados preenchidos pelo usuário no formulário, redirecionando-os para o serviço da API correspondente.

Portanto, ao executar a primeira requisição, o sistema verifica se há a necessidade de envio de algum parâmetro. Em caso afirmativo, é montado um formulário, codificado no padrão XForms, para que o usuário possa fornecer os valores de entrada. A adoção deste padrão permite que o ambiente seja utilizado pelos mais diversos clientes, dando a estes a liberdade de escolher a maneira como vão apresentar o formulário ao usuário.

Uma vez que o usuário tenha fornecido os parâmetros para execução do método, uma nova requisição ao sistema de integração é feita, possibilitando que este a repasse ao servidor da aplicação integrada. O retorno é então enviado pela aplicação ao núcleo, de acordo com o atributo mediaType descrito no arquivo WADL. Ao receber o retorno, o sistema converte-o para um formato apresentável (XHTML, por exemplo) enviando-o ao cliente. Esta conversão é realizada levando em consideração a descrição WADL dos parâmetros de retorno e a configuração da aplicação realizada pelo usuário. Caso este retorno seja uma mensagem de erro que indique a necessidade de autenticação para execução do serviço solicitado, cabe à aplicação

cliente solicitar conta e senha ao usuário e refazer a requisição original, acrescentando estes novos parâmetros.

Para exemplificar este processo, imaginemos a chamada para adicionar um favorito na API do del.icio.us: "**https://api.del.icio.us/v1/posts/add**". Supondo que a aplicação Del.icio.us já foi incorporada no sistema com o nome "delicious", a chamada deste método através do sistema de integração seria realizada em duas etapas, através das seguintes requisições:

1. "**GET /executor/delicious/v1/posts/add/form**", para obter o formulário com os parâmetros de entrada desta função, como a url, o título, as tags e a descrição do favorito a ser incluído.

2. Uma vez que o usuário tenha fornecido estes parâmetros através do formulário, uma nova requisição deve ser realizada – "**POST /executor/delicious/v1/posts/add**" – para enviar os dados do formulário e redirecionar a requisição para a API do Del.icio.us.

De forma a permitir a seleção das funcionalidades disponíveis, o ambiente oferece ainda um método que retorna um RSS de índice da aplicação com uma lista com todos os métodos da aplicação integrada. Isso faz com que o cliente seja capaz de navegar pela aplicação a partir daí, como numa aplicação REST, desde que a API original interligue as demais representações de recursos.

Além disso, o cliente não precisa conhecer a priori os parâmetros esperados por cada chamada, pois eles são fornecidos pelo próprio ambiente através da requisição do formulário para cada método existente.

A principal vantagem desta abordagem é permitir o acesso automático a uma API não REST.

5.2.6.1. Requisição de formatos específicos de saída

Os formatos de representação de recursos utilizados originalmente pelo ambiente são RSS para retornar uma lista de informações e XForms para definição de parâmetros através de formulários. Embora estes padrões sejam originalmente definidos baseados na sintaxe XML, pode ser conveniente oferecê-los em outros formatos como JSON ou até mesmo XHTML. Sendo assim, ao requisitar uma URL REST, é possível opcionalmente determinar um formato de saída específico no qual o cliente espera receber o retorno de sua requisição. Isto é feito acrescentando uma extensão (por exemplo: .xml, .xhtml ou .json) à URL que identifica o recurso requisitado.

No caso de uma requisição a uma API externa através do ambiente, caberá ao núcleo a conversão do retorno recebido da API no formato solicitado pelo cliente.

Este modelo favorece a criação de diferentes clientes que sejam capazes de consumir os serviços oferecidos pelo ambiente, sejam eles oferecidos diretamente pelo núcleo, por uma aplicação interna, ou por uma aplicação externa.

A Figura 5.13 apresenta um exemplo de RSS utilizando uma representação XML correspondente a uma listagem de turmas, obtido através de uma requisição REST ao ambiente. Este mesmo recurso pode ser solicitado em formato json, acrescentando a extensão .json à requisição original. A nova resposta obtida é apresentada na Figura 5.14.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0">
<channel>
  <title>Listagem de Turmas</title>
  <link>/IesAvance/rest/axG3hEhnuOhg/turma</link>
  <description>Turmas do Usuario Mauricio Bomfim</description>
  <language>pt</language>
  <item>
    <title>2007-2</title>
    <link>/IesAvance/rest/axG3hEhnuOhg/turma/2</link>
    <description></description>
    <author>aluno</author>
  </item>
  <item>
    <title>HTML-2007-2</title>
    <link>/IesAvance/rest/axG3hEhnuOhg/turma/4</link>
    <description></description>
    <author>professor</author>
  </item>
  <item>
    <title>xpto</title>
    <link>/IesAvance/rest/axG3hEhnuOhg/turma/11</link>
    <description></description>
    <author>professor</author>
  </item>
</channel>
</rss>
```

Figura 5.13 - Exemplo de uma representação XML

```
{
    rss:{
        version:2,
        channel:{
            title:'Listagem de Turmas',
            link:'/IesAvance/rest/axG3hEhnuOhg/turma',
            description:'Turmas do Usuario Mauricio Bomfim',
            language:'pt',
            item:[
                {
                    title:'2007-2',
                    link:'/IesAvance/rest/axG3hEhnuOhg/turma/2',
                    description:{
                    },
                    author:'aluno'
                },
                {
                    title:'HTML-2007-2',
                    link:'/IesAvance/rest/axG3hEhnuOhg/turma/4',
                    description:{
                    },
                    author:'professor'
                },
                {
                    title:'xpto',
                    link:'/IesAvance/rest/axG3hEhnuOhg/turma/11',
                    description:{
                    },
                    author:'professor'
                }
            ]
        }
    }
}
```

Figura 5.14 - Exemplo de uma representação JSON

Da mesma forma, as Figuras 5.15 e 5.16 ilustram um formulário definido através do padrão XForms utilizando os formatos XML e JSON.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="/IesAvance/xform.xsl"?>
<xforms>

<model>
<submission id="DeliciousWADL"
action="/IesAvance/Executor/DeliciousWADL/v1/posts/get"
method="urlencoded-post"/>
</model>

<input ref="tag">
<label>tag: </label>
</input>
<input ref="dt">
<label>dt: </label>
</input>
<input ref="url">
<label>url: </label>
</input>
<submit submission="DeliciousWADL">
<label>Enviar</label>
</submit>

</xforms>
```

Figura 5.15 - Exemplo de um formulário XForms codificado em XML

```
{
    xforms:{
        model:{
            submission:{
                id:'DeliciousWADL',
                action:'/IesAvance/Executor/DeliciousWADL/v1/posts/get',
                method:'urlencoded-post'
            }
        },
        input:[
            {
                ref:'tag',
                label:'tag: '
            },
            {
                ref:'dt',
                label:'dt: '
            },
            {
                ref:'url',
                label:'url: '
            }
        ],
        submit:{
            submission:'DeliciousWADL',
            label:'Enviar'
        }
    }
}
```

Figura 5.16 - Exemplo de um formulário XForms codificado em JSON

5.3. Considerações Finais

Este capítulo propõe um modelo de integração automática de aplicações baseado numa camada intermediária capaz de interpretar a descrição WADL de aplicações externas e oferecer mecanismos que permitam facilitar a sua integração a outras aplicações Web 2.0 como, por exemplo, permitindo que aplicações clientes possam saber quais são os métodos disponíveis na respectiva API e como estes devem ser chamados.

Este tipo de abordagem pode favorecer o desenvolvimento de um ambiente de aprendizagem onde os usuários incorporem aplicações com o intuito de integrar suas experiências e permitir a interação com outras pessoas com os mesmos interesses.

Esta proposta de integração de aplicações resultou na implementação do ambiente AvaNCE, apresentado no Capítulo 6, onde é possível incorporar automaticamente aplicações com APIs REST híbridas.

Capítulo 6
AvaNCE – Um Ambiente de Aprendizagem Baseado na Web 2.0

Este capítulo apresenta o desenvolvimento do AvaNCE (BOMFIM, ASSIS e SAMPAIO, 2007; BOMFIM, SAMPAIO e ASSIS, 2008), um protótipo do ambiente de aprendizagem baseado nas idéias da Web 2.0 discutidas nesta dissertação. Este ambiente foi construído com o propósito de verificar a viabilidade das propostas apresentadas no Capítulo 5, sendo capaz de integrar aplicações externas automaticamente a partir da descrição com WADL de suas APIs.

Além de poder incorporar novas aplicações, a facilidade de exportação de suas aplicações através do uso de formatos padronizados pode favorecer a incorporação de seus serviços em outros ambientes. Estas características são particularmente interessantes num cenário onde os aprendizes adquirem a liberdade de selecionar as ferramentas mais adequadas às suas necessidades, não ficando restritos a um conjunto predefinido de aplicações.

6.1. Ambiente de Desenvolvimento

De acordo com o modelo de arquitetura proposto no Capítulo 5, o AvaNCE foi desenvolvido em duas camadas: uma interface cliente e uma camada servidora responsável por implementar os serviços REST. Para a implementação da interface cliente foram utilizados a linguagem javascript e AJAX. Já a implementação da camada servidora foi realizada na linguagem Java / J2EE.

A opção realizada pelo uso da linguagem Java no servidor se deveu principalmente aos seguintes fatores:

- A portabilidade desta linguagem, isto é a possibilidade de executar o ambiente em diferentes plataformas;
- O fato desta linguagem constituir um ambiente livre e gratuito, facilitando sua operação em qualquer instituição, sem a necessidade de adquirir licenças de software;
- A existência de bibliotecas prontas e gratuitas para a realização de funções específicas como a geração de RSS e a autenticação através de openID;
- O conhecimento prévio dos pesquisadores nestas tecnologias.

Como ambiente de desenvolvimento foi utilizado o software Eclipse[78] com plugin ObjectWeb/Lomboz[79] versão 3.1.2, voltado para o desenvolvimento de aplicações cliente-servidor através de J2EE, em ambiente Windows com Java 6.

O ambiente onde é executada a camada servidora, é uma máquina com sistema operacional Linux Ubuntu 8.04 e servidor de aplicação Caucho/Resin[80] versão 2.1.17 com máquina virtual Java 6. O servidor de banco de dados é o MS-SQL Server 2000 em Windows Server 2003 Enterprise Edition.

A camada cliente é totalmente Web, não exigindo nenhuma instalação especial além de um navegador.

[78] http://www.eclipse.org
[79] http://lomboz.objectweb.org
[80] http://www.caucho.com

6.2. Arquitetura Interna

A Figura 6.1 ilustra a arquitetura do ambiente AvaNCE, baseada na arquitetura genérica proposta no Capítulo 5.

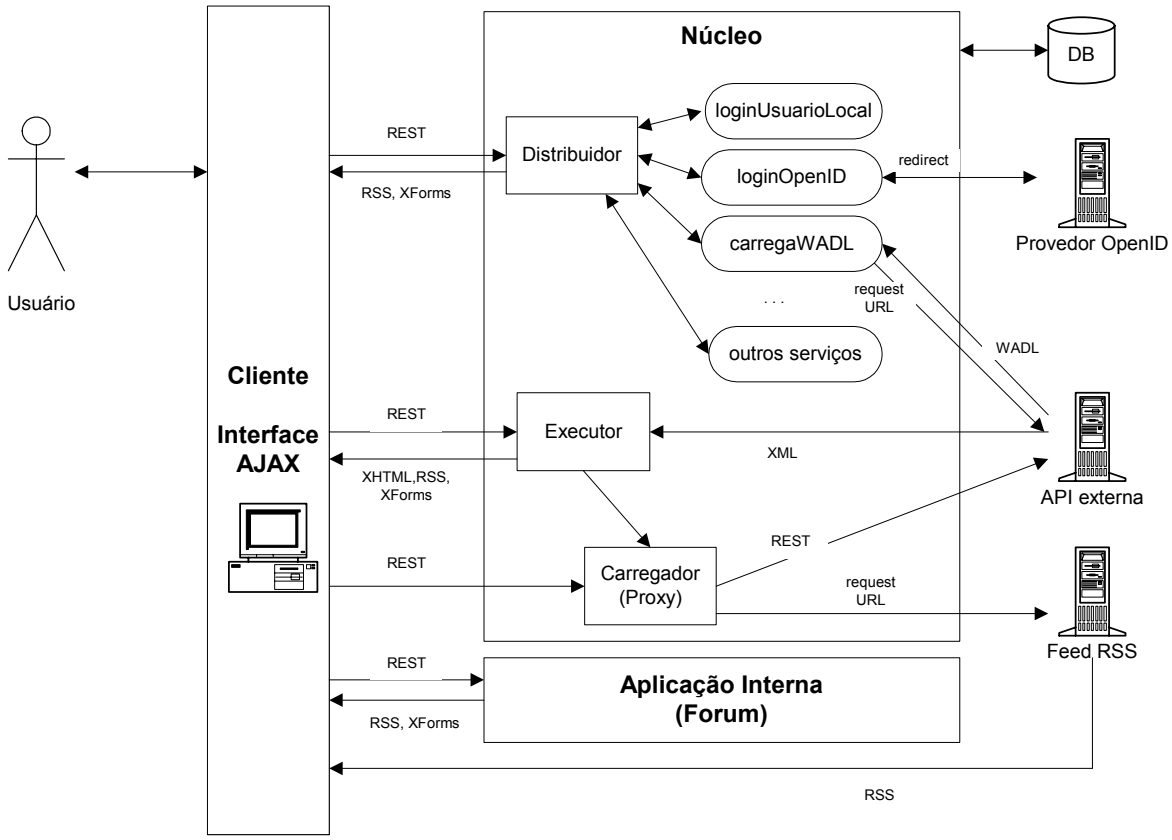

Figura 6.1 – Arquitetura do ambiente AvaNCE

6.2.1. Desenvolvimento do módulo cliente

O cliente AJAX é responsável por apresentar ao usuário a interface do sistema. É ele quem solicita autenticação ao usuário para entrada no sistema, e implementa o menu principal através do qual são acessadas as funcionalidades básicas. O cliente também é responsável por interpretar os formatos RSS e XForms recebidos como resposta das requisições ao servidor e por solicitar conta e senha para acesso a recursos de aplicações externas que exijam autenticação.

Todas as funcionalidades são executadas através de requisições REST assíncronas realizadas pelo cliente. Quem responde a estas requisições é o módulo servidor.

Para facilitar a criação da interface gráfica e a realização das requisições assíncronas, utilizou-se a biblioteca Yahoo User Interface (YUI) [81] versão 2.2.0. O YUI é uma biblioteca gratuita desenvolvida pelo Yahoo em JavaScript, composta de um conjunto de módulos e classes que implementam controles para facilitar a construção de aplicações Web interativas, utilizando técnicas como DHTML, AJAX e manuseio do conteúdo da página através do DOM.

No momento, os únicos métodos utilizados do YUI são o YAHOO.util.Connect.asyncRequest e o YAHOO.util.Connect.setForm. Estes métodos servem para realizar uma requisição assíncrona, independentemente do navegador utilizado. Embora na versão atual o AvaNCE não apresente uma interface gráfica muito trabalhada, a utilização do YUI pode vir a facilitar muito esta atividade no futuro.

6.2.2. Desenvolvimento do módulo servidor

O servidor consiste de um núcleo e de um conjunto de aplicações internas. Para ilustrar o mecanismo de desenvolvimento de aplicações internas, foi desenvolvido um fórum de discussões incorporado ao AvaNCE. O Fórum, apesar de ser uma aplicação independente do núcleo, têm acesso ao mesmo banco de dados e às suas classes na forma de um arquivo JAR externo.

Tanto o núcleo quanto as aplicações internas definem suas próprias APIs REST para que suas funcionalidades possam ser invocadas pelo cliente. O retorno destas invocações é realizado na forma de um arquivo XML padronizado (XHTML, XForms ou RSS) ou de uma mensagem HTTP de erro. Esta abordagem visa facilitar a incorporação de serviços do AvaNCE em outros ambientes.

Os principais módulos do núcleo AvaNCE são o Distribuidor, o Executor e o Carregador, implementados na forma de Servlets Java.

O Distribuidor é a porta de entrada para os serviços do Núcleo. É ele quem analisa a chamada REST e a decodifica, repassando a execução para o método correspondente. Diferentes serviços podem ser executados através do Distribuidor como, por exemplo, a autenticação de usuários e a interpretação das descrições WADL das aplicações externas.

[81] http://developer.yahoo.com/yui/

O Carregador é um módulo que funciona como um *Proxy*, realizando as requisições às URLs externas que, por medida de segurança, não podem ser realizadas diretamente pelo cliente. Desta forma, é possível que o cliente requisite *feeds* RSS externos.

O Executor é responsável por realizar as chamadas às APIs externas, repassando para estas, através do Carregador, as requisições realizadas pelo cliente. O Executor utiliza o conhecimento das APIs externas armazenado no banco de dados do sistema e que foi obtido através da interpretação da descrição WADL realizada previamente. O resultado do processamento é devolvido para o próprio Executor, que utiliza os parâmetros de configuração da API para determinar a forma de apresentação dos resultados esperada pelo cliente.

As seguintes bibliotecas foram utilizadas pelo módulo servidor:

- **sqljdbc.jar** (Microsoft SQL Server 2005 JDBC Driver 1.1)[82] – É um driver JDBC desenvolvido pela Microsoft capaz de prover acesso a bancos de dados SQL Server a partir de aplicações Java.

- **jrss** (Simple Java RSS Feed Generator)[83] e Xerces Java Parser[84] – O jrss é uma API em Java capaz de produzir documentos no formato RSS, versão 2.0. O Xerces é um interpretador XML para Java utilizado pelo jrss.

- **openid4java** (OpenID 2.0 Java Libraries)[85] – É uma biblioteca de classes que permite a criação de um consumidor OpenID, que deve ser incorporado numa aplicação Web para que esta possa fazer autenticação de usuários através de um provedor OpenID.

6.3. Especificação do Ambiente AvaNCE

Esta especificação é composta da descrição funcional e dos casos de uso do ambiente, seguidos de uma descrição das classes implementadas e das chamadas das APIs REST oferecidas. Além do núcleo do ambiente, é documentado também o fórum de discussão, desenvolvido como uma aplicação interna do AvaNCE.

---

[82] http://msdn.microsoft.com/en-us/data/aa937724.aspx
[83] http://jrss.sourceforge.net/
[84] http://xerces.apache.org/xerces-j/
[85] http://code.google.com/p/openid4java/

6.3.1. Descrição funcional do núcleo

A tela do AvaNCE (Figura 6.2) é composta de uma área de login, de um menu de opções e de uma área de exibição de informações à direita. A autenticação de usuários pode ser realizada internamente ao AvaNCE através de CPF e senha ou através de OpenID. Na identificação com OpenID o usuário digita sua URI fornecida pelo seu provedor de identidade, sendo remetido então a este para que ele solicite sua senha e autorize o acesso a aplicação (Figuras 6.3 e 6.4). Uma vez que o usuário tenha se identificado, o sistema exibe a listagem de turmas nas quais o usuário tem alguma participação, seja como aluno ou como professor (Figura 6.5).

Ao selecionar uma turma nesta listagem, o usuário visualiza a página da turma onde são listadas todas as aplicações que fazem parte dela (Figura 6.6). Estas aplicações podem ser internas, externas ou feeds RSS.

Todas as aplicações externas que fazem parte desta lista têm os seus recursos exibidos de forma que o usuário possa executar cada um dos métodos disponíveis na API da aplicação correspondente. O processo de execução dos métodos de APIs externas será exemplificado no Capítulo 7. Feeds RSS externos podem ser visualizados diretamente no ambiente (Figura 6.7).

