UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

UTILIZAÇÃO DE OBJETOS DE APRENDIZAGEM SCORM COM A REDE SOCIAL ORKUT

LEONARDO RAFAEL MORASTONI

BLUMENAU
2011

2011/1-25

LEONARDO RAFAEL MORASTONI

UTILIZAÇÃO DE OBJETOS DE APRENDIZAGEM SCORM COM A REDE SOCIAL ORKUT

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos , Doutor - Orientador

BLUMENAU
2011

2011/1-25

UTILIZAÇÃO DE OBJETOS DE APRENDIZAGEM SCORM COM A REDE SOCIAL ORKUT

Por

LEONARDO RAFAEL MORASTONI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Mauro Marcelo Mattos, Doutor – Orientador, FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB
Membro:	Prof. Adilson Vahldick, Mestre – UDESC

Blumenau, 30 de junho de 2011

Dedico esse trabalho à memória de Norita Vicente Morastoni, minha mãe, por me dar a vida. A Augusta Vicente, minha nona, por seu amor.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À toda minha família, que esteve presente em todos os momentos especialmente ao meu tio Aldírio Vicente e a nona Augusta pelo apoio incondicional em todos os momentos de minha vida.

Ao meu orientador, Mauro Marcelo Mattos, pelo conhecimento transmitido e modelo a ser seguido.

Ao professor Adilson Vahldick por todo o apoio e compreensão no desenvolvimento desse trabalho.

A minha namorada Ana, pela presença e compreensão.

Aos meus amigos e pessoas que de certa forma contribuíram para realização desse trabalho.

Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes.

Isaac Newton

RESUMO

A web é muito mais interessante quando é possível construir aplicativos que interajam facilmente com amigos e colegas. Essa tendência também criou uma lista crescente de APIs específicas para sites, que os desenvolvedores precisam aprender. A OpenSocial define uma API comum para desenvolver aplicativos sociais que irão funcionar em diversos sites. O objetivo final dessa API é que qualquer site de relacionamentos seja capaz de implementar a API e hospedar aplicativos sociais de terceiros. Este trabalho apresenta uma forma de utilizar redes sociais através da API OpenSocial criando um ambiente de aprendizagem gerenciável que utiliza objetos de aprendizagem padronizados segundo a especificação SCORM. O objetivo principal desse ambiente é possibilitar aos usuários de redes sociais aprender, dividir e difundir seu conhecimento pela rede social. O ambiente permite adição de objetos SCORM, criar e remover cursos bem como seus usuários. Foi utilizado o LMS denominado BRUCE e realizadas as adaptações para utilizar os recursos da API OpenSocial.

Palavras-chave: Rede Social. API. OpenSocial. SCORM.

ABSTRACT

The web is more interesting when is possible to build apps that easily interact with friends and colleagues. But with the trend towards more social applications also comes a growing list of site-specific APIs that developers must learn. OpenSocial defines a common API for social applications across multiple websites. The ultimate goal is for any social website to be able to implement the API and host 3rd party social applications. This work shows a way of how social networks can be used through OpenSocial API creating a learning management system, that uses learning objects that follows SCORM specifications and standardize. The primary purpose of this environment is make possible that users learn, share and disseminate their own knowledge through social network. The environment permits adding SCORM object, create and remove courses as well their users. It was used a LMS called BRUCE and made changes on it to support OpenSocial API resources.

Key-words: Social network. OpenSocial. SCORM.

LISTA DE ILUSTRAÇÕES

Quadro 1 – Características da topologia de redes 19
Quadro 2 – Características dos *sites* sociais 24
Figura 3 – Estrutura Social Mashup 26
Quadro 3 – Exemplo de visualização de perfil e tela 27
Quadro 4 – Exemplo de owner e viewer 28
Figura 4 – Retorno exemplo de *owner* e *viewer* 28
Figura 5 – Estrutura de Aplicação social 30
Figura 6 – Fluxo de requisição 31
Figura 7 – Estrutura Social Website 31
Quadro 5 – Listagem da classe BruceWeb 32
Figura 8 – Estrutura Modular do protótipo de sistema de gerenciamento de aprendizagem virtual interativa (LMS) 33
Quadro 6 – Algumas pessoas que podem causar problemas de segurança e os motivos para fazê-lo 34
Quadro 7 – Requisitos funcionais 37
Quadro 8 – Requisitos não funcionais 37
Figura 9 – Diagrama de casos de uso do usuário 38
Quadro 9 – Detalhamento do caso de uso Registrar 38
Quadro 10 – Detalhamento do caso de uso Administrar pacote 39
Quadro 11 – Detalhamento do caso de uso Compartilhar pacote 40
Quadro 13 – Detalhamento do caso de uso Listar objetos 40
Quadro 14 – Detalhamento do caso de uso Selecionar objeto dos amigos 40
Quadro 15 – Detalhamento do caso de uso Selecionar objeto para interagir 40
Quadro 16 – Detalhamento do caso de uso Registrar objetos públicos ou sugeridos 41
Quadro 17 – Diagrama de caso de detalhado de administração de usuário 41
Quadro 18 – Diagrama de caso de detalhado de administração de pacote 41
Figura 12 – Diagrama de atividades do nível administrador 42
Figura 13 – Diagrama de atividades do nível de usuário 42
Figura 14 – Diagrama de classes 44
Figura 15 – Modelo entidade relacionamento 45
Figura 16 – Exemplo de objetos do *gadget* 48

Quadro 19– Desenvolvimento do *gadget* 49
Figura 17 – Tela de saudação do *gadget* 50
Quadro 20 – Implementação da página de saudação 52
Figura 18 – Página de importação de curso 53
Quadro 21 – Implementação da página de importação de curso 55
Figura 19 – Tela inicial do *gadget* de nível *admin* 56
Figura 20 – Tela inicial do *gadget* de nível usuário 57
Figura 21 – Tela de cursos registrados 57
Figura 22 – Tela de registro de curso 58
Figura 23 – Tela de manutenção de curso 59
Figura 24 – Tela de importação de curso 59
Figura 25 – Tela de gerenciamento de usuários 60
Quadro 22 – Características do BRUCE e dos trabalhos correlatos 61

LISTA DE SIGLAS

API – *Application Programming Interface*

BRUCE – *Basic Reference of Using Celine*

HTML – *Hyper Text Markup Language*

LMS – *Learning Management System*

SCORM – *Sharable Content Object Reference Model*

XML – *eXtensible Markup Language*

WWW – *World Wide Web*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS DO TRABALHO 14
1.2 ESTRUTURA DO TRABALHO 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 REDES SOCIAIS 16
2.1.1 Atores 18
2.1.2 Conexões 18
2.1.3 Topologias de redes sociais na internet 18
2.1.4 Dinâmicas das redes sociais na internet 19
2.1.4.1 Cooperação, Competição e Conflito 20
2.1.4.2 Ruptura e Agregação 21
2.1.4.3 Adaptação e auto-organização 21
2.1.5 Tipos de redes sociais na internet 22
2.1.5.1 Redes Sociais Emergentes 22
2.1.5.2 Redes de filiação ou Redes Associativas 22
2.1.6 *Sites* de redes sociais 23
2.1.7 Comunidades em redes sociais 23
2.1.8 Comunidades Virtuais 23
2.1.9 Sobre os *sites* de redes sociais 24
2.2 OPENSOCIAL 25
2.2.1 Social Mashup 26
2.2.2 Social Application 29
2.2.2.1 Pré-requisitos 29
2.2.3 Social Website/Social Mobile Application 31
2.3 BRUCE 32
2.4 CONCEPÇÃO DE UM LMS 33
2.5 TRABALHOS CORRELATOS 34
2.5.1 SCORM Sample RTE 35
2.5.2 ReadingSocial 35
2.5.3 UdutuTeach 36
3 DESENVOLVIMENTO 37

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 37
3.2 ESPECIFICAÇÃO 38
3.2.1 Diagrama de casos de uso 38
3.2.2 Diagrama de Atividades 41
3.2.3 Diagrama de Classes 43
3.2.4 MODELO ENTIDADE RELACIONAMENTO 44
3.3 IMPLEMENTAÇÃO 45
3.3.1 Técnicas e ferramentas utilizadas 46
3.3.1.1 Desenvolvimento do *gadget* BRUCE 46
3.3.1.2 Desenvolvimento do LMS 50
3.3.2 Operacionalidade da implementação 55
3.4 RESULTADOS E DISCUSSÃO 60
4 CONCLUSÕES 63
4.1 EXTENSÕES 64
REFERÊNCIAS BIBLIOGRÁFICAS 65

1 INTRODUÇÃO

Atualmente, em função das necessidades sociais de um mundo interligado através de grandes redes estão sendo acelerados os processos de mudança, transformando a economia, globalizando processos, rompendo barreiras e diminuindo distâncias (SCHLEMMER; SACCOL; GARRIDO, 2006, p. 1). Além de destruir barreiras, foi necessário encontrar uma forma de disseminar o conhecimento na mesma velocidade dos processos de mudança e aproveitando o meio interligado das redes, o que aconteceu com a organização do conhecimento em repositórios *online*. Estes repositórios são organizados de forma acessível e organizada, alinhando com os atributos elencados por Schlemmer, Saccol e Garrido (2006, p. 2) que eram abrangência, a complexidade e a disposição em forma de rede conhecidos como ambientes de aprendizagem.

Segundo Santarosa (1999), a criação destes ambientes de aprendizagem são o reflexo mais atual deste novo enfoque da aprendizagem, fundamentada nas idéias de desenvolvimento cognitivo individual. Ainda diz-se sobre estes ambientes, que *Learning Management System* (LMS) são denominações utilizadas para softwares desenvolvidos para o gerenciamento da aprendizagem via web (SCHLEMMER, 2002, p. 8).

Os LMS constituem uma forma mais recente de sistemas, que se diferenciam dos tradicionais ambientes de aprendizagem por possuírem integração com outros tipos de sistemas de apoio (ADL, 2006, p. 23).

Baseando-se nas premissas descritas anteriormente e nos atributos definidos por Schlemmer, Saccol e Garrido (2006), neste trabalho desenvolveu-se um LMS utilizando objetos de aprendizagem como recursos de apresentação de conteúdos e atividades padronizados segundo a especificação *Sharable Content Object Reference Model* (SCORM).

SCORM é o nome da especificação que visa promover a criação de conteúdo reusável para aprendizagem como objetos instrucionais com uma estrutura técnica universal para o aprendizado baseado em computador e na web (ADL, 2006, p. 12).

CELINE é um componente que permite a execução de pacotes SCORM, assim como interferir durante a interação do estudante com estes pacotes (VAHLDICK; RAABE, 2008, p. 4). Este componente conectado à aplicação web intermedia vários recursos necessários para atender a estes objetivos.

Com o desenvolvimento das ferramentas tecnológicas, emergem na sociedade novas formas de relação, comunicação e organização das atividades humanas, entre elas, merecem

destaque o estudo de redes sociais virtuais (MACHADO; TIJIBOY, 2005, p. 2).

Segundo Machado e Tijiboy (2005, p. 3), os softwares sociais são programas que funcionam como mediadores sociais e que favorecem a criação de redes de relacionamentos através de espaços onde o usuário pode juntar pessoas do seu círculo de relacionamentos, conhecer outras que compartilhem os mesmos interesses e discutir temas variados, construindo diferentes elos entre os eu privado e o eu público.

Neste panorama foi utilizado como software social a *Application Programming Interface* (API) OpenSocial, uma plataforma de desenvolvimento para aplicativos que utilizam os conceitos de relacionamento de redes sociais, implementada pela Google (TAKAMOTO, NAVARRO, VAHL, 2009, p. 54) .

O serviço OpenSocial define uma API comum para desenvolver aplicativos sociais que irão funcionar em diversos *sites*. Elas possibilitam que os desenvolvedores criem aplicativos utilizando JavaScript e *Hyper Text Markup Language* (HTML) padrão para acessar amigos de uma rede social e atualizar *feeds*[1] (GOOGLE, 2009a).

Os *gadgets* são miniaplicativos criados com o uso de HTML, além de JavaScript, Flash ou Silverlight para um comportamento dinâmico. Como os *gadgets* podem ser executados em diversos sites e plataformas, há *tags* e bibliotecas especiais que funcionam em lugares diferentes (GOOGLE, 2009b).

Neste trabalho realizou-se comunicação entre o LMS e o *gadget*, criando assim um ambiente de gerenciamento de ensino integrado com *sites* de relacionamentos.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi criar um LMS que possa ser acessado por ambientes de redes sociais compatíveis com a OpenSocial API.

Os objetivos específicos do trabalho são:

a) disponibilizar um LMS que administre pacotes SCORM;

b) disponibilizar um *gadget* que se comunique com este LMS.

[1] Recurso de alguns *sites* que, aliado a um software específico, permite alertar os visitantes quando há conteúdo novo. Também conhecido como *feed Really Simple Syndication* (RSS) (FOLHAONLINE, 2009).

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos. No primeiro capítulo tem-se a introdução, a justificativa do trabalho, o objetivo geral e objetivos específicos. No segundo capítulo é exposta a fundamentação teórica, onde são abordados temas relevantes como Redes Sociais, OpenSocial e BRUCE, o LMS desenvolvido por Vahldick (2008), além de trabalhos correlatos. No capítulo terceiro é abordado o desenvolvimento do trabalho apresentando requisitos principais, especificação, implementação, técnicas e ferramentas utilizadas e resultados relativos ao LMS e o *gadget*. No último capítulo apresenta-se a conclusão final e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica necessária para compreensão do trabalho e a apresentação de uma aplicação similar ao sistema desenvolvido.

2.1 REDES SOCIAIS

Desde os primórdios o homem tem a necessidade de ampliar sua forma de interação. Aproximadamente 8000 a.C. apareceram os primeiros indícios de forma escrita da comunicação, desde então, as mudanças que ocorrerem com a humanidade exigem o rompimento de barreiras para aumentar seus horizontes. Associada a esse panorama, a evolução tecnológica e do conhecimento proporcionaram os meios que efetivamente contribuíram para transpor as barreiras entre as nações (RECUERO, 2009).

De acordo com Recuero (2009) a rede de computadores interconectados (internet) assume um papel fundamental nesse propósito. A internet surgiu na década de 90 como conseqüência de uma situação de competição e monitoramento entre Estados Unidos e União Soviética. A expansão mundial deste sistema ficou conhecida como *World Wide Web* (www). Esta inovação possibilitou às pessoas a comunicação e a transmissão de informações de qualquer localidade com um nível de segurança e com um grau de confiabilidade suficiente para dar credibilidade a esse meio de interação sem a necessidade da presença física.

A internet possibilitou, no momento de sua criação, apenas a transferência de dados. Com o aprimoramento da sua utilização, foi gradativamente evoluindo para formas mais próximas de comunicação, nas quais as pessoas interagem de maneira compartilhada e conjunta (RECUERO, 2009).