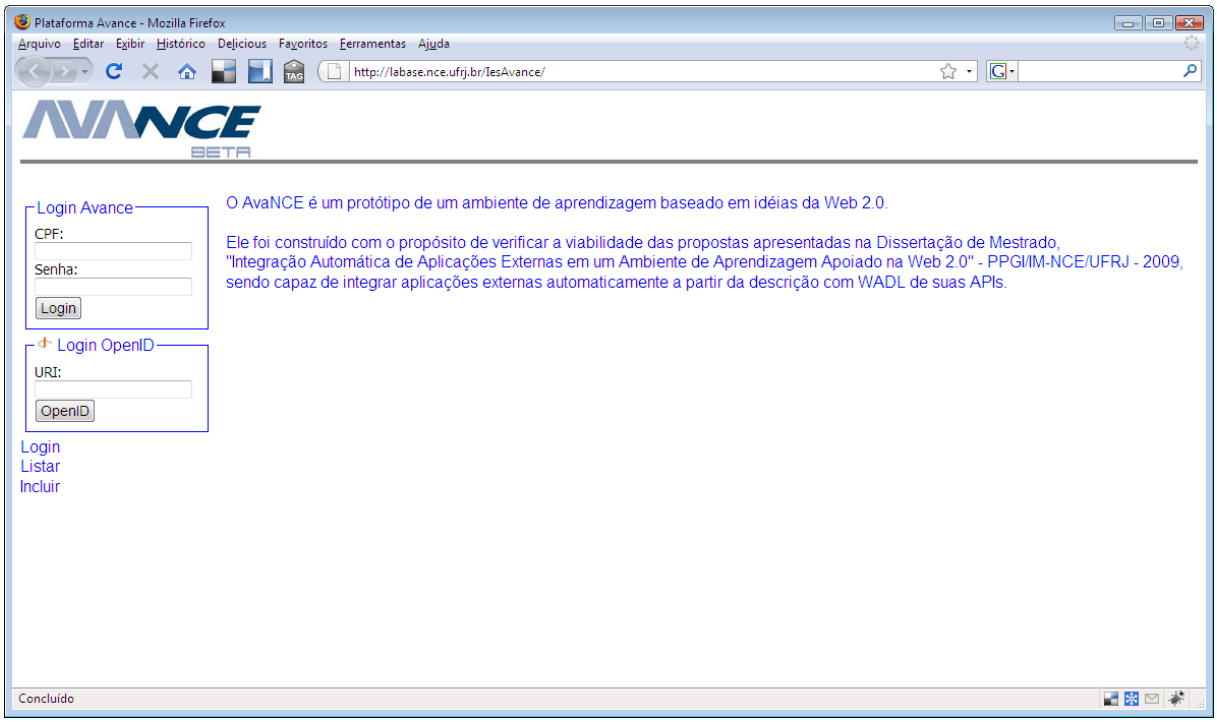

Figura 6.2 – Tela inicial do AvaNCE

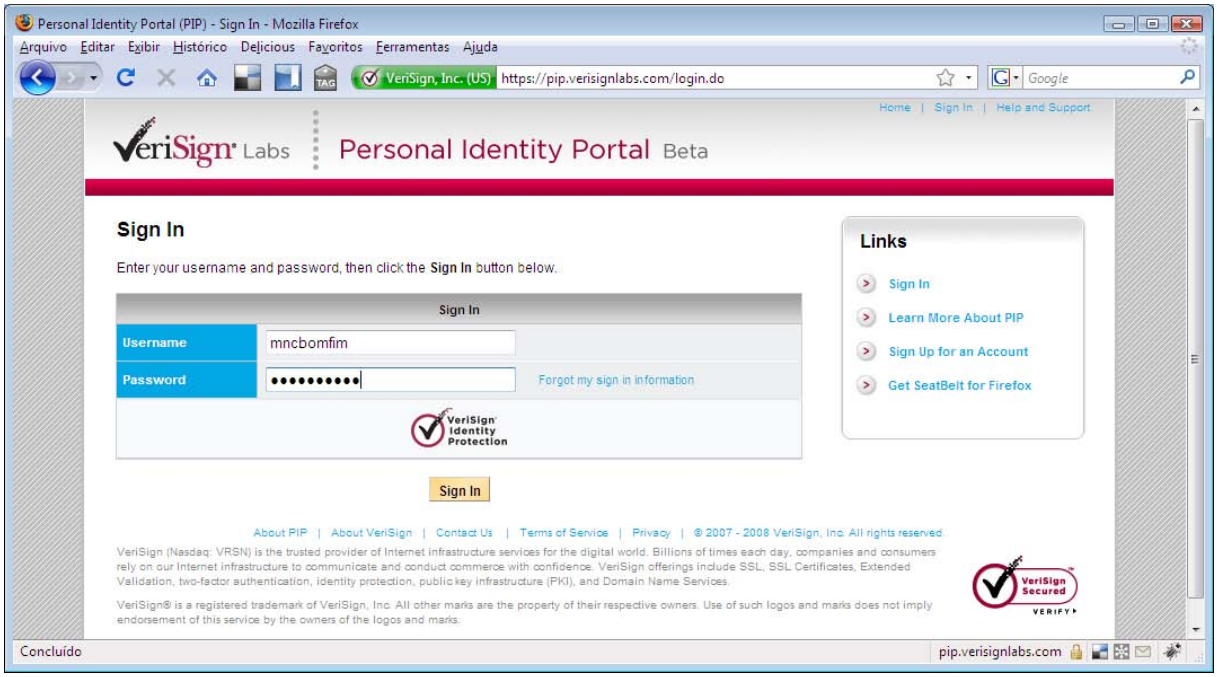

Figura 6.3 – Solicitação de conta e senha no provedor OpenID

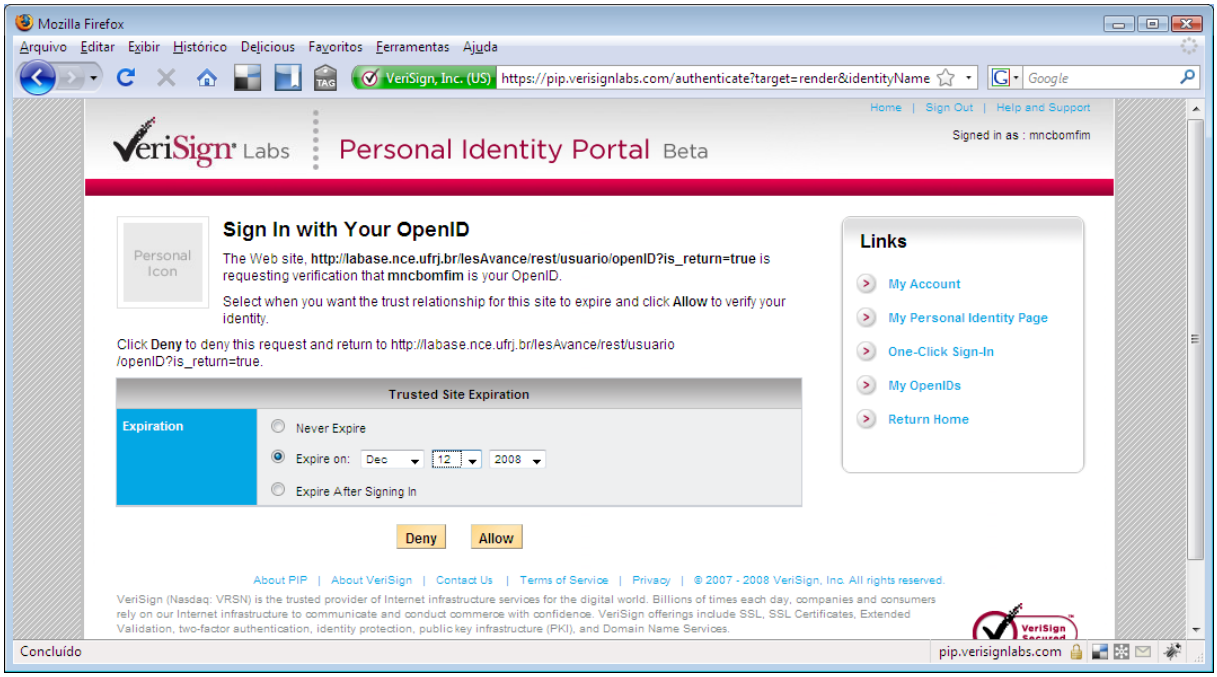

Figura 6.4 – Determinação do prazo de expiração da autenticação

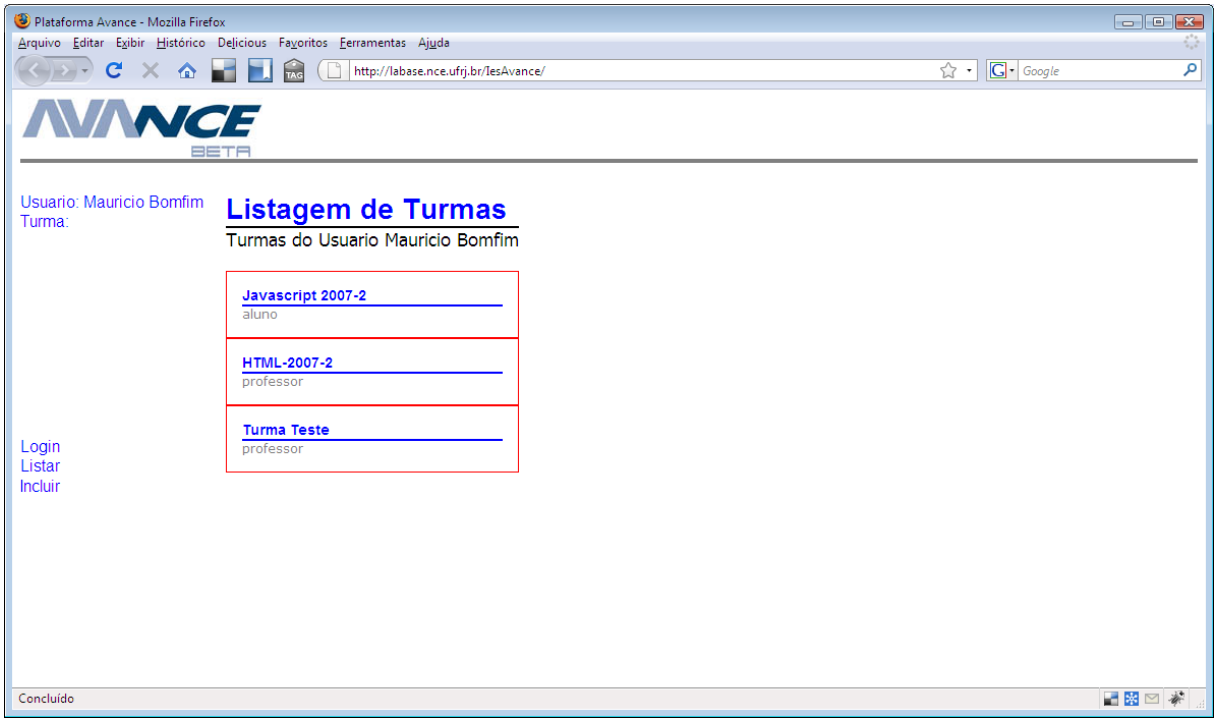

Figura 6.5 – Listagem de turmas de um usuário

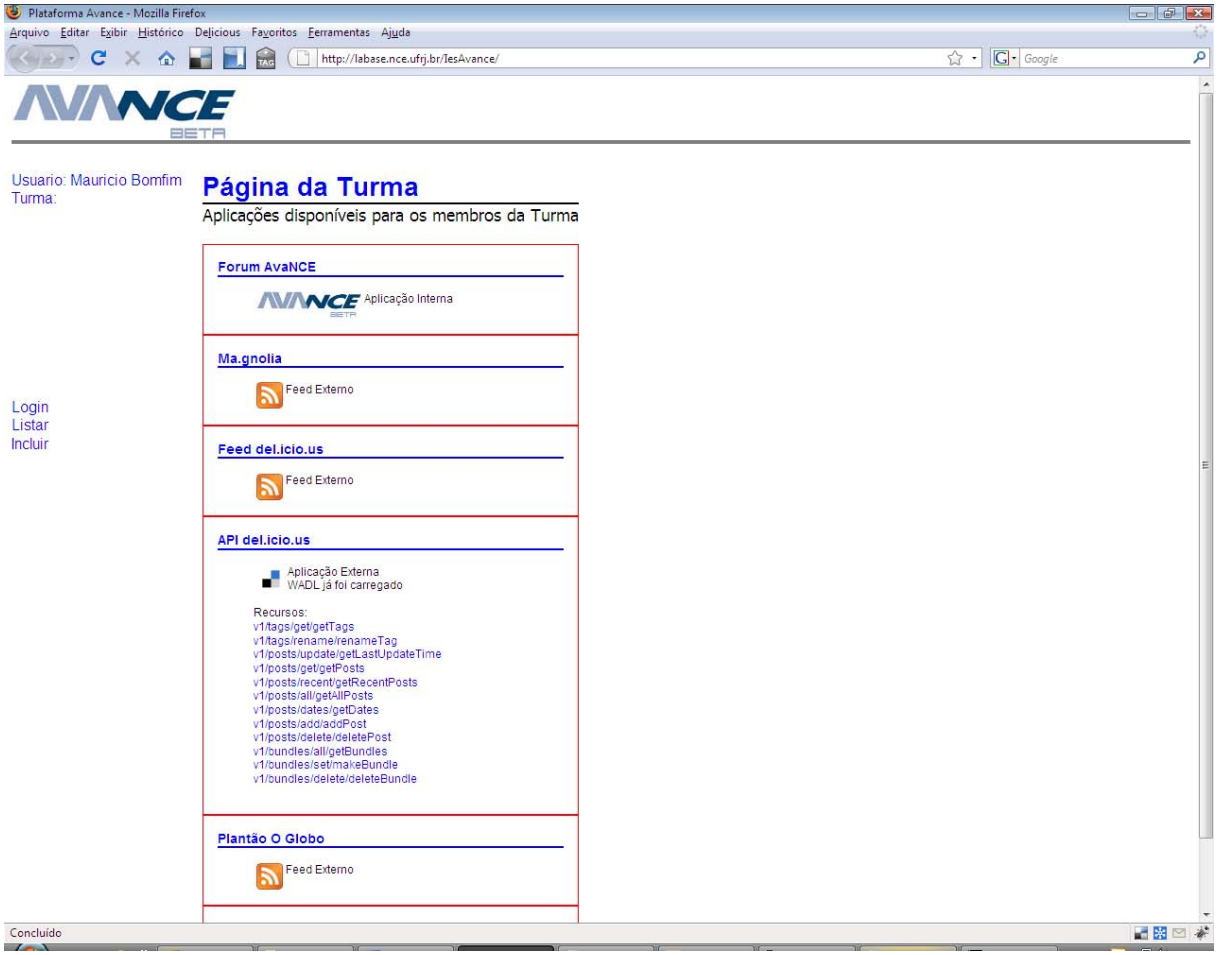

Figura 6.6 – Página de uma turma

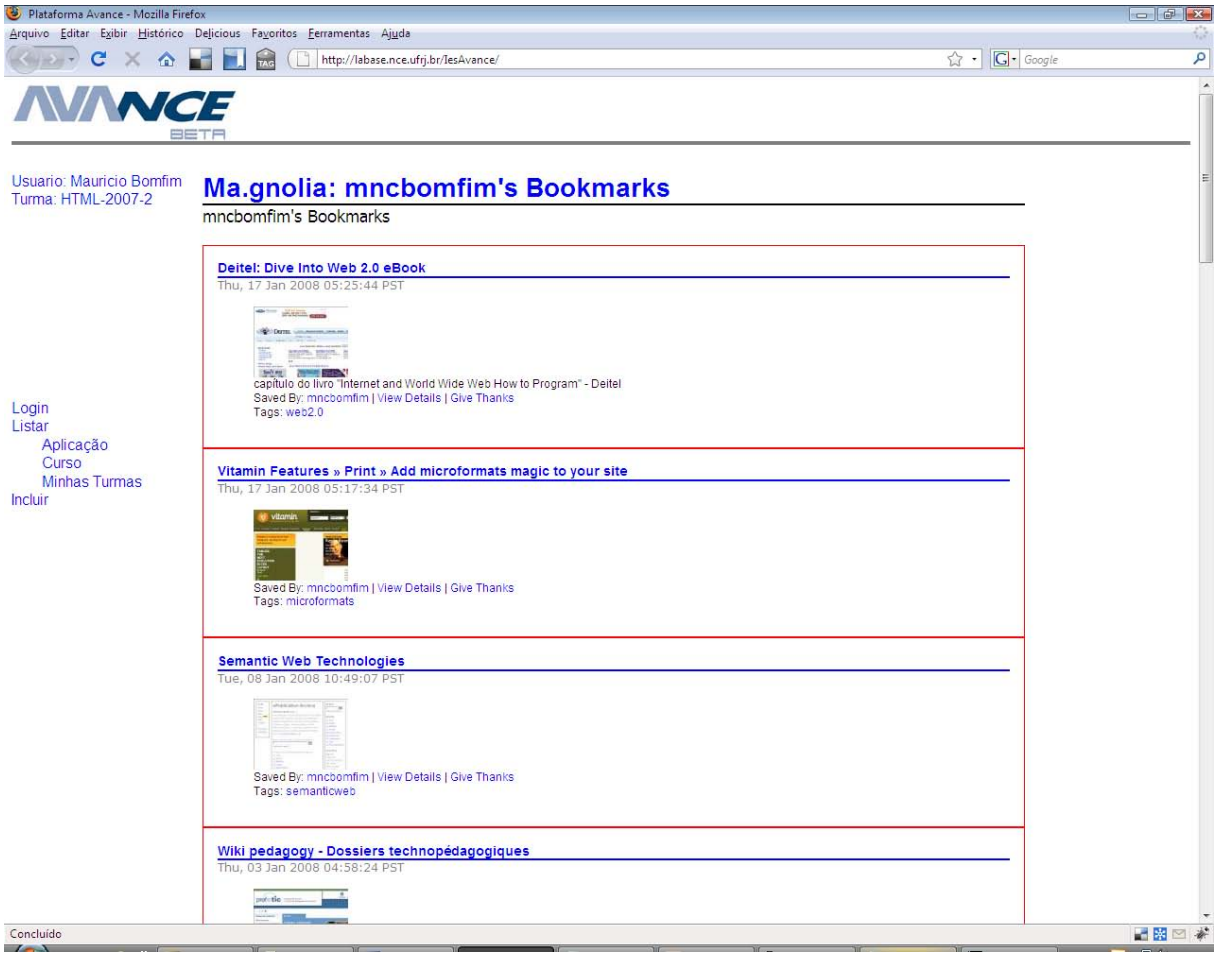

Figura 6.7 – Visualização de um RSS externo

O processo de incorporação de aplicações externas é realizado a partir do menu "Incluir -> Aplicação". O usuário fornece então o nome da aplicação, a sua URL e o seu tipo (Figura 6.8).

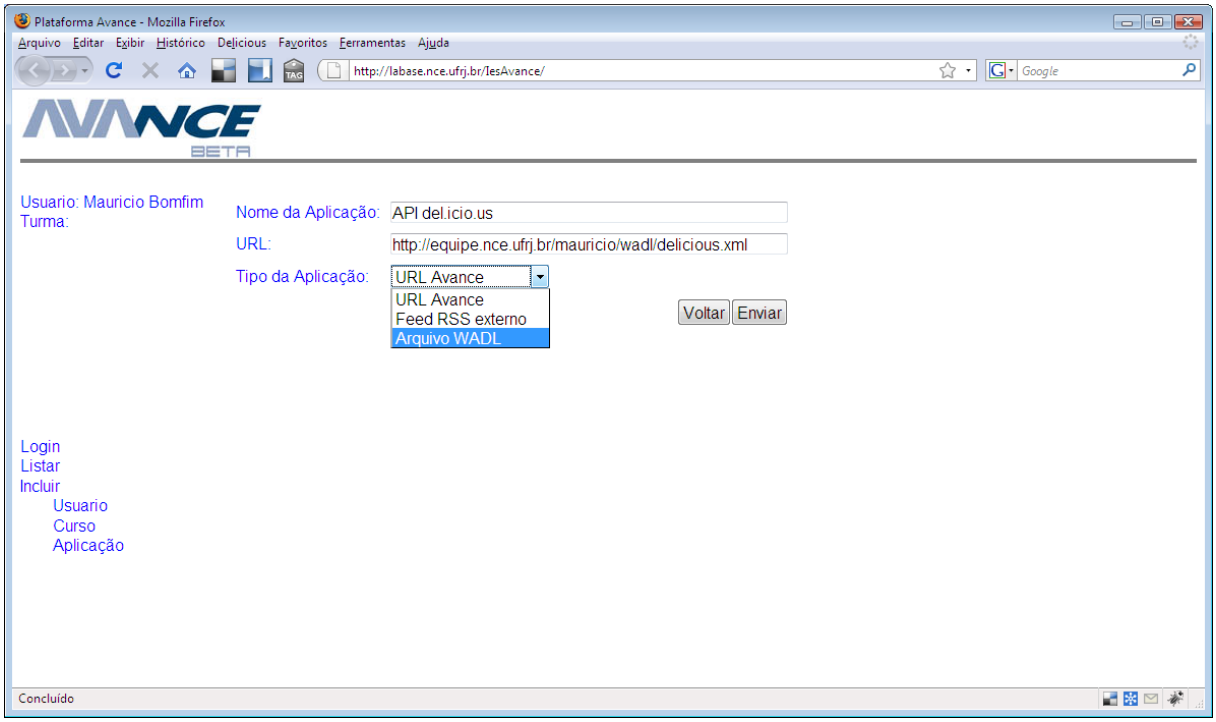

Figura 6.8 – Incorporação de uma aplicação externa

Para carregar o WADL da aplicação, o usuário deve pressionar o botão correspondente na lista de aplicações obtida a partir de "Listar -> Aplicação" (Figura 6.9). Para incluir a aplicação numa turma, o usuário obtém a lista de aplicações existentes a partir de Listar -> Curso, navegando por disciplina, turma e aplicação da turma até poder incluir uma nova aplicação na turma (Figura 6.10). A aplicação é então selecionada numa lista com as aplicações existentes (Figura 6.11) e, uma vez incluída, passa a figurar na página da turma correspondente.

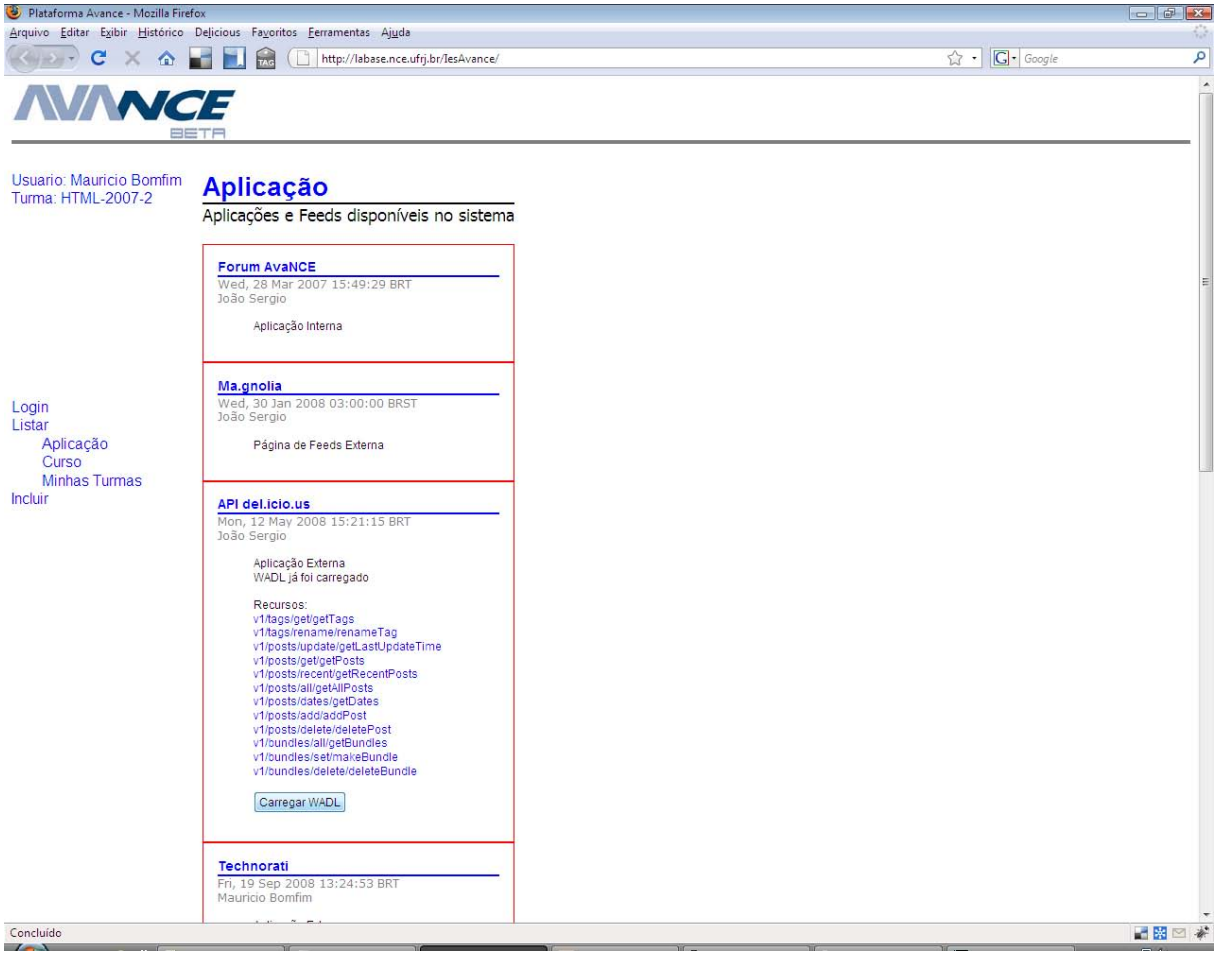

Figura 6.9 – Interpretação da descrição WADL de uma aplicação externa

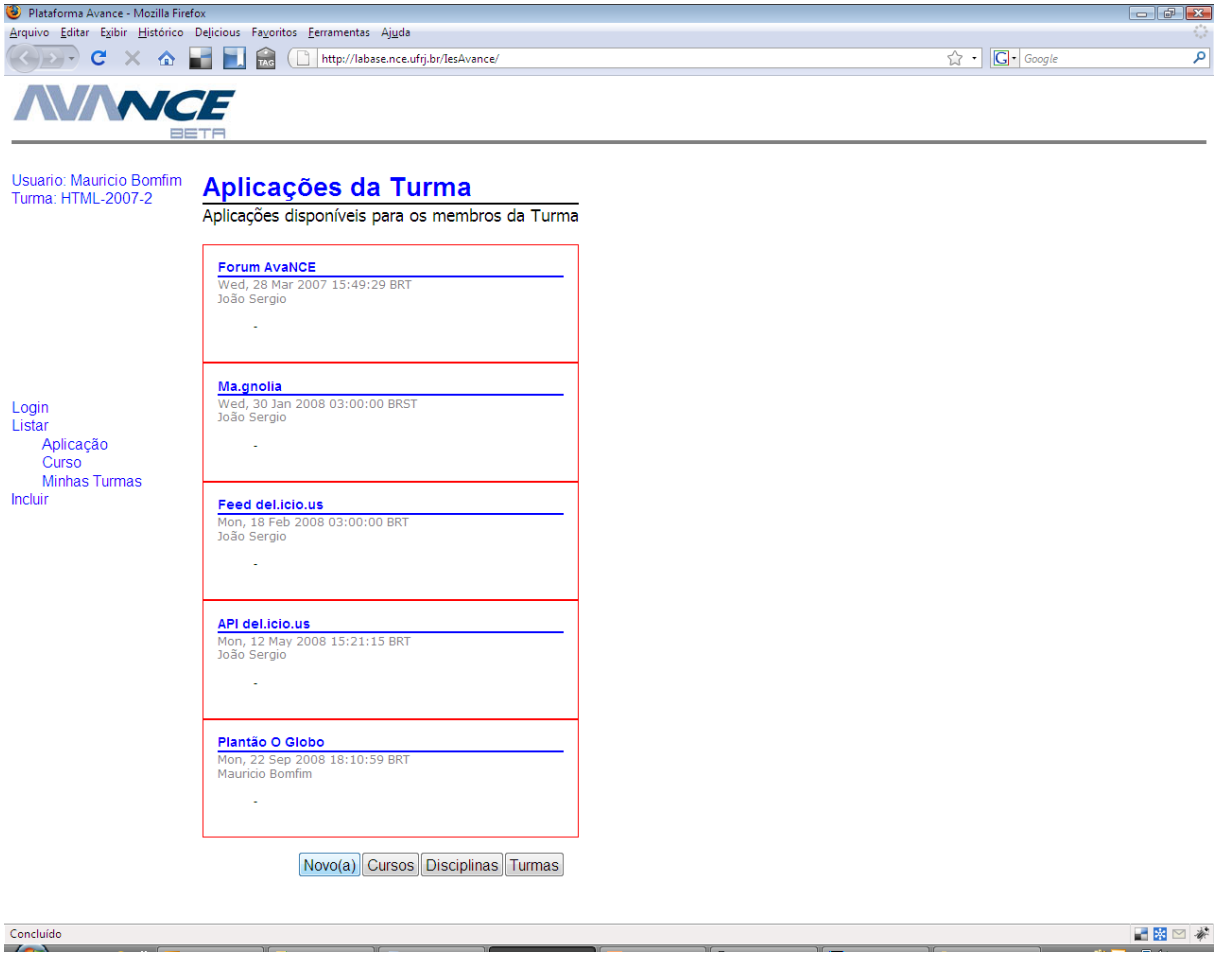

Figura 6.10 – Inclusão de uma aplicação em uma turma

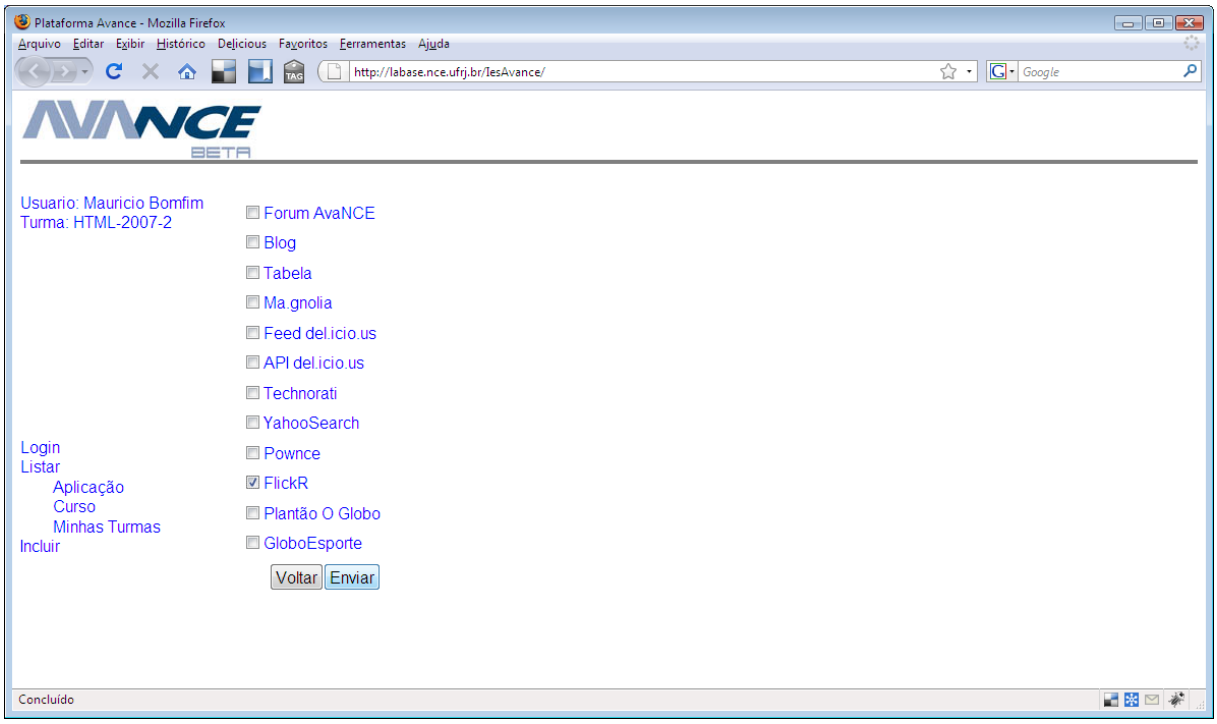

Figura 6.11 – Selecionando a aplicação a ser incluída

6.3.2. Descrição funcional do fórum de discussão

As informações do fórum estão organizadas em quatro níveis: Temas, Assuntos, Tópicos e Mensagens.

Temas e Assuntos são utilizados pelo criador do fórum, seja ele o professor de uma turma ou o dono de uma comunidade, para organizar hierarquicamente (como nos capítulos e seções de um livro), o conteúdo a ser estudado. Dentro de um Assunto, os usuários podem criar Tópicos que por sua vez são compostos de Mensagens. Um Tópico corresponde a uma nova discussão ou, simplesmente, uma nova pergunta colocada por um usuário. As Mensagens do tópico são as respostas ou participações individuais dos demais usuários nesta discussão. A Figura 6.12 exemplifica a utilização destes quatro níveis na organização das informações num fórum específico.

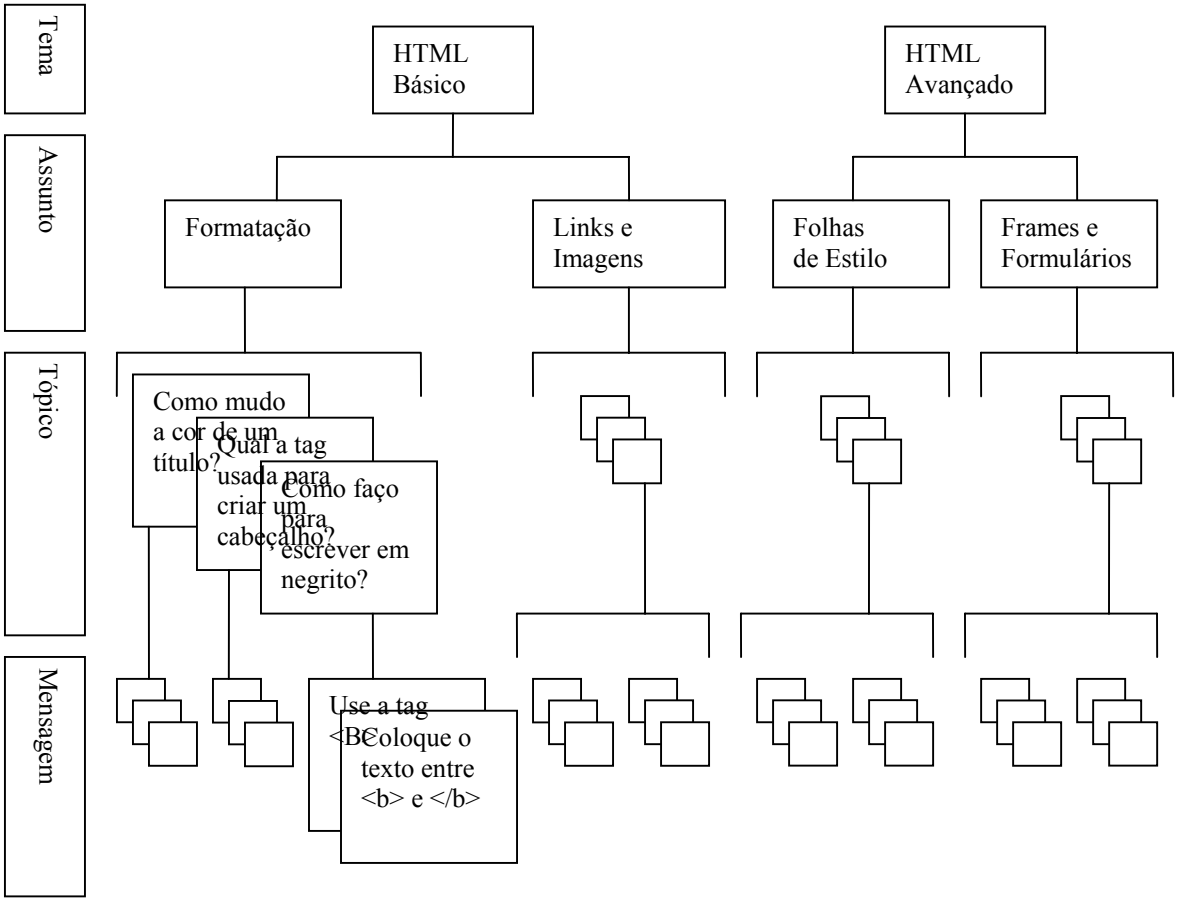

Figura 6.12 – Hierarquia do Fórum do AvaNCE

Ao entrar no fórum de discussão a partir da página de uma turma, o usuário pode visualizar a lista de temas (Figura 6.13). Da mesma forma, ao selecionar um tema o usuário obtém a lista de

assuntos deste tema (Figura 6.14), ao selecionar um assunto obtém a lista de tópicos (Figura 6.15) e, ao selecionar um tópico obtém a lista de mensagens (Figura 6.16). Abaixo de cada uma destas listas existem botões que podem ser utilizados para incluir um novo item ou para retornar aos níveis anteriores. Ao pressionar o botão de inclusão, o formulário correspondente é exibido solicitando as informações necessárias (Figura 6.17).