Um reflexo da evolução da internet, foi a criação de ambientes de aprendizagem, nascidos para disseminar o conhecimento e organizá-lo em repositórios on-line. Os mesmo ainda podem ter auxilio de outras ferramentas de apoio para que os conhecimentos extraídos desses ambientes sejam aproveitados ao máximo.

Segundo Aranha (2004), neste panorama surgiram os ambientes de conhecimento e lazer denominados redes sociais. O primeiro evento que caracteriza este tipo de comunicação foi a lista de e-mails que viabilizou a criação de comunidades, utilizada para socializar uma

informação de modo informal, dinâmico e rápido. Esse modelo avançou e resultou em uma ferramenta para criação de um perfil pessoal que difundia a idéia de interação entre pessoas que permitissem publicar seus dados pessoais e adicionar mais usuários. Apesar de sua curta duração, aproximadamente três anos, instigou a criação de outras redes sociais.

A dinâmica das redes sociais atuais tornou-se de uso tão intenso que atinge a vida da grande maioria da população. Ela foi adicionada ao cotidiano da maioria das pessoas que a utilizam para as mais variadas finalidades. As redes sociais, além do perfil, propiciam compartilhamentos de fotos, vídeos, entre outros, criando espaços onde outros usuários podem acessar e participar. A rede é utilizada também como meio de trabalho para muitas profissões, como por exemplo, jornalistas que escrevem suas colunas disponibilizando-as para leituras e comentários. Empresas utilizam essas redes como divulgação de seus produtos e serviços e de um modo geral, as pessoas utilizam a rede para contribuir, expandir informações. A cultura e o lazer também são estimulados por meio da rede social, na medida em que seus usuários acessam e compartilham jogos, filmes, livros, entre outros (BOYD; ELLISON, 2007).

Para Recuero (2004) o interesse no estudo de redes complexas permeia todo o século XX. Iniciado pelas ciências exatas, matemáticos e físicos trouxeram as maiores contribuições para o estudo das redes, que depois foram absorvidas pela sociologia, na perspectiva da análise estrutural das redes sociais.

Na tentativa de elucidar o conceito de rede social, Aguiar (2007) apresenta a definição na visão de diversos autores.

> Redes sociais são, antes de tudo, relações entre pessoas, estejam elas interagindo em causa própria, em defesa de outrem ou em nome de uma organização, mediadas ou não por sistemas informatizados; são métodos de interação que sempre visam algum tipo de mudança concreta na vida das pessoas, no coletivo e/ou nas organizações participantes (AGUIAR, 2007).

Recuero (2007) com base em diversos autores, pondera que a rede social é “um conjunto de dois elementos: atores (pessoas, instituições ou grupos – os nós da rede) e suas conexões (interações ou laços sociais). Neste aspecto, argumenta que a rede é uma metáfora que permite observar os padrões das conexões de grupos sociais que são estabelecidos nas conexões entre os atores desses grupos.

2.1.1 Atores

Os atores são os nós ou nodos, representados pelas pessoas envolvidas na rede. A partir de sua interação é que se formam as estruturas sociais por meio da interação e dos laços sociais criados nesta interação. Por causa do distanciamento, inerente na comunicação mediada pelo computador (CMC), os atores não são imediatamente discerníveis, portanto, lida-se com construções feitas a partir de si no ciberespaço. Desta um ator pode ser representado por um *weblog*, *fotolog*, *twitter*, o*rkut*, entre outros (RECUERO, 2009).

2.1.2 Conexões

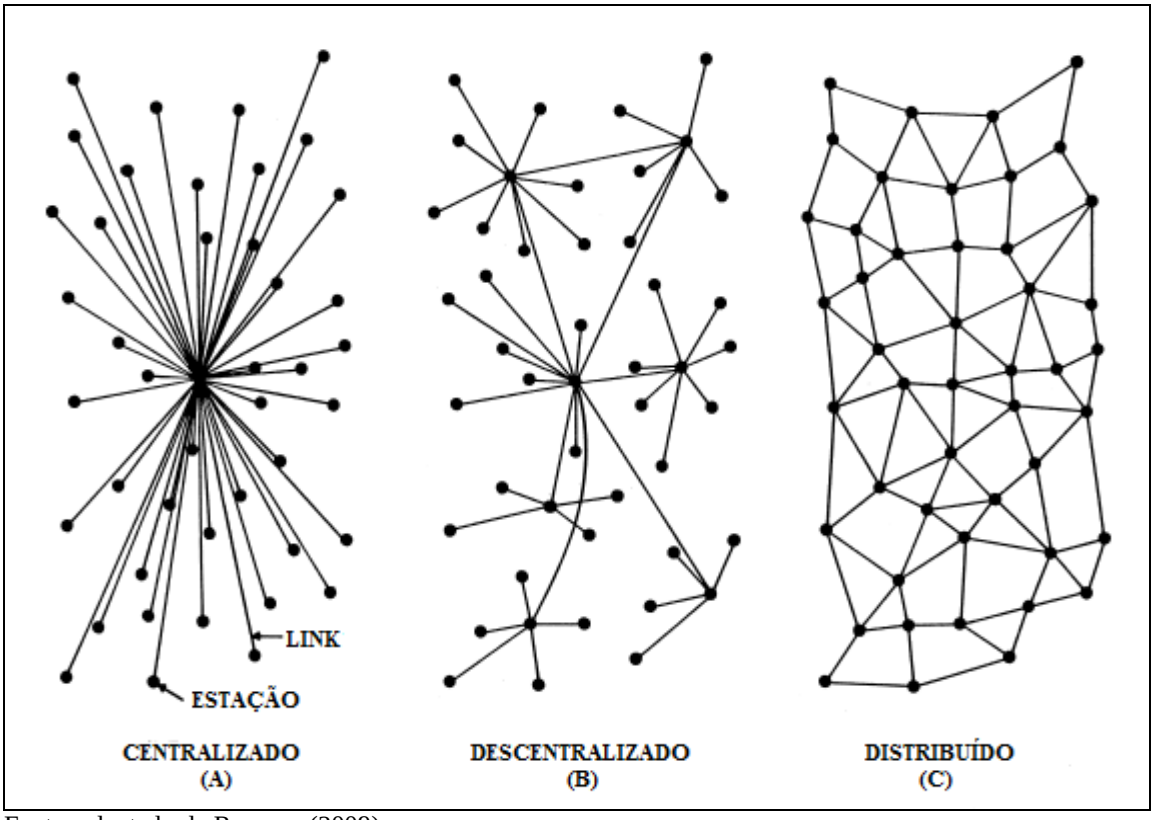
As conexões de uma rede são constituídas dos laços sociais, os quais são formados a partir da interação social entre os atores, assim, podem ser consideradas o principal elemento no estudo das redes sociais, visto que, a alteração da estrutura dos grupos depende da variação das conexões. (RECUERO,2009).

2.1.3 Topologias de redes sociais na internet

As redes apresentam uma estrutura formata a partir das interações ou laços sociais estabelecidos entre os atores. Franco (2008 apud RECUERO, 2009) argumenta que as topologias de redes são importantes para a compreensão das redes sociais. Neste sentido, apoiando-se em estudos de Baran (1964 apud RECUERO, 2008) apresenta três tipos de topologias de redes: distribuída, centralizada e descentralizada como poder ser visualizado no Quadro 1.

Tipo de Topologia	Característica
Distribuída	Todos os nós possuem mais ou menos a mesma quantidade de conexões, sem apresentar valoração hierárquica desses nós.
Centralizada	Um nó centraliza a maior parte das conexões. Apresenta o formato de estrela (observado na Figura 1)
Descentralizada	Possui vários centros, não é mantida por um único nó, mas por um grupo pequeno de nós, conecta vários outros grupos.

Quadro 1 – Características da topologia de redes



Fonte: adaptado de Recuero (2009).

Figura 1 – Topologia de redes

Essas topologias são interessantes para o estudo de vários elementos das redes sociais, tais como os processos de difusão de informações, porém, trata-se de modelos fixos e uma mesma estrutura de rede social pode apresentar características de vários deles. (RECUERO, 2009).

2.1.4 Dinâmicas das redes sociais na internet

As redes sociais, mesmo na internet, sofrem alterações, modificações, ao longo tempo e esse movimento acaba contribuindo na formação de padrões para a compreensão da rede. Neste sentido Recuero (2010, p.79) afirma que "essas dinâmicas são dependentes das interações que abarcam uma rede e podem influenciar diretamente na sua estrutura." Watts (2003 apud RECUERO, 2009) argumenta que as redes não param no tempo e no espaço, estão em constante transformação propiciadas pelas interações. Assim, essas transformações podem estar embasadas em interações para somar e construir um determinado laço ou para enfraquecer ou destruir outro laço.

Recuero (2009) aponta que o processo dinâmico nas redes sociais é considerado um processo emergente e capaz de impactar sua estrutura. Neste aspecto, a autora elenca alguns elementos importantes para a compreensão deste processo.

2.1.4.1 Cooperação, Competição e Conflito

A cooperação, competição e conflito são elementos dinâmicos que influenciam a rede social denominados de processos sociais. Processo social é definido por Ogburn e Nimkoff (1975 apud RECUERO, 2009,p.81) como a interação que existe entre os homens. Neste sentido os autores afirmam que: "interação social é compreendida como geradora de processos sociais a partir de seus padrões na rede, classificados em competição, cooperação e conflito."

A cooperação é a base para que exista uma estrutura social, na medida em que é refletido um modo de agir organizado. Assim a cooperação é gerada pelo interesse individual, pelo capital social envolvido e pelas finalidades do grupo.

Já a competição, de certa forma pode ser entendida como a vontade de superar o outro. Nesse sentido pode ser percebida como cooperação de um determinado grupo para superar o outro grupo. (RECUERO, 2009).

O conflito gera situações confusas que contribuem para o desgaste e ruptura de uma rede social, podendo inclusive estar associado à agressão e à violência.

De acordo com Recuero (2009, p.82) "a cooperação, a competição e o conflito não são, necessariamente, processos distintos e não relacionados. São sim, fenômenos naturais emergentes das redes sociais." Em síntese, pode-se dizer que enquanto a cooperação contribui para formação e manutenção da estrutura social, a competição e o conflito apontam para o desequilíbrio dessa estrutura. Desta forma, a cooperação está vinculada à criação dos grupos na internet, que dependem da cooperação dos atores participantes na atualização, leituras de comentários e dividir as informações como meios de manutenção para que possam continuar a existir. Pode-se citar como exemplo: *weblogs* coletivos, *fotologs*, entre outros.

2.1.4.2 Ruptura e Agregação

É comum em uma rede social, as pessoas aderirem e romperem nas interações com grupos sociais. Essa dinâmica é esperada e classificada pelos estudiosos das redes como clusterização. Neste sentido, Barabási e Albert (1999 apud RECUERO, 2009, p.86) afirmam que existem atores que executam muito mais conexões com outras pessoas, do que a maioria dos outros atores do grupo. Esses indivíduos são denominados conectores, e eles existem em todas as redes. "Conectores são um componente extremamente importante de nossas redes sociais. Eles criam tendências e modas, fazem negócios importantes, espalham boatos ou auxiliam a lançar um restaurante" (RECUERO, 2009, p.87).

Sendo os conectores 'espalhadores' das informações em um grupo, assumem papel fundamental na topologia das redes. A clusterização, tende a produzir nós muito mais densos do que o restante da rede, provocando o aparecimento de comunidades.

Em redes sociais onde há conflitos, é natural que ocorra outro movimento dinâmico: a ruptura, considerada uma ação 'natural' por estudiosos do assunto. Há outros autores que afirmam que existem um limite na quantidade de conexões que uma pessoa é capaz de manter. Dunbar (1993 apud RECUERO, 2009) discute que por questões biológicas, o limite está em torno de 150 conexões (conhecido como 'Dunbar number'). Neste contexto, considera-se que o conflito é tão importante quanto a cooperação para garantir um tamanho de grupo em que todos os membros possam interagir socialmente.

2.1.4.3 Adaptação e auto-organização

Parson (1969) e Holland (1996 apud RECUERO, 2009) afirma que a adaptação é uma questão a ser resolvida nos sistemas sociais. Esses autores afirmam ainda que a construção de novas estruturas, pelos sistemas sociais, aumentam sua capacidade adaptativa, além disso, os sistemas sociais mais evoluídos, são mais adaptativos, consideram, ainda, a adaptação uma evolução dos sistema. O conceito de adaptação está diretamente ligado ao de auto-organização. A mediação pelo computador, por exemplo, gerou outras formas de estabelecimento de relações sociais.

> As pessoas adaptaram-se aos novos tempos utilizando a rede para formar novos padrões de interação e criando novas formas de sociabilidade e novas organizações

> sociais. Como essas formas de adaptação e auto-organização são baseadas em interação e comunicação, é preciso que exista circularidade nessa informações, para que os processos sociais coletivos possam manter a estrutura social e as interações possam continuar acontecendo. Como a comunicação mediada por computador proporciona que essas interações sejam transportadas a um novo espaço, que é o ciberespaço, novas estruturas sociais e grupos que não poderiam interagir livremente tendem a surgir (RECUERO, 2009, p. 89).

2.1.5 Tipos de redes sociais na internet

Segundo Recuero (2009) as redes analisadas pela internet podem ser de dois tipos: as redes emergentes e as redes de filiação ou redes de associação.

2.1.5.1 Redes Sociais Emergentes

São as redes que aparecem a partir das interações entre os atores sociais. Essas redes emergem através das trocas realizadas pela interação social e pela conversação através da mediação por computador. Essa forma seria caracterizada pela construção do grupo através da interação, por exemplo, nos comentários de um *weblog* ou *fotolog*. As redes emergentes, geralmente, são pequenas visto a necessidade de investimento financeiro e de tempo disponível para que as trocas sociais aconteçam (RECUERO, 2009, p. 103). No Orkut, por exemplo, podem ser vistas no livros de recados.

2.1.5.2 Redes de filiação ou Redes Associativas

Redes de apenas um conjunto de atores, porém, chamadas redes de dois modos. Isto se justifica devido ao estudo de um conjunto de eventos aos quais um ator pertence. As duas variáveis medidas são: os atores e eventos. Desta forma essas redes são constituídas de dois tipos de nós: os atores e os grupos, mantendo uma relação de pertencimento. “As redes sociais de filiação ou associativas na internet são aquelas derivadas das conexões “estáticas” entre os atores, ou seja, das interações reativas, como exemplo, a lista de amigos no Orkut.” (RECUERO, 2009, p. 103).

2.1.6 *Sites* de redes sociais

Segundo Recuero (2009,p.102) os sites de redes sociais (SRSs) "são uma consequência da apropriação das ferramentas de comunicação mediadas pelo computador pelos atores sociais." Neste caso, O Orkut, Facebook, etc., embora sejam frequentemente citados como exemplo, não são os únicos tipos de sites de redes sociais.

Esses sites são espaços utilizados para a expressão das redes sociais na internet, definidos como sistemas que permitem: 1) a construção de uma *persona* através de um perfil ou página pessoal; 2) a integração através de comentários; 3) a expressão pública da rede social de cada ator. (BOYD e ELLISON, 2007 apud RECUERO, 2009, p. 110).

A diferença entre os sites de redes sociais e outras formas de comunicação mediada por computador reside no modo de visibilidade, articulação e manutenção dos laços sociais estabelecidos no espaço off-line, como por exemplo, o flickr e o fotolog; os weblog, Twiter, Plurk, Orkut e Facebook. Esses site permitem criar um perfil das pessoas, mostram publicamente a rede social de cada ator e permitem interações.

Neste contexto, BOYD e ELLISON (2007 apud RECUERO, 2009), apresenta uma definição de apropriação (sistema utilizado para manter redes sociais e dar sentido as mesmas) e a estrutura – apresentando a principal característica de exposição pública da rede.

2.1.7 Comunidades em redes sociais

"A estrutura básica da comunidade da rede social é aquela de um cluster ou seja, de um aglomerado de nós com maior densidade de conexões" (RECUERO, 2009).

2.1.8 Comunidades Virtuais

"O processo de expansão das interações sociais começa com o surgimento dos meios de transporte e de comunicação". Neste sentido, cita-se o evento das cartas, telefone e outros meios de comunicação mediada, ocorrem trocas comunicacionais sem a presença física. (RECUERO, 2009, p. 135).

> As comunidades virtuais são agregados sociais que surgem da Rede (internet), quando uma quantidade suficiente de gente leva adiante essas discussões públicas durante um tempo suficiente, com suficientes sentimentos humanos, para formar redes de relações pessoais no ciberespaço. (RHEINGOLD, 1995 apud RECUERO, 2009, p. 137).

Assim, a comunidade virtual é composta por: discussões públicas, pessoas que se encontram e reencontram, ou continuam em contato para dar prosseguimento a discussão, o tempo e o sentimento. Esses elementos, através do ciberespaço formam redes de relações sociais, constituindo-se em comunidades (RECUERO, 2009).

2.1.9 Sobre os *sites* de redes sociais

Os *sites* sociais apresentam características bastante semelhantes mas que diferem na forma de sua utilização. O quadro 2 demonstra algumas destas características.

SITE	CARACTERISTICAS
ORKUT	Sistema criado por Orkut Buyukkokten – funcionário do Google – desenvolvido em 2001 e lançado em 2004. Este site combinava diversas características de sites de redes sociais anteriores como: criação do perfil focados no interesse, a criação de comunidades e a mostra dos membros da rede social de cada ator.
FOTOLOG	O fotolog é um sistema de publicação que possibilitam ao usuário publicar fotografias acompanhadas de pequenos textos e receber comentários.
FLIKER	Inicialmente criado para publicação de fotografias. Recentemente acrescentou a publicação de vídeos.
FACEBOOK	Criado por Mark Zuckerberg com o objetivo de focar em alunos que saiam do segundo grau (nos EUA) e aqueles que estavam entrando na universidade. Lançado em 2004, hoje é um dos sistemas de maior base de usuários do mundo.
MySPACE	Lançado em 2003, permitia a mostra de redes sociais e a interação com outros usuários através da construção de perfis, blogs, grupos e fotos, música e vídeos.
TWITTER	É um site popularmente denominado de um serviço de microblogging. Permite que sejam escritos pequenos textos. É estruturado com seguidores e pessoas a seguir onde cada twitter pode escolher quem deseja seguir e ser seguido por outros.
PLUNK	Semelhante ao twitter, permite aos usuários publicar mensagens que são visíveis a quem os segue (amigos e fãs).

Quadro 2 – Características dos *sites* sociais

2.2 OPENSOCIAL

Segundo Google (2009c), pode-se definir a OpenSocial como um conjunto de APIs JavaScript comuns para desenvolver aplicativos sociais. Este serviço define uma API que possibilita que os desenvolvedores criem aplicativos utilizando JavaScript e HTML padrão para a criação de redes sociais (GOOGLE, 2009a).

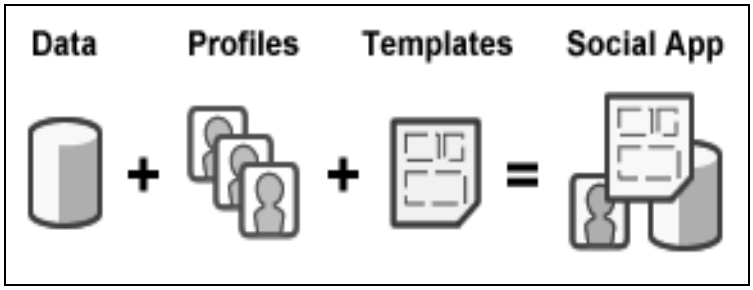
Esta nova plataforma tem como objetivos, além de reduzir o tempo de desenvolvimento do programador, também potencializar quantidade de usuários de uma aplicação (TAKAMOTO; NAVARRO; VAHL, 2009, p. 54).

Com foco nas pessoas, a API do OpenSocial permite o acesso a serviços da rede, assim sendo, possui três áreas de funcionamento (TAKAMOTO; NAVARRO; VAHL, 2009, p. 55):

a) pessoas e relacionamentos: representa uma parte muito importante do sistema, responsável por fornecer as informações do usuário e de seus relacionamentos com outros usuários do sistema;
b) atividades: representa as atividades de um usuário dentro do sistema para que seja mantido um histórico de suas ações e elas sejam divulgadas aos amigos e visitantes do perfil;
c) persistência: a API disponibiliza capacidade de persistência para que os aplicativos possam ler e gravar dados dos usuários específicos de cada aplicação.

O objetivo final é que qualquer *site* de relacionamentos seja capaz de implementar a API e hospedar aplicativos sociais de terceiros (GOOGLE, 2009a).

Segundo OpenSocial (2010), a OpenSocial fornece algumas formas para desenvolver aplicações. A forma escolhida dependerá dos requisitos de cada projeto. Contudo, a grande maioria dos projetos sociais tem uma estrutura comum. Na figura 2 pode-se visualizar essa estrutura.



Fonte: OPENSOCIAL (2010).

Figura 2 – Estrutura básica de um projeto social

Em sistemas baseados na OpenSocial, os componentes da figura 2, são providos de

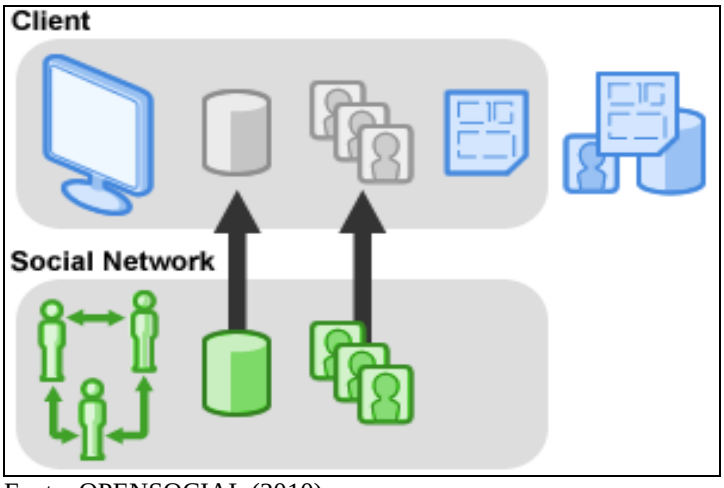
vários lugares. As aplicações clientes podem usar JavaScript para renderizar dados em um modelo. As redes sociais podem armazenar esses dados sociais e dados da aplicação, já o lado servidor pode aproveitar os bancos de dados e frameworks para produzir saídas renderizadas.

A API divide-se em três padrões para o desenvolvimento de aplicações, e cada padrão provê solução para todos os seus requisitos. Cada um dos padrões será aprofundado em seguida.

2.2.1 Social Mashup

Social Mashup é uma aplicação leve que executa internamente nas redes sociais. Essas aplicações normalmente apresentam um bom desempenho mas são limitadas em capacidade de armazenamento e processamento de dados. Uma Mashup é tipicamente criada utilizando HTML, JavaScript, CSS, OpenSocial Templates, e/ou Flash. Como é demonstrado na figura 3, o *gadget* retorna a seu usuário todas as requisições feitas por ele, sem necessidade de um servidor adicional.

Para construir uma Social Mashup, é necessário um local onde armazenar o arquivo XML de especificação do *gadget*. Este arquivo deve ficar em um local que tenha acesso pela web e não estar protegido por um *firewall*. Precisa-se também um local para testar o *gadget*. Normalmente as redes sociais disponibilizam um local para testes. No Orkut, esse local foi denominado *sandbox* e o mesmo pode ser acessado pelo link www.sandbox.orkut.com. Para acesso ao ambiente é necessário um cadastro na Google.



Fonte: OPENSOCIAL (2010).

Figura 3 – Estrutura Social Mashup

Para a criação da visualização de perfil e de tela do *gadget*, as aplicações baseadas na OpenSocial podem gerar várias interfaces para o contexto em que cada *gadget* se encontra.

Por exemplo, quando um usuário da rede social acessa o perfil de outro usuário, como visualização do *gadget* no perfil desse segundo usuário, é apresentado o tipo de visualização chamado perfil. Já quando um usuário adiciona ou utiliza um *gadget* é gerada a visualização de tela, muito diferente e mais detalhada que a visualização de perfil. Na visualização de tela é onde pode ser visto o *gadget* totalmente carregado. No quadro 3 é demonstrado a forma como as configurações são criadas.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
     <ModulePrefs title="Social Mashup Tutorial - Gifts">
     </ModulePrefs>
     <Content type="html" view="profile">
           <![CDATA[
               Hello, Gifts! (profile view)
             ]]>
     </Content>
     <Content type="html" view="canvas, default">
           <![CDATA[
               Hello, Gifts! (canvas view)
             ]]>
     </Content>
</Module>
```

Fonte: OPENSOCIAL (2010).

Quadro 3 – Exemplo de visualização de perfil e tela

A OpenSocial define 2 atores: o dono do *gadget* e o visualizador. O proprietário é quem gerencia o conteúdo de uma página particular, enquanto o visualizador está autenticado como usuário visualizando uma página.

De acordo com OpenSocial (2010) existem dois modos de solicitar informação de usuário, a primeira já se tem conhecimento do que se quer saber então a requisição já pode ser feita na especificação da *gadget*. Já na segunda forma de requisição necessita de dados do visualizador e a partir dos dados informados processar a página. Nesse caso é utilizada API JavaScript para recarregar a página automaticamente de acordo com a solicitação do usuário.

O quadro 4 demonstra um exemplo de proprietário do *gadget* e visualizador onde o *gadget* faz requisição a um XML passando as informações do *owner* (dono) e *viewer* (visualizador) e retornando essas informações. Como a requisição está na visualização perfil, somente se acessar o perfil de um usuário que tenha a *gadget* adicionada irá conseguir visualizar o retorno. O resultado pode ser visto na figura 4 o retorno do quadro 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ModulePrefs title="Social Mashup Tutorial - Gifts (owner and
viewer)">
    <Require feature="OpenSocial-data"/>
    <Require feature="OpenSocial-templates"/>
  </ModulePrefs>
  <Content type="html" view="profile">
    <![CDATA[
      <script type="text/os-data"
              xmlns:os="http://ns.OpenSocial.org/2008/markup">
      <os:ViewerRequest key="viewer" fields="displayName"/>
      <os:OwnerRequest key="owner" fields="displayName"/>
      </script>
      <script type="text/os-template"
              xmlns:os="http://ns.OpenSocial.org/2008/markup">
        Hello<os:If condition="${viewer.displayName!=null}">,
                              ${viewer.displayName}</os:If>!

        <br/> You're looking at ${owner.displayName}'s profile.
      </script>
    ]]>
  </Content>
  <Content type="html" view="canvas">
    <![CDATA[
      Hello, Gifts! (canvas view)
    ]]>
  </Content>
</Module>
```

Fonte: OPENSOCIAL (2010).

Quadro 4 – Exemplo de owner e viewer

Fonte: OPENSOCIAL (2010).

Figura 4 – Retorno exemplo de *owner* e *viewer*

2.2.2 Social Application

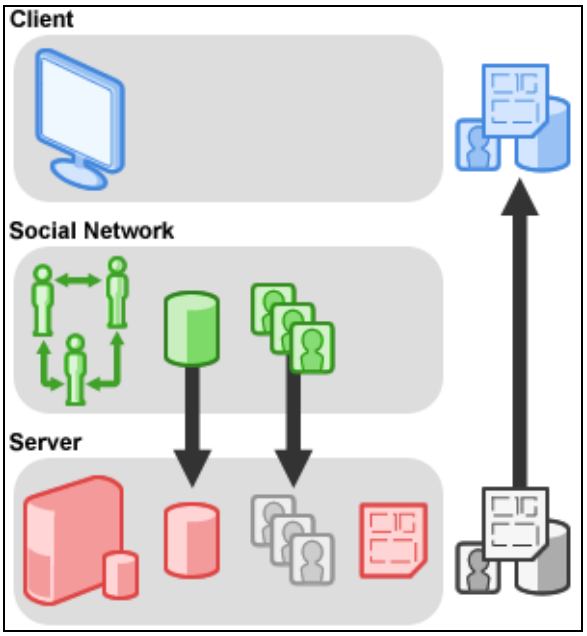
Segundo OpenSocial (2010), uma aplicação social executa na rede social mas se baseia em um servidor externo para processamento e renderização de dados. Esse tipo de aplicação provê funcionalidades avançadas, mas podem ter problemas de desempenho quando se tornam muito populares. Aplicações sociais podem ser criadas utilizando tecnologias como HTML, JavaScript, CSS, OpenSocial Templates, Flash, PHP, Python, Java, Perl, .NET, ou Ruby.

Muitas aplicações sociais combinam dados sociais obtidos nas redes sociais com dados de aplicações que estão hospedadas em servidor na web (OPENSOCIAL, 2010).

2.2.2.1 Pré-requisitos

Os pré-requisitos do Social Application são os mesmos pré-requisitos aplicados ao Social Mashup (item 2.2.1.1). Entretanto no Social Application, devido às requisições feitas ao servidor externo, além dos dados sociais permanecerem armazenados internamente, é necessário um local para armazenar os dados resultantes das requisições dos aplicativos sociais.

No modelo de aplicação social, a rede social envia os dados para o servidor externo e esse servidor fica responsável por apresentar ao usuário uma informação relativa aos dados recebidas da rede social. A estrutura desse tipo de aplicação pode ser verificado na figura 5.



Fonte: OPENSOCIAL (2010).