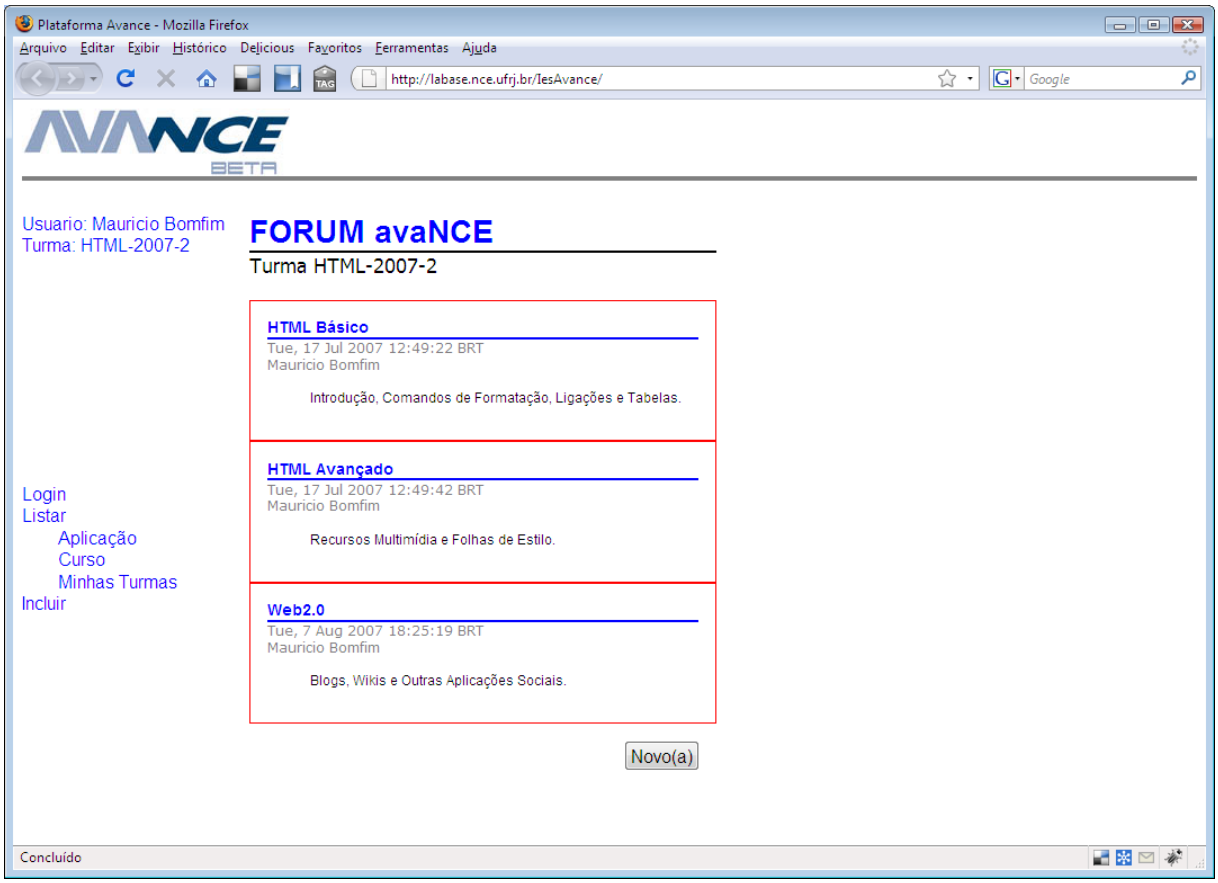

Figura 6.13 – Lista de temas do fórum de uma turma

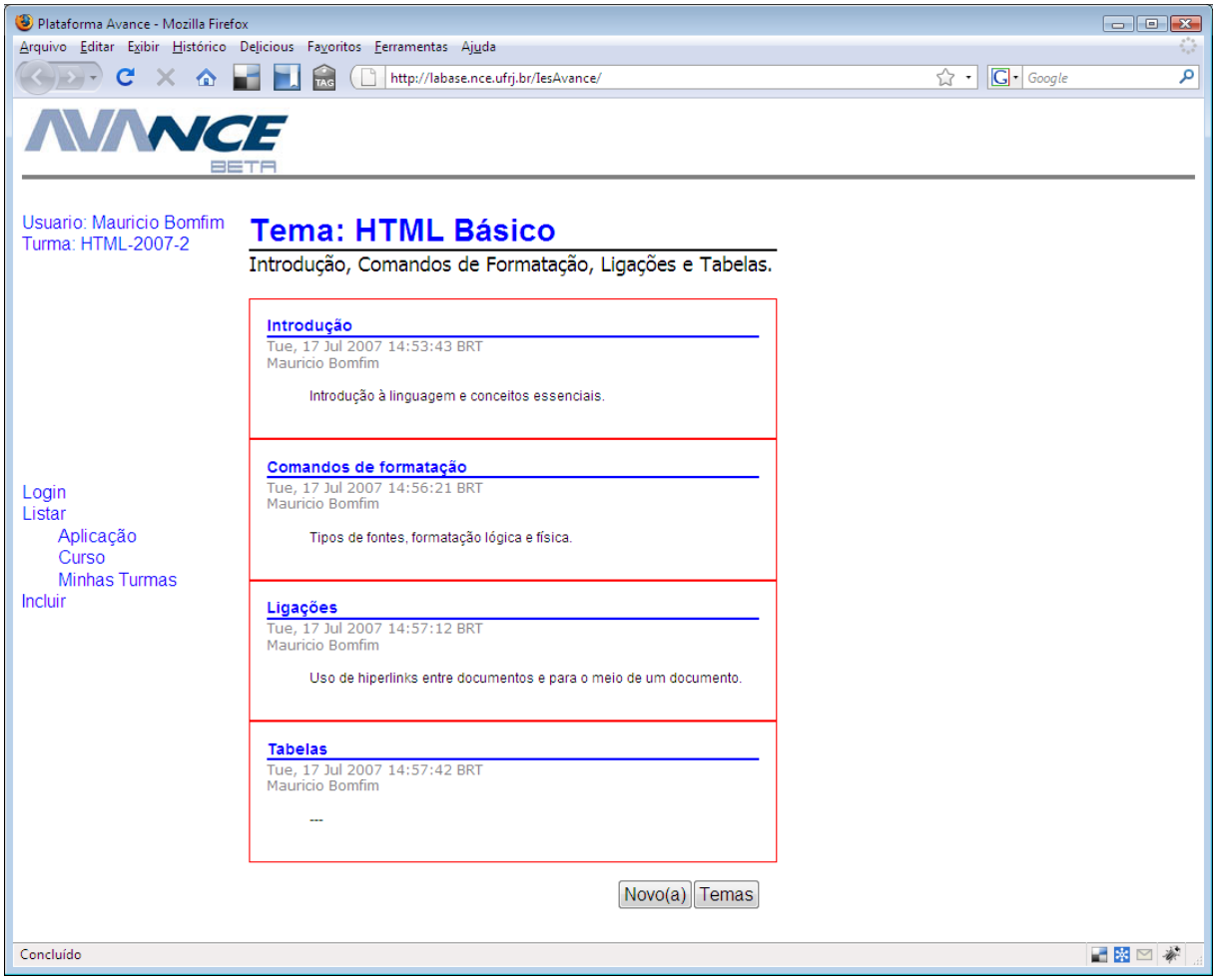

Figura 6.14 – Lista de assuntos de um tema

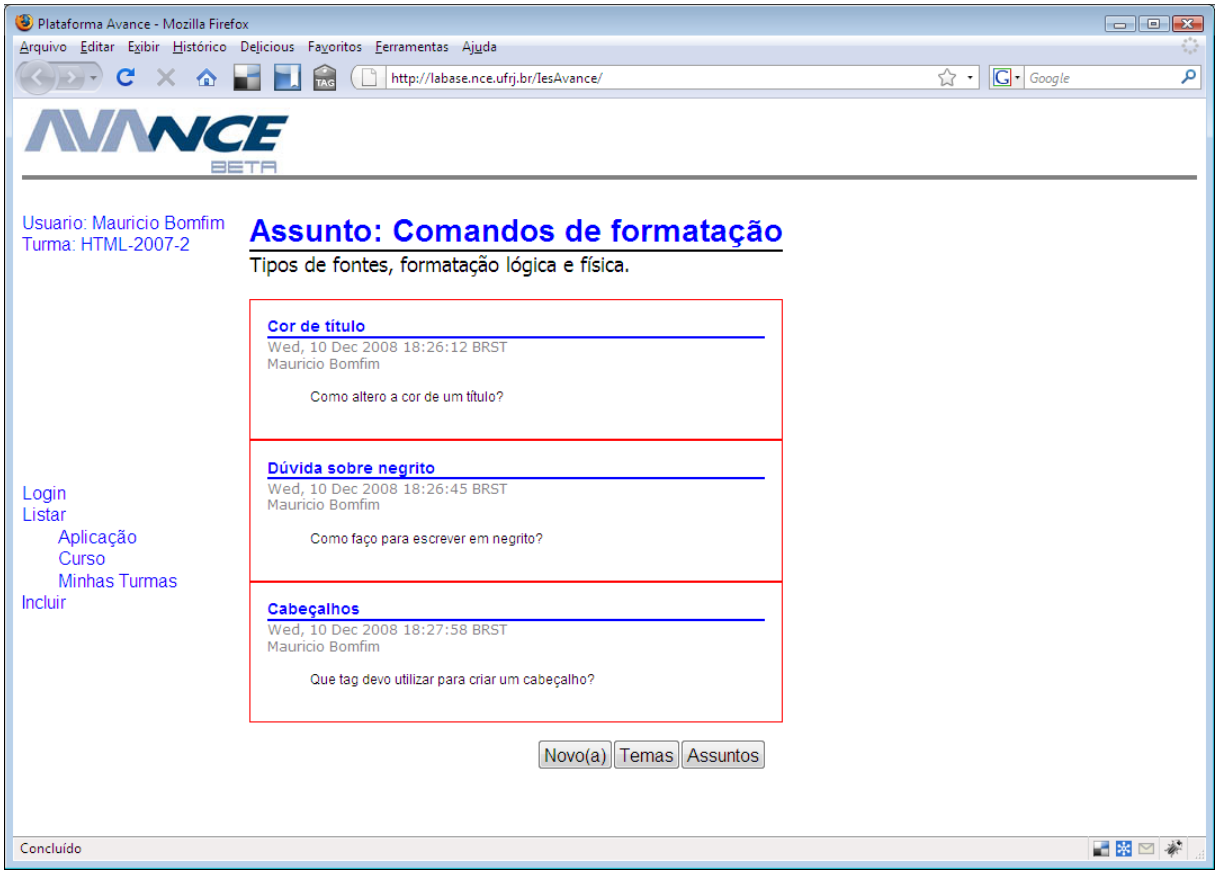

Figura 6.15 – Lista de tópicos de um assunto

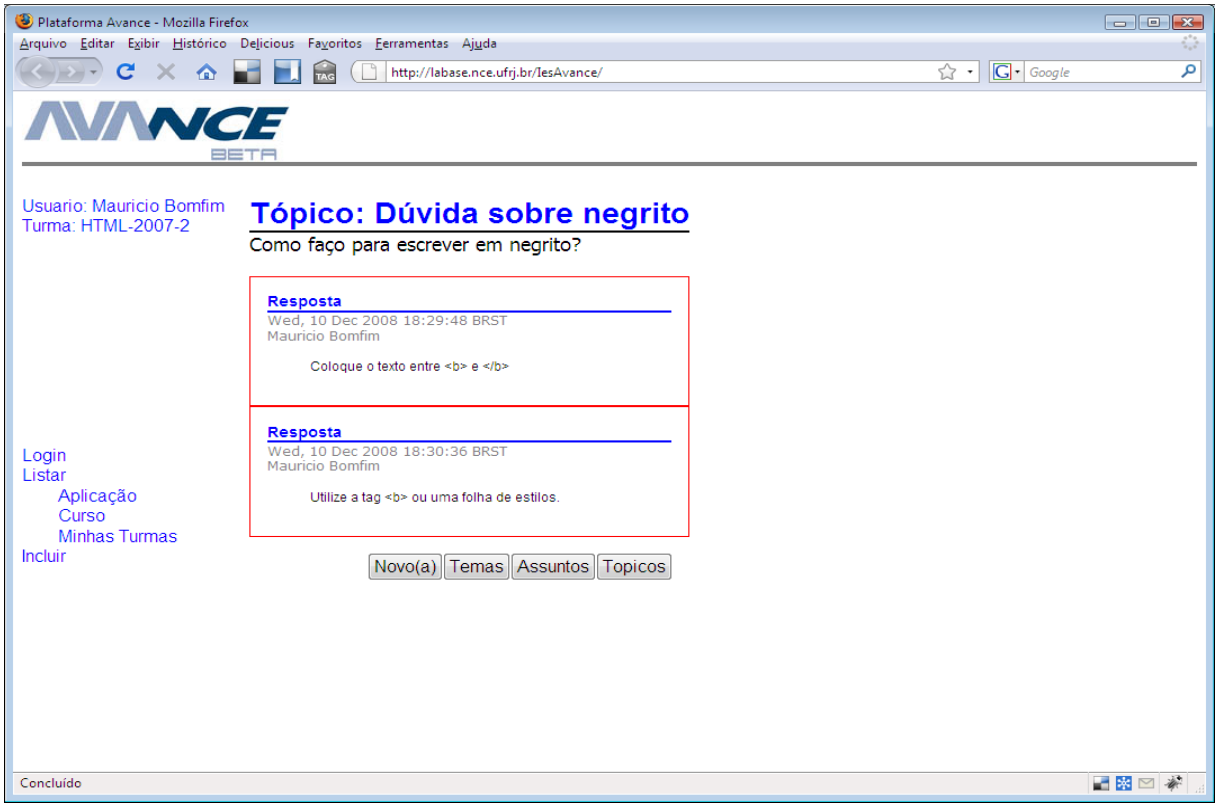

Figura 6.16 – Lista de mensagens de um tópico

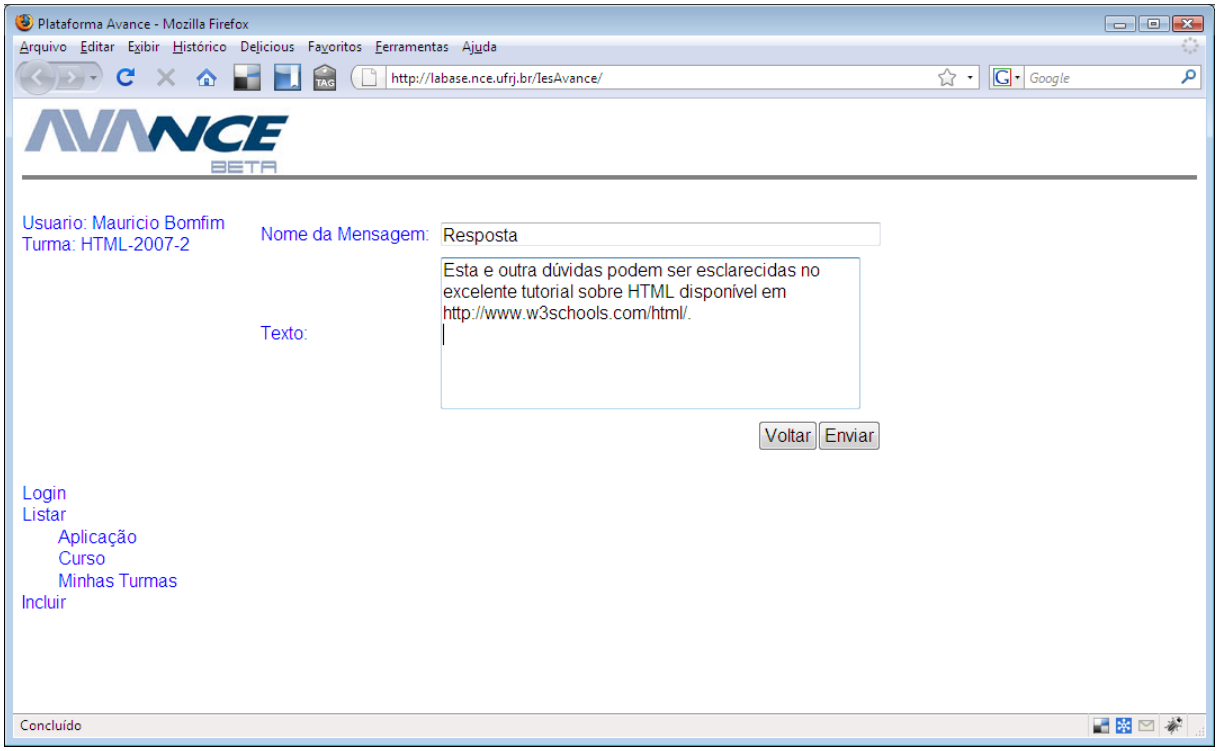

Figura 6.17 – Inclusão de uma nova mensagem num tópico

6.3.3. Casos de uso

Para diferenciar o usuário autenticado do usuário anônimo, os casos de uso estão associados a dois atores diferentes: Pessoa e Usuário. Qualquer pessoa não identificada que acesse o ambiente pode cadastrar-se no ambiente (UC03) ou autenticar-se (UC01 e UC02). Uma vez autenticado, um usuário pode executar todos os demais casos de uso do sistema. São eles: listar as turmas em que estejam envolvidos (UC06), entrar na página destas turmas (UC07) e executar aplicações associadas a estas turmas (UC08), listar as aplicações externas cadastradas no sistema (UC09), listar as aplicações de uma turma (UC15), incluir uma aplicação em uma turma (UC18) e configurar uma aplicação externa de acordo com sua utilização por uma turma (UC11), listar e incluir cursos (UC12 e UC04), listar e incluir disciplinas (UC13 e UC16), listar e incluir turmas (UC14 e UC17), incluir aplicações no ambiente (UC05) e carregar a descrição WADL destas aplicações (UC10). A descrição destes casos de uso encontra-se no Apêndice A deste trabalho.

A Figura 6.18 apresenta o diagrama dos casos de uso implementados pelo AvaNCE.

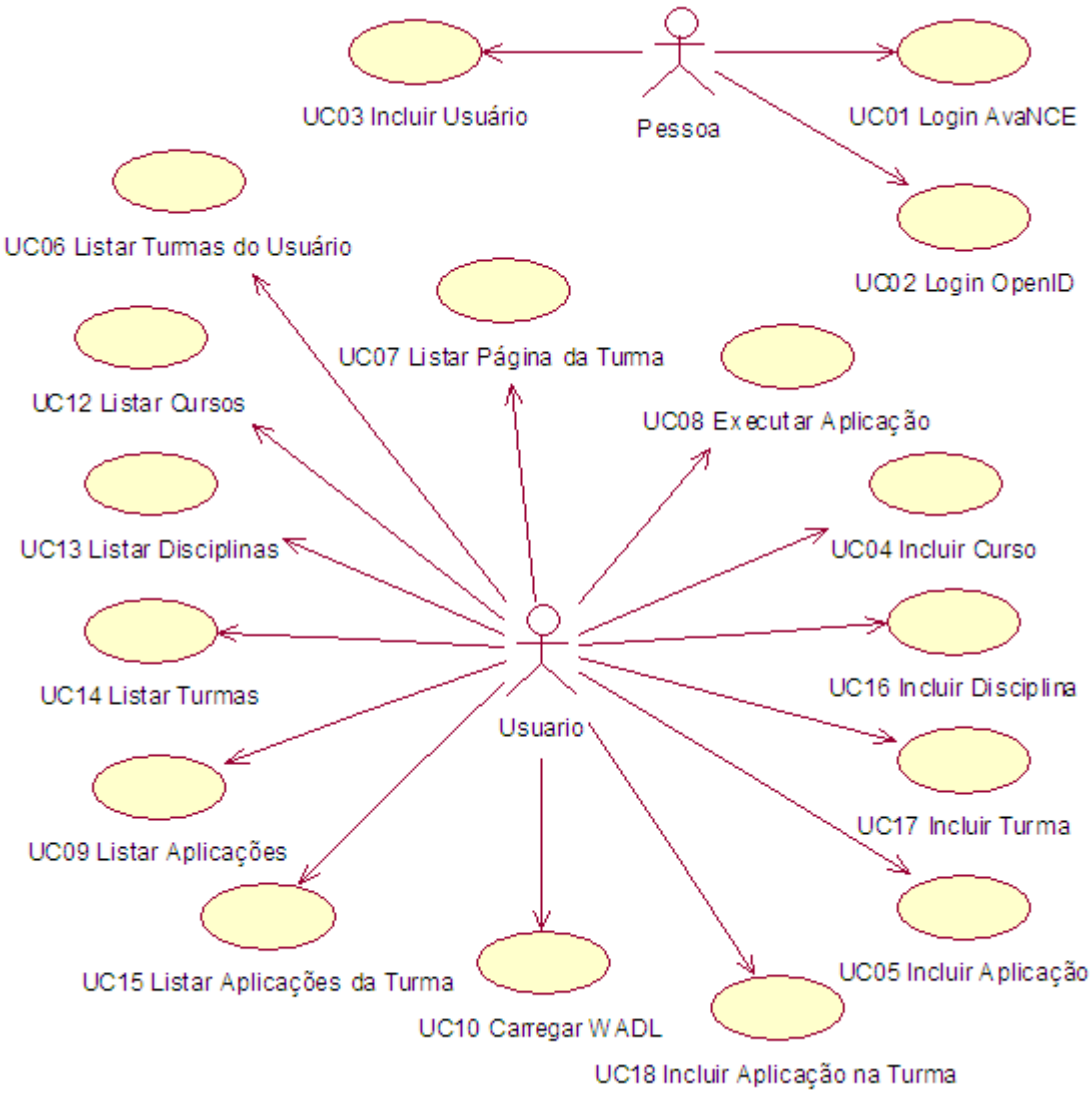

Figura 6.18 - Diagrama de casos de uso implementados pelo AvaNCE

Os casos de uso do fórum baseiam-se na consulta e inclusão de Temas, Assuntos, Tópicos e Mensagens, conforme apresentado no diagrama da Figura 6.19.

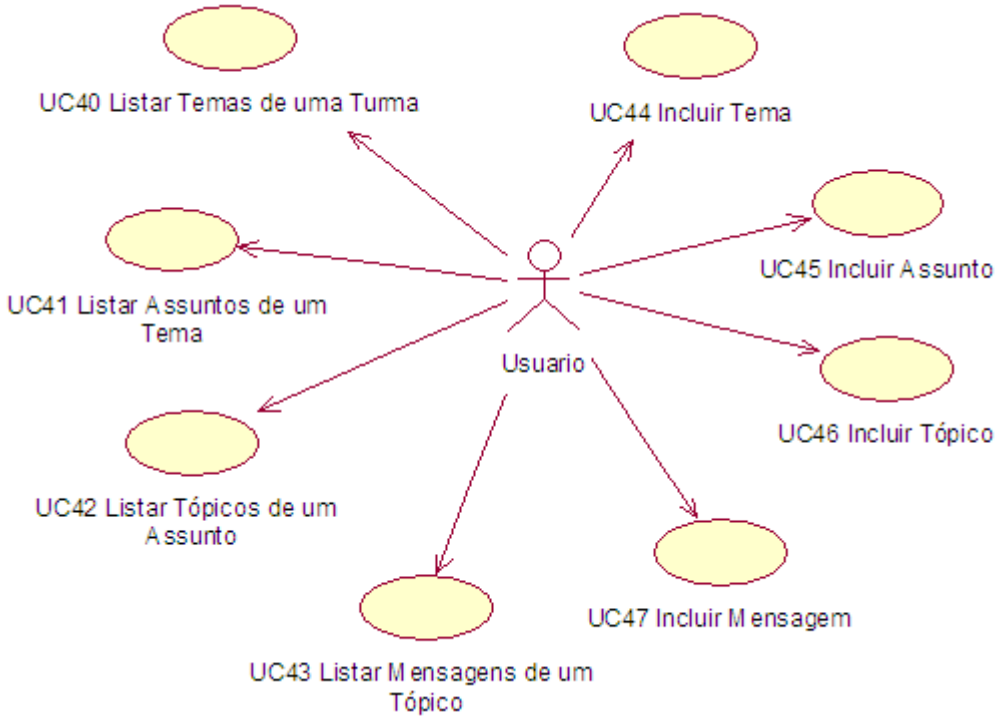

Figura 6.19 - Diagrama de casos de uso do Fórum do AvaNCE

6.3.4. Descrição das classes implementadas

6.3.4.1. Pelo núcleo do ambiente

O núcleo do ambiente AvaNCE é uma aplicação J2EE composta dos três Servlets apresentados na Seção 5.4. As principais classes utilizadas pelo núcleo do AvaNCE são *Curso*, *Disciplina*, *Turma*, *AplicacaoDaTurma* e *Aplicacao*. Um curso pode possuir várias disciplinas que por sua vez podem possuir várias turmas. Uma Turma é composta de várias Aplicações que podem ser de três tipos: *feeds* RSS e aplicações internas ou externas.

- *Feeds* RSS são URLs externas de arquivos RSS que podem ser exibidos dentro do contexto da turma.

- Aplicações internas são aplicações J2EE independentes do AvaNCE mas que podem compartilhar seu banco de dados e suas classes.

- Aplicações externas são representadas através de URLs externas de arquivos WADL que, se carregados no ambiente, produzem uma estrutura de dados que representa os recursos expostos pela API da aplicação, seus parâmetros e formato de retorno. Para representar esta estrutura existem as classes *Recurso*, *Método*, *Parâmetro* e *Resposta* associadas aos elementos utilizados no arquivo WADL. Além destas, existem também as classes

Autenticacao que representa o modelo de Autenticação utilizado pela aplicação e *NoArvoreResposta* que representa o formato de saída do XML produzido por cada requisição da API.

As classes do AvaNCE estão contidas nos seguintes pacotes:

- **avance** – É o pacote principal onde estão os servlets do ambiente e a classe ConsumerOpenID, responsável por fazer a autenticação com OpenID. Suas classes são: AvanceRestParser, Carrega, Distribuidor, Executor, FinalizaLoginOPenID e ConsumerOpenID.

- **nucleo.nível** – É o pacote que implementa os objetos básicos do ambiente. Suas classes são: Nivel, NivelAvance, Aplicacao, AplicacaoDaTurma, Curso, Disciplina e Turma.

- **nucleo.lista** – É o pacote que implementa listas de objetos básicos. Suas classes são: ListaNivelAvance, ListaAplicacao, ListaAplicacaoDaTurma, ListaCurso, ListaDisciplina e ListaTurma.

- **nucleo.output** – É o pacote onde são produzidas as saídas de cada objeto básico, sejam elas arquivos RSS ou XForms. Suas classes são: GeraSaida, GeraSaidaAvance, GeraSaidaUsuario, GeraSaidaAplicacao, GeraSaidaAplicDaTurma, GeraSaidaCurso, GeraSaidaDisciplina e GeraSaidaTurma.

- **nucleo.persist** – É o pacote que implementa a camada de persistência dos objetos básicos. Suas classes são: NivelAvancePersist, AplicacaoDaTurmaPersist, AplicacaoPersist, CursoPersist, DisciplinaPersist e TurmaPersist.

- **nucleo.usuario** – É o pacote que implementa objetos referentes ao controle de Usuários e Sessões. Suas classes são: Sessao, SessaoLog, SessaoPersist, Usuario e UsuarioPersist.

- **nucleo.wadl** – É o pacote que implementa a interpretação do arquivo WADL e sua persistência no banco de dados. Suas classes são: NivelWADL, Autenticacao, Metodo, Parametro, Recurso, Resposta, NivelWADLPersist, AutenticacaoPersist, MetodoPersist, ParametroPersist, RecursoPersist, RespostaPersist, NoArvoreResposta, NoElemento, NoPropriedade, NoArvoreRespostaPersist, NoElementoPersist e NoPropriedadePersist.

- **util.arquivo** – É o pacote que implementa a leitura de arquivos externos. Suas classes são: Arquivo e ExcecaoArquivo.

- **util.banco** – É o pacote que implementa funções básicas da conexão ao banco. Suas classes são: ExcecaoBanco e Persistencia.

- **util.interpretador** – É o pacote que implementa a interpretação da string REST através de uma máquina de estados e a interpretação das respostas fornecidas pelas APIs externas. Suas classes são: ContextList, ContextNode, IntContextNode, StrContextNode, DomParser, ExcecaoParser, ParseNode, ResponseParser e RestParser.

A Figura 6.20 apresenta um diagrama com as principais classes implementadas no AvaNCE.

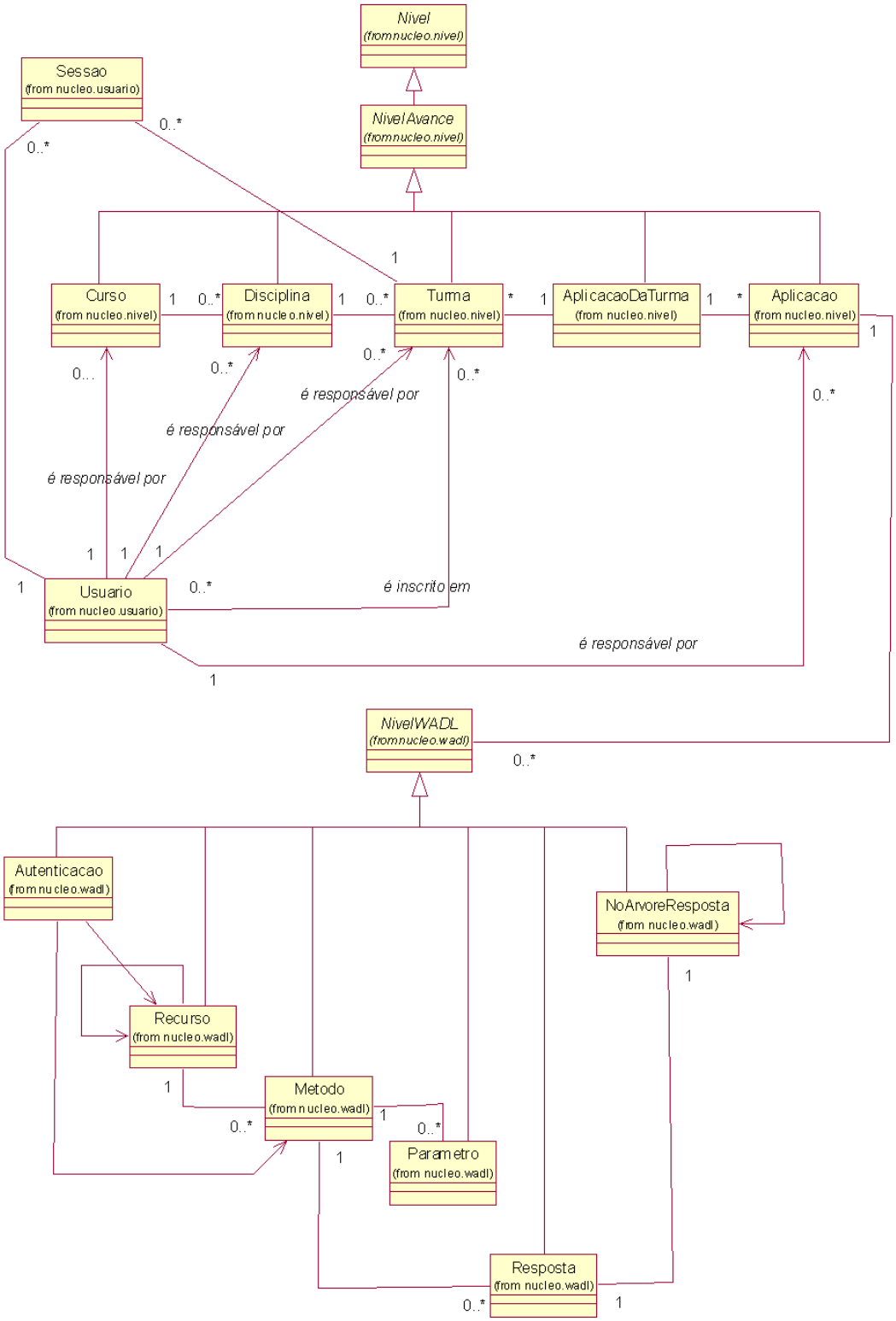

Figura 6.20 - Diagrama com as principais classes do AvaNCE

6.3.4.2. Pelo fórum de discussão

O Fórum é uma aplicação J2EE independente do núcleo, mas que como mencionado na Seção 5.4.2, pode reutilizar classes implementadas por este.

O Distribuidor do fórum é um Servlet Java responsável por decodificar a String da requisição REST e acionar o método responsável por executar a ação correspondente. Isto é feito através da classe ForumRestParser que implementa uma máquina de estados

As classes do Fórum AvaNCE estão contidas nos seguintes pacotes:

- **forum** – É o pacote principal do fórum. Suas classes são: Distribuidor e ForumRestParser.

- **forum.nivel** – É o pacote que implementa os objetos básicos do Fórum. Suas classes são: Assunto, Mensagem, NivelForum, Tema e Tópico.

- **forum.lista** – É o pacote que implementa listas de objetos básicos. Suas classes são: ListaAssuntos, ListaMensagem, ListaNivelForum, ListaTemas e ListaTopicos.

- **forum.output** – É o pacote onde são produzidas as saídas de cada objeto básico, sejam elas arquivos RSS ou XForms. Suas classes são: GeraSaidaAssunto, GeraSaidaForum, GeraSaidaMensagem, GeraSaidaTema e GeraSaidaTopico.

- **forum.persist** – É o pacote que implementa a camada de persistência dos objetos básicos. Suas classes são: AssuntoPersist, MensagemPersist, NivelForumPersist, TemaPersist e TopicoPersist.

A Figura 6.21 apresenta o diagrama com as principais classes implementadas no Fórum AvaNCE. Neste diagrama é possível perceber a reutilização das classes Turma e Usuario definidas pelo núcleo e importadas por esta aplicação.