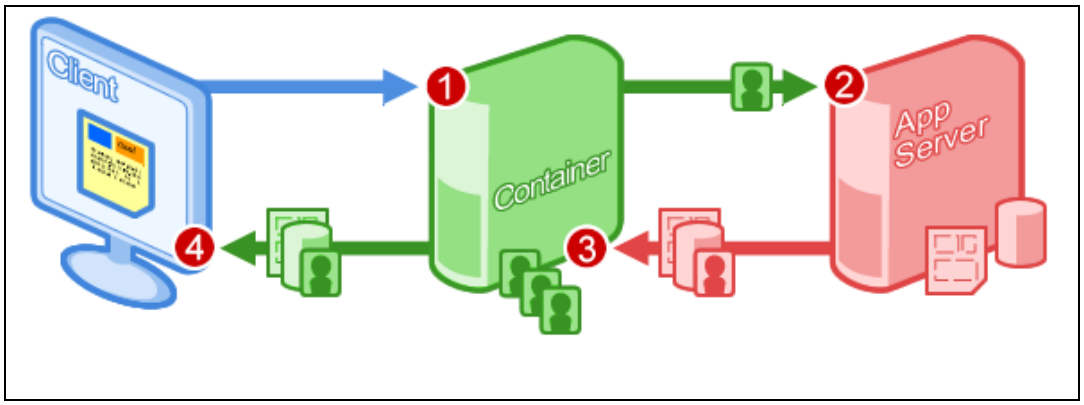
Figura 5 – Estrutura de Aplicação social

A figura 6 apresenta a forma como a requisição é processada. O processamento da requisição segue os seguintes passos:

a) o cliente solicita a visualização de uma aplicação;
b) a rede social envia os dados para o servidor remoto;
c) o servidor remoto combina dados sociais e de aplicação e retorna;
d) a rede social envia o conteúdo ao cliente.

Segundo OpenSocial (2010), para devolver a requisição, o servidor remoto precisa utilizar o método de segurança OAuth da OpenSocial. Esse método é subdividido em dois: OAuth2Legged e OAuth3Legged. O primeiro método é utilizado quando as requisições são recebidas de um *gadget* já publicado na rede social. Ele utiliza os dados de verificação obtidos na validação de propriedade do *gadget* para acessar a rede social. O modo OAuth3Legged faz acesso direto a rede social, não utilizando as informações do *gadget*, mas a autenticação de acesso do servidor a rede social.

O modo OAuth2Legged é largamente utilizado pelos desenvolvedores de *gadgets* por seu desempenho em autenticação e resposta às requisições.

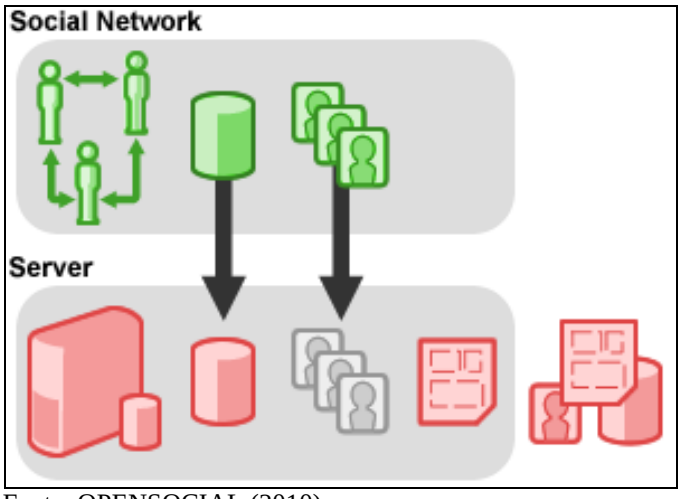


Fonte: OPENSOCIAL (2010).

Figura 6 – Fluxo de requisição

2.2.3 Social Website/Social Mobile Application

Sites sociais e aplicações móveis sociais não executam na rede social mas acessam-na através de RPC e REST APIs. Os usuários dessas aplicações garantem acesso aos dados sociais utilizando a autenticação OAuth3Legged sem ter que adicionar aplicativos em uma rede social. O *design* padrão garante muita flexibilidade, muitas plataformas e várias linguagens conseguem utilizar esse padrão. Na figura 7 pode-se visualizar a estrutura do social website. Percebe-se que as requisições são feitas pelo aplicativo que esta executando externamente e são devolvidas pela OpenSocial para que, após processadas, possam ser visualizadas pelo usuário.



Fonte: OPENSOCIAL (2010).

Figura 7 – Estrutura Social Website

2.3 BRUCE

Segundo Vahldick (2008), o LMS BRUCE foi criado com o propósito de servir de exemplo e documentação para os desenvolvedores que desejarem conhecer o componente CELINE.

Segundo Vahldick e Raabe (2008), o BRUCE foi montado contendo uma *servlet* que simplesmente verifica se o usuário está autenticado ou não. Caso esteja autenticado apresenta as telas solicitadas do contrário, retorna a tela de login.

Vahldick e Raabe (2008), o funcionamento do BRUCE parte do princípio de redirecionamento de links e formulários para a *servlet* que por sua vez direciona para a página responsável pela apresentação.

O Quadro 5 demonstra o código implementado para a servlet.

```
public class BruceWeb extends HttpServlet {

   protected void service(HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException {
     String nextURL = "";
     User user = UserAdministration.getUser(request);
     if (user == null) {
        nextURL = "dontlogin.jsp";
     }else {
       nextURL = request.getServletPath().replaceFirst(".do", ".jsp");
     }
     RequestDispatcher rd=request.getRequestDispatcher("
                                                    WEB-INF/views/"
                                                    +nextURL);
     rd.forward(request, response);
   }
}
```

Fonte: Vahldick (2008).

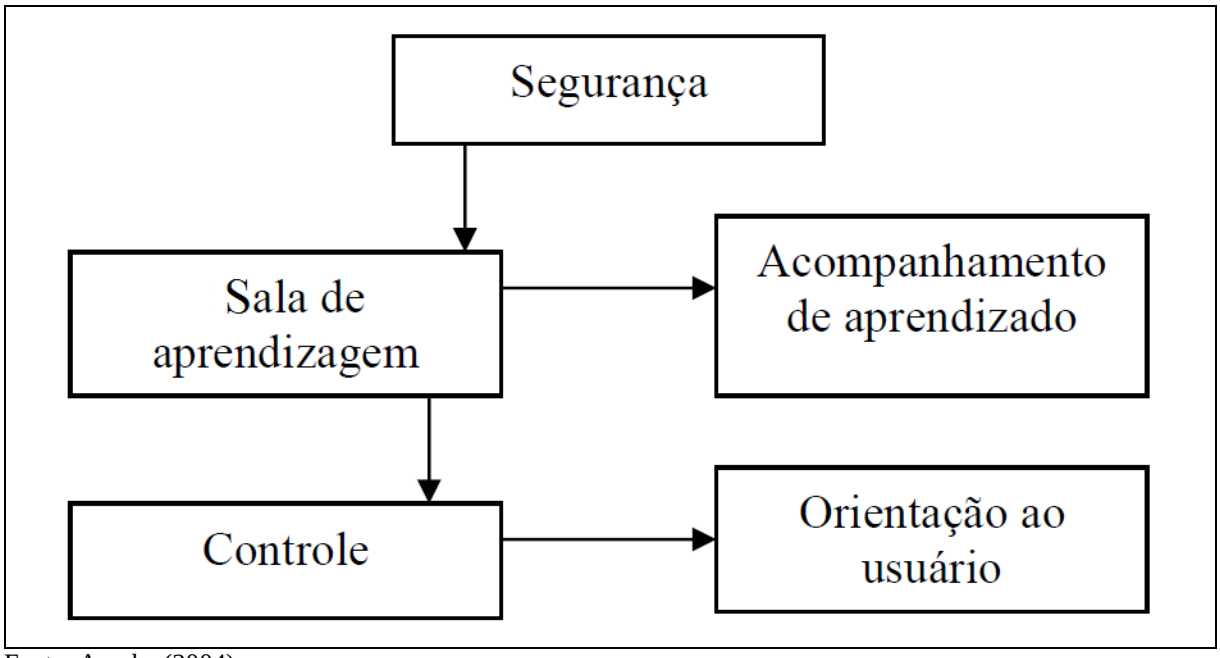
Quadro 5 – Listagem da classe BruceWeb

Segundo Vahldick e Raabe(2008), o BRUCE foi testado tanto com o mecanismo de persistência padrão de XML, quanto com o banco de dados MySQL, e executado no servidor Apache Tomcat.

Com a criação do BRUCE pode-se concluir que é possível desenvolver aplicações web utilizando os recursos do CELINE para o gerenciamento e uso de pacotes SCORM.

2.4 CONCEPÇÃO DE UM LMS

Segundo Aranha (2004), o LMS foi concebido e integrado em cinco módulos a saber: segurança, aprendizagem virtual, acompanhamento da aprendizagem, controle e orientação ao usuário, como pode ser visualizado na figura 8.



Fonte: Aranha (2004).

Figura 8 – Estrutura Modular do protótipo de sistema de gerenciamento de aprendizagem virtual interativa (LMS)

Segundo Aranha (2004) o módulo de segurança realiza a administração da segurança e o controle de acessos dos usuários ao sistema. Ele está estruturado para administrar três categorias de usuários: o aluno, o professor ou tutor e o administrador do sistema.

Para Tanenbaum (2003) a segurança nas redes envolve inúmeros problemas desde garantir que pessoas mal intencionadas não leiam até modificar secretamente as mensagens enviadas de outros destinatários. Além disso, a preocupação com segurança envolve também questões com pessoas que tentam ter acesso a serviços remotos para os quais não estão autorizadas. A segurança evita que mensagens legítimas sejam capturadas e reproduzidas indevidamente. Lida também com situações de pessoas que tentam negar o envio de determinadas mensagens. “A maior parte dos problemas de segurança é causada intencionalmente por pessoas maliciosas que tentam obter algum benefício, chamar a atenção ou prejudicar alguém” (TANENBAUM, 2003, p.767). O autor relaciona os principais invasores conforme o quadro 6.

Adversário	Objetivo
Estudante	Divertir-se em bisbilhotar mensagens de correio eletrônico de outros pessoas.
Cracker	Testar o sistema de segurança de alguém, roubar dados.
Representante de Vendas	Tentar representar toda a Europa e não apenas Andorra.
Executivo	Descobrir a estratégia de marketing do concorrente.
Ex-funcionário	Vingar-se por ter sido demitido
Contador	Desviar dinheiro de uma empresa
Corretor de Valores	Negar uma promessa feita ao cliente através de um mensagem de correio eletrônico.
Vigarista	Roubar números de cartões de créditos e vendê-los
Espião	Descobrir segredos militares ou industriais de um inimigo
Terrorista	Roubar segredo de armas bacteriológicas

Fonte: Tanenbuam (2003, p. 768).

Quadro 6 – Algumas pessoas que podem causar problemas de segurança e os motivos para fazê-lo

O módulo de sala de aprendizagem virtual foi desenhado utilizando o conceito de metáforas para proporcionar e facilitar a utilização das funções pelo aluno durante a aprendizagem a distância.

O módulo de acompanhamento da aprendizagem deve gerar dados automáticos sobre o acompanhamento da aprendizagem do estudante para posterior consulta.

O módulo de orientação deve orientar os usuários (professor ou tutor, alunos e administrador do sistema) em relação a utilização dos comandos e funções do sistema. O mesmo foi concebido baseado nas dificuldades que o usuário tem em operar o sistema, foi desenvolvido para atender usuários iniciantes e experientes.

O módulo de controle deve ser responsável por gerar dados sobre todas as operações internas e externas efetuadas pelo sistema e também aquelas executadas pelo estudante e professor.

2.5 TRABALHOS CORRELATOS

Existem alguns ambientes capazes de executar pacotes SCORM. No item 2.5.1 será apresentado sobre a SCORM Sample RTE criado pela organização ADL (2006).

Quanto a redes sociais, será citado no item 2.5.2 o *gadget* ReadingSocial desenvolvido por LivingSocial (2009) e utilizado pelo *site* de relacionamentos Orkut. No item 2.5.3 será citado o *gadget* UdutuTeach na visão de Hill (2008), implementado para o Facebook.

2.5.1 SCORM Sample RTE

Segundo Vahldick (2008, p. 49-50), SCORM Sample RTE trata-se de um exemplo da implementação de referência do SCORM. Este ambiente foi desenvolvido em Java e está disponível para *download* no *site* da ADL. Nesta ferramenta existem somente dois atores: o administrador e o usuário (VAHLDICK, 2008, p. 50).

O administrador tem atividades como importar curso, excluir curso, comentar atividades do curso, gerenciar usuários e gerenciar objetos globais (VAHLDICK, 2008, p. 46-47). Algumas tarefas como registrar-se em um curso, ver os cursos registrados, visualizar os estados nos cursos e alterar dados cadastrais, foram citadas por Vahldick (2008, p. 47) como atividades do usuário no sistema.

2.5.2 ReadingSocial

Segundo Google (2009d), a ferramenta ReadingSocial é um organizador dos históricos de leitura. A ferramenta é capaz de armazenar os livros lidos, que estão sendo lidos ou que ainda serão lidos.

Segundo LivingSocial (2009), o ambiente é capaz de manter a comunicação entre usuários que estão lendo o mesmo livro, podendo dar dicas de leitura ou até disponibilizar o histórico a todos os usuários.

A aplicação foi hospedada em um servidor disponibilizado pela empresa Google e a ferramenta está sendo executada em massa por usuários do *site* de relacionamentos Orkut. Podem ser realizadas pesquisas de livros no *site* da empresa e os resultados desta pesquisa, retornados em *eXtensible Markup Language* (XML), adicionados diretamente no *gadget*. Conforme LivingSocial (2009) afirma que são adicionados vários livros e duas descrições no *site* e aplicação todos os dias. Uma proposta segundo LivingSocial (2009) é que os próprios usuários possam futuramente adicionar livros ao ambiente.

2.5.3 UdutuTeach

Segundo Hill (2008), UdutuTeach é uma aplicação com a utilidade de um verdadeiro negócio para redes sociais desenvolvida para o *site* Facebook. O ambiente é um LMS que habilita negociadores, educadores e entidades do governo a gerenciar o processo de *e-learning* do início ao fim sem o custo de softwares corporativos ou infra-estrutura. Além disso, é a primeira aplicação que não dependente de propaganda que fornece uma maneira das empresas obterem receita com a plataforma de redes sociais.

O UdutuTeach foi construído para integrar facilmente com outras redes sociais como MySpace, LinkedIn e Bebo, permitindo que as pequenas e grandes corporações, órgãos governamentais, escolas e outros tipos de negócios, conduzam formação e/ou certificação de cursos para tirar proveito dos recursos disponíveis e funcionalidade do Facebook. Com o LMS, os usuários podem publicar material didático, controle de acesso, gravar o processo do aprendiz, recolher o pagamento e ver os resultados, uma vez que os usuários tenham completado uma sessão, tudo sem sair do Facebook.

Segundo Hill (2008), a ferramenta permite professores e alunos utilizarem o Facebook, ou *sites* similares, a executar tarefas como:

a) carregar cursos (pacotes SCORM) para a plataforma de redes sociais ou desenvolver diretamente no Facebook;
b) definir os indivíduos ou grupos de indivíduos que são membros de redes sociais, como alunos ou professores, podendo utilizar os pacotes de um determinado curso;
c) gravação do processo de aprendizado do aluno através da SCORM API;
d) geração de relatórios do processo de aprendizagem (disponível apenas para os administradores do curso);
e) emissão de certificados aos formandos que já tenham concluído o curso da maneira apropriada;
f) *feedback* de aprendizado entre alunos e professores sobre vários cursos;
g) prover avaliações públicas dos cursos formuladas pelos alunos;
h) procura de cursos baseados no conteúdo ou no desempenho dos outros membros.

3 DESENVOLVIMENTO

As seções seguintes descrevem a especificação e implementação do *gadget* e do LMS (BRUCE). Por fim são indicados os resultados obtidos com esse trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Na sequência são apresentados os Requisitos Funcionais (RF) (quadro 7) e os Requisitos Não Funcionais (RNF) (quadro 8) que são atendidos por este trabalho

Requisitos Funcionais (RF)	**Caso de Uso**
RF01: o *gadget* deverá permitir que o usuário do Orkut se registre.	UC01
RF02: o LMS deverá permitir que o usuário faça manutenção de seus objetos SCORM: adição, substituição e remoção de pacotes SCORM.	UC02
RF03: o LMS deverá permitir que o pacote possa ser sugerido pelos amigos ou torná-lo público e avisar todos os usuários quando um pacote público for adicionado.	UC03
RF04: o LMS deverá permitir que o usuário liste todos os seus objetos.	UC04
RF05: o LMS deverá permitir que o usuário liste os objetos de cada um de seus amigos.	UC05
RF06: o LMS deverá permitir que o usuário selecione um objeto para interagir.	UC06
RF07: o LMS deverá permitir que o usuário registre-se em objetos públicos ou sugeridos por seus amigos.	UC07
RF08: o LMS deverá permitir a administração de usuários.	UC08
RF09: o LMS deverá permitir a administração de pacotes.	UC09

Quadro 7 – Requisitos funcionais

Requisitos Não Funcionais (RNF)
RNF01: utilizar OpenSocial API para implementar o *gadget*.
RNF02: ser compatível com o Orkut.
RNF03: o LMS deverá ser compatível com o Apache Tomcat.
RNF04: o LMS deverá ser compatível com o MySQL.
RNF05: o LMS deverá utilizar o componente CELINE para as funções de administração de pacotes SCORM.

Quadro 8 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Na sequência é apresentada a especificação do *gadget*, que foi modelado na ferramenta Entreprise Architect. Foram utilizados conceitos da orientação a objetos e da Unified Modeling Language (UML) para a criação dos diagramas de casos de uso, atividades, classe e de sequência.

3.2.1 Diagrama de casos de uso

Na figura 9 tem-se os casos de uso sobre a atividade do usuário.

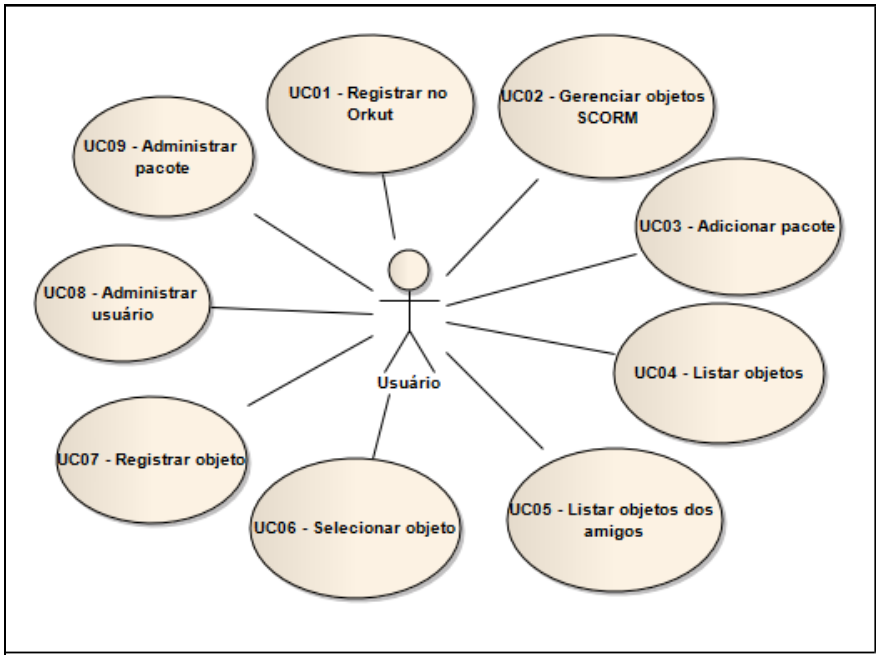


Figura 9 – Diagrama de casos de uso do usuário

O primeiro caso de uso (Quadro 9), designado como Registrar, apresenta a autenticação do usuário no *gadget*.

UC01 – Registrar	
Descrição	Autenticação do usuário
Pré-condições	O usuário deve estar logado no Orkut; O usuário deve ter o *gadget* adicionado a seu perfil.
Cenário principal	01) O usuário faz acesso a *gadget*; 02) O sistema verifica a permissão do usuário; 03) O *gadget* grava os dados; 04) O usuário visualiza a tela de saudação.
Pós-condições	O usuário está apto a utilizar o sistema.

Quadro 9 – Detalhamento do caso de uso Registrar

O caso de uso de número 2 (Quadro 10), designado como Administrar pacotes, apresenta as operações que podem ser realizadas com os pacotes de aprendizado SCORM. Esse caso é constituído pelo cenário principal, dois cenários alternativos e um cenário de exceção.

O caso de uso 3 (Quadro 11), denominado Compartilhar pacotes, define as operações para que os usuários da rede social possam sugerir cursos para seus amigos ou assumir cursos públicos.

O caso de uso 5 (Quadro 13), nomeado Listar objetos, apresenta a operação que o usuário quando quer verificar foram indicados por seus amigos.

UC02 – Administrar pacotes	
Descrição	Permite adição, substituição e remoção de pacotes SCORM
Pré-condições	O usuário deve estar logado ao *gadget* com uma conta que disponha de privilegio de administrador.
Cenário principal	01) O usuário acessa o menu “Manage courses”; 02) O *gadget* lista os cursos existentes com a respectiva opção de removê-los e habilita a opção “Import Course”; 03) O usuário faz a operação necessária; 04) O sistema processa a solicitação e retorna ao menu inicial;
Fluxo Alternativo 01	No passo 02 do cenário principal, caso o usuário clique em “DELETE”: 02.01) O sistema exclui o pacote e retorna ao passo 03.
Fluxo Alternativo 02	No passo 02 do cenário principal, caso o usuário clique em “Import Courses”: 02.01) O usuário informa um ID para o curso; 02.02) O usuário informa um nome para o curso; 02.03) O usuário escolhe um pacote SCORM no formato zip e faz upload do arquivo; 02.04) O usuário clica na opção “Send” 02.05) O *gadget* grava os dados; 02.06) O *gadget* retorna a pagina inicial.
Exceção 01	No passo 03 do fluxo alternativo 02, caso o usuário não adicione um pacote no padrão SCORM, o sistema gera uma mensagem de erro e solicita ao usuário que tente novamente.

Quadro 10 – Detalhamento do caso de uso Administrar pacote

O caso de uso 6 (Quadro 14), nomeado Listar objetos dos amigos, apresenta a operação que o usuário executa quando deseja verificar os cursos adicionados por seus amigos.

UC03 – Compartilhar pacote e avisar amigo	
Descrição	Compartilhar pacotes com os amigos ou deixar um pacote público
Pré-condições	O usuário deve estar utilizando o *gadget*.
Cenário principal	1) O usuário acessa o item “Import Courses”; 2) O usuário registra um novo curso; 3) O usuário define o curso como público; 4) O usuário finaliza a importação do curso; 5) O usuário retorna ao menu inicial. 6) O *gadget* gera uma atualização no perfil de todos os usuários cadastrados avisando que foi adicionado um novo curso e retorna ao menu inicial.
Fluxo Alternativo 01	No passo 2, quando o usuário registra um novo curso, mas não define como público, o usuário finaliza o curso e retorna ao passo 4
Fluxo Alternativo 02	No passo 5, após retornar ao menu principal, o usuário acessa o item “Your registered courses” e sugere o curso importado anteriormente a um amigo.

Quadro 11 – Detalhamento do caso de uso Compartilhar pacote

UC05 – Listar objetos	
Pré-condições	O usuário deve estar logado ao *gadget*.
Cenário principal	01) O usuário acessa o menu “Register to a new course”; 02) O *gadget* lista os cursos existentes; 03) O usuário faz a operação necessária; 04) O sistema processa a solicitação e retorna ao menu inicial;

Quadro 13 – Detalhamento do caso de uso Listar objetos

UC06 – Listar objetos dos amigos	
Pré-condições	O usuário deve estar logado ao *gadget*.
Cenário principal	01) O usuário acessa o menu “*Your registered courses*”; 02) O usuário acessa o menu “Friends Courses” 03) O *gadget* lista todos os cursos cadastrados para os amigos desse usuário; 04) O sistema aguarda ação do usuário;

Quadro 14 – Detalhamento do caso de uso Selecionar objeto dos amigos

O caso de uso 7 (Quadro 15), nomeado Selecionar objeto para interagir, apresenta a operação que o usuário executa quando deseja iniciar um curso.

UC07 – Selecionar objeto para interagir	
Pré-condições	O usuário deve estar logado ao *gadget*.
Cenário principal	01) O usuário acessa o menu “*Your registered courses*”; 02) O sistema lista os cursos do usuário; 03) O usuário seleciona um curso para interação; 04) O usuário inicia o curso; 05) O usuário finaliza o curso; 06) O sistema retorna a tela inicial.

Quadro 15 – Detalhamento do caso de uso Selecionar objeto para interagir

O caso de uso 8 (Quadro 16), nomeado Registrar objetos públicos ou sugeridos, apresenta a operação que o usuário quando quer verificar foram indicados por seus amigos.

UC08 – Registrar objetos públicos ou sugeridos	
Pré-condições	O usuário deve estar logado ao *gadget*.
Cenário principal	01) O usuário acessa o menu “Register to a new course”; 02) O *gadget* lista os cursos existentes; 03) O usuário registra-se nos cursos disponíveis ou nos sugeridos de seus amigos; 04) O sistema retorna ao menu inicial;

Quadro 16 – Detalhamento do caso de uso Registrar objetos públicos ou sugeridos

O caso de uso de número 9 (Quadro 17) apresenta a forma que o LMS gerencia os usuários.

UC9 – Administrar usuário	
Pré-condições	O usuário deve estar logado ao *gadget* ter privilegio de administrador.
Cenário principal	01) O usuário acessa o menu “Manage users”; 02) O *gadget* lista os usuários cadastrados; 03) O usuário pode criar, editar ou remover um usuário; 04) O usuário executa a tarefa; 05) O sistema processa a solicitação e retorna ao menu inicial;

Quadro 17 – Diagrama de caso de detalhado de administração de usuário

O caso de uso de número 10 (Quadro 18) apresenta a forma que o LMS gerencia os pacotes.

UC10 – Administrar pacote	
Pré-condições	O usuário deve estar logado ao *gadget* ter privilegio de administrador.
Cenário principal	01) O usuário acessa o menu “Manage courses”; 02) O *gadget* lista os cursos cadastrados; 03) O usuário pode criar, editar ou remover um curso; 04) O usuário executa a tarefa; 05) O sistema processa a solicitação e retorna ao menu inicial;

Quadro 18 – Diagrama de caso de detalhado de administração de pacote

3.2.2 Diagrama de Atividades

A figura 12 e figura 13 fornece uma visão de como todas as atividades são executadas, de forma a clarificar o funcionamento do *gadget* e LMS.