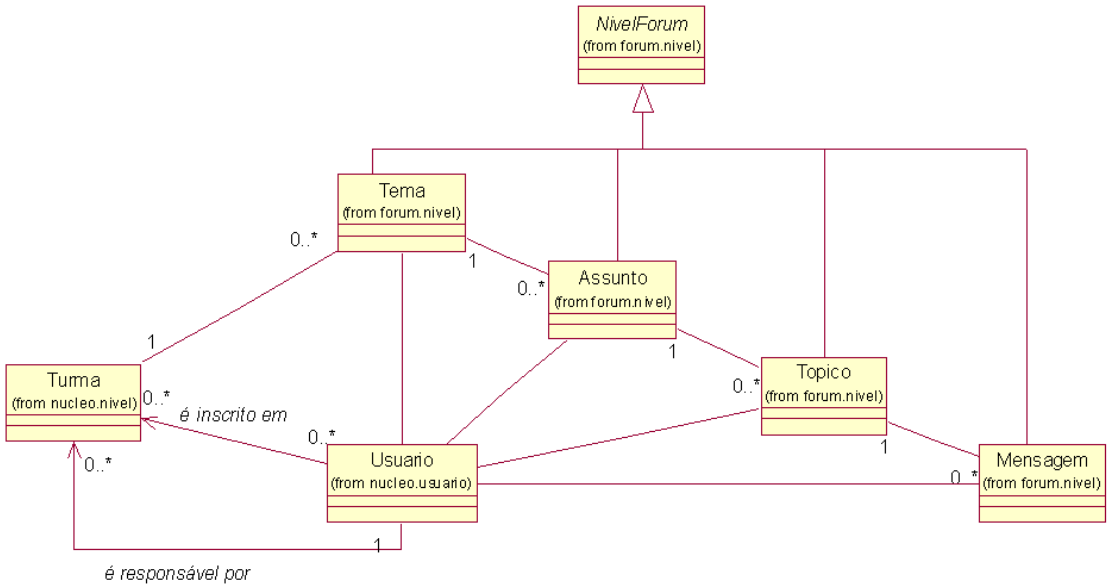

Figura 6.21 - Diagrama com as principais classes do Fórum AvaNCE

6.3.5. Descrição da API REST

O AvaNCE dispõe de uma API REST que possui um conjunto de operações, onde cada uma delas executa uma funcionalidade do núcleo. As operações podem ser de consulta, realizadas através de uma única requisição, ou de inclusão, realizadas através de duas requisições. A Figura 6.22 apresenta a relação de requisições da API REST do AvaNCE. Uma descrição mais completa desta API pode ser encontrada no Apêndice B desta dissertação.

```
Autenticação
GET /Avance/rest/sessao/form
POST /Avance/rest/sessao
POST /Avance/rest/openid

Listar cursos
GET /Avance/rest/[idSessao]/curso

Listar disciplinas de um curso
GET /Avance/rest/[idSessao]/curso/[idCurso]

Listar turmas de uma disciplina
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]

Listar aplicações disponíveis para os membros de uma turma
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]

Listar turmas de um usuário
GET /Avance/rest/[idSessao]/turma

Listar a página da turma (com aplicações disponíveis)
GET /Avance/rest/[idSessao]/turma/[idTurma]

Listar aplicações cadastradas no sistema
GET /Avance/rest/[idSessao]/aplicacao

Criar usuário
GET /Avance/rest/usuario/form
POST /Avance/rest/usuario

Criar curso
GET /Avance/rest/[idSessao]/curso/form
POST /Avance/rest/[idSessao]/curso

Criar disciplina
GET /Avance/rest/[idSessao]/curso/[idCurso]/form
POST /Avance/rest/[idSessao]/curso/[idCurso]

Criar turma
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/form
POST /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]

Criar inscrição de aluno em turma
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/ ►
       inscricao/[idAluno]/form
POST /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/ ►
       inscricao/[idAluno]

Inserir aplicação no sistema
GET /Avance/rest/[idSessao]/aplicacao/form
POST /Avance/rest/[idSessao]/aplicacao

Carrega WADL da aplicação
GET /Avance/rest/[idSessao]/aplicacao/[idAplic]

Incluir uma aplicação numa turma
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/form
POST /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]
```

Figura 6.22 – Relação de chamadas à API REST do núcleo AvaNCE

6.3.5.1. Operações de consulta

As operações de consulta são realizadas através de uma requisição simples usando o método HTTP GET que retornam sempre um arquivo RSS com o resultado da consulta. Além disso, algumas chamadas, como as utilizadas para obter a relação de cursos, disciplinas e turmas, aproveitam-se da relação hierárquica existente entre estes conceitos. Por exemplo:

- GET /[idSessao]/curso - É utilizado para listar todos os cursos existentes.
- GET /[idSessao]/curso/[idCurso] - É utilizado para listar todas as disciplinas de um determinado curso cujo identificador (idCurso) foi fornecido na própria requisição.
- GET /[idSessao]/curso/[idCurso]/[idDisc] - É utilizado para listar todas as turmas de uma determinada disciplina cujo identificador (idDisc) foi fornecido na própria requisição.

Esta maneira de construir o caminho da requisição facilita a inclusão de ligações entre as representações dos recursos permitindo a navegabilidade entre elas. No exemplo anterior, o usuário obtém a lista de cursos. Ao clicar em um dos cursos da lista, obtém agora a lista de disciplinas daquele curso. Ao clicar numa das disciplinas, obtém a lista de turmas daquela disciplina, e assim por diante.

6.3.5.2. Operações de inclusão

As operações de inclusão exigem o envio de informações adicionais (por exemplo, para a inclusão de um novo curso é necessário fornecer o nome do curso, o professor responsável, etc). Como a solução aqui apresentada deve ser capaz de ser exportada para outros clientes, estas operações são realizadas em dois passos:

- Primeiro executa-se uma requisição do tipo GET para obter o formulário através do qual, os dados serão fornecidos pelo usuário. Este formulário é enviado no formato padronizado XForms, permitindo sua apresentação por diferentes clientes.
- Uma vez que este formulário seja submetido pelo usuário, será realizada uma nova requisição do tipo POST, enviando os valores dos parâmetros fornecidos, para que então a operação de inclusão possa ser efetivamente realizada.

Por exemplo, para criar uma turma de uma determinada disciplina, é necessário requisitar o recurso:

GET /[idSessao]/curso/[idCurso]/[idDisc]/form

para obter o formulário para solicitação dos dados da turma a ser criada. Posteriormente realiza-se a submissão do formulário através da requisição:

POST /[idSessao]/curso/[idCurso]/[idDisc]

que vai inserir nova turma, e retornar um arquivo RSS com a descrição da mesma.

6.3.5.3. Operações de alteração e exclusão

Embora não implementadas no escopo deste trabalho, as operações de alteração e exclusão poderiam ser oferecidas por esta API através de requisições utilizando os métodos PUT e DELETE.

Desta forma, a operação DELETE sobre um recurso identificado por sua URI leva a exclusão do respectivo recurso. Já a alteração, assim como a operação de inclusão vista anteriormente, precisa ser realizada em dois passos: numa primeira requisição utilizando o método GET o cliente solicita o formulário para alteração do recurso e, numa requisição seguinte utilizando o método PUT, o recurso é alterado a partir dos dados fornecidos através do formulário. De forma a permitir que a requisição GET para obtenção do formulário de alteração possa ser diferenciada daquela utilizada para o formulário de inclusão, é necessário criar um novo nome de recurso para definir o formulário de alteração. A Figura 6.23 apresenta alguns exemplos de operações de exclusão e alteração utilizando esta solução, onde se definiu o nome de recurso <form-alteracao> com este objetivo.

```
Excluir um curso
DELETE /Avance/rest/[idSessao]/curso/[idCurso]

Excluir uma disciplina
DELETE /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]

Excluir um turma
DELETE /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]

Alterar usuário
GET /Avance/rest/usuario/[idUsuario]/form-alteracao
PUT /Avance/rest/usuario/[idUsuario]

Alterar curso
GET /Avance/rest/[idSessao]/curso/[idCurso]/form-alteracao
PUT /Avance/rest/[idSessao]/curso/[idCurso]

Alterar disciplina
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/form-alteracao
PUT /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]

Alterar turma
GET /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/ ►
      form-alteracao
PUT /Avance/rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]
```

Figura 6.23 – Alteração e exclusão através da API REST do núcleo AvaNCE

6.3.5.4. Autenticação

Todas as requisições que exigem autenticação esperam que seja enviado um identificador de sessão. Este identificador é obtido com uma requisição especifica de autenticação que cria uma nova sessão. O processo de autenticação é realizado da mesma forma que as demais operações de inclusão. Uma primeira requisição obtém o formulário de login que solicita conta e senha do usuário. Quando do envio deste formulário uma nova requisição do tipo POST solicita a criação de uma nova sessão.

6.3.5.5. Formatos de saída e conectividade entre os recursos

Os formatos de saída mencionados pelo AvaNCE são XForms para descrição de formulários e RSS para listar informações.

A Figura 6.24 ilustra a interligação entre as representações dos recursos obtidos das chamadas à API. As funcionalidades podem ser acessadas pelo usuário a partir do menu principal da interface cliente, representado na figura pela barra vertical a esquerda. A partir daí, os recursos são interligados permitindo a navegação entre as representações dos mesmos.

Para permitir a inclusão de links adicionais nos arquivos RSS foram criados itens especiais que apresentam descrição vazia. Estes itens são considerados pela interface cliente como botões que

se pressionados produzem uma transição para outros recursos. É o caso dos botões para voltar e para incluir informações.

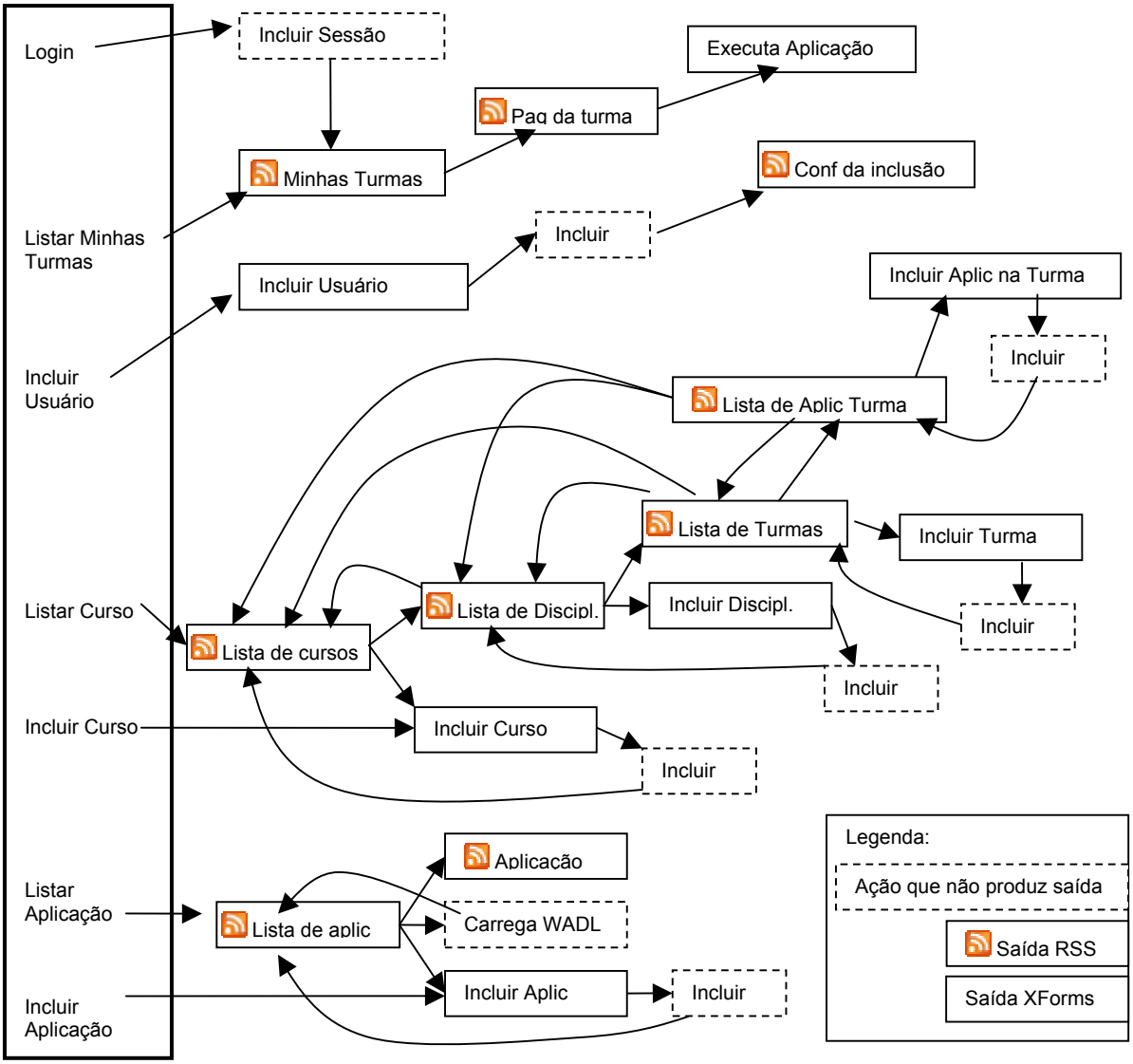

Figura 6.24 – Interligação entre as representações dos recursos da API REST do AvaNCE

As Figuras 6.25 e 6.26 apresentam dois exemplos de arquivos de saída produzidos AvaNCE. O primeiro é um RSS com uma lista de disciplinas e o segundo é um XForms para inclusão de um novo curso.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0">
<channel>
  <title>Disciplinas</title>
  <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11</link>
  <description>Disciplinas do curso 11</description>
  <language>pt</language>
  <item>
    <title>Criação de Páginas Web com HTML</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11/2</link>
    <description>-</description>
    <author>Mauricio Bomfim</author>
    <pubDate>Wed, 28 Mar 2007 14:20:36 BRT</pubDate>
  </item>
  <item>
    <title>Linguagem Javascript</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11/3</link>
    <description>-</description>
    <author>João Sergio</author>
    <pubDate>Wed, 28 Mar 2007 14:21:37 BRT</pubDate>
  </item>
  <item>
    <title>Linguagem Java</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11/14</link>
    <description>-</description>
    <author>Mauricio Bomfim</author>
    <pubDate>Mon, 17 Nov 2008 18:53:57 BRST</pubDate>
  </item>
  <item>
    <title>JSP e Servlets</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11/15</link>
    <description>-</description>
    <author>Mauricio Bomfim</author>
    <pubDate>Mon, 17 Nov 2008 18:54:21 BRST</pubDate>
  </item>
  <item>
    <title>Fundamentos de XML</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11/16</link>
    <description>-</description>
    <author>Mauricio Bomfim</author>
    <pubDate>Mon, 17 Nov 2008 18:57:30 BRST</pubDate>
  </item>
  <item>
    <title>Novo(a)</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso/11/form</link>
    <description></description>
  </item>
  <item>
    <title>Curso</title>
    <link>/IesAvance/rest/aFMU6QMQ_1pb/curso</link>
    <description></description>
  </item>
</channel>
</rss>
```

Figura 6.25 – Exemplo de arquivo RSS com uma relação de disciplinas

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="/IesAvance/xform.xsl"?>
<xforms>
   <model>
      <instance>
         <f_nome/>
         <f_instituicao/>
      </instance>
      <submission id="CriarCurso"
                  action="/IesAvance/rest/aFMU6QMQ_1pb/curso"
                  method="urlencoded-post"/>
   </model>
   <input ref="f_nome">
      <label>Nome do Curso:</label>
   </input>
   <input ref="f_instituicao">
      <label>Instituição:</label>
   </input>
   <submit submission="CriarCurso">
      <label>Enviar</label>
   </submit>
</xforms>
```

Figura 6.26 – Exemplo de arquivo XForms para inclusão de um novo curso

6.3.5.6. API REST do fórum de discussão

A exemplo da API REST do núcleo do AvaNCE onde cursos, disciplinas e turmas possuem uma relação hierárquica, aqui, na representação dos recursos relacionados aos conceitos de Tema, Assunto, Tópico e Mensagem esta relação também é explorada. Desta forma, para listar os temas de uma turma utiliza-se "GET /[idSessao]/[idTurma]"; para listar os assuntos de um tema utiliza-se "GET /[idSessao]/[idTurma]/[idTema]"; para listar todos os tópicos de um assunto utiliza-se "GET /[idSessao]/[idTurma]/[idTema]/[idAssunto]" e para listar todas as mensagens de um tópico utiliza-se "GET /[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]".

A Figura 6.27 apresenta uma relação das chamadas à API REST implementada pelo Fórum do AvaNCE.

```
Listar Temas
GET /Forum/rest/[idSessao]/[idTurma]

Listar Assuntos
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]

Listar Tópicos
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]

Listar Mensagens
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]

Inserir Tema
GET /Forum/rest/[idSessao]/[idTurma]/form
POST /Forum/rest/[idSessao]/[idTurma]

Inserir Assunto
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/form
POST /Forum/rest/[idSessao]/[idTurma]/[idTema]

Inserir Tópico
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/form
POST /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]

Inserir Mensagem
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]/form
POST /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]
```

Figura 6.27 – Relação de chamadas à API REST do Fórum AvaNCE

Assim como no Núcleo, também no Fórum, apenas operações de consulta e inclusão foram implementadas. Utilizando a mesma solução proposta na Seção 6.3.5.3, a Figura 6.28 apresenta as possíveis chamadas à API REST para executar operações de alteração e exclusão.

```
Alterar Tema
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/form-alteracao
PUT /Forum/rest/[idSessao]/[idTurma]/[idTema]

Alterar Assunto
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/form-alteracao
PUT /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]

Alterar Tópico
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]/ ►
       form-alteracao
PUT /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]

Alterar Mensagem
GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]/ ►
       [idMensagem]/form-alteracao
PUT /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]/ ►
       [idMensagem]

Remover Tema
DELETE /Forum/rest/[idSessao]/[idTurma]/[idTema]

Remover Assunto
DELETE /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]

Remover Tópico
DELETE /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]

Remover Mensagem
DELETE /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]/ ►
       [idMensagem]
```

Figura 6.28 – Alteração e exclusão através da API REST do Fórum AvaNCE

6.4. Considerações Finais

O AvaNCE é um protótipo de um ambiente de aprendizagem baseado em idéias da Web 2.0 que foi construído com o propósito de verificar a viabilidade do modelo de integração apresentado nesta dissertação. Suas principais características são: a possibilidade de importação de aplicações externas através de suas APIs a partir da interpretação do arquivo WADL contendo sua descrição; a exportação de serviços através de formatos padronizados; e a possibilidade do desenvolvimento de outros clientes que utilizem a sua infra-estrutura com o intuito de incorporar aplicações externas.

Entre as funcionalidades implementadas pelo AvaNCE estão: dois modelos de autenticação de usuários, sendo um deles baseado em OpenID; a possibilidade de criação de cursos, disciplinas, turmas e usuários; uma interface simplificada capaz de permitir o acesso a suas funcionalidades e de interpretar os formatos RSS e XForms que devem ser exibidos para o usuário; e um serviço de fórum de discussão, desenvolvido na forma de uma aplicação interna.

Capítulo 7
AvaNCE – Exemplos de Integração

Para exemplificar o funcionamento do mecanismo de integração proposto nesta dissertação e implementado no ambiente AvaNCE apresentado no Capítulo 6, foram descritas as APIs de duas aplicações Web 2.0 típicas de forma a permitir que as mesmas sejam incorporadas a este ambiente. Além disso, como prova de conceito de que é possível realizar a integração de serviços do AvaNCE em um ambiente externo, foi desenvolvido um módulo capaz de integrar o AvaNCE ao Orkut.

7.1. Exemplos da Integração de Aplicações Externas ao AvaNCE

7.1.1. Del.icio.us

O Del.icio.us é uma aplicação Web 2.0 para armazenamento remoto e compartilhamento de favoritos. Com ele é possível armazenar links favoritos (chamados de *posts*), classificá-los com palavras-chaves criadas livremente pelos usuários (chamadas de *tags*), além de organizar suas *tags* em grupos (chamados de *bundles*).

O Del.icio.us dispõe de uma API muito simples onde é possível executar as suas principais funcionalidades. Assim, através desta API, é possível: listar, adicionar e remover *posts*; listar e renomear *tags*; e listar, criar e remover *bundles*.

Esta API foi analisada segundo a metodologia descrita no Capítulo 4, sendo considerada híbrida quanto ao estilo de arquitetura, pois não apresenta conectividade entre os seus recursos, utiliza URIs associadas aos métodos oferecidos em lugar de recursos, e define um vocabulário próprio de operações, não utilizando uma interface uniforme como preconizado pelo estilo de arquitetura REST. Quanto ao modelo de autenticação, o Del.icio.us utiliza HTTP Basic Authentication sobre HTTPS.

A descrição WADL completa desta API encontra-se no Apêndice D desta dissertação. A partir dela, o Del.icio.us pode ser integrado ao AvaNCE, que passou a oferecer seus recursos indiretamente. Caso algum outro cliente deseje integrar o Del.icio.us utilizando o AvaNCE como camada intermediária, poderá fazê-lo utilizando para isso as requisições correspondentes definidas na Figura 7.1.

Esta solução permite que o cliente obtenha uma relação com todos os recursos/métodos da API, além de obter a relação de parâmetros para cada método. Para isso o AvaNCE oferece algumas requisições adicionais, não disponíveis na API original.

	Delicious API original		**Delicious API através do AvaNCE**
	(Índice da aplicação)		
GET	https://api.del.icio.us/v1/delicious.wadl	GET	/executor/delicious – RSS com lista de métodos e links da aplicação.
	(Posts)		
GET	https://api.del.icio.us/v1/posts/get	GET	/executor/delicious/v1/posts/get/form
		POST	/executor/delicious/v1/posts/get
GET	https://api.del.icio.us/v1/posts/recent	GET	/executor/delicious/v1/posts/recent/form
		POST	/executor/delicious/v1/posts/recent
GET	https://api.del.icio.us/v1/posts/all	GET	/executor/delicious/v1/posts/all/form
		POST	/executor/delicious/v1/posts/all
GET	https://api.del.icio.us/v1/posts/dates	GET	/executor/delicious/v1/posts/dates/form
		POST	/executor/delicious/v1/posts/dates
GET	https://api.del.icio.us/v1/posts/update	GET	/executor/delicious/v1/posts/update/form
		POST	/executor/delicious/v1/posts/update
GET	https://api.del.icio.us/v1/posts/add	GET	/executor/delicious/v1/posts/add/form
		POST	/executor/delicious/v1/posts/add
GET	https://api.del.icio.us/v1/posts/delete	GET	/executor/delicious/v1/posts/delete/form
		POST	/executor/delicious/v1/posts/delete
	(Tags)		
GET	https://api.del.icio.us/v1/tags/get	GET	/executor/delicious/v1/tags/get/form
		POST	/executor/delicious/v1/tags/get
GET	https://api.del.icio.us/v1/tags/rename	GET	/executor/delicious/v1/tags/rename/form
		POST	/executor/delicious/v1/tags/rename
	(Bundles)		
GET	https://api.del.icio.us/v1/tags/bundles/all	GET	/executor/delicious/v1/tags/bundles/all/form
		POST	/executor/delicious/v1/tags/bundles/all
GET	https://api.del.icio.us/v1/tags/bundles/set	GET	/executor/delicious/v1/tags/bundles/set/form
		POST	/executor/delicious/v1/tags/bundles/set
GET	https://api.del.icio.us/v1/tags/bundles/delete	GET	/executor/delicious/v1/tags/bundles/delete/form
		POST	/executor/delicious/v1/tags/bundles/delete

Figura 7.1 – Requisições da API do Del.icio.us realizadas através do AvaNCE

Do ponto de vista do usuário, uma vez que a aplicação seja associada a uma turma, seus recursos passam a ser oferecidos para seus participantes através de uma lista de links na página da própria turma (Figura 7.2). As Figuras a seguir exemplificam o processo de execução do método *Posts/All* que retorna todos os *Posts* de um usuário com uma determinada tag fornecida. Uma vez que o usuário selecione esta opção no menu, o AvaNCE executa o respectivo método, primeiramente solicitando ao usuário os parâmetros necessários para sua execução (Figura 7.3), posteriormente solicitando a autenticação na aplicação externa (Figura 7.4), e finalmente produzindo a resposta da execução solicitada (Figura 7.5).

Neste exemplo, os resultados são apresentados na forma de uma tabela onde cada registro retornado pela consulta aparece numa linha. As ligações entre as respostas produzidas (que foram definidas através de links no arquivo WADL) são representadas nesta tela através de botões que

permitem ao usuário a navegação para outros recursos como aconteceria numa API aderente ao estilo REST.

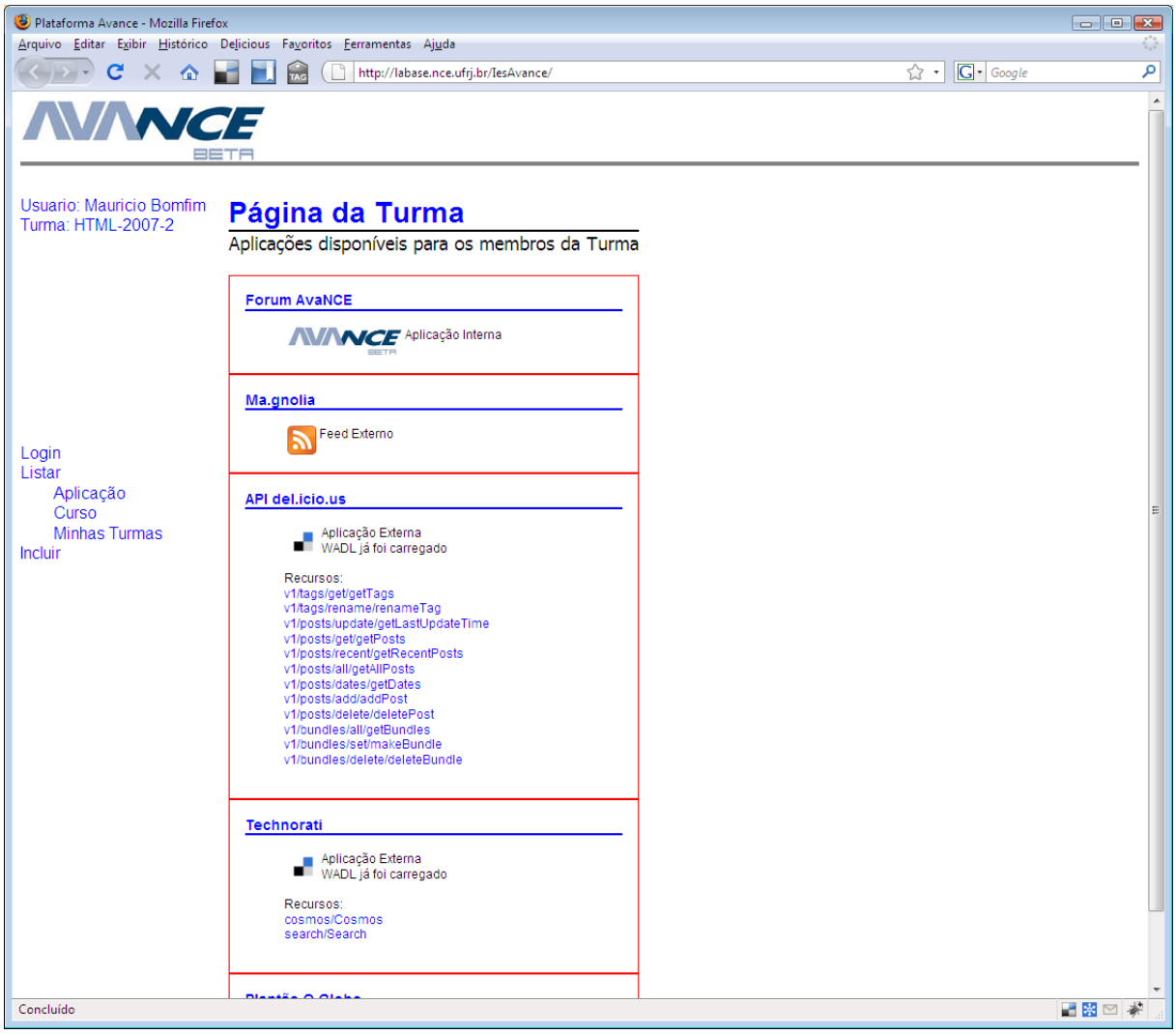

Figura 7.2 – Página de uma turma com menu de funcionalidades das aplicações externas

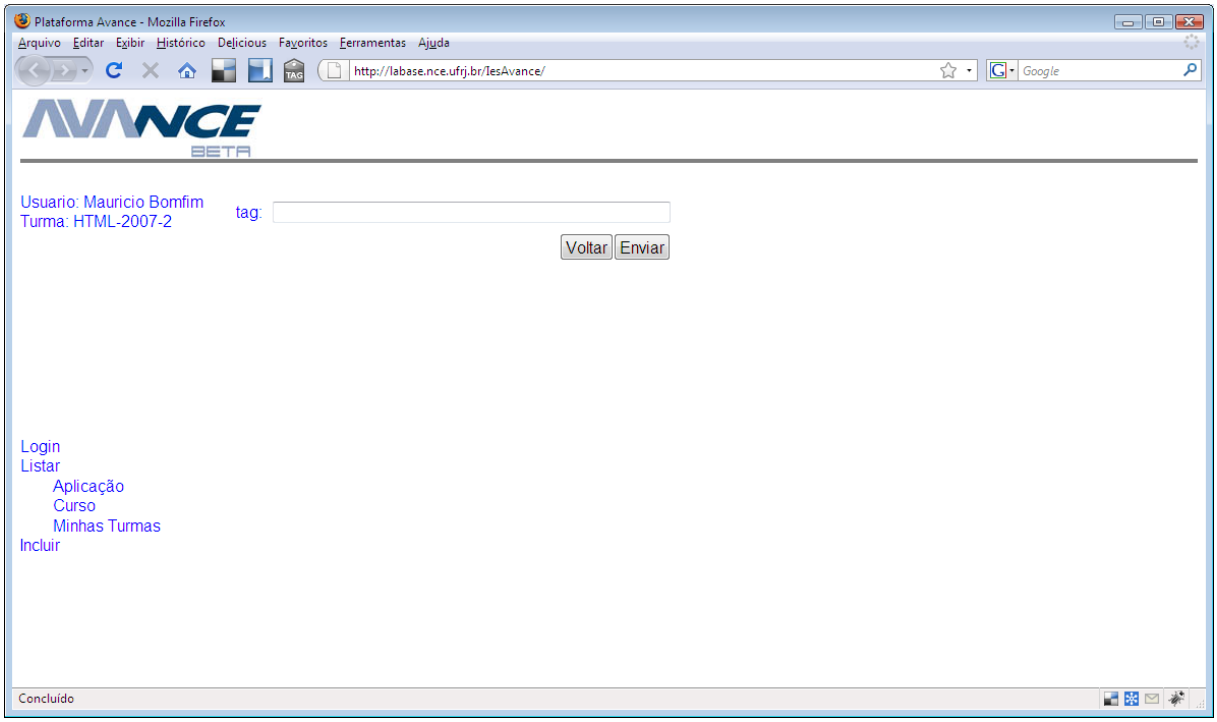

Figura 7.3 – Solicitação de parâmetro para a execução do método *posts/all* do Del.icio.us

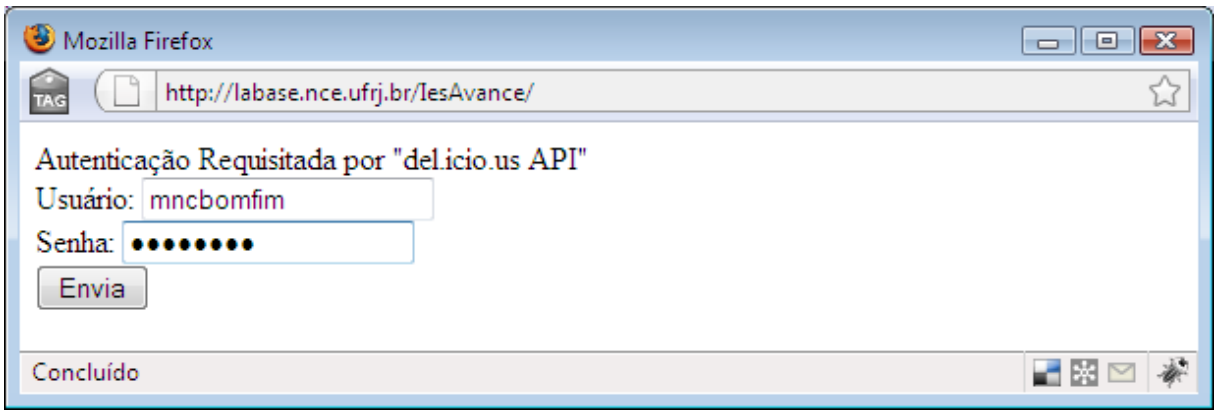

Figura 7.4 – Solicitação de autenticação pelo Del.icio.us

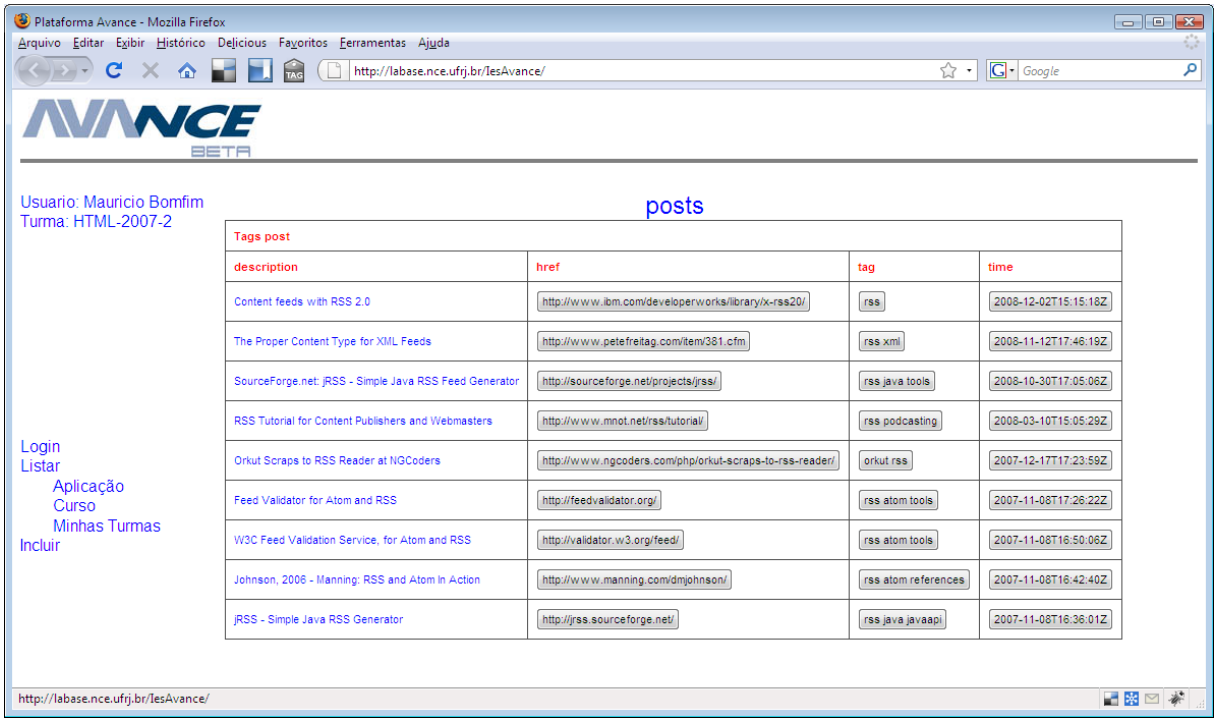

Figura 7.5 – Resultado da execução do método *posts/all* do Del.icio.us

7.1.2. Technorati

O Technorati é um serviço de busca em blogs, fotos e vídeos que oferece uma API onde suas funcionalidades são expostas para qualquer aplicação que deseje consumi-las. Como as consultas realizadas por esta aplicação são públicas, não existe nenhum processo de autenticação para utilizá-las.

A exemplo da Figura 7.1 onde foram apresentados os serviços do Del.icio.us e sua correspondência com as mesmas requisições realizadas através do AvaNCE, a Figura 7.6 apresenta a relação de serviços oferecidos pela API do Technorati. Através desta solução, uma aplicação que deseje integrar o Technorati através do AvaNCE pode utilizar-se de algumas requisições adicionais, não disponíveis na API original, onde é possível obter uma relação com todos os recursos/métodos da API, além de obter a relação de parâmetros para cada método.

	Technorati API original		**Technorati API através do AvaNCE**
	(Índice da aplicação)		
GET	http://api.technorati.com /technorati.wadl	GET	/executor/technorati – RSS com lista de métodos e links da aplicação.
	(relação de métodos)		
GET	http://api.technorati.com/cosmos	GET	/executor/technorati/cosmos/form
		POST	/executor/technorati/cosmos
GET	http://api.technorati.com/search	GET	/executor/technorati/search/form
		POST	/executor/technorati/search
GET	http://api.technorati.com/tag	GET	/executor/technorati/tag/form
		POST	/executor/technorati/tag
GET	http://api.technorati.com/dailycounts	GET	/executor/technorati/dailycounts/form
		POST	/executor/technorati/dailycounts
GET	http://api.technorati.com/toptags	GET	/executor/technorati/toptags/form
		POST	/executor/technorati/toptags
GET	http://api.technorati.com/bloginfo	GET	/executor/technorati/bloginfo/form
		POST	/executor/technorati/bloginfo
GET	http://api.technorati.com/blogposttags	GET	/executor/technorati/blogposttags/form
		POST	/executor/technorati/blogposttags
GET	http://api.technorati.com/getinfo	GET	/executor/technorati/getinfo/form
		POST	/executor/technorati/getinfo

Figura 7.6 – Requisições da API do Technorati realizadas através do AvaNCE

A descrição WADL desta API se encontra no Apêndice E desta dissertação. Para efeito deste exemplo, apenas dois métodos da API foram descritos. O primeiro procura por blogs que apontam para uma URL dada (*cosmos*), enquanto o segundo procura por blogs que contenham as palavras fornecidas pelo usuário (*search*).

Partindo da página da turma (apresentada na Figura 7.2) o usuário pode selecionar uma funcionalidade da aplicação. O AvaNCE então executa o respectivo método, primeiramente solicitando ao usuário os parâmetros necessários para sua execução (Figura 7.7) e, posteriormente, produzindo a resposta da execução solicitada (Figuras 7.8 e 7.9).

Como o formato do XML produzido como resposta desta requisição é mais complexo (contendo vários níveis de tags aninhadas), sua representação através de uma tabela simples como no caso do Delicious, não é tão simples de ser obtida. Sendo assim, a apresentação dos resultados realiza-se num formato alternativo onde a estrutura do XML recebido é reproduzida graficamente através de tabelas aninhadas.

É possível perceber ainda que as requisições, em geral, retornam muito mais informações do que o usuário normalmente esperaria receber como resposta. O processo de configuração poderia, nestes casos, permitir que o usuário estabelecesse filtros que omitissem as informações

irrelevantes produzindo uma tela de resposta mais limpa e de acordo com as expectativas reais dos usuários.

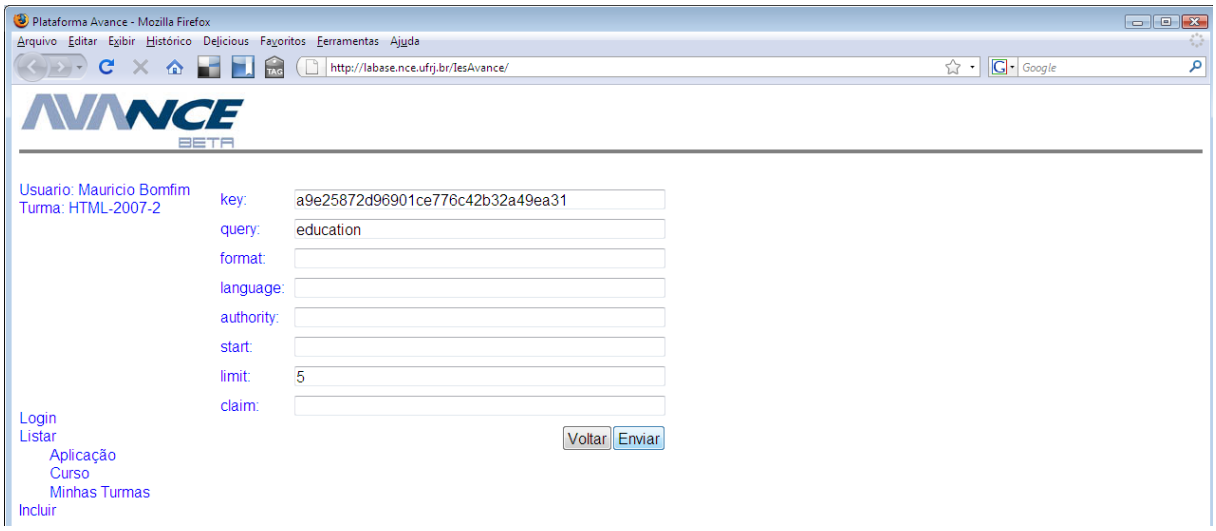

Figura 7.7 – Solicitação de parâmetros para a execução do método *search* do Technorati

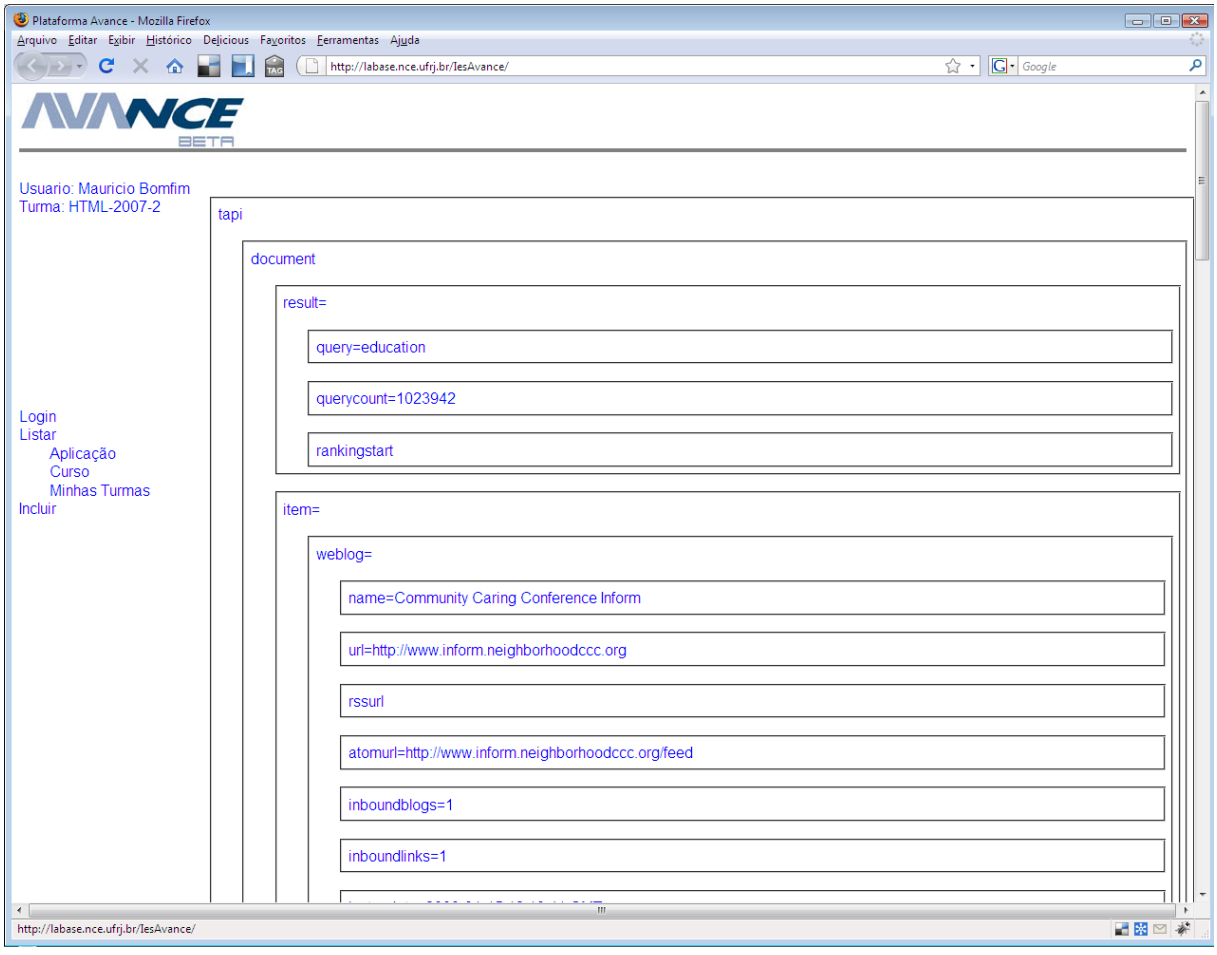

Figura 7.8 – Resultado da execução do método *search* do Technorati

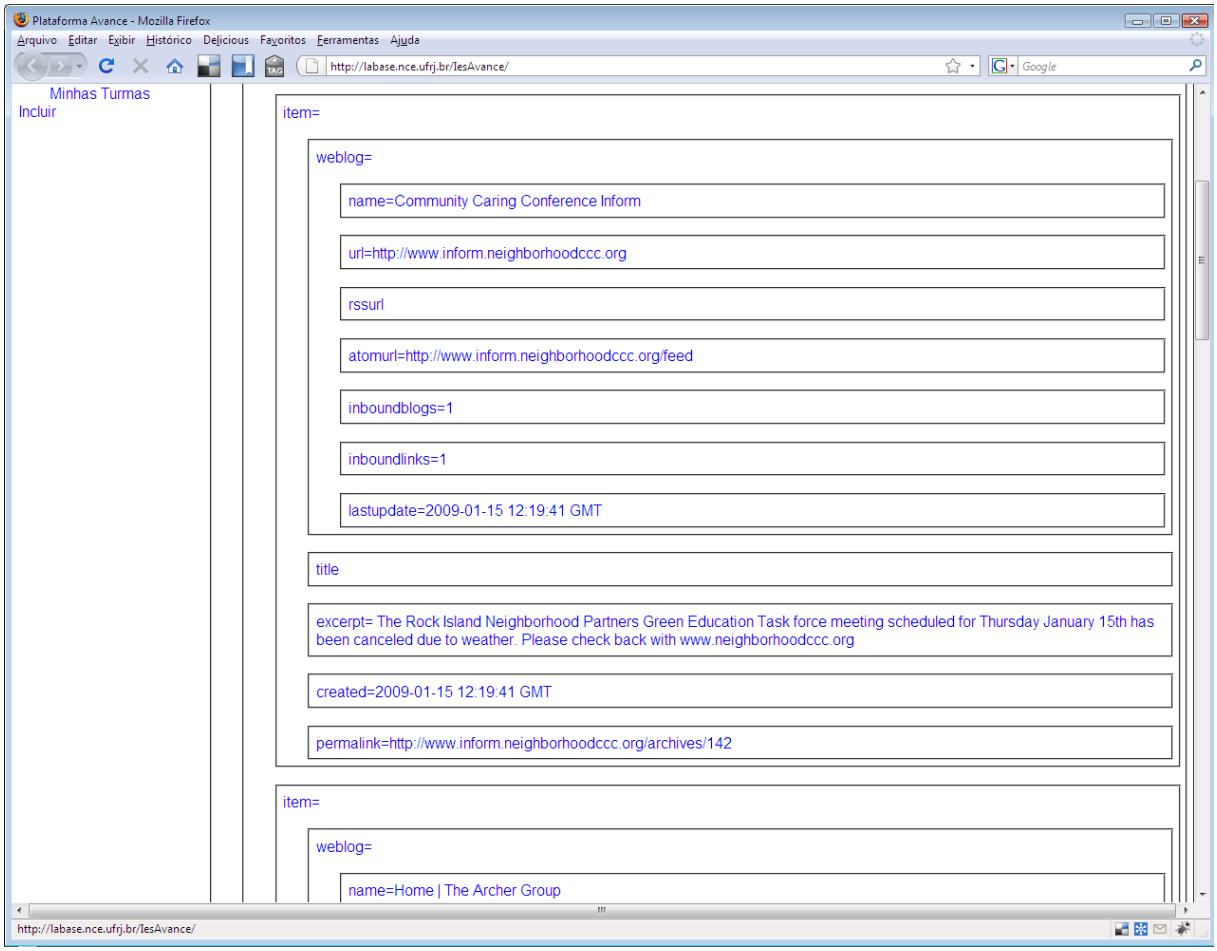

Figura 7.9 – Resultado da execução do método *search* do Technorati (continuação)

7.2. Exemplos de Integração do AvaNCE em outros Ambientes

Para exemplificar a possibilidade de integração dos serviços do AvaNCE em outros ambientes, foi desenvolvido um módulo capaz de ser integrado em uma rede social, utilizando para isso as APIs OpenSocial[86] e Google Gadgets[87]. Estas APIs permitem a criação de módulos, compostos de alguns blocos estruturais simples onde é possível combinar códigos XML, HTML e Javascript, que podem ser executados em diversos recipientes como Orkut[88], MySpace[89], LinkedIn[90] e Hi5[91].

[86] http://code.google.com/apis/opensocial/
[87] http://code.google.com/intl/pt-BR/apis/gadgets/
[88] http://orkut.com
[89] http://www.myspace.com
[90] http://www.linkedin.com
[91] http://hi5.com

No teste aqui realizado, o recipiente utilizado foi o Orkut, para o qual foram desenvolvidos dois módulos de acesso ao AvaNCE. A inclusão destes módulos como aplicações no Orkut é ilustrada nas Figuras 7.10 e 7.11.

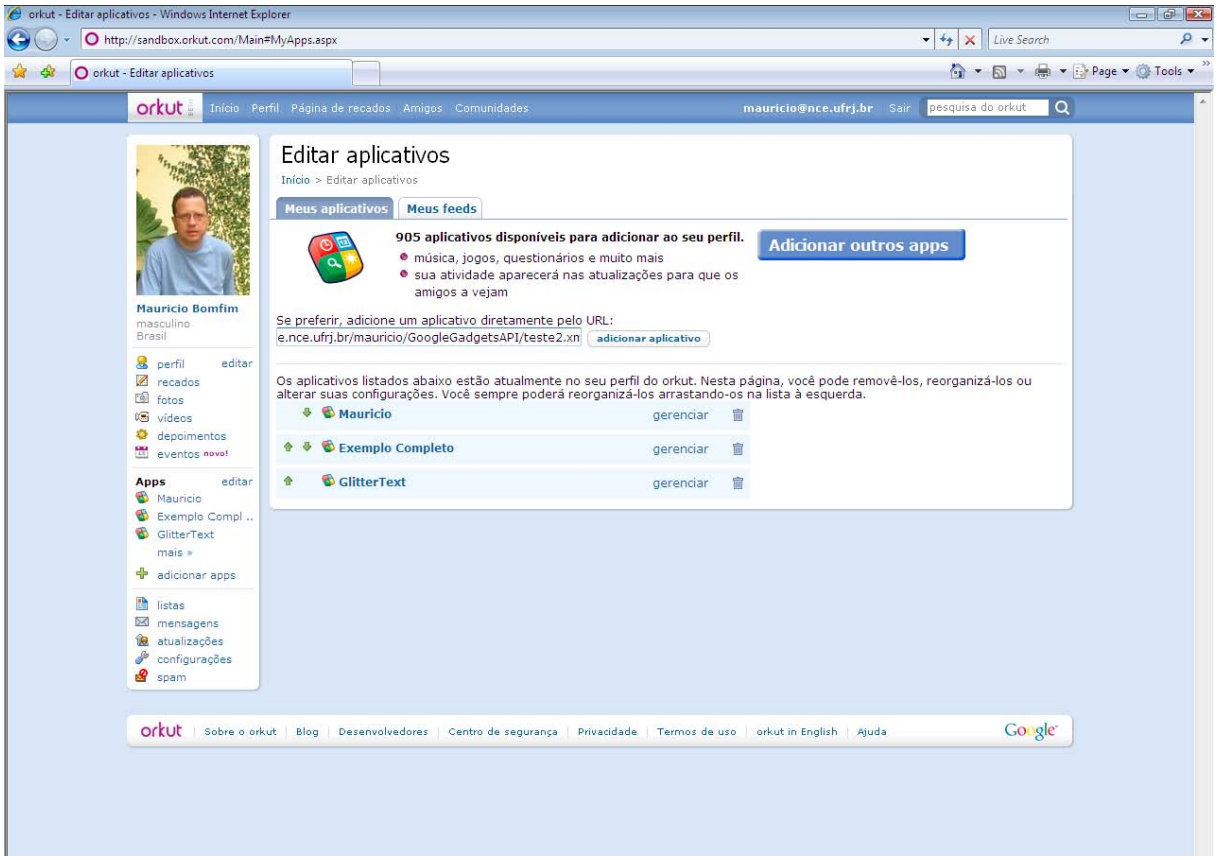

Figura 7.10 – Adicionando um aplicativo ao Orkut

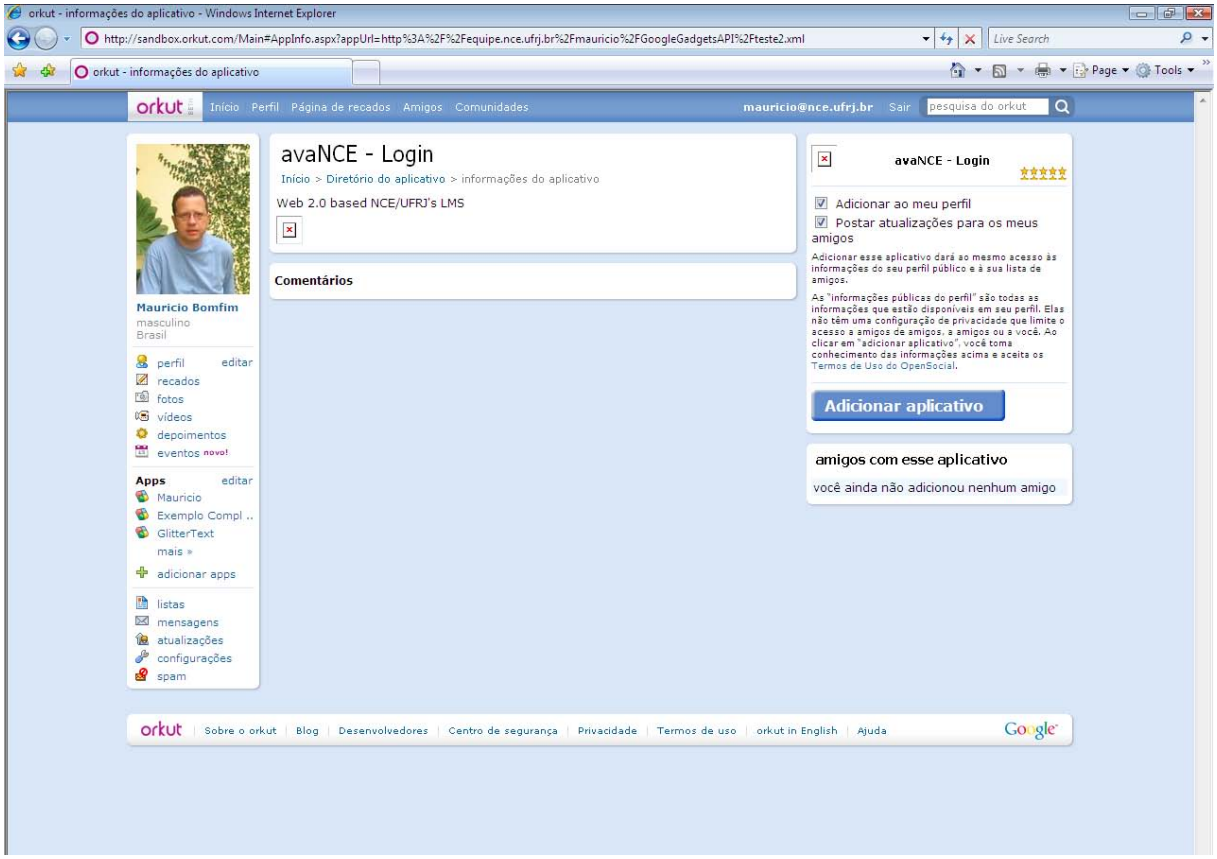

Figura 7.11 – Adicionando um aplicativo ao Orkut

O teste da integração do AvaNCE neste ambiente, pode ser realizado utilizando duas abordagens diferentes. Na primeira, foi criado um módulo que encapsula o formulário de *login* (Figura 7.12). Uma vez que o usuário preencha e envie este formulário, o AvaNCE vai validar as informações fornecidas, iniciando uma sessão e produzindo o arquivo RSS com a relação de turmas do usuário. A partir daí, a navegação na aplicação ocorre naturalmente através dos links existentes nos próprios arquivos RSS gerados pelo AvaNCE. Entretanto, existem dois problemas com esta solução: a apresentação da listagem do RSS não pode ser personalizada, pois estes arquivos são nativamente interpretados pelo navegador e apresentados de maneira padrão (Figura 7.13). Além disso, como os navegadores atuais ainda não são capazes de reconhecer e interpretar o formato XForms, é inviável a execução de funcionalidades que produzam formulários esperando que o usuário forneça parâmetros de entrada.

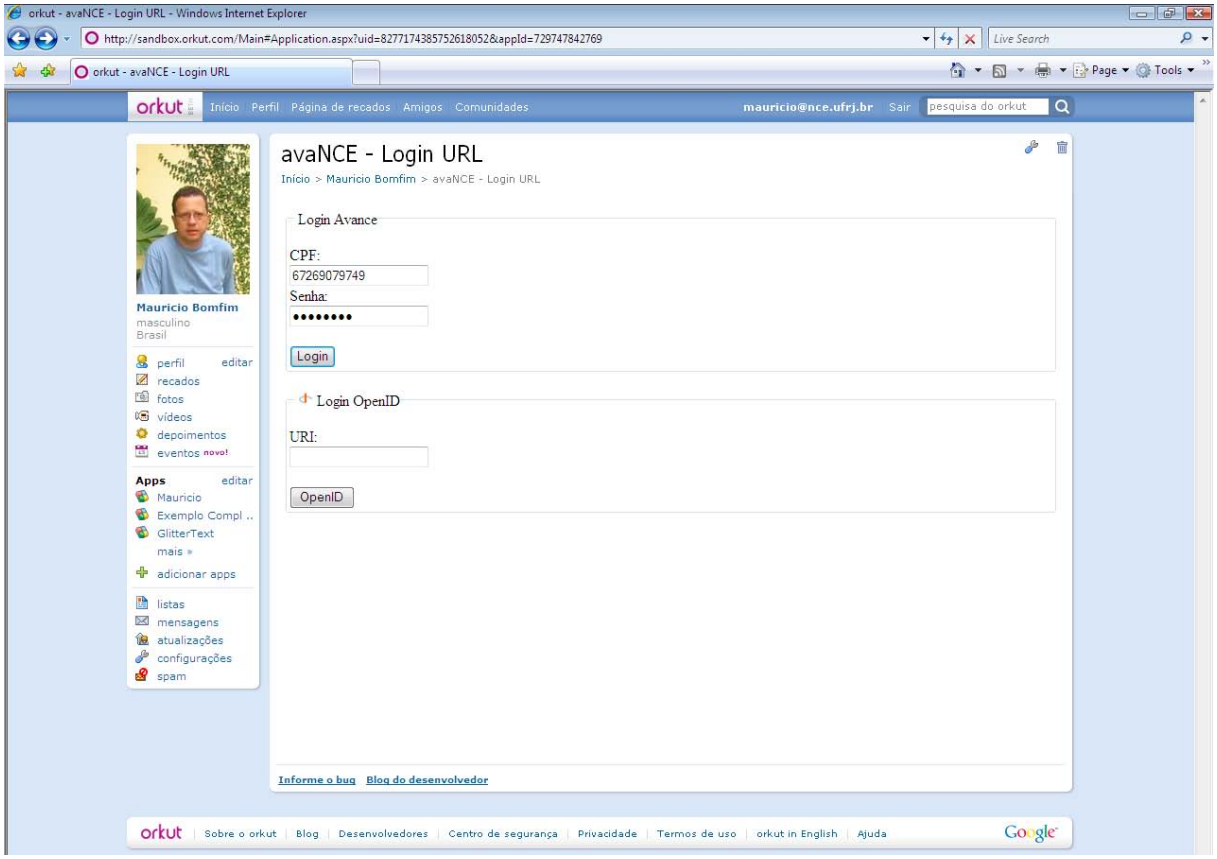

Figura 7.12 – Fornecendo identificação no AvaNCE

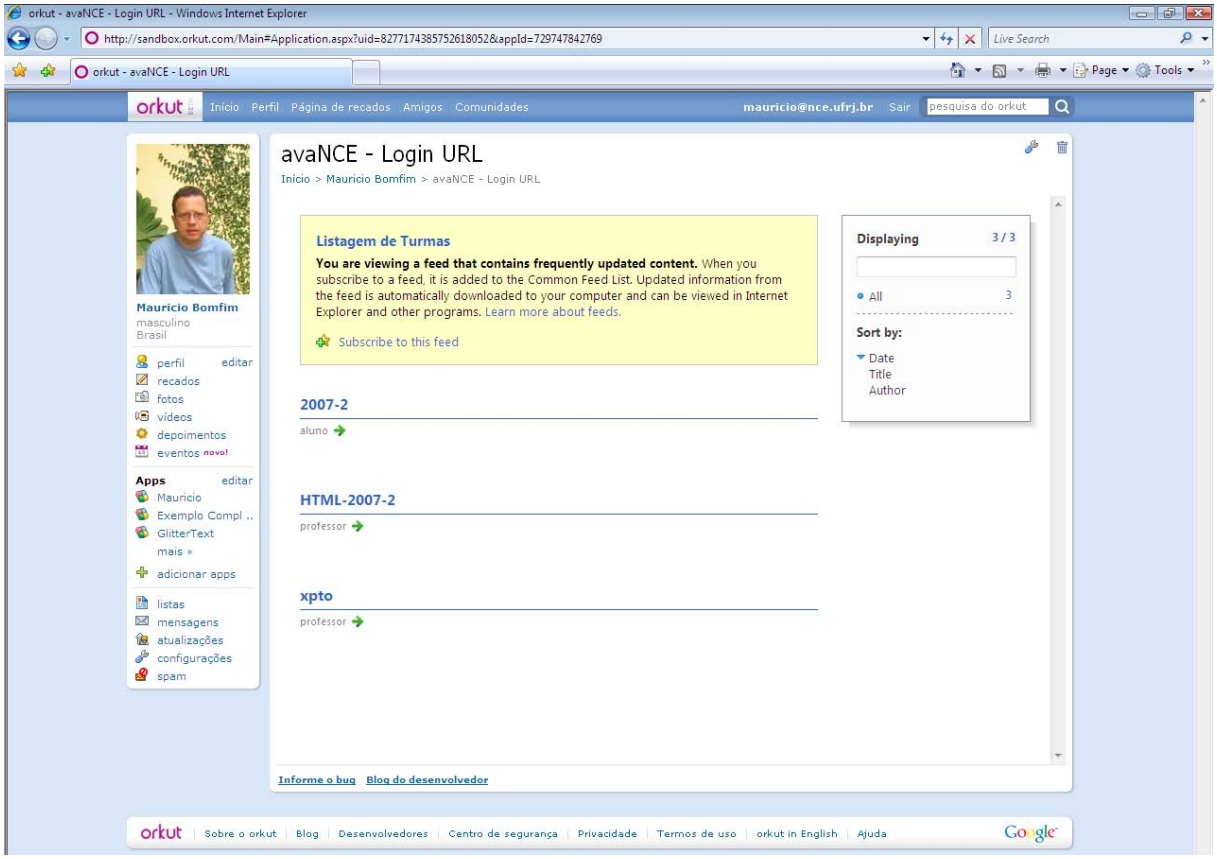

Figura 7.13 – RSS com listagem de turmas (interpretado pelo navegador através do Orkut)

Na segunda abordagem, foi utilizada uma solução mais completa que consiste em embutir no próprio módulo uma programação na linguagem Javascript que seja capaz de interpretar os formatos RSS e XForms. Neste caso, o módulo cliente recebe cpf e senha como preferências da aplicação (Figura 7.14) fazendo uma chamada REST ao AvaNCE para iniciar a sessão. Ao receber as respostas RSS ou XForms, o módulo deve interpretá-los produzindo uma apresentação dos mesmos através de formatação definida numa folha de estilos (Figura 7.15). A navegação na aplicação prossegue, da mesma forma que no teste anterior, através dos links existentes nos próprios arquivos gerados pelo AvaNCE.