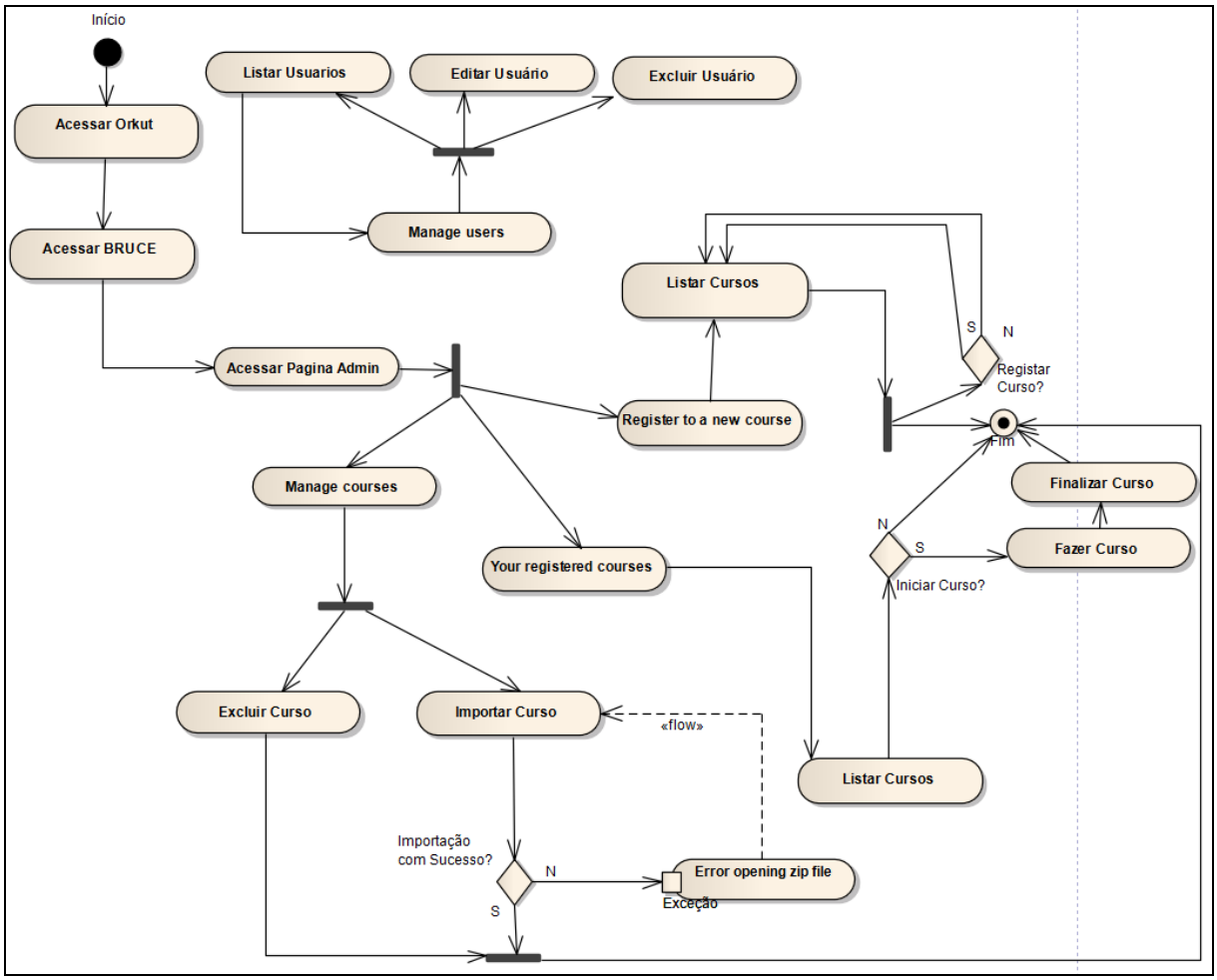


Figura 12 – Diagrama de atividades do nível administrador

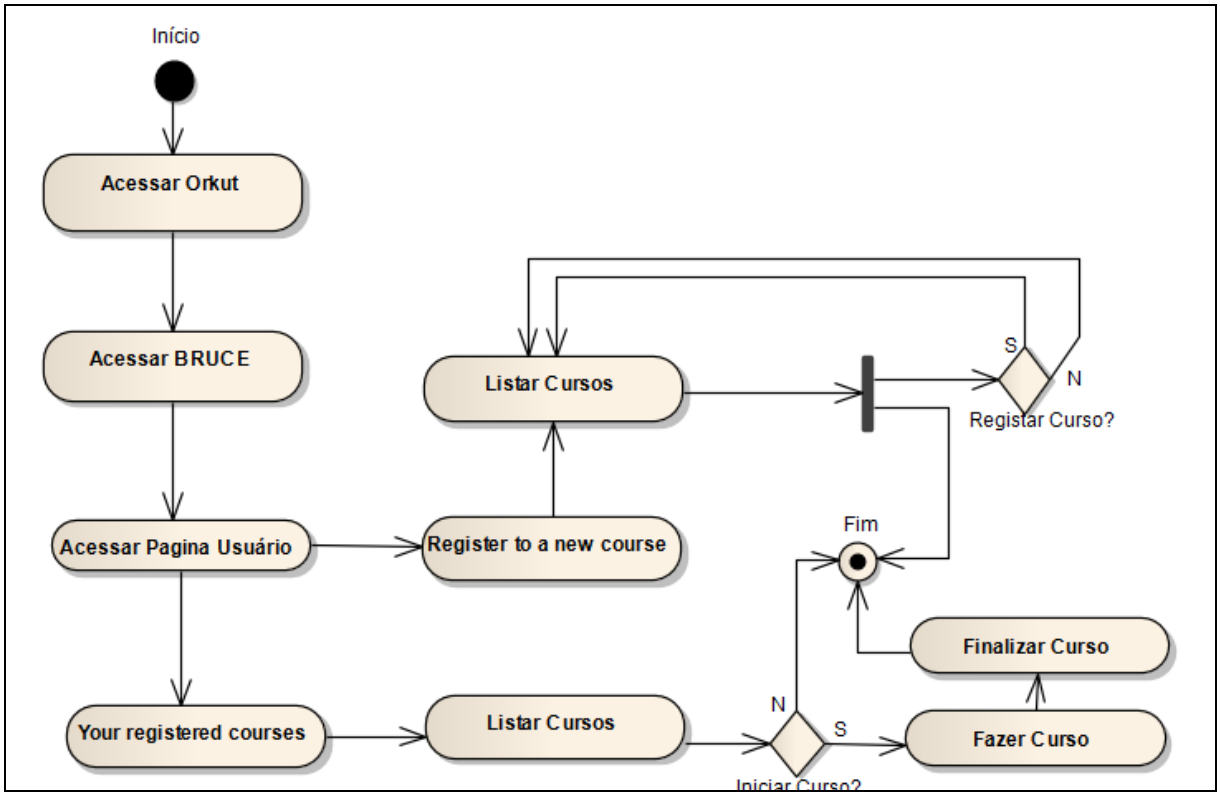


Figura 13 – Diagrama de atividades do nível de usuário

De início o usuário deve acessar o Orkut e adicionar o *gadget* a seu perfil. Feito isso, o usuário deve acessar o *gadget*. É feito uma verificação se o usuário é do tipo administrador ou não. Se o usuário for administrador ele pode exercer as seguintes opções: gerenciar cursos, gerenciar usuários, registrar em cursos, listar os cursos registrados. Já o usuário do tipo usuário tem recursos reduzidos que são: registrar-se em cursos e listar os cursos registrados.

Na função de gerenciar os cursos, o usuário pode escolher entre incluir um novo curso ou excluir um curso existente. Na opção de inserir um novo curso, caso seja escolhido um curso SCORM inválido, o usuário é redirecionado a tela de gerenciar curso novamente. A opção de excluir curso, após um curso ser selecionado, o exclui e retorna a tela inicial do usuário onde são listadas todas as funções.

Na função de gerenciar usuários, pode-se optar por excluir ou editar um usuário. Se a opção excluir for escolhida, após o usuário ser selecionado, o sistema retorna a tela de gerenciamento de usuários. Caso a opção escolhida seja editar usuário, após o usuário ser selecionado, pode-se alterar a permissão do usuário selecionado para administrador. Feito esse processo o sistema retorna a tela inicial do usuário.

Na função listar os cursos registrados, o usuário pode iniciar um curso e realizar as atividades oferecidas por esse curso. Ao finalizar o curso o usuário é direcionado novamente a tela de listagem de cursos.

Na função registrar-se a cursos, o usuário visualiza cursos públicos disponíveis no gadget, ele pode se registrar em um curso e/ou indicar um curso ao um amigo já registrado no *gadget*. Feito esse processo o usuário retorna a pagina inicial.

3.2.3 Diagrama de Classes

Na figura 14 pode se verificar o diagrama de classes que fornece uma visão de como as classes estão estruturadas e relacionadas, de forma a facilitar a estruturação e a relação entre elas.

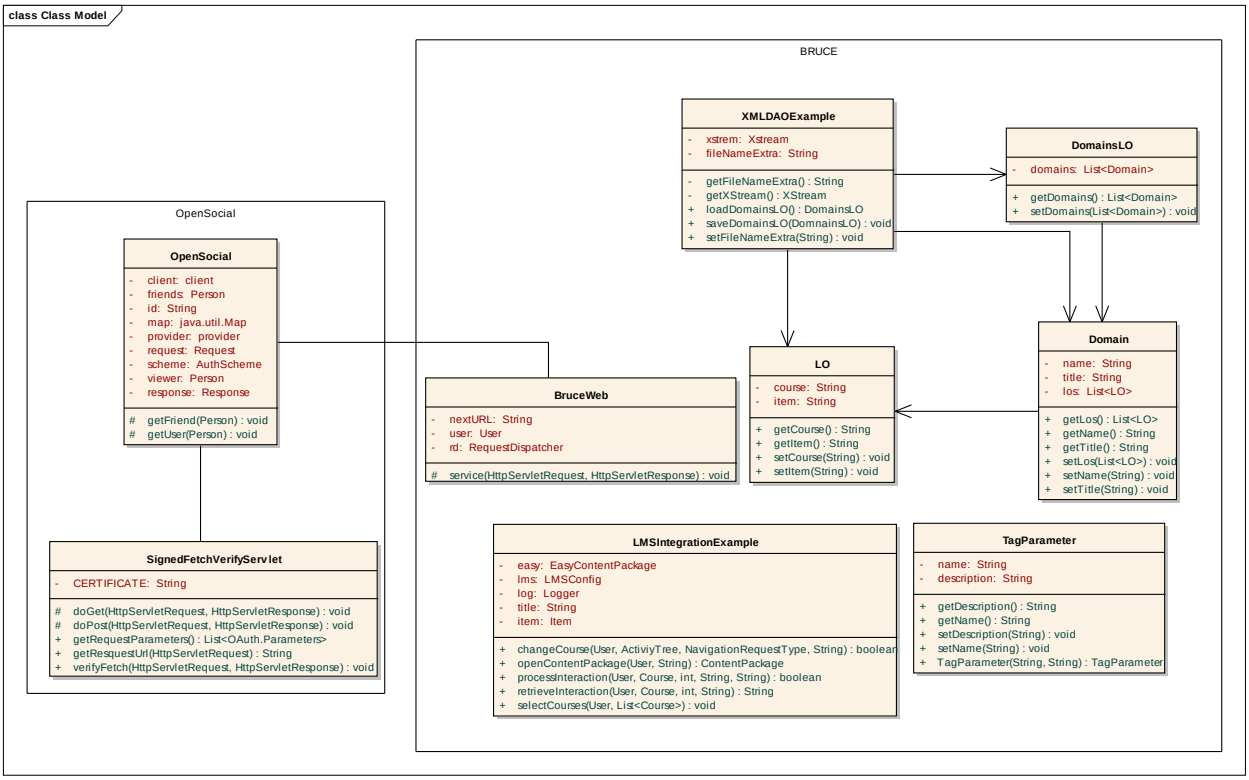


Figura 14 – Diagrama de classes

Além das classes implementadas por Vahldick (2008), que são aquelas contidas dentro do quadro BRUCE na figura 13, foram implementadas duas classes para o LMS relacionar-se com o container via OpenSocial. As duas classes implementadas para o desenvolvimento deste trabalho serão descritas abaixo.

A classe `OpenSocial` foi desenvolvida para que cada conexão com o container Orkut não tivesse que ser refeita em cada requisição realizada ao ambiente.

A classe `SignedFetchVerifyServlet` foi desenvolvida para validar as requisições feitas pelo *gadget* para acessar os recursos do LMS. A classe verifica o método de envio de requisição com os métodos `doGet` e `doPost`. Além de validar a requisição, a classe é capaz de obter a requisição em forma de lista de objetos ou somente um objeto, utilizando o método `getRequest`.

3.2.4 MODELO ENTIDADE RELACIONAMENTO

A figura 15 apresenta o modelo de entidade e relacionamento do LMS e do *gadget*. Além do modelo de tabelas proposto por Vahldick (2008), foi adicionado uma tabela de

cursos públicos, vinculado a tabela de cursos. Essa tabela armazena o atributo público utilizado para listagem do cursos disponíveis a todos os usuários do *gadget*.

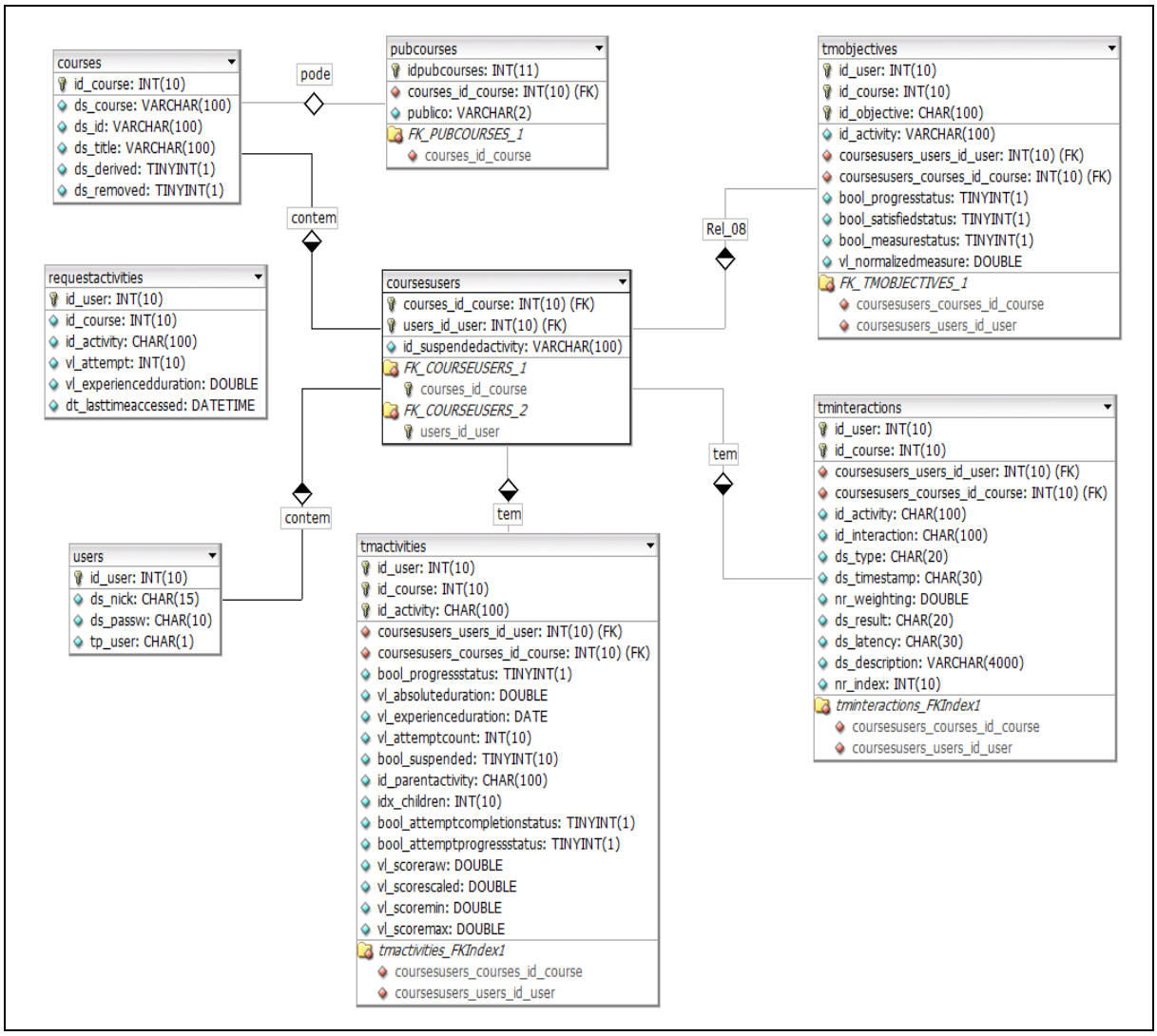


Figura 15 – Modelo entidade relacionamento

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para desenvolvimento do *gadget* BRUCE, foi utilizada a API OpenSocial da Google, uma API de código aberto. O BRUCE faz requisições ao LMS Bruce, hospedado no servidor da FURB. Partindo do ponto de que as requisições só são realizadas pela API, foi implementado a partir do LMS uma classe que faz solicitações ao *container* OpenSocial Orkut. Com essa implementação é possível capturar dados dos usuários do Orkut, como nome e amigos. Cinco novas bibliotecas foram adicionadas ao LMS: *OpenSocial* (versão 1.0), o*auth*, *junit* (versão 4.5), *json* (versão 1.1),*commons codec* (versão 1.3). Essas bibliotecas utilizadas são de código aberto. Para criação dessa classe o LMS foi importado no ambiente de desenvolvimento web Eclipse (versão Helios Service Release 1).