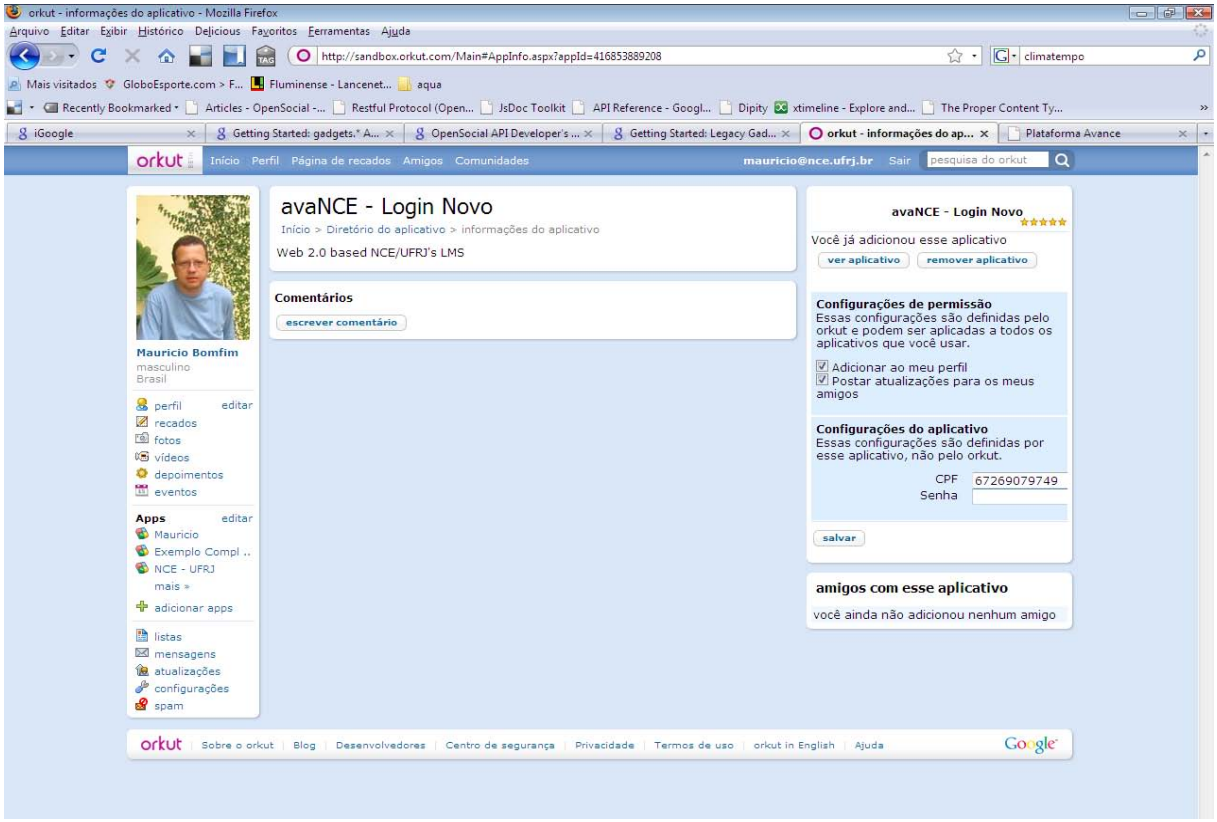

Figura 7.14 – Fornecendo identificação no AvaNCE através de preferências da aplicação

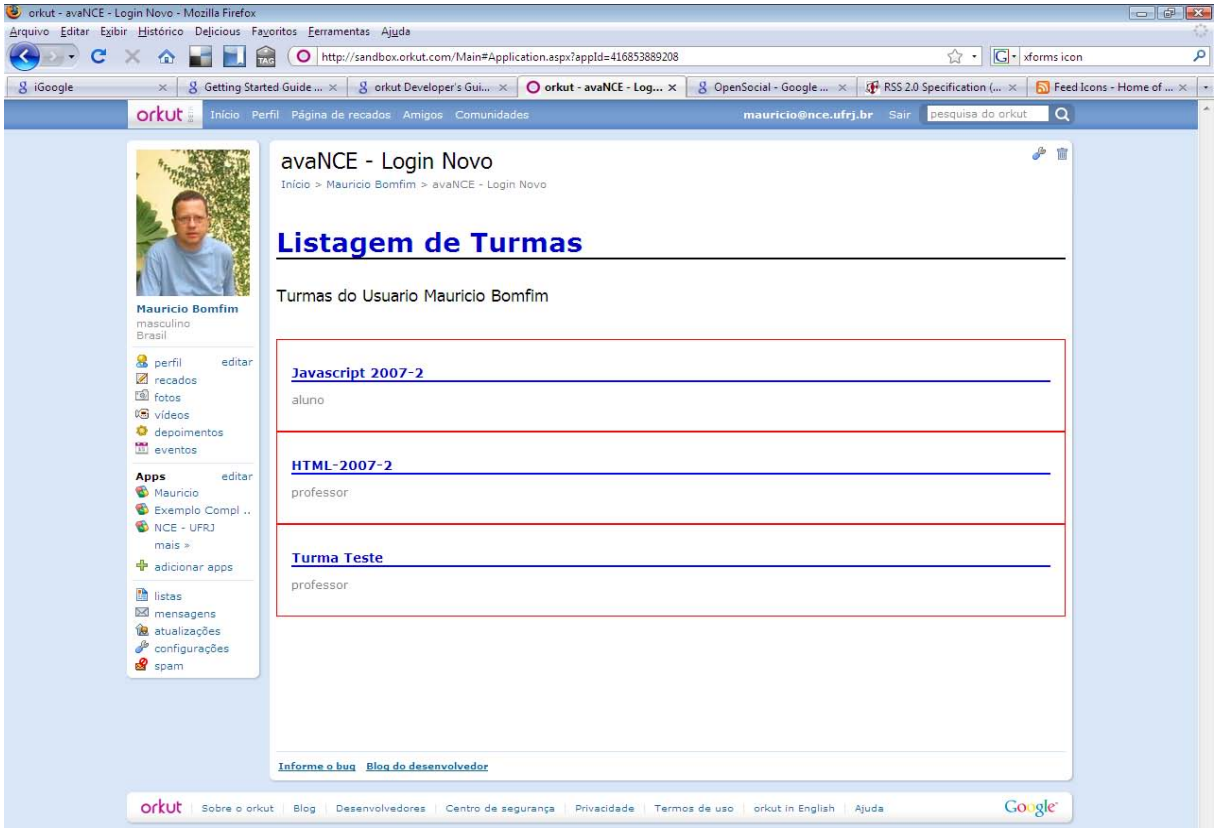

Figura 7.15 – RSS com listagem de turmas (interpretado pelo módulo desenvolvido)

7.3. Considerações Finais

O AvaNCE é um ambiente de aprendizagem que oferece ao usuário a possibilidade de incorporar automaticamente, qualquer aplicação externa que possua uma API aberta descrita formalmente com WADL Além disso, ele oferece ainda a possibilidade de exportar suas aplicações para outros ambientes.

Uma prova de conceito é realizada com o intuito de demonstrar a viabilidade da integração de aplicações nestes dois sentidos, onde são apresentados a integração de duas aplicações externas (o Del.icio.us e o Technorati) ao AvaNCE e, a integração dos serviços do AvaNCE a um ambiente externo (o Orkut). Para isso foi necessário descrever as APIs do Del.icio.us e do Technorati em WADL e desenvolver um módulo do Orkut capaz de interpretar os formatos RSS e XForms utilizados pelo AvaNCE.

Capítulo 8
Considerações Finais e Trabalhos Futuros

Este capítulo apresenta as conclusões desta dissertação, identificando suas contribuições, e os trabalhos futuros com vistas ao prosseguimento desta pesquisa.

8.1. Resumo do Trabalho

A incorporação das idéias da Web 2.0 nos Ambientes Virtuais de Aprendizagem tradicionais parece ser um caminho a ser seguido. A utilização desta abordagem, além de permitir ganhos no processo de aprendizado, uma vez que vai ao encontro das idéias construtivistas e sócio-interacionistas, oferece aos aprendizes a oportunidade de desempenhar um papel mais ativo no processo de aprendizagem, contemplando também as necessidades do aprendizado informal.

Nestes novos ambientes espera-se que o aluno possua uma liberdade maior, permitindo a escolha e a incorporação de conteúdos e funcionalidades de diferentes ferramentas e serviços oferecidos pela Web, sejam eles provenientes de provedores de educação formal ou de fontes informais de conhecimento. Além disso, a possibilidade de exportar seus próprios serviços para outras aplicações pode fazer com que a convivência entre diferentes ambientes seja facilitada.

Acreditando na hipótese de que a definição de uma abordagem genérica de integração automática de aplicações Web 2.0, a partir de suas APIs, pode facilitar a criação de um ambiente de aprendizagem onde os alunos sejam capazes de escolher suas próprias ferramentas, este trabalho propõe um modelo de integração que viabiliza a criação de um ambiente que seja de fácil utilização por usuários não técnicos, e que permita a incorporação de qualquer aplicação escolhida livremente.

Um estilo de arquitetura que vem cada vez mais se tornando uma tendência no desenvolvimento de serviços e aplicações na Web 2.0 e que poderia facilitar a integração automática de serviços é o REST. Entretanto as aplicações existentes não seguem todos os preceitos desta arquitetura. Esta constatação foi um dos resultados do estudo exploratório, comparando algumas APIs existentes, realizado no escopo deste trabalho. Para isso foi criada uma rede sistêmica organizando os principais aspectos relacionados à componentização das aplicações da Web 2.0, através da qual foi possível a classificação destas APIs de acordo com o grau de adesão de cada uma delas à arquitetura REST e seus mecanismos de autenticação. As conclusões apresentadas por este estudo indicam que a maior parte das aplicações existentes não adere totalmente ao REST, dificultando a definição de uma abordagem genérica que pudesse ter utilização prática.

O modelo de integração proposto para a solução deste problema é baseado numa camada intermediária, que permite a incorporação automática de aplicações Web 2.0, a partir da descrição formal de suas APIs utilizando para isso a linguagem WADL. Este modelo é apresentado primeiramente de maneira genérica, onde diferentes clientes podem ser integrados com as

aplicações Web 2.0. Posteriormente discute-se como um ambiente de aprendizagem pode ser desenvolvido utilizando estas idéias, de forma que o usuário seja capaz de integrar aplicações escolhidas livremente de maneira simples e fácil.

Para demonstrar a viabilidade da aplicação desta solução, foi desenvolvido o AvaNCE - um protótipo do ambiente proposto - que oferece ao usuário a possibilidade de incorporar automaticamente, qualquer aplicação externa que possua uma API aberta descrita formalmente com WADL Além disso, ele oferece ainda a possibilidade de exportar suas aplicações para outros ambientes.

Uma prova de conceito foi realizada através da integração de duas aplicações externas (o Del.icio.us e o Technorati) ao AvaNCE e, a da integração dos serviços do AvaNCE a um ambiente externo (o Orkut). Para isso foi necessário descrever as APIs do Del.icio.us e do Technorati em WADL e desenvolver um módulo do Orkut capaz de interpretar os formatos RSS e XForms utilizados pelo AvaNCE.

8.2. Contribuições desta Dissertação

A principal contribuição deste trabalho é a definição de uma arquitetura capaz de facilitar a integração automática entre aplicações Web 2.0, baseada na descrição formal das APIs das aplicações a serem integradas através da linguagem WADL. A aplicação desta arquitetura a um ambiente de aprendizagem que permita incorporar aplicações externas, assim como exportar seus próprios serviços, resultou na definição da proposta apresentada no Capítulo 5.

A revisão da literatura apresentada nos Capítulos 2 e 3 sistematiza, apesar da pequena documentação existente até o momento sobre alguns assuntos abordados, o estado da arte das tecnologias envolvidas no desenvolvimento de aplicações para a Web 2.0. Além disso, estes capítulos também realizam uma análise crítica da utilização dos Ambientes Pessoais de Aprendizagem e do uso de novas tecnologias no apoio à educação.

Outra contribuição foi a publicação de um relatório técnico (BOMFIM e SAMPAIO, 2008) e de dois trabalhos em congressos: na Conferência Ibero-americana WWW/Internet, 2007 (BOMFIM, ASSIS e SAMPAIO, 2007) e no XIX Simpósio Brasileiro de Informática na Educação – SBIE (BOMFIM, SAMPAIO e ASSIS, 2008).

A Rede Sistêmica construída para análise das APIs das aplicações da Web 2.0, quanto à aderência aos preceitos do estilo de arquitetura REST e quanto aos modelos de autenticação utilizados, é

mais uma contribuição desta dissertação. Além disso, os resultados do estudo exploratório realizado a partir da utilização desta rede sistêmica para a classificação de APIs existentes comprovam a existência de uma tendência errônea de chamar de REST qualquer API baseada em HTTP, independente de todos os critérios que definem este estilo de arquitetura serem satisfeitos.

Quanto ao desenvolvimento do protótipo de um ambiente que facilite a integração de novas ferramentas dentro da UFRJ (AvaNCE), espera-se que a sua evolução possa levar a um produto que seja utilizável em situações reais de ensino conforme descrito na seção a seguir.

8.3. Trabalhos Futuros

A principal linha de evolução desta pesquisa deve ser no sentido de aprofundar o estudo dos processos de configuração das aplicações integradas. É através deles que a integração automática pode ser de grande valia para o usuário não técnico. Conforme discutido na Seção 5.2.5.3, uma solução simplificada para este problema seria a implementação de um mecanismo através do qual o usuário pudesse selecionar os métodos e parâmetros da aplicação integrada que deseje utilizar.

É possível ainda permitir a configuração do formato de apresentação das informações retornadas pelas aplicações integradas de forma que o usuário definisse um modelo que pudesse ser utilizado na conversão do retorno XML para a página de resposta produzida. Uma maneira de implementar esta idéia poderia, por exemplo, basear-se no trabalho de Borges (2002) onde uma interface gráfica permite que o usuário forneça um documento exemplo que, juntamente com o documento XML original sejam utilizados para produzir um conjunto de regras XSLT capazes de fazer a conversão desejada.

Além da configuração das aplicações, o protótipo AvaNCE precisa ainda evoluir para um produto cuja utilização prática possa ser avaliada em situações reais de ensino. Para isso é necessário:

- Melhorar a interface;
- Implementar outros métodos de autenticação como, OAuth, Google Authsub e FlickRAuth, de forma a permitir a integração de um maior número de aplicações diferentes;

- Implementar os casos de uso definidos no Capítulo 5, que permitem que o ambiente seja utilizado como um APA, envolvendo a utilização de comunidades;

- Implementar a conversão de tipos entre os formatos de saída.

Outros aspectos que podem ser considerados referem-se ao uso desta ferramenta dentro da UFRJ, permitindo sua integração com bancos de dados institucionais de gestão acadêmica, centralizando e, conseqüentemente facilitando, o acesso da comunidade a estas informações.

A existência de uma plataforma baseada nestas idéias, desenvolvida na própria UFRJ, pode permitir a incorporação de novas ferramentas a partir de outros projetos que explorem aspectos específicos do processo de aprendizado apoiado pelo computador. Assim, esta plataforma poderá funcionar como um laboratório de pesquisa, fazendo com que futuros trabalhos possibilitem o desenvolvimento de novos módulos, que possam ser incorporados à versão da plataforma, em uso pela própria comunidade da UFRJ, transformando-se num programa de pesquisa-ação a distância conforme proposto por Elia e Sampaio (2001).

Em 2006, o GINAPE criou um grupo de estudos com a finalidade de pesquisar a possibilidade do desenvolvimento de ferramentas educacionais apoiadas na Web 2.0. Esta dissertação se insere neste contexto, apresentando resultados que serão de grande valia para trabalhos subseqüentes. No momento, existem mais cinco projetos em andamento, estudando temas como: sistemas de combinação social (SILVA et al, 2008; SILVA, 2009), sistemas de recomendação (ZANETTI, 2008), reputação em comunidades de prática (CRUZ, 2008; CRUZ et al, 2008) e, a definição de diferentes modelos de avaliação educacional (GONÇALVES e ELIA, 2008). Como resultado destas pesquisas, será produzido um conjunto de serviços educacionais, que poderão no futuro, ser integrados ao AvaNCE.

Outra iniciativa, é o Ambiente Interativo para o Trabalho Integrado e Virtual (ActivUFRJ), um portal de comunidades virtuais inicialmente desenvolvido num ambiente proprietário (Microsoft), que está sendo portado de acordo com as tecnologias abertas da Web 2.0. Seu objetivo principal é facilitar a formação e a manutenção de comunidades de interesses na UFRJ, proporcionando o encontro entre grupos heterogêneos de pessoas que possam ter interesse pelos mesmos temas dentro da universidade (HILDENBRAND, 2006). O ActivUFRJ, a exemplo do AvaNCE, também é um ambiente central, capaz de integrar o desenvolvimento de diversas pesquisas relacionadas. A utilização da arquitetura de integração proposta por esta dissertação também favorecerá a interoperabilidade entre estes dois ambientes.

Por fim, questões relativas às implicações pedagógicas da utilização deste ambiente como um APA também ficam em aberto, podendo ser alvo de futuras investigações que tentem comprovar a eficácia destas novas abordagens para o aprendizado.

Referências Bibliográficas

ALEXANDER, B. Web 2.0: A new wave of innovation for teaching and learning? **Educase Review**. Boulder, v. 41, n. 2, p. 32-44, Mar./Apr. 2006. Disponível em: http://net.educause.edu/ir/library/pdf/ERM0621.pdf. Acesso em: jul. 2008.

ANDERSON, P. What is Web 2.0? ideas, technologies and implications for education**. JISC** Technology and Standards Watch, Feb. 2007. Disponível em: <http://www.jisc.ac.uk/media/documents/techwatch/tsw0701b.pdf >. Acesso em: ago. 2007.

ASLEESON, R. ; SCHUTTA, N. **Foundations of Ajax.** Berkeley: Apress, 2006. 273p.

ASPIN, D. ; CHAPMAN, J. Lifelong learning: concepts, theories and values. In: ANNUAL CONFERENCE OF SCUTREA, 31., 2001. London. **Proceedings …** London: University of East London, 2001. . Disponível em: http://www.leeds.ac.uk/educol/documents/00002564.doc. Acesso em: set. 2008.

ATTWELL, G. Personal learning environments – the future of eLearning? **eLearning Papers**, Barcelona, v.2, n.1, Jan. 2007. ISSN 1887-1542. Disponível em: http://www.elearningeuropa.info/files/media/media11561.pdf. Acesso em: out. 2007.

ATWOOD, M. et al. (OAuth Core Workgroup). **OAuth Core 1.0.** specification. 04/12/2007. Disponível em: http://oauth.net/core/1.0. Acesso em: jul. 2008.

BALTAZAR, N. ; AGUADED, I. Weblogs como recurso tecnológico numa nova educação. In: CONGRESSO DA ASSOCIAÇÃO PORTUGUESA DE CIÊNCIAS DA COMUNICAÇÃO,4. 2005, Aveiro. **Anais ...** Aveiro: Universidadede de Aveiro, 2005.

BERNERS-LEE, T.; MASINTER, L. ; McCAHILL, M. **Uniform resource locators (URL**). Dec. 1994. Disponível em: http://www.ietf.org/rfc/rfc1738.txt Acesso em: dez. 2007.

BERNERS-LEE, T. ; FIELDING, R. ; MASINTER, L. **Uniform resource identifiers (URI)**: Generic Syntax, RFC 3986. Jan. 2005. Disponível em: ftp://ftp.isi.edu/in-notes/rfc3986.txt Acesso: em dez. 2007.

BLISS, J. ; MONK, M. ; OGBORN, J. **Qualitative data analysis for educational research:** a guide to uses of systemic networks. London: Croom Helm, 1983.

BOMFIM, M. N. C. ; ASSIS, J. ; SAMPAIO, F. F. Avance: um ambiente de ensino e aprendizagem baseado na web 2.0. In: CONFERÊNCIA IBERO-AMERICANA WWW/INTERNET, 2007, Vila Real. **Proceedings...** Vila Real: Universidade Trás dos Montes e Alto Douro, 2007. p. 262-265. ISBN: 978-972-8924-43-0.

BOMFIM, M. N. C. ; SAMPAIO, F. F. ; ASSIS, J. Avance: ambiente pessoal de aprendizagem na web 2.0. SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 19., 2008, Fortaleza. **Anais...** Fortaleza: SBC, 2008.

BOMFIM, M. N. C. ; SAMPAIO, F. F. **A Web 2.0, suas tecnologias e aplicações educacionais**. Rio de Janeiro, NCE/UFRJ, 2008. (Relatório Técnico NCE 02/08).

BOOTH, D. et al. **Web services architecture**. W3C Working Group Note. 11 Feb. 2004. Disponível em: http://www.w3.org/TR/ws-arch/. Acesso em: dez. 2007.

BOOTH, D. ; LIU, C. K. **Web services description language (WSDL) version 2.0 part 0**: Primer. W3C Recommendation. 26 June 2007. Disponível em: http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/. Acesso em: dez. 2007.

BORGES, R. T. **Apresentação de conteúdos XML através de exemplos**. 2002. 77 p. Dissertação (Mestrado em Computação) – PPGC, Universidade Federal do Rio Grande do Sul, , Porto Alegre, 2002.

BOYER, J. **XForms 1.1 w3c candidate recommendation.** 29 November 2007. Disponível em: http://www.w3.org/TR/2007/CR-xforms11-20071129/. Acesso em: jul. de 2008.

BRAY, T. et al. **Extensible markup language (XML) 1.0**. w3c recommendation, 4. ed. 16 August 2006. Disponível em: http://www.w3.org/TR/REC-xml/. Acesso em: jul. 2008.

BROWN, M. Mashing up the once and future CMS. **Educase Review**. Boulder, v. 42, n. 2, Mar./Apr. 2007. Disponível em: http://www.educause.edu/apps/er/erm07/erm0725.asp?bhcp=1. Acesso em: abr. de 2007.

BRUCHEZ, E. **XForms**: an Alternative to Ajax? XTech 2006: "building web 2.0" — 16-19 May 2006, Amsterdam, The Netherlands. Dsiponível em: http://xtech06.usefulinc.com/schedule/paper/133. Acessado em junho de 2008.

BULL, G. Collaboration in a web 2.0 environment. **Learning & Leading with Technology,** Eugene , v. 33, n. 7 p. 23-24, Apr. 2006. Disponível em: http://www.iste.org/Content/NavigationMenu/EducatorResources/YourLearningJourney/Web20/collaboration-in-web-20-environment.pdf. Acesso em: jul. 2008.

CAMPOS, F. C. A. et al. **Cooperação e aprendizagem on-line**. Rio de Janeiro: DP&A, 2003. 168 p.

CLARK, J. **XSL transformations (XSLT) version 1.0. w3c recommendation.** 16 Nov. 1999. Disponível em: http://www.w3.org/TR/xslt. Acesso em: jul. 2008

CROCKFORD, D. JSON: the fat-free alternative to XML. In: XML 2006. 2006, Boston **Proceedings:…** Boston: Just Systems, IBM, 2006. Disponível em: http://www.json.org/fatfree.html. Acesso em jun. de 2008.

CROSS, J. **Informal learning:** rediscovering the natural pathways that inspire innovation and performance. San Francisco: Pfeiffer, 2006. 320 p. ISBN-10: 0787981699 ISBN-13: 978-0787981693

CRUZ, C. C. P. **ReCoP um modelo para reputação em comunidades de prática.** 2008, Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica , Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008.

CRUZ, C. C. P et al. Um estudo sobre reputação baseado no grau de concordância entre os membros de comunidades de prática. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 19., 2008, Fortaleza. **Anais ...** Fortaleza: SBC, 2008.

DOWNES, S. Educational blogging. **EDUCAUSE Review**, Boulder, v. 39, n. 5, p. 14-26, Sept./Oct. 2004. Disponível em: <http://www.educause.edu/pub/er/erm04/erm0450.asp>. Acesso em: dez. 2006.

DOWNES, S. E-learning 2.0. **eLearn Magazine**, New York, v. 2005, n. 10, Oct. 2005.

EBERSBACH, A. ; GLASER, M. ; HEIGL, R. **Wiki -** web collaboration. Berlin: Springer-Verlag, 2006. 383 p.

ECMA. **ECMAScript language specification**. Standard ECMA-262, 3rd Edition, December 1999. 188p. Disponível em: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf. Acesso em: dez. 2006.

ELIA, M. F. ; SAMPAIO, F. F. Plataforma interativa para a internet: uma proposta de pesquisa-ação a distância para professores. SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. 2001, Vitória. **Anais ...** Vitória: SBIE, 2001. p.102-105.

FELDSTEIN, M. ; MASSON, P. Unbolting the chairs: making learning management systems more flexible. **eLearn Magazine**, New York, v. 2006, n. 1, Jan. 2006. Disponível em: http://www.elearnmag.org/subpage.cfm?section=tutorials&article=22-1. Acesso em: abr. 2007.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures.** 2000. Dissertation (PhD) - University of California, Irvine, 2000. Disponível em: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. Acesso em: maio 2007.

FIELDING, R. T. ; TAYLOR, R. N. Pricipled design of the modern web architecture. **ACM Transactions on Internet Technology**, New York, v. 2, n. 2, p. 115-150, May 2002.

FRANKS, J. et al. **HTTP authentication**: basic and digest access authentication. RFC 2617, June 1999. Disponível em: http://tools.ietf.org/html/rfc2617 Acesso em: jun. 2008.

FUMERO A. et al. Next-generation educational web. In: INTERNATIONAL CONFERENCE ON CONCURRENT ENTERPRISING, 12,. 2006, Milan. **Proceedings …** Milan: Eso Net, 2006. Disponível em: http://jungla.dit.upm.es/~saguirre/publications/NewGeneration.pdf. Acesso em: abr. de 2007.

GARRETT, J. J. **Ajax**: a new approach to web applications. 2005. Disponível em: http://www.adaptivepath.com/ideas/essays/archives/000385.php. Acesso em: abr. 2007.

GOMES, M. Blogs: um recurso e uma estratégia pedagógica. In: SIMPÓSIO INTERNACIONAL DE INFORMÁTICA EDUCATIVA, 7., 2005, Leiria. **Anais ...** Leiria: Escola Superior de Educação de Instituto Politécnico de Leiria, 2005. Disponível em: http://creazeitao.googlepages.com/BlogsUtilEducUNIVMINHO.pdf. Acesso em: nov. de 2007.

GONÇALVES, M. V. F. ; ELIA, M. F. Arquitetura de avaliação educacional em fórum de discussão. In: WORKSHOP SOBRE INFORMÁTICA NA ESCOLA (WIE2008).2008. Belém. **Anais ...** Belém: SBC, 2008.

GOTTSCHALK, K. ; GRAHAM, S. Introduction to web services architecture. **IBM Systems Journal**, Armonk, v. 41, n. 2, p.178-198. 2002. Disponível em: http://www.research.ibm.com/journal/sj/412/gottschalk.pdf. Acesso em: dez. 2007.

GOULART, I. B. **Piaget**: experiências básicas para utilização pelo professor. 14. ed. Petrópolis: Vozes, 1998. 148 p. ISBN 85.326.0386-6.

GREGORIO, J. **Do we need WADL?** 05/06/2007. Disponível em: http://bitworking.org/news/193/Do-we-need-WADL Acesso em: jun. 2008.

HAAS, H. Reconciling web services and REST services. IEEE EUROPEAN CONFERENCE ON WEB SERVICES, 3., 2005, Växjö, **Proceedings …** Växjö: IEEE, 2005. Disponível em: http://www.w3.org/2005/Talks/1115-hh-k-ecows. Acesso em: jun. 2008.

HADLEY, M. **Web application description language (WADL) specification**. Sun Microsystems Inc . November 9, 2006. Disponível em: https://wadl.dev.java.net/. Acesso em: jun. 2008.

HAMMOND, T. Social bookmarking tools (I) a general review. **D-Lib Magazine**. Reston, v. 11, n. 4, Apr. 2005. ISSN 1082-9873. Disponível em: http://www.dlib.org/dlib/april05/hammond/04hammond.html. Acesso em: jun. 2008.

HARMELEN, M. Personal learning environments. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES, 6., 2006, Kerkrade. **Proceedings …** Kerkrade: IEEE, 2006. p. 815-816.

HILDENBRAND, B. **ActivUFRJ**: ambiente colaborativo de trabalho integrado e virtual. 2006. 128 f. Trabalho de Conclusão de Curso. (Graduação em Ciência da Computação) – Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006..

IETF. **The atom syndication format**. RFC-4287. December, 2005. Disponível em: http://tools.ietf.org/html/rfc4287. Acesso em: jun. 2008.

_____ **The atom publishing protocol**. RFC-5023. October, 2007. Disponível em: http://tools.ietf.org/html/rfc5023. Acesso em: jun. 2008.

JOHNSON, D. **RSS and atom in action:** web 2.0 building blocks. Greenwich: Manning Publications, 2006. 368p. ISBN 1932394494.

KOPER, R. ; TATTERSALL, C. New directions for lifelong learning using network technologies. **British Journal of Educational Technology**. West Sussex, v. 35, n. 6, p. 689-700, 2004. Disponível em: http://www3.interscience.wiley.com/journal/118747749/abstract. Acesso em: nov. 2008.

LÉVY, P. **A inteligência coletiva:** por uma antropologia do ciberespaço. São Paulo: Loyola, 1998. 212p.

LIBER, O. Colloquia – a conversation manager. **Campus-Wide Information Systems**. Bingley, v. 17, n. 2, p. 56-62, 2000. ISSN: 1065-0741. Disponível em: http://www.emeraldinsight.com/10.1108/10650740010326618. Acesso em: out. 2007.

LIMA, C. R. M. ; SANTINI, R. M. Creative commons e produção cultural colaborativa no Brasil. **Comunicação e Espaço Público**, Brasília, v. 9, n. 1-2, 2006. Disponível em http://www.unb.br/fac/posgraduacao/revista2006/14_a_clovisrose.pdf. Acesso em: out. de 2008.

LUND, A. ; SMØRDAL, O. Is there a space for the teacher in a WIKI? In: 2006 INTERNATIONAL SYMPOSIUM ON WIKIS, 2006, Odense. **Proceedings …** New York: ACM, 2006. p. 37-46. ISBN: 1-59593-413-8.

LUND, B. et al. Social bookmarking tools (II) a case study – connotea. **D-Lib Magazine**. Reston, v. 11, n. 4, Apr. 2005. ISSN 1082-9873. Disponível em: http://www.dlib.org//dlib/april05/lund/04lund.html. Acesso em: dez. 2007.

McCANDLESS, M. The MP3 revolution. **IEEE Intelligent Systems**, New York, v.14, n.3, p.8-9, May/Jun 1999.

MILLER, J. **Yadis specification**. Technical report, March 2006. Disponível em: http://yadis.org/papers/yadis-v1.0.pdf. Acesso em: dez. 2007.

MILLIGAN, C. et al. Developing a reference model to describe the personal learning environment. In: BEJDL, W. ; TOCHTERMANN, K. **Innovative Approaches for Learning and Knowledge Sharing** Berlin: Springer, 2006. p. 506-511. (Lecture Notes in Computer Science, 4227). Disponível em: http://www.springerlink.com/content/u04836n0460j2678/fulltext.pdf. Acesso em: ago. 2008.

MITRA, N. ; LAFON, Y. **SOAP** version 1.2 Part 0: primer (Second Edition). W3C recommendation. 27 April 2007. Disponível em: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/. Acesso em: ago. 2008.

MOREIRA, M. A. **Teorias de aprendizagem**. São Paulo: EPU, 1999. ISBN 85-12-32140-7.

MOTTA, C. ; BORGES, M. TEAMWORKS: um ambiente para apoio à cooperação nas equipes de trabalho. In: SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E HIPERMÍDIA, 6., 2000, Natal. **Anais...** Natal: SBC, 2000. p. 259-272.

MYLLYNIEMI, A. Identity management systems: a comparison of current solutions. In: SEMINAR ON NETWORK SECURITY . 2006. Spoo. **Proceedings ...** Spoo: Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, 2006. Disponível em: http://www.tml.tkk.fi/Publications/C/22/papers/Myllyniemi_final.pdf Acesso em: dez. 2007.

NELSON, B. J. **Remote procedure call**. 1981. Dissertation (Ph.D.) _ Carnegie-Mellon University, Pittsburg, 1981. (CMU-CS-81-119).

NUNES, S. ; DAVID, G. Uma arquitetura web para serviços web. In: XATA 2005 – XML: APLICAÇÕES E TECNOLOGIAS ASSOCIADAS. 2005. Braga, **Actas ...** Braga: Universidade do Minho, Universidade do Porto, 2005. p.205-215. Disponível em: https://www.fe.up.pt/si/publs_pesquisa.formview?p_id=12085. Acesso em: dez. 2007.

OASIS. **Extensible resource identifier (XRI) resolution version 2.0** - Committee Draft 02. 25 November 2007. Disponível em: http://docs.oasis-open.org/xri/2.0/specs/cd02/xri-resolution-V2.0-cd-02.pdf Acesso em: dez. 2007.

OBLINGER, D. G. Boomers, gen-xers & millenials: understanding the new students. **Educause Review**, Boulder, v. 38, n. 4, Jul./Aug. 2003. Disponível em: http://net.educause.edu/ir/library/pdf/ERM0342.pdf. Acesso em: dez. 2007.

OPEN GROUP. **Single sign-on**. 2005. Disponível em: http://www.opengroup.org/security/sso/. Acesso em: dez. 2007.

O'REILLY, T. **What is web 2.0**? design patterns and business models for the next generation of software. 2005. Disponível em: <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>. Acesso em: out. 2006.

PAGALTZIS, A. **Does REST need a service description language?** 27/05/2007. Disponível em: http://plasmasturm.org/log/460/ Acesso em: jun. 2008.

PIAGET, J. **Psicologia e epistemologia por uma teoria do conhecimento**. 2. ed. Rio de Janeiro: Forense Universitária, 1978.

PITNER, T. ; DRASIL, P. An e-learning 2.0 environment - principles, technology and prototype. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE MANAGEMENT, 6., 2006. Graz. **Proceedings …** Graz: Graz University of Technology, 2006. Disponível em: http://i-know.tugraz.at/blog/wp-content/uploads/2008/11/66_an-e-learning-20-environment.pdf. Acesso em: jun. 2008

POWELL, A ; RECORDON, D. OpenID: decentralised single sign-on for the web. **Ariadne**, Bath, n. 51, Apr. 2007. Disponível em: http://www.ariadne.ac.uk/issue51/powell-recordon/ Acesso em: dez. 2007.

PRENSKY, M. Digital natives, digital immigrants. **On the Horizon**. Bingley, v. 9, n. 5, Sep./Oct. 2001. Disponível em: http://www.emeraldinsight.com/Insight/viewPDF.jsp?contentType=Article&Filename=html/Output/Published/EmeraldFullTextArticle/Pdf/2740090501.pdf. Acesso em: dez. 2007.

RAGGETT, D. ; LE HORS, A. ; JACOBS, I. **HTML 4.01 specification**. W3C recommendation 24 December 1999. Disponível em: http://www.w3.org/TR/html401/. Acesso em: jun. 2008.

RESCORLA, E. **SSL and TLS:** designing and building secure systems**.** Reading: Addison-Wesley, 2001. ISBN 0-201-61598-3.

RESNICK, P., VARIAN, H. R. Recommender Systems. **Communications of the ACM**, vol. 40, n. 3, 1997.

REZENDE, F. As novas tecnologias na prática pedagógica sob a perspectiva construtivista. **Ensaio** – pesquisa em educação em ciências. Belo Horizonte, v.2, n.1, março de 2002.

RICHARDSON, L. ; RUBY, S. **RESTful web services**. 1. ed. Sebastopol: O'Reilly, 2007. 446 p. ISBN-10: 0-596-52926-0. ISBN-13: 978-0-596-52926-0

ROLLETT, H. et al. The Web 2.0 way of learning with technologies. **International Journal of Learning Technology,** Geneve, v. 3, n. 1 / 2007. p 87-107. Disponível em: http://www.cs.toronto.edu/~mstrohm/documents/2007_JoLT_Learning.pdf. Acesso em abr. 2007.

SCHWARTZ, L. et al. Educational wikis: features and selection criteria. **International Review of Research in Open and Distance Learning,** Edmonton, v. 5, n. 1. Apr. 2004. ISSN: 1492-3831. Disponível em: http://www.irrodl.org/index.php/irrodl/article/view/163/692. Acesso em: jul. 2008.

SILVA, S. P. A. et al. Oraculous: um serviço de combinação social apoiando ambientes pessoais de aprendizagem. SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 19., 2008, Fortaleza. **Anais ...** Fortaleza: SBC, 2008.

SILVA, F. E. O. **Uma abordagem de combinação social para apoiar a formação de equipes**. 2009. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal Rio de Janeiro, Rio de Janeiro, 2009.

STENBACK, J. ; HENINGER, A. **Document object model (DOM) level 3 load and save specification.** Version 1.0. W3C recommendation. Disponível em: http://www.w3.org/TR/DOM-Level-3-LS/, 07 April 2004. Acesso em: jul. 2008.

TAKASE, T. et al. **Definition languages for RESTful web services**: WADL vs. WSDL 2.0. April 22, 2008. Disponível em: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-wadlwsdl/WADLWSDLpaper20080422.pdf. Acesso em: jun. de 2008.

VYGOTSKY, L. **A formação social da mente:** o desenvolvimento dos processos psicológicos superiores. 6. ed. São Paulo: Martins Fontes, 1998.

W3C. **Public-web-http-desc@w3.org mail archives**. May 2005 to December 2007. Disponível em: http://lists.w3.org/Archives/Public/public-web-http-desc/. Acesso em: jun. 2008.

W3C. **XHTML™ 1.0 the extensible hypertext markup language** (Second Edition) - a reformulation of HTML 4 in XML 1.0. W3C recommendation 26 January 2000, revised 1 August 2002. Disponível em: http://www.w3.org/TR/xhtml1/. Acesso em: jun. 2008.

W3C INTERACTION DOMAIN. **The forms working group**. Disponível em: http://www.w3.org/MarkUp/Forms/. Acesso em: jun. 2008.

W3C XFORMS GROUP WIKI PUBLIC. **XForms implementations**. Disponível em: http://www.w3.org/MarkUp/Forms/wiki/XForms_Implementations. Acesso em: jun. 2008.

WANG, C. ; TURNER, D. Extending the wiki paradigm for use in the classroom. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: CODING AND COMPUTING, 2004. Las Vegas. **Proceedings …** Los Alamitos: IEEE, 2004. v. 1. p. 255-259. ITCC 2004. Disponível em: http://ieeexplore.ieee.org/iel5/9035/28682/01286462.pdf?isnumber=28682∏=STD&arnumber=1286462&arnumber=1286462&arSt=+255&ared=+259+Vol.1&arAuthor=Chien-min+Wang%3B+Turner%2C+D. Acesso em: jun. 2008.

WILSON, S. et al. Personal learning environments: challenging the dominant design of educational systems. In: INTERNATIONAL WORKSHOP ON LEARNER-ORIENTED KNOWLEDGE MANAGEMENT AND KM-ORIENTED LEARNING, 2., 2006. Crete **Proceedings …** Hannover: L3S Research Center / PROLEARN, 2006. p. 67-76. In conjunction with the First European Conference on Technology-Enhanced Learning (ECTEL 06), 2006. Disponível em: < http://hdl.handle.net/1820/727 >. Acesso em: dez. 2006.

YEE, R. **Pro Web 2.0 Mashups Pro Web 2.0 Mashups**: remixing data and web services. Berkeley: Apress, 2008. 603 p. ISBN 159059858X, 9781590598580

YOSHIZAKI, K. Development of social networking site for learning purposes. In: IASTED INTERNATIONAL CONFERENCE ON WEB-BASED EDUCATION, 5., 2006. Puerto Vallarta. **Proceedings …** Anaheim: ACTA Press, 2006. p. 106-111. ISSN:1482-7905 , 0-88986-541-8.

ZANETTE, L. R.. **Sistema de recomendação de itens baseado na confiança do usuário**. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008.

Apêndices

Apêndice A - Casos de Uso do AvaNCE

A.1) Núcleo AvaNCE

UC-01 Autenticar Usuário (Login AvaNCE)

Objetivo: Validar o usuário do AvaNCE
Ator: Usuário do sistema
Pré-condição:

Cenário Principal
1. Sistema solicita CPF do usuário e senha
2. Usuário digita CPF e senha
3. Sistema verifica se a senha foi preenchida
4. Sistema verifica se a senha corresponde à conta informada
5. Sistema cria uma nova sessão.
6. Sistema executa UC-06 Listar Turmas do Usuário.

Variações
3.a. Se a senha não foi informada
3.a.1 Sistema retorna erro 400: "Senha não informada"
3.a.2 Sistema retorna ao passo 6 do fluxo principal

4.a. Se senha não corresponde ao CPF informado
4.a.1 Sistema retorna erro 401: "Senha inválida ou login inválido"
4.a.2 Sistema retorna ao passo 6 do fluxo principal

UC-02 Autenticar Usuário (Login OpenID)

Objetivo: Validar o usuário do AvaNCE através de OpenID
Atores: Usuário do sistema, provedor OpenID do usuário
Pré-condição:

Cenário Principal
1. Sistema solicita URI do usuário
2. Usuário digita sua URI
3. Sistema redireciona para o provedor OpenID do usuário
4. Provedor OpenID solicita username e password
5. Usuário fornece username e password
6. Provedor OpenID solicita o prazo para expiração da autenticação
7. Usuário fornece o prazo para expiração da autenticação
8. Provedor OpenID redireciona o usuário de volta para o AvaNCE
9. Sistema cria uma nova sessão.
10. Sistema executa UC-06 Listar Turmas do Usuário.

Variações
3.a. Se a URI é inválida ou o sistema não consegue se conectar com o provedor
3.a.1 Sistema retorna erro 500: "Erro ao redirecionar para OpenID"

3.a.2 Sistema retorna ao passo 10 do fluxo principal

4.a. Se username ou password estiverem incorretos
4.a.1 Sistema retorna ao passo 4 do fluxo principal

UC-03 Incluir Usuário

Objetivo: Cadastrar uma pessoa como usuária do AvaNCE
Ator: Qualquer pessoa
Pré-condição:

Cenário Principal
1. Sistema solicita informações de cadastro (CPF, senha, nome, e-mail, telefone)
2. Usuário digita informações solicitadas
3. Sistema inclui cadastro do usuário no banco
4. Sistema produz RSS confirmando a inclusão com informações do usuário cadastrado.
5. Sistema encerra a funcionalidade

Variações
3.a. Caso o CPF informado já tenha sido cadastrado
3.a.1 Sistema retorna erro 500: "Usuário já existente"
3.a.2 Sistema retorna ao passo 5 do fluxo principal

UC-04 Incluir Curso

Objetivo: Cadastrar um novo curso no AvaNCE
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema solicita informações do curso (Nome do curso, instituição)
2. Usuário digita informações solicitadas
3. Sistema inclui cadastro do curso no banco
4. Sistema executa UC 12 Listar Cursos

UC-05 Incluir Aplicação

Objetivo: Cadastrar um nova aplicação no AvaNCE
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema solicita informações da aplicação (Nome da aplicação, URL, Tipo)
2. Usuário digita informações solicitadas
3. Sistema inclui cadastro da aplicação no banco
4. Sistema executa UC 09 Listar Aplicações.

UC-06 Listar Turmas do Usuário

Objetivo: Listar as turmas do usuário
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de turmas do usuário, com link para a página de cada turma (UC-07).
2. Sistema encerra a funcionalidade

UC-07 Listar Página da Turma

Objetivo: Listar a página de uma turma
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de aplicações de uma turma, com link para executar cada aplicação (UC-08). Se a aplicação for externa, apresenta menu de opções da API.
2. Sistema encerra a funcionalidade

UC-08 Executar Aplicação

Objetivo: Executar uma aplicação nativa ou externa ao AvaNCE.
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema executa a aplicação dependendo do seu tipo.
2. Se for interna ao AvaNCE, desvia para a URL da aplicação.
3. Se for feed RSS externo, obtém arquivo RSS na URL da aplicação e exibe-o para o usuário.
4. Se for aplicação WADL externa, requisita o método através do Executor.
5. Sistema encerra a funcionalidade

UC-09 Listar Aplicações

Objetivo: Listar as aplicações cadastradas no sistema
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema produz RSS com relação de aplicações cadastradas no sistema.
2. Caso a aplicação seja externa, apresenta link para carregar WADL (UC-10).
3. Se o WADL já estiver carregado, apresenta as funções da API.
4. Sistema encerra a funcionalidade

UC-10 Carregar WADL

Objetivo: Carregar os métodos de uma API através de sua descrição WADL
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema obtém arquivo WADL na URL cadastrada.
2. Caso o WADL desta aplicação já tenha sido carregado anteriormente, apaga a descrição da API armazenada no banco.
3. Sistema interpreta o arquivo WADL e armazena informações da API no banco.
4. Sistema executa UC 09 Listar Aplicações

UC-11 Configurar Aplicação Externa

Objetivo: Permitir que o usuário configure uma aplicação externa
Ator: Usuário do sistema
Pré-condição: Usuário autenticado, WADL Carregado

Cenário Principal
1. Sistema exibe formulário para seleção dos métodos utilizados e respectivos parâmetros de entrada e saída.
2. Usuário fornece informações solicitadas.
3. Sistema armazena configuração da aplicação no banco.
4. Sistema encerra a funcionalidade

UC-12 Listar Cursos

Objetivo: Listar os cursos cadastrados no sistema
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema produz RSS com relação de cursos cadastrados no sistema, com link para listar as disciplinas de cada curso (UC-13).
2. Sistema encerra a funcionalidade

UC-13 Listar Disciplinas

Objetivo: Listar as disciplinas de um curso
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema produz RSS com relação de disciplinas de um curso com link para listar as turmas de cada disciplina (UC-14).
2. Sistema encerra a funcionalidade

UC-14 Listar Turmas

Objetivo: Listar as turmas de uma disciplina
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema produz RSS com relação de turmas de uma disciplina com link para listar as aplicações de cada turma (UC-15).
2. Sistema encerra a funcionalidade

UC-15 Listar Aplicações da Turma

Objetivo: Listar as aplicações de uma turma
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema produz RSS com relação de aplicações de uma turma.
2. Sistema encerra a funcionalidade

UC-16 Incluir Disciplina

Objetivo: Cadastrar uma nova disciplina de um curso
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema solicita informações da disciplina (Nome da disciplina, Carga horária, Ementa)
2. Usuário digita informações solicitadas
3. Sistema inclui cadastro da disciplina no banco
4. Sistema executa UC 13 Listar Disciplinas.

UC-17 Incluir Turma

Objetivo: Cadastrar uma nova turma de uma disciplina
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema solicita informações da turma (Nome da turma, Data início, Data final e professor)
2. Usuário digita informações solicitadas
3. Sistema inclui cadastro da turma no banco
4. Sistema executa UC 14 Listar Turmas.

UC-18 Incluir Aplicação na Turma

Objetivo: Incluir uma aplicação numa turma
Ator: Administrador do sistema
Pré-condição: Administrador autenticado

Cenário Principal
1. Sistema apresenta lista com as aplicações cadastradas no sistema
2. Usuário seleciona uma aplicação
3. Sistema inclui aplicação na turma
4. Sistema executa UC 15 Listar Aplicações da Turma

UC-19 Incluir Aluno em Turma

Objetivo: Incluir usuário como aluno de uma turma
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita nome da turma na qual o usuário deseja se inscrever
2. Usuário informa o nome da turma
3. Sistema inclui aluno na turma
4. Sistema executa UC 20 Listar Alunos de uma Turma

Variações
3.a. Caso a turma informada seja inexistente
3.a.1 Sistema retorna erro 500: "Turma inexistente"
3.a.2 Sistema encerra a funcionalidade

UC-20 Listar Alunos numa Turma

Objetivo: Listar os alunos pertencentes a uma turma
Ator: Professor da turma
Pré-condição: Professor autenticado

Cenário Principal
1. Sistema produz RSS com relação de alunos pertencentes a uma turma.
2. Sistema encerra a funcionalidade

UC-21 Incluir Comunidade

Objetivo: Cadastrar uma nova comunidade
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita informações da comunidade (Nome da comunidade e Descrição)
2. Usuário digita informações solicitadas

3. Sistema inclui cadastro da comunidade no banco
4. Sistema executa UC 24 Listar Comunidades.

UC-22 Incluir Aplicação na Comunidade

Objetivo: Incluir uma aplicação numa comunidade
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema apresenta lista com as aplicações cadastradas no sistema
2. Usuário seleciona uma aplicação
3. Sistema inclui aplicação na comunidade
4. Sistema executa UC 25 Listar Aplicações de uma Comunidade

UC-23 Incluir Usuário em Comunidade

Objetivo: Incluir um usuário numa comunidade
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita nome da comunidade na qual o usuário deseja se inscrever
2. Usuário informa o nome da comunidade
3. Sistema inclui aluno na comunidade
4. Sistema executa UC 28 Listar Pessoas numa Comunidade

Variações
3.a. Caso a comunidade informada seja inexistente
3.a.1 Sistema retorna erro 500: "Comunidade inexistente"
3.a.2 Sistema encerra a funcionalidade

UC-24 Listar Comunidades

Objetivo: Listar as comunidades existentes
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de comunidades com link para listar as aplicações de uma comunidade (UC-25).
2. Sistema encerra a funcionalidade

UC-25 Listar Aplicações da uma Comunidade

Objetivo: Listar as aplicações de uma comunidade
Ator: Usuário do sistema

Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de aplicações de uma comunidade.
2. Sistema encerra a funcionalidade

UC-26 Listar Comunidades do Usuário

Objetivo: Listar as comunidades do usuário
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de comunidades do usuário, com link para a página de cada comunidade (UC-27).
2. Sistema encerra a funcionalidade

UC-27 Listar Página da Comunidade

Objetivo: Listar a página de uma comunidade
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de aplicações de uma comunidade, com link para executar cada aplicação (UC-08).
2. Sistema encerra a funcionalidade

UC-28 Listar Pessoas numa Comunidade

Objetivo: Listar as pessoas pertencentes a uma comunidade
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de usuários pertencentes a uma comunidade.
2. Sistema encerra a funcionalidade

A.2) Fórum de Discussão AvaNCE

UC-29 Listar Temas de uma Turma

Objetivo: Listar os temas do fórum de uma turma do AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado.

Cenário Principal
1. Sistema produz RSS com relação de temas (título, data, autor e descrição) e link em cada tema para listar a relação de assuntos do tema. Ao final da lista de temas, a saída RSS deve possuir um item sem descrição para identificar um link para inclusão de novo tema.
2. Sistema encerra a funcionalidade

UC-30 Listar Assuntos de um Tema

Objetivo: Listar os assuntos de uma tema do fórum AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de assuntos (título, data, autor e descrição) e link em cada assunto para listar a relação de tópicos do assunto. Ao final da lista de assuntos, a saída RSS deve possuir um item sem descrição para identificar um link para inclusão de novo assunto e outro item para retornar à lista de temas.
2. Sistema encerra a funcionalidade

UC-31 Listar Tópicos de um Assunto

Objetivo: Listar os tópicos de um assunto do fórum do AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de tópicos (título, data, autor e descrição) e link em cada tópico para listar a relação de mensagens do tópico. Ao final da lista de tópicos, a saída RSS deve possuir um item sem descrição para identificar um link para inclusão de novo tópico, um para retornar à lista de temas e outro para retornar à lista de assuntos.
2. Sistema encerra a funcionalidade

UC-32 Listar Mensagens de um Tópico

Objetivo: Listar as mensagens de um tópico do fórum do AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema produz RSS com relação de mensagens (data, autor e mensagem). Ao final da lista de mensagens, a saída RSS deve possuir um item sem descrição para identificar um link para inclusão de nova mensagem, um para retornar à lista de temas e outro para retornar à lista de assuntos, e outro para retornár à lista de tópicos .
2. Sistema encerra a funcionalidade

UC-33 Incluir Tema

Objetivo: Incluir um tema no fórum de uma turma do AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita informações do tema (nome do tema e descrição)
2. Usuário digita informações solicitadas
3. Sistema inclui o tema no banco
4. Sistema executa UC 29 Listar Temas de uma Turma

UC-34 Incluir Assunto

Objetivo: Incluir um assunto num temas do fórum do AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita informações do assunto (nome do assunto e descrição)
2. Usuário digita informações solicitadas
3. Sistema inclui o assunto no banco
4. Sistema executa UC 30 Listar Assuntos de um Tema

UC-35 Incluir Tópico

Objetivo: Incluir um tópico num assunto do fórum do AvaNCE
Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita informações do tópico (nome do tópico e descrição)
2. Usuário digita informações solicitadas
3. Sistema inclui o tópico no banco
4. Sistema executa UC 31 Listar Tópicos de um Assunto

UC-36 Incluir Mensagem

Objetivo: Incluir uma mensagem num tópico do fórum do AvaNCE

Ator: Usuário do sistema
Pré-condição: Usuário autenticado

Cenário Principal
1. Sistema solicita informações da mensagem (nome da mensagem e texto)
2. Usuário digita informações solicitadas
3. Sistema inclui a mensagem no banco
4. Sistema executa UC 32 Listar Mensagens de um Tópico

Apêndice B - Descrição das APIs REST do AvaNCE

B.1) API REST do núcleo do AvaNCE

Autenticação do usuário

Caso de uso: UC01 e UC02
URL: GET /rest/sessao/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para que o usuário seja autenticado.
Privilégio: Nenhum
Retorno: XForms com formulário de autenticação

URL: POST /rest/sessao
Descrição: Verifica usuário e senha e cria uma sessão.
Privilégio: Nenhum
POST Data: Usuario; senha.
Retorno: RSS com turmas do usuário

URL: POST /rest/OpenID
Descrição: Redireciona para provedor openID do usuário para fazer autenticação e cria uma sessão.
Privilégio: Nenhum
POST Data: OpenID_url
Retorno: RSS com turmas do usuário

Cadastramento de um usuário

Caso de uso: UC03
URL: GET /rest/usuario/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para o cadastramento de um novo usuário.
Privilégio: Nenhum
POST Data: Cpf; Senha; Confirmação da senha; Nome; E-mail; Telefone; É professor; É aluno; É tutor; É administrador;
Retorno: XForms com formulário de cadastramento

URL: POST /rest/usuario
Descrição: Recebe os dados de um novo usuário, criando um registro para o mesmo.
Privilégio: Nenhum
POST Data: Cpf; Senha; Confirmação da senha; Nome; E-mail; Telefone; É professor; É aluno; É tutor; É administrador;
Retorno: RSS com descrição do usuário cadastrado.

Listar turmas de um usuário

Caso de uso: UC06
URL: GET /rest/[idSessao]/turma
Privilégio: Usuário
Retorno: RSS com turmas do usuário.

Listar a página de uma turma

Caso de uso: UC07
URL: GET /rest/[idSessao]/turma/[idTurma]
Privilégio: Usuário pertencente à turma
Retorno: RSS com aplicações disponíveis para os membros de uma turma

Listar cursos existentes

Caso de uso: UC12
URL: GET /rest/[idSessao]/curso
Privilégio: Administrador
Retorno: RSS com os cursos cadastrados no ambiente

Listar disciplinas do curso

Caso de uso: UC13
URL: GET /rest/[idSessao]/curso/[idCurso]
Privilégio: Administrador
Retorno: RSS com as disciplinas de um determinado curso

Listar turmas de uma disciplina

Caso de uso: UC14
URL: GET /rest/[idSessao]/curso/[idCurso]/[idDisc]
Privilégio: Administrador
Retorno: RSS com as turmas de uma determinada disciplina

Listar aplicações de uma turma

Caso de uso: UC15
URL: GET /rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]
Privilégio: Administrador
Retorno: RSS com as aplicações de uma determinada turma

Criar novo curso

Caso de uso: UC04
URL: GET /rest/[idSessao]/curso/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para o cadastramento de um novo curso.
Privilégio: Administrador
Retorno: XForms com formulário de cadastramento de curso

URL: POST /rest/[idSessao]/curso
Descrição: Recebe os dados de um novo curso, criando um registro para o mesmo.
Privilégio: Administrador
POST Data: Nome; Instituição; Responsável
Retorno: RSS com lista de cursos existentes

Criar disciplina de um curso

Caso de uso: UC16
URL: GET /rest/[idSessao]/curso/[idCurso]/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para o cadastramento de uma nova disciplina.
Privilégio: Administrador
Retorno: XForms com formulário de cadastramento de disciplina

URL: POST /rest/[idSessao]/curso/[idCurso]
Descrição: Recebe os dados de uma nova disciplina, criando um registro para a mesma.
Privilégio: Administrador
POST Data: Nome; Carga horária; Ementa; Responsável.
Retorno: RSS com lista de disciplinas do curso no qual a disciplina foi criada

Criar turma de uma disciplina

Caso de uso: UC17
URL: GET /rest/[idSessao]/curso/[idCurso]/[idDisc]/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para o cadastramento de uma nova turma.
Privilégio: Administrador
Retorno: XForms com formulário de cadastramento de turma

URL: POST /rest/[idSessao]/curso/[idCurso]/[idDisc]
Descrição: Recebe os dados de uma nova turma, criando um registro para a mesma.
Privilégio: Administrador
POST Data: Nome; Data início; Data final; Professor; Tutor; Senha.
Retorno: RSS com lista de turmas da disciplina na qual a turma foi criada

Listar alunos numa turma

Caso de uso: UC37

URL: GET /rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/inscricao
Privilégio: Professor
Retorno: RSS com os alunos de uma turma

Criar inscrição de aluno em uma turma

Caso de uso: UC19
URL: POST /rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/inscricao/[idAluno]
Privilégio: Professor
Retorno: RSS com os alunos da turma

Incluir aplicação no sistema

Caso de uso: UC05
URL: GET /rest/[idSessao]/aplicacao/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para o cadastramento de uma nova aplicação.
Privilégio: Administrador
Retorno: XForms com formulário de inclusão de aplicação no sistema

URL: POST /rest/[idSessao]/aplicacao
Descrição: Recebe os dados de uma nova aplicação, criando um registro para a mesma.
Privilégio: Administrador
POST Data: Nome; Externa; Url.
Retorno: RSS com lista de cursos existentes

Incluir aplicações numa turma

Caso de uso: UC18
URL: GET /rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]/form
Descrição: Obtém o formulário XForms com uma lista de checkboxes com as aplicações cadastradas no sistema.
Privilégio: Professor
Retorno: XForms com formulário de inclusão de aplicações em turma

URL: POST /rest/[idSessao]/curso/[idCurso]/[idDisc]/[idTurma]
Descrição: Atualiza a lista de aplicações utilizadas pela turma.
Privilégio: Professor
POST Data: Lista de aplicações selecionadas, dentre as cadastradas no sistema.
Retorno: RSS com lista das aplicações da turma

Carregar WADL da aplicação

Caso de uso: UC10
URL: GET /rest/[idSessao]/aplicacao/[idAplic]
Descrição: Lê e interpreta o arquivo WADL da aplicação carregando no banco de dados a sua descrição, de forma a permitir a execução posterior de seus métodos.