3.3.1.1 Desenvolvimento do *gadget* BRUCE

No desenvolvimento da *gadget*, foi desenvolvido um arquivo XML de acordo com as especificações para o desenvolvimento de *gadgets* da OpenSocial e do Orkut. O foco da *gadget* é a rede social Orkut, pois das redes sociais estudadas é a que tem menos particularidades e tem sua especificação mais próxima da especificação da OpenSocial. Para criação do XML foi utilizada a ferramenta Notepad++.

O *gadget* foi implementada como apresentado no quadro 19, as variáveis do gadget são:

a) linha 3. `<ModulePrefs>`: *tag* que representa as informações básicas da aplicação, como nome, autor, descrição, email do autor. Essa *tag* também é responsável por elementos que informam a versão da API OpenSocial que será utilizada (require) e a latência dos arquivos estáticos na rede social, fazendo que assim que o tempo de carregamento da *gadget* seja diminuído, a configuração de cache é opcional para *gadgets*;
b) linha 4. `title="BRUCE"`: nome que será concedido a *gadget* (figura 16(1));
c) linha 5. `directory_tile="BRUCE"`: nome do aplicativo no diretório de aplicativos;
d) linha 6. `description`: breve descrição sobre o aplicativo(figura 16(2));
e) linha 8. `author`: nome do desenvolvedor da *gadget*;

f) linha 9. `author_email`: email do desenvolvedor da *gadget*;
g) linha 10. `author_affiliation`: empresa ou instituição onde o desenvolvedor esta afiliado;
h) linha 11. `author_location`: local físico onde o *gadget* foi desenvolvido;
i) linha 12. `screenshot`: imagem de pré-visualização do *gadget*, será visualizada pelo usuário quando o mesmo adicionar o *gadget*(figura 16(3));
j) linha 13. `thumbnail`: imagem reduzida utilizada para localizar a *gadget* no diretório de aplicativos da rede social(figura 16(4));
k) linha 14. `<Require feature="OpenSocial-0.8"/>`: fazendo a chamada da API, com o parâmetro `Require` pode-se chamar varias APIs internas ou externas;
l) linha 15. `<Require feature="OpenSocial-data"/>`: chamada para a obtenção de dados do usuários da API;
m) linha 16. `<Require feature="OpenSocial-templates"/>`: chamada para a obtenção dos *templates* de interface da API;
n) linha 17. `<Require feature="views" />`: chamada para obter as funções de visualizações de objetos do usuário;
o) linha 18. `<Param name="process-on-server">true</Param>`: parâmetro para que todas as execuções sejam feitas localmente (servidor Orkut);
p) linha 20. `<Content view="profile" >`: exibe seu resultado no perfil do usuário que tem o *gadget* atrelado a seu perfil;
q) linha 22. `<!-- codigo removido-->`: string de validação de propriedade da *gadget*. Antes de atrelar a *gadget* a uma rede social, é necessária confirmar de que a *gadget* é de propriedade de quem esta tentando colocá-la na rede social. Para tal, GOOGLE (2011), solicita que se coloque o caminho de acesso ao XML (como por exemplo: http://campeche.inf.furb.br/lrmorastoni/bruce.xml) para validação. O site retorna uma chave e solicita que essa chave seja colocada dentro da *tag* `<Content>` do seu XML. Feito isso, o *gadget* está verificado e pronto para ser colocado no Orkut;
r) linha 24. `<Content type="html">`: exibe o *gadget* em si, tudo que estiver dentro dessa *tag*, será a própria execução e interface do *gadget*, também informa o tipo de conteúdo de que o mini aplicativo é formado;
s) linha 27. `function carregaViewer()`: função em *javascript* que requisita os dados do usuário do *gadget* ao servidor do OpenSocial;

t) linha 34. `function onCarregaMe(ret)`: função em *javascript* que recebe como parâmetro os dados do usuário do gadget e invoca um *iframe* com a aplicação remota;

u) linha 42. `function init()`: função em *javascript* que é responsável pela execução de outras funções;

v) linha 44. `gadgets.util.registerOnLoadHandler(init)`: função da API que faz o carregamento da função `init` para que o gadget seja iniciado.

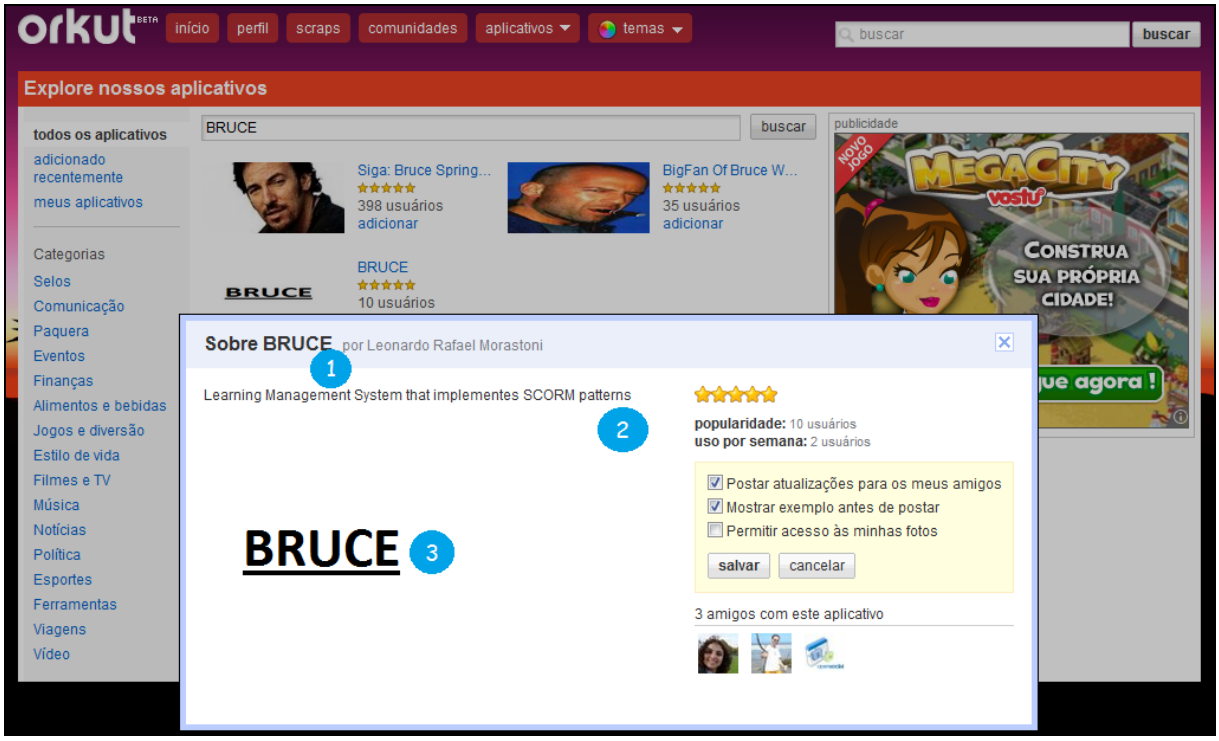


Figura 16 – Exemplo de objetos do *gadget*

```
1.<?xml version="1.0" encoding="ISO-8859-1"?>
2.<Module>
3.    <ModulePrefs
4.    title="BRUCE"
5.    directory_title="BRUCE"
6.    description="Learning Management System that implements SCORM
7.    patterns"
8.    author="Leonardo Rafael Morastoni"
9.    author_email="leomoras@gmail.com"
10.   author_affiliation="FURB"
11.   author_location="Blumenau, SC"
12.   screenshot="/gadget/bruce.png"
13.   thumbnail="/gadget/bruce.png">
14.  <Require feature="OpenSocial-0.8"/>
15.  <Require feature="OpenSocial-data"/>
16.  <Require feature="OpenSocial-templates"/>
17.   <Require feature="views" />
18.   <Param name="process-on-server">true</Param>
19.  </ModulePrefs>
20.  <Content view="profile" >
21.       Bem vindo ao Bruce!
22.     <!-- codigo removido-->
23.  </Content>
24. <Content type="html">
25.    <![CDATA[
26.     <script type="text/javascript">
27.     function carregaViewer(){
28.         var req = OpenSocial.newDataRequest();
29.         req.add(req.newFetchPersonRequest(
30.            OpenSocial.IdSpec.PersonId.VIEWER), 'viewer');
31.         var idspec = OpenSocial.newIdSpec({'userId':'VIEWER',
32.        'groupId':'SELF'});
33.         req.send(onCarregaMe);}
34.     function onCarregaMe(ret){
35.         var viewer = ret.get('viewer').getData();
36.         var source =
37.         "http://campeche.inf.furb.br/lrmorastoni/orkut.jsp?id='+
38.         viewer.getId() + '"
39.         document.getElementById('frm').innerHTML='<iframe
40.         height=100% width=100% frameborder="0"
41.         src=source></iframe>'; }
42.     function init() {carregaViewer();}
43.     gadgets.util.registerOnLoadHandler(init);
44.     </script>
45.     <div id='frm'>
46.         Hello! Your application is being loaded...
47.     </div>
48.     </html>]]>
49.  </Content>
50. </Module>
```

Quadro 19– Desenvolvimento do *gadget*

3.3.1.2 Desenvolvimento do LMS

O desenvolvimento do LMS é descrito em detalhes em Vahldick (2008). No contexto deste trabalho serão demonstradas as paginas *Java Server Pages* (JSP) que fazem acesso a OpenSocial e os códigos que foram alterados no LMS para que essa adequação à API funcionasse corretamente.

O quadro 20 apresenta a página de saudação do LMS, alterada para apresentar o cumprimente ao usuário do Orkut com o seu nome (figura 17(1)). Esse cumprimento somente é exibido após o usuário o adicionar o *gadget* a seu perfil. Este, por sua vez, faz o cadastro implicitamente nos usuários do LMS. Por uma questão de segurança foram removidas as chaves de acesso ao Orkut, pois as mesmas são de uso pessoal e intransferível.

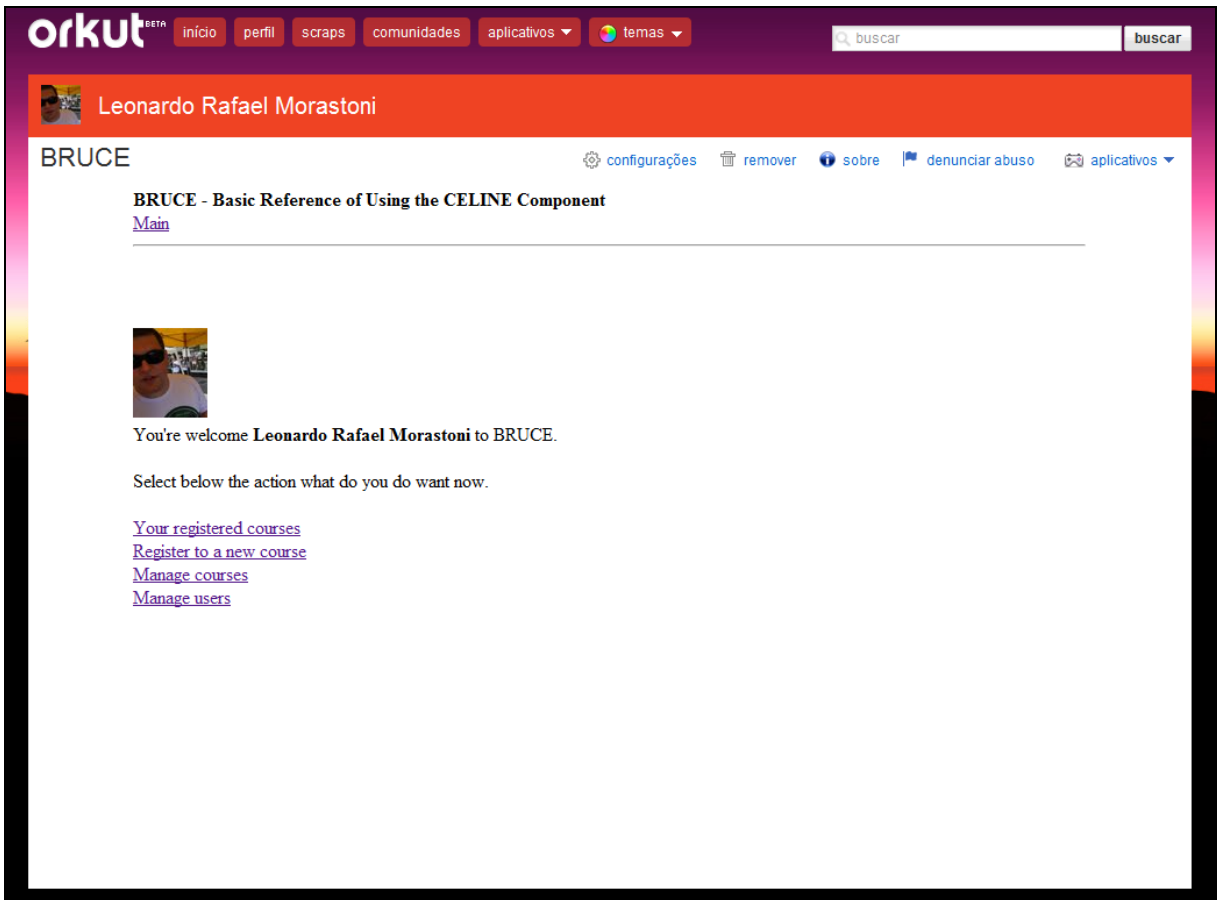


Figura 17 – Tela de saudação do *gadget*

No quadro 20, alem dos códigos implementados por Vahldick (2008), foram incluídos comandos que fazem o acesso ao container Orkut utilizando a OpenSocial. Os dois comandos de maior relevância nesse código são:

a) `AuthScheme scheme = new OAuth2LeggedScheme("orkut.com:xxxxxxx", "xx",id);` : cria um novo esquema utilizando segurança `OAuth2Legged` (os

parâmetros foram escondidos pois pertencem a dados particulares do desenvolvedor). O esquema recebe como parâmetro uma chave de consumidor, uma chave de segurança e a OpenSocialid do visualizador;

b) `Welcome, <b><%= viewer.getDisplayName() %></b>!`: imprime no gadget uma saudação ao visualizador.