Privilégio: Administrador
Retorno: RSS com lista de cursos existentes

Configurar aplicação externa

Caso de uso: UC11
URL: GET /rest/[idSessao]/aplicacao/[idAplic]/configuracao/form
Descrição: Obtém o formulário XForms com os parâmetros para a configuração da aplicação especificada.
Privilégio: Usuário
Retorno: XForms com formulário de configuração da aplicação.

URL: POST /rest/[idSessao]/aplicacao/[idAplic]/configuracao
Descrição: Armazena a configuração da aplicação.
Privilégio: Usuário
Retorno: RSS com a configuração da aplicação.

Executar aplicação externa

Caso de uso: UC08
URL: GET /executor/[nome da aplicação externa]/[recurso1]/[recurso2]/.../[recursoN]/form
Descrição: Obtém o formulário XForms com os parâmetros para a requisição do recurso especificado.
Privilégio: Usuário
Retorno: XForms com formulário de inclusão de aplicações em turma

URL: POST /executor/[nome da aplicação externa]/[recurso1]/[recurso2]/.../[recursoN]
Descrição: Executa a requisição à API externa.
Privilégio: Usuário
POST Data: Lista de parâmetros do recurso requisitado.
Retorno: XHTML convertido a partir do retorno recebido da aplicação externa, de acordo com a configuração da aplicação.

B.2) API REST do fórum do AvaNCE

Listar Tema do Fórum

Caso de uso: UC40
URL: GET /Forum/rest/[idSessao]/[idTurma]
Privilégio: Usuário pertencente à turma
Retorno: RSS com os temas do fórum de uma turma

Listar Assunto

Caso de uso: UC41
URL: GET /Forum/rest/[idSessao]/[idTurma]/[idTema]
Privilégio: Usuário pertencente à turma
Retorno: RSS com os assuntos de um tema

Listar Tópico

Caso de uso: UC42
URL: GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]
Privilégio: Usuário pertencente à turma
Retorno: RSS com os tópicos de um assunto

Listar Mensagem

Caso de uso: UC43
URL: GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]
Privilégio: Usuário pertencente à turma
Retorno: RSS com as mensagens de um tópico

Inserir Tema

Caso de uso: UC44
URL: GET /Forum/rest/[idSessao]/[idTurma]/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para a inclusão de um novo tema.
Privilégio: Usuário pertencente à turma
Retorno: XForms com formulário de inclusão de tema

URL: POST /Forum/rest/[idSessao]/[idTurma]/
Descrição: Recebe os dados de um novo tema, criando um registro para o mesmo.
Privilégio: Usuário pertencente à turma
POST Data:
Retorno: RSS com lista de temas existentes

Inserir Assunto

Caso de uso: UC45
URL: GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para a inclusão de um novo assunto.
Privilégio: Usuário pertencente à turma
Retorno: XForms com formulário de inclusão de assunto

URL: POST /Forum/rest/[idSessao]/[idTurma]/[idTema]/
Descrição: Recebe os dados de um novo assunto, criando um registro para o mesmo.
Privilégio: Usuário pertencente à turma
POST Data:
Retorno: RSS com lista de assuntos existentes

Inserir Tópico

Caso de uso: UC46
URL: GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para a inclusão de um novo tópico.
Privilégio: Usuário pertencente à turma
Retorno: XForms com formulário de inclusão de tópico

URL: POST /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]
Descrição: Recebe os dados de um novo tópico, criando um registro para o mesmo.
Privilégio: Usuário pertencente à turma
POST Data:
Retorno: RSS com lista de tópico existentes

Inserir Mensagem

Caso de uso: UC47
URL: GET /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]/form
Descrição: Obtém o formulário XForms com os campos que precisam ser fornecidos para a inclusão de uma nova mensagem.
Privilégio: Usuário pertencente à turma
Retorno: XForms com formulário de inclusão de mensagem

URL: POST /Forum/rest/[idSessao]/[idTurma]/[idTema]/[idAssunto]/[idTopico]
Descrição: Recebe os dados de uma nova mensagem, criando um registro para a mesma.
Privilégio: Usuário pertencente à turma
POST Data:
Retorno: RSS com lista de mensagens existentes

Apêndice C – Esquema XML do formato WADL com a extensão proposta pelo AvaNCE

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3 targetNamespace="http://research.sun.com/wadl/2006/10"
4 xmlns:tns="http://research.sun.com/wadl/2006/10"
5 xmlns:xml="http://www.w3.org/XML/1998/namespace"
6 elementFormDefault="qualified">
7
8 <xs:import namespace="http://www.w3.org/XML/1998/namespace"
9 schemaLocation="http://www.w3.org/2001/xml.xsd"/>
10
11 <xs:element name="application">
12 <xs:complexType>
13 <xs:sequence>
14 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
15 <xs:element ref="tns:grammars" minOccurs="0"/>
16 <xs:element ref="tns:resources" minOccurs="0"/>
17 <xs:choice minOccurs="0" maxOccurs="unbounded">
18 <xs:element ref="tns:resource_type"/>
19 <xs:element ref="tns:method"/>
20 <xs:element ref="tns:representation"/>
21 <xs:element ref="tns:fault"/>
22 </xs:choice>
23 <xs:any namespace="##other" processContents="lax" minOccurs="0"
24 maxOccurs="unbounded"/>
25 </xs:sequence>
26 </xs:complexType>
27 </xs:element>
28
29 <xs:element name="doc">
30 <xs:complexType mixed="true">
31 <xs:sequence>
32 <xs:any namespace="##other" processContents="lax" minOccurs="0"
33 maxOccurs="unbounded"/>
34 </xs:sequence>
35 <xs:attribute name="title" type="xs:string"/>
36 <xs:attribute ref="xml:lang"/>
37 <xs:anyAttribute namespace="##other" processContents="lax"/>
38 </xs:complexType>
39 </xs:element>
40
41 <xs:element name="grammars">
42 <xs:complexType>
43 <xs:sequence>
44 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
45 <xs:element minOccurs="0" maxOccurs="unbounded" ref="tns:include"/>
46 <xs:any namespace="##other" processContents="lax" minOccurs="0"
47 maxOccurs="unbounded"/>
48 </xs:sequence>
49 </xs:complexType>
50 </xs:element>
51
25
52 <xs:element name="resources">
53 <xs:complexType>
```

```
<xs:sequence>
<xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="tns:resource" maxOccurs="unbounded"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="base" type="xs:anyURI"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>

<xs:element name="resource">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="tns:param" minOccurs="0" maxOccurs="unbounded"/>
<xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element ref="tns:method"/>
<xs:element ref="tns:resource"/>
</xs:choice>
<xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax"/>
</xs:sequence>
<xs:attribute name="id" type="xs:ID"/>
<xs:attribute name="type" type="tns:resource_type_list"/>
<xs:attribute name="queryType" type="xs:string"
default="application/x-www-form-urlencoded"/>
<xs:attribute name="path" type="xs:string"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>

<xs:simpleType name="resource_type_list">
<xs:list itemType="xs:anyURI"/>
</xs:simpleType>

<xs:element name="resource_type">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="tns:param" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="tns:method" maxOccurs="unbounded"/>
<xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax"/>
</xs:sequence>
<xs:attribute name="id" type="xs:ID"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>

<xs:element name="method">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="tns:request" minOccurs="0"/>
<xs:element ref="tns:response" minOccurs="0"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="id" type="xs:ID"/>
<xs:attribute name="name" type="tns:Method"/>
```

```
115 <xs:attribute name="href" type="xs:anyURI"/>
116 <xs:anyAttribute namespace="##other" processContents="lax"/>
117 </xs:complexType>
118 </xs:element>
119
120 <xs:simpleType name="Method">
121 <xs:union memberTypes="tns:HTTPMethods xs:NMTOKEN"/>
122 </xs:simpleType>
123
124 <xs:simpleType name="HTTPMethods">
125 <xs:restriction base="xs:NMTOKEN">
126 <xs:enumeration value="GET"/>
127 <xs:enumeration value="POST"/>
128 <xs:enumeration value="PUT"/>
129 <xs:enumeration value="HEAD"/>
130 <xs:enumeration value="DELETE"/>
131 </xs:restriction>
132 </xs:simpleType>
133
134 <xs:element name="include">
135 <xs:complexType>
136 <xs:sequence>
137 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
138 </xs:sequence>
139 <xs:attribute name="href" type="xs:anyURI"/>
140 <xs:anyAttribute namespace="##other" processContents="lax"/>
141 </xs:complexType>
142 </xs:element>
143
144 <xs:element name="request">
145 <xs:complexType>
146 <xs:sequence>
147 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
148 <xs:element ref="tns:param" minOccurs="0" maxOccurs="unbounded"/>
149 <xs:element ref="tns:representation" minOccurs="0"
150 maxOccurs="unbounded"/>
151 <xs:any namespace="##other" processContents="lax" minOccurs="0"
152 maxOccurs="unbounded"/>
153 </xs:sequence>
154 <xs:anyAttribute namespace="##other" processContents="lax"/>
155 </xs:complexType>
156 </xs:element>
157
158 <xs:element name="response">
159 <xs:complexType>
160 <xs:sequence>
161 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
162 <xs:element ref="tns:param" minOccurs="0" maxOccurs="unbounded"/>
163 <xs:choice minOccurs="0" maxOccurs="unbounded">
164 <xs:element ref="tns:representation"/>
165 <xs:element ref="tns:fault"/>
166 </xs:choice>
167 <xs:any namespace="##other" processContents="lax" minOccurs="0"
168 maxOccurs="unbounded"/>
169 </xs:sequence>
170 <xs:anyAttribute namespace="##other" processContents="lax"/>
171 </xs:complexType>
172 </xs:element>
173
174 <xs:simpleType name="uriList">
175 <xs:list itemType="xs:anyURI"/>
```

```
176 </xs:simpleType>
177
178 <xs:complexType name="representation_type">
179 <xs:sequence>
180 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
181 <xs:element ref="tns:param" minOccurs="0" maxOccurs="unbounded"/>
182 <xs:any namespace="##other" processContents="lax" minOccurs="0"
183 maxOccurs="unbounded"/>
184 </xs:sequence>
185 <xs:attribute name="id" type="xs:ID"/>
186 <xs:attribute name="element" type="xs:QName"/>
187 <xs:attribute name="status" type="tns:statusCodeList"/>
188 <xs:attribute name="mediaType" type="xs:string"/>
189 <xs:attribute name="href" type="xs:anyURI"/>
190 <xs:attribute name="profile" type="tns:uriList"/>
191 <xs:anyAttribute namespace="##other" processContents="lax"/>
192 </xs:complexType>
193
194 <xs:simpleType name="statusCodeList">
195 <xs:list itemType="xs:unsignedInt"/>
196 </xs:simpleType>
197
198 <xs:element name="representation" type="tns:representation_type"/>
199
200 <xs:element name="fault" type="tns:representation_type"/>
201
202 <xs:simpleType name="ParamStyle">
203 <xs:restriction base="xs:string">
204 <xs:enumeration value="plain"/>
205 <xs:enumeration value="query"/>
206 <xs:enumeration value="matrix"/>
207 <xs:enumeration value="header"/>
208 <xs:enumeration value="template"/>
209 </xs:restriction>
210 </xs:simpleType>
211
```

```
<xs:simpleType name="AuthStyle">
<xs:restriction base="xs:string">
<xs:enumeration value="none"/>
<xs:enumeration value="basic"/>
<xs:enumeration value="oauth"/>
<xs:enumeration value="authsub"/>
<xs:enumeration value="flickr"/>
</xs:restriction>
</xs:simpleType>
```

```
212 <xs:element name="param">
213 <xs:complexType>
214 <xs:sequence>
215 <xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
216 <xs:element ref="tns:option" minOccurs="0" maxOccurs="unbounded"/>
217 <xs:element ref="tns:link" minOccurs="0"/>
218 <xs:any namespace="##other" processContents="lax" minOccurs="0"
219 maxOccurs="unbounded"/>
220 </xs:sequence>
221 <xs:attribute name="name" type="xs:NMTOKEN" use="required"/>
222 <xs:attribute name="style" type="tns:ParamStyle" use="required"/>
223 <xs:attribute name="id" type="xs:ID"/>
224 <xs:attribute name="type" type="xs:QName" default="xs:string"/>
225 <xs:attribute name="default" type="xs:string"/>
```

```
<xs:attribute name="required" type="xs:boolean" default="false"/>
<xs:attribute name="repeating" type="xs:boolean" default="false"/>
<xs:attribute name="fixed" type="xs:string"/>
<xs:attribute name="path" type="xs:string"/>

    <xs:attribute name="authmode" type="tns:AuthStyle"/>

<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>

<xs:element name="option">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="value" type="xs:string" use="required"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>

<xs:element name="link">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:doc" minOccurs="0" maxOccurs="unbounded"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="resource_type" type="xs:anyURI"/>
<xs:attribute name="rel" type="xs:token"/>
<xs:attribute name="rev" type="xs:token"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:element>

</xs:schema>
```

Apêndice D – Descrição no formato WADL da API do Del.icio.us

```
<?xml version="1.0"?>
<!--This is a bootleg WADL file for the del.icio.us API. -->
<!--Modified to represent the possible responses using the parameter tag-->
<!--                          July 7, 2008 - Mauricio Bomfim & Joao Assis -->

<!--                                    December 3, 2008 - Mauricio Bomfim -->

<application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns="http://research.sun.com/wadl/2006/07">

 <resources base="https://api.del.icio.us">
  <doc xml:lang="en" title="The del.icio.us API v1">
   Post or retrieve your bookmarks from the social networking website.
   Limit requests to one per second.
  </doc>

  <resource path="v1">

   <param name="Authorization" style="header" required="true"
          authmode="basic">
    <doc xml:lang="en">All del.icio.us API calls must be authenticated
    using Basic HTTP auth.</doc>
   </param>

   <resource path="tags">
    <resource path="get"><method href="#getTags" /></resource>
    <resource path="rename"><method href="#renameTag" /></resource>
   </resource>

   <resource path="posts">
    <resource path="update"><method href="#getLastUpdateTime" /></resource>
    <resource path="get"><method href="#getPosts" /></resource>
    <resource path="recent"><method href="#getRecentPosts" /></resource>
    <resource path="all"><method href="#getAllPosts" /></resource>
    <resource path="dates"><method href="#getDates" /></resource>
    <resource path="add"><method href="#addPost" /></resource>
    <resource path="delete"><method href="#deletePost" /></resource>
   </resource>

   <resource path="bundles">
    <resource path="all"><method href="#getBundles" /></resource>
    <resource path="set"><method href="#makeBundle" /></resource>
    <resource path="delete"><method href="#deleteBundle" /></resource>
   </resource>
  </resource>
 </resources>

 <!-- Methods -->

 <!-- "update" method -->

 <method id="getLastUpdateTime" name="GET">
  <response>
   <representation mediaType="application/xml" element="update">
```

```
    <param name="dt" style="plain" path="@time">
    <link href="#getPosts" rel="" rev="" />

    </param>
   </representation>
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <!-- "tags" methods -->

 <method id="getTags" name="GET">
  <response>

   <representation mediaType="application/xml" element="tags">
    <param name="count" style="plain" path="/tags/tag/@count" />
    <param name="tag" style="plain" path="/tags/tag/@tag">
    <link href="#getPosts" rel="" rev="" />
    </param>
   </representation>
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <method id="renameTag" name="POST">
  <request>
   <param name="old" required="true" style="form">
    <doc xml:lang="en" title="Tag to rename." />
   </param>
   <param name="new" required="true" style="form">
    <doc xml:lang="en" title="New name." />
   </param>

  </request>

  <response>
   <representation href="#result" />
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <!-- "posts" methods part I: ways of getting posts -->

 <method id="getPosts" name="GET">

  <doc xml:lang="en" title="Returns posts matching the arguments.">
   If no date or url is given, most recent date will be used.
  </doc>

  <request>
   <param name="tag" style="form">
    <doc xml:lang="en" title="Filter by this tag." />
   </param>
   <param name="dt" style="form">
    <doc xml:lang="en" title="Filter by this date (CCYY-MM-DDThh:mm:ssZ)."
/>

   </param>
   <param name="url" style="form">
    <doc xml:lang="en" title="Filter by this URL." />
   </param>
```

```
  </request>
  <response>
   <representation mediaType="application/xml" element="posts">
    <param name="dt" style="plain" path="@dt" />
    <param name="tag" style="plain" path="@tag" />

    <param name="user" style="plain" path="@user" />

    <param name="description" style="plain" path="post/@description" />
    <param name="url" style="plain" path="post/@href" >
    <link href="#getPosts" rel="" rev=""> </link>
    </param>
    <param name="extended" style="plain" path="post/@extended" />
    <param name="hash" style="plain" path="post/@hash" />
    <param name="others" style="plain" path="post/@others" />

    <param name="tag" style="plain" path="post/@tag" />
    <param name="dt" style="plain" path="post/@time"  >
    <link href="#getPosts" rel="" rev="" />
    </param>
   </representation>
  </response>
 </method>

 <method id="getRecentPosts" name="GET">

  <doc xml:lang="en" title="Returns a list of the most recent posts.">
   Filtered by argument. Maximum 100.
  </doc>

  <request>
   <param name="tag" style="form">
    <doc xml:lang="en" title="Filter by this tag." />
   </param>
   <param name="count" style="form" default="15">

    <doc xml:lang="en" title="Number of items to retrieve.">Maximum:
100</doc>
   </param>
  </request>

  <response>
   <representation href="#postList" />
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <method id="getAllPosts" name="GET">
  <doc xml:lang="en" title="Returns all posts">
   Please use sparingly. Call the update function to see if you need
   to fetch this at all.
  </doc>

  <request>
   <param name="tag" style="form">
    <doc xml:lang="en" title="Filter by this tag." />
   </param>

  </request>

  <response>
```

```
   <representation href="#postList" />
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <method id="getDates" name="GET">
  <doc xml:lang="en"
   title="Returns a list of dates with the number of posts at each date."
/>

  <request>
   <param name="tag" style="form">
    <doc xml:lang="en" title="Filter by this tag." />
   </param>
  </request>
  <response>
   <representation mediaType="application/xml" element="dates">
    <param name="tag" style="plain" path="@tag" />

    <param name="user" style="plain" path="@user" />
    <param name="count" style="plain" path="date/@count" />
    <param name="dt" style="plain" path="date/@date">
    <link href="#getPosts" rel="" rev="" />
    </param>
   </representation>
  </response>
 </method>

 <!-- "posts" methods part II: ways of manipulating posts -->

 <method id="addPost" name="GET">
  <doc xml:lang="en" title="Add a post to del.icio.us" />
  <request>
   <param name="url" required="true" style="form">
    <doc xml:lang="en" title="The URL of the item." />
   </param>

   <param name="description" required="true" style="form">

    <doc xml:lang="en" title="The description of the item." />
   </param>

   <param name="extended" style="form">
    <doc xml:lang="en" title="Notes for the item." />
   </param>

   <param name="tags" style="form">
    <doc xml:lang="en" title="Tags for the item.">Space delimited</doc>

   </param>

   <param name="dt" style="form">
   <doc xml:lang="en" title="Datestamp of the item.">
     Format: "CCYY-MM-DDThh:mm:ssZ". Requires a LITERAL "T" and "Z"
     like in
     <html:a href="http://www.cl.cam.ac.uk/~mgk25/iso-
time.html">ISO8601</html:a>.
     For example: "1984-09-01T14:21:31Z"
    </doc>
   </param>
```

```
   <param name="replace" default="" style="form">
    <doc xml:lang="en"
     title="Unless set to &quot;no&quot;, a post will overwrite an
            earlier post with the same URL." />

    <option value="" />
    <option value="no" />
   </param>

   <param name="shared" style="form">
    <doc xml:lang="en" title="Set to &quot;no&quot; to make the item
private." />
    <option value="" />
    <option value="no" />
   </param>

  </request>

  <response>
   <representation href="#resultCode" />
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <method id="deletePost" name="GET">
  <doc xml:lang="en" title="Delete a post from del.icio.us" />

  <request>
   <param name="url" required="true" style="form">
    <doc xml:lang="en" title="The URL of the item." />
   </param>
  </request>

  <response>
   <representation href="#resultCode" />
   <fault href="#AuthorizationRequired" />

  </response>
 </method>

 <!-- "bundles" methods -->

 <method id="getBundles" name="GET">
  <doc xml:lang="en" title="Retrieve all of a user's bundles." />

  <response>
   <representation mediaType="application/xml" element="bundles">

    <param name="name" style="plain" path="bundle/@name" />
    <param name="tags" style="plain" path="bundle/@tags" />
   </representation>
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <method id="makeBundle" name="GET">
  <doc xml:lang="en" title="Assign a set of tags to a single bundle.">
   Wipes away previous settings for bundle.
  </doc>

  <request>
```

```
   <param name="bundle" style="form">
    <doc xml:lang="en" title="The bundle name." />
   </param>
   <param name="tags" required="true" style="form">
    <doc xml:lang="en" title="List of tags.">Space-separated.</doc>
   </param>
  </request>

  <response>
   <representation href="#result" />
   <fault href="#AuthorizationRequired" />
  </response>
 </method>

 <method id="deleteBundle" name="GET">
  <doc xml:lang="en" title="Deletes a bundle." />
  <request>

   <param name="bundle" style="form">
    <doc xml:lang="en" title="The bundle name." />
   </param>
  </request>

  <response>
   <representation href="#resultCode" />
   <fault href="#AuthorizationRequired" />
  </response>

 </method>

 <!-- Commonly used representations -->

 <representation id="postList" mediaType="text/xml" element="posts">
    <param name="tag" style="plain" path="@tag" />
    <param name="user" style="plain" path="@user" />

    <param name="description" style="plain" path="post/@description" />
    <param name="url" style="plain" path="post/@href" >
       <link href="#getPosts" rel="" rev="" />
    </param>
    <param name="tag" style="plain" path="post/@tag">
       <link href="#getPosts" rel="" rev="" />
    </param>
    <param name="dt" style="plain" path="post/@time"  >
       <link href="#getPosts" rel="" rev="" />
    </param>
 </representation>

 <representation id="result" mediaType="application/xml" element="result">
    <param name="result" style="plain" path="." />
 </representation>

 <representation id="resultCode" mediaType="application/xml"
element="result">
    <param name="tag" style="plain" path="@code" />
 </representation>

 <fault id="AuthorizationRequired" status="401" />

</application>
```

Apêndice E – Descrição no formato WADL da API do Technorati

```
<?xml version="1.0"?>
<!-- This is a bootleg WADL file for the technorati API.                -->
<!--                   September 18, 2008 - Mauricio Bomfim & Joao Assis -->

<application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns="http://research.sun.com/wadl/2006/07">

 <resources base="http://api.technorati.com">
    <resource path="cosmos"><method href="#Cosmos" /></resource>
    <resource path="search"><method href="#Search" /></resource>
    <!--
    <resource path="tag"><method href="#Tag" /></resource>
    <resource path="dailycounts"><method href="#DailyCounts" /></resource>
    <resource path="toptags"><method href="#TopTags" /></resource>
    <resource path="bloginfo"><method href="#BlogInfo" /></resource>
    <resource path="blogposttags"><method href="#BlogPostTags"
/></resource>
    <resource path="getinfo"><method href="#GetInfo" /></resource>
    -->
 </resources>

 <!-- Methods -->

 <!-- "cosmos" method -->

 <method id="Cosmos" name="GET">
  <request>
      <param name="key" required="true" style="form">
      <doc xml:lang="en" title="Technorati API key." />
      </param>
      <param name="url" required="true" style="form">
      <doc xml:lang="en" title="The base target URL for which you are
searching." />
      </param>

      <param name="type" style="form">
      <doc xml:lang="en" title="" />
      <option value="link">
         <doc>returns the freshest links referencing your target URL.</doc>
      </option>
      <option value="weblog">
         <doc>returns the last set of unique weblogs referencing your
target URL.</doc>
      </option>
      </param>

      <param name="limit" style="form">
      <doc xml:lang="en" title="Set this to a number larger than 0 and
smaller or equal to 100 and it will return limit number of links for a
query." />
      </param>

      <param name="start" style="form">
```

```
      <doc xml:lang="en" title="Set this to a number larger than 0 and
you'll get the start+20 freshest items (links or blogs)" />
      </param>

      <param name="current" style="form">
      <doc xml:lang="en" title="" />
      <option value="yes">
         <doc>Technorati returns links that are currently on a weblog's
homepage.</doc>
      </option>
      <option value="no">
         <doc>Returns all links to the given URL regardless of their
current placement on the source blog.</doc>
      </option>
      </param>

      <param name="format" style="form">
      <doc xml:lang="en" title="This allows you to request an output
format, which by default is set to xml." />
      </param>

      <param name="claim" style="form">
      <doc xml:lang="en" title="Set this to 1 to have each link result
embellished with any user information associated with a link result's
parent blog." />
      </param>

      <param name="highlight" style="form">
      <doc xml:lang="en" title="The default setting of 1 highlights the
citation of the given URL within the weblog excerpt. Set this parameter to
0 to apply no special markup to the blog excerpt." />
      </param>
  </request>

  <response>
      <representation mediaType="application/xml" element="tapi">

      <param name="searchurl"    style="plain" path="document/result/url"
/>
      <param name="name"         style="plain"
path="document/result/weblog/name" />
      <param name="url"          style="plain"
path="document/result/weblog/url" />
      <param name="rssurl"       style="plain"
path="document/result/weblog/rssurl" />
      <param name="atomurl"      style="plain"
path="document/result/weblog/atomurl" />
      <param name="inboundblogs" style="plain"
path="document/result/weblog/inboundblogs" />
      <param name="inboundlinks" style="plain"
path="document/result/weblog/inboundlinks" />
      <param name="lastupdate"   style="plain"
path="document/result/weblog/lastupdate" />
      <param name="rank"         style="plain"
path="document/result/weblog/rank" />

      <param name="name"         style="plain"
path="document/item/weblog/name" />
      <param name="url"          style="plain"
path="document/item/weblog/url" />
```

```
      <param name="rssurl"       style="plain"
path="document/item/weblog/rssurl" />
      <param name="atomurl"      style="plain"
path="document/item/weblog/atomurl" />
      <param name="inboundblogs" style="plain"
path="document/item/weblog/inboundblogs" />
      <param name="inboundlinks" style="plain"
path="document/item/weblog/inboundlinks" />
      <param name="lastupdate"   style="plain"
path="document/item/weblog/lastupdate" />

      <param name="nearestpermalink" style="plain"
path="document/item/nearestpermalink" />
      <param name="excerpt"          style="plain"
path="document/item/excerpt" />
      <param name="linkcreated"      style="plain"
path="document/item/linkcreated" />
      <param name="linkurl"          style="plain"
path="document/item/linkurl" />
      </representation>

      <fault href="#Error" />
  </response>
 </method>

 <!-- "search" method -->

 <method id="Search" name="GET">
  <request>
      <param name="key" required="true" style="form">
      <doc xml:lang="en" title="Technorati API key." />
      </param>
      <param name="query" required="true" style="form">
      <doc xml:lang="en" title="Set this to the words you are searching
for. Separate words with '+' as usual." />

      </param>

      <param name="format" style="form">
      <doc xml:lang="en" title="This allows you to request an output
format, which by default is set to xml." />
      </param>

      <param name="language" style="form">
      <doc xml:lang="en" title="Set this to a ISO 639-1 two character
language code to retrieve results specific to that language." />
      </param>

      <param name="authority" style="form">
      <doc xml:lang="en" title="Set this to filter results to those from
blogs with at least the Technorati Authority specified." />
      <option value="n">
         <doc>Any authority: All results.</doc>
      </option>
      <option value="a1">
         <doc>A little authority: Results from blogs with at least one
link.</doc>
      </option>
      <option value="a4">
         <doc>Some authority: Results from blogs with a handful of
links.</doc>
```

```
      </option>
      <option value="a7">
         <doc>A lot of authority: Results from blogs with hundreds of
links.</doc>
      </option>
      </param>

      <param name="start" style="form">
      <doc xml:lang="en" title="Set this to a number larger than 0 and
you'll get the start+20 freshest items (links or blogs)" />
      </param>

      <param name="limit" style="form">

      <doc xml:lang="en" title="Set this to a number larger than 0 and
smaller or equal to 100 and it will return limit number of links for a
query." />
      </param>

      <param name="claim" style="form">
      <doc xml:lang="en" title="Set this to 1 to have each link result
embellished with any user information associated with a link result's
parent blog." />
      </param>
  </request>

  <response>
      <representation mediaType="application/xml" element="tapi">

      <param name="query_string"  style="plain"
path="document/result/query" />
      <param name="matches"       style="plain"
path="document/result/querycount" />
      <param name="duration"      style="plain"
path="document/result/querytime" />
      <param name="ranking_start" style="plain"
path="document/result/rankingstart" />

      <param name="name"         style="plain"
path="document/item/weblog/name" />

      <param name="url"          style="plain"
path="document/item/weblog/url" />
      <param name="rssurl"       style="plain"
path="document/item/weblog/rssurl" />
      <param name="atomurl"      style="plain"
path="document/item/weblog/atomurl" />
      <param name="inboundblogs" style="plain"
path="document/item/weblog/inboundblogs" />
      <param name="inboundlinks" style="plain"
path="document/item/weblog/inboundlinks" />
      <param name="lastupdate"   style="plain"
path="document/item/weblog/lastupdate" />

      <param name="title"     style="plain" path="document/item/title" />
      <param name="excerpt"   style="plain" path="document/item/excerpt" />

      <param name="created"   style="plain" path="document/item/created" />
      <param name="permalink" style="plain" path="document/item/permalink"
/>
      </representation>
```

```
      <fault href="#Error" />
   </response>
  </method>

  <!-- Commonly used representations -->

  <fault id="Error" status="401">
      <representation mediaType="application/xml" element="tapi">
      <param name="errormsg" style="plain" path="document/result/error" />
      </representation>
  </fault>

</application>
```