Para que os usuários possam liberar pacotes SCORM para seus amigos foi implementada uma página JSP que retornasse todos os amigos na página onde são importados os objetos. Caso a pessoa que esteja importando um novo curso queira liberar esse mesmo curso para todos os indivíduos participantes do *gadget*, o usuário pode optar por escolher que o pacote seja público(assim todos os usuários terão acesso ao curso disponibilizado).

```
<%@taglib prefix="celine" uri="http://www.univali.br/celine/tags" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page import="org.OpenSocial.Client"%>
<%@ page import="org.OpenSocial.providers.Provider"%>
<%@ page import="org.OpenSocial.providers.OrkutProvider"%>
<%@ page import="org.OpenSocial.auth.AuthScheme"%>
<%@ page import="org.OpenSocial.auth.OAuth2LeggedScheme"%>
<%@ page import="org.OpenSocial.Request"%>
<%@ page import="org.OpenSocial.Response"%>
<%@ page import="org.OpenSocial.services.PeopleService"%>
<%@ page import="org.OpenSocial.models.Person" %>
<%@ page import="java.util.HashMap" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.Map" %>

<%
Person viewer = null;
List<Person> friends = null;
Provider provider = new OrkutProvider();
AuthScheme scheme = new OAuth2LeggedScheme("orkut.com:xxxxxxx",
"xxxxxxxxxxxx","13367845281276129166");
Client client = new Client(provider, scheme);
try {
    Map<String, Request> requests = new HashMap<String, Request>();
    requests.put("friends", PeopleService.getFriends());
    requests.put("viewer", PeopleService.getViewer());

    Map<String, Response> responses = client.send(requests);
    friends = responses.get("friends").getEntries();
    viewer = responses.get("viewer").getEntry();
  } catch (Exception e) {
  }
%>
<div>
        Welcome, <b><%= viewer.getDisplayName() %></b>!

    </div>
You're welcome to BRUCE.<br/><br/>
Select below the action what do you do want now.<br/><br/>
<celine:user/> <!-- attrib the user data to some page variables -->

<div>
<a href='listcourses.do'>Your registered courses</a><br/>
<a href='regcourses.do'>Register to a new course</a><br/>
</div>
<c:if test="${useradmin}">
   <a href='managecourses.do'>Manage courses</a><br/>
   <a href='manageusers.do'>Manage users</a><br/>
</c:if>
<br>
<div>
<a href='logoff.do'>Logoff</a><br/>
</div>
<script>
//window.parent.frames[0].document.getElementById('main').style.display = "";
//window.parent.frames[0].document.getElementById('exitCourse').style.display =
"hidden";
//window.parent.frames[0].document.getElementById('suspendCourse').style.display
= "hidden";
</script>
```

Quadro 20 – Implementação da página de saudação

Na figura 18, pode-se verificar o layout da página de importação de cursos e no quadro 21 o código que gera esse *layout*.

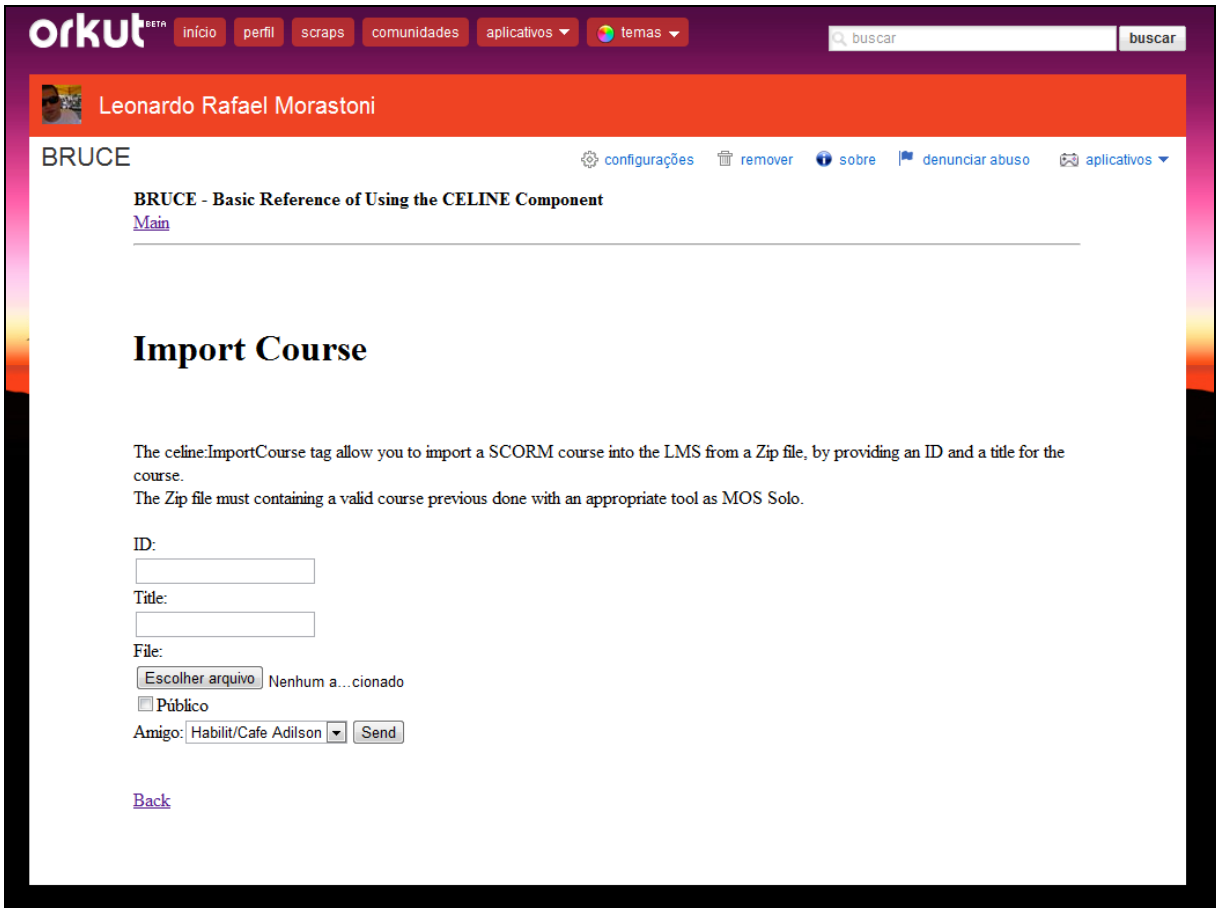


Figura 18 – Página de importação de curso

Destacam-se no quadro 21 os seguintes aspectos:

a) `Person viewer = null;`: cria um objeto do tipo pessoa chamado viewer, esse usuário está fazendo acesso do *gadget* via Orkut;

b) `String id = request.getParameter("id");`: um dado do tipo string id, recebe o `OpenSocialid` do visualizador ;

c) `Provider provider = new OrkutProvider();`: cria um novo provedor, indica para a aplicação onde a conexão será feita(nesse caso, Orkut);

d) `AuthScheme scheme = new OAuth2LeggedScheme("orkut.com:xxxxxxx", "xx",id);`: cria um novo esquema utilizando segurança `OAuth2Legged`(os parâmetros foram escondidos pois pertencem a dados particulares do desenvolvedor). O esquema recebe como parâmetro uma chave de consumidor, uma chave de segurança e `OpenSocialid` do visualizador;

e) `Client client = new Client(provider, scheme);`: cria uma nova conexão cliente utilizando os dados do provedor e do esquema de segurança;

f) `requests.put("friends", PeopleService.getFriends());` faz uma requisição ao provedor onde solicita os amigos do usuário;

```
<%@include file="conecta.jsp" %>
<%@page import="java.util.*"%>
<%@taglib prefix="celine" uri="http://www.univali.br/celine/tags" %>
<%@ page import="org.opensocial.Client"%>
<%@ page import="org.opensocial.providers.Provider"%>
<%@ page import="org.opensocial.providers.OrkutProvider"%>
<%@ page import="org.opensocial.auth.AuthScheme"%>
<%@ page import="org.opensocial.auth.OAuth2LeggedScheme"%>
<%@ page import="org.opensocial.Request"%>
<%@ page import="org.opensocial.Response"%>
<%@ page import="br.furb.opensocial.*"%>
<%@ page import="org.opensocial.models.Person" %>
<%@ page import="java.util.HashMap" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.Map" %>
<html>
      <head>
            <title>CELINE - Import Course</title>
      </head>
      <body>
            <h1>Import Course</h1><br/><br/>
            The celine:ImportCourse tag allow you to import a SCORM
course into the LMS from a Zip file, by providing an ID and a title for
the course.<br/>
            The Zip file must containing a valid course previous done
with an appropriate tool as MOS Solo. <br/><br/>
      <celine:formImportCourse nextURL="salvarpublico.do" >
            <input type="checkbox" name="publico" value="s"/>Público<br/>
            <%
            String id = session.getAttribute("id").toString();
            Person viewer = null;
            List<Person> friends = null;
            Provider provider = new OrkutProvider();
            AuthScheme scheme = new
OAuth2LeggedScheme("orkut.com:xxxxxx", "xxxxxxxx",id);
            Client client = new Client(provider, scheme);
            try {
                Map<String, Request> requests = new HashMap<String,
Request>();
                requests.put("friends",
SignedFetchVerifyServlet.getFriends());
                requests.put("viewer",
SignedFetchVerifyServlet.getViewer());

                Map<String, Response> responses = client.send(requests);
                friends = responses.get("friends").getEntries();
                viewer = responses.get("viewer").getEntry();
              } catch (Exception e) {}          %>
              Amigo:<select id="person" name="person">
              <option value="<%="0"%>"><%="-"%></option>
                <%
                for (Person friend : friends) {
                ArrayList <String> user= new ArrayList<String>();
                PreparedStatement select = con.prepareStatement("select
ds_nick from USERS");
                ResultSet selectrs = select.executeQuery();
```

```
                    while (selectrs.next()) {
                       user.add(selectrs.getString(1));}
                                select.close();
                                selectrs.close();
                                for(int i=0; i < user.size();i++){
                                if(friend.getId().equals(user.get(i))){
                    %>

              <option value="<%= friend.getId()
%>"><%=friend.getDisplayName() %></option>
                    <%
                     }}}
                    %>

              </celine:formImportCourse>
              <br/><a href="managecourses.do">Back</a>

       </body>
</html>
```

Quadro 21 – Implementação da página de importação de curso

g) `requests.put("viewer", PeopleService.getViewer());` : solicita ao provedor o valor da `OpenSocialid` do visualizador;

h) `friends = responses.get("friends").getEntries();`: recebe o retorno do provedor com os amigos do visualizador;

i) `<option value=" <%=friend.getDisplayName()%>"> <%= friend.getDisplayName()%></option>` : imprime no gadget os nomes de todos os amigos do visualizador.

3.3.2 Operacionalidade da implementação

Esta seção tem por objetivo demonstrar a operacionalidade e funcionalidade do *gadget* através de um estudo de caso.

Inicialmente o usuário deve estar utilizando o Orkut e adicionar o mini-aplicativo BRUCE, feito isso ele irá visualizar a figura 19 se tiver permissão de *admin* ou a figura 20 caso o usuário tenha permissão comum. A permissão de *admin* é concedida caso o usuário solicite ao desenvolvedor. Uma vez concedida esta permissão além de executar funções de um usuário comum, como realizar cursos ou registrar-se em cursos novos, este tipo de usuário tem a capacidade de adicionar e gerenciar cursos e usuários.

Ao acessar o item "*Your registered courses*" visualizado pelo nível *admin* e pelo nível usuário, o usuário visualiza uma lista de cursos já adicionados (figura 21). Já estão disponíveis nessa mesma tela os cursos adicionados pelos amigos do usuário ou que

indicarem esses cursos a ele. Para a implementação dessa tela foi alterada uma *tag* no componente CELINE permitindo a utilização de do id do curso e a liberação para trabalhar com os dados recebidos do LMS.

Nos cursos indicados por amigos do visualizador além de exibir o nome do curso, exibirá também o nome do amigo indicador do curso correspondente.

Na tela de cursos registrados o usuário pode iniciar o curso ou simplesmente retornar ao menu inicial.

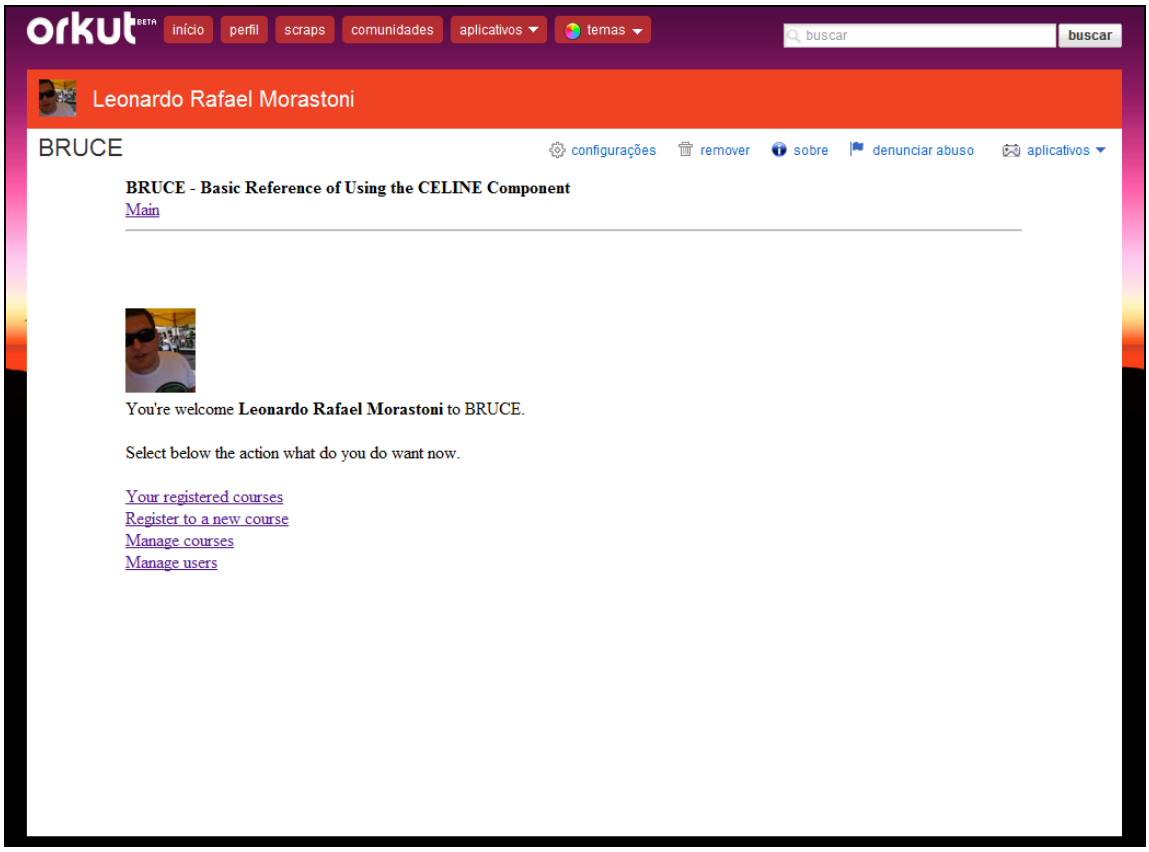


Figura 19 – Tela inicial do *gadget* de nível *admin*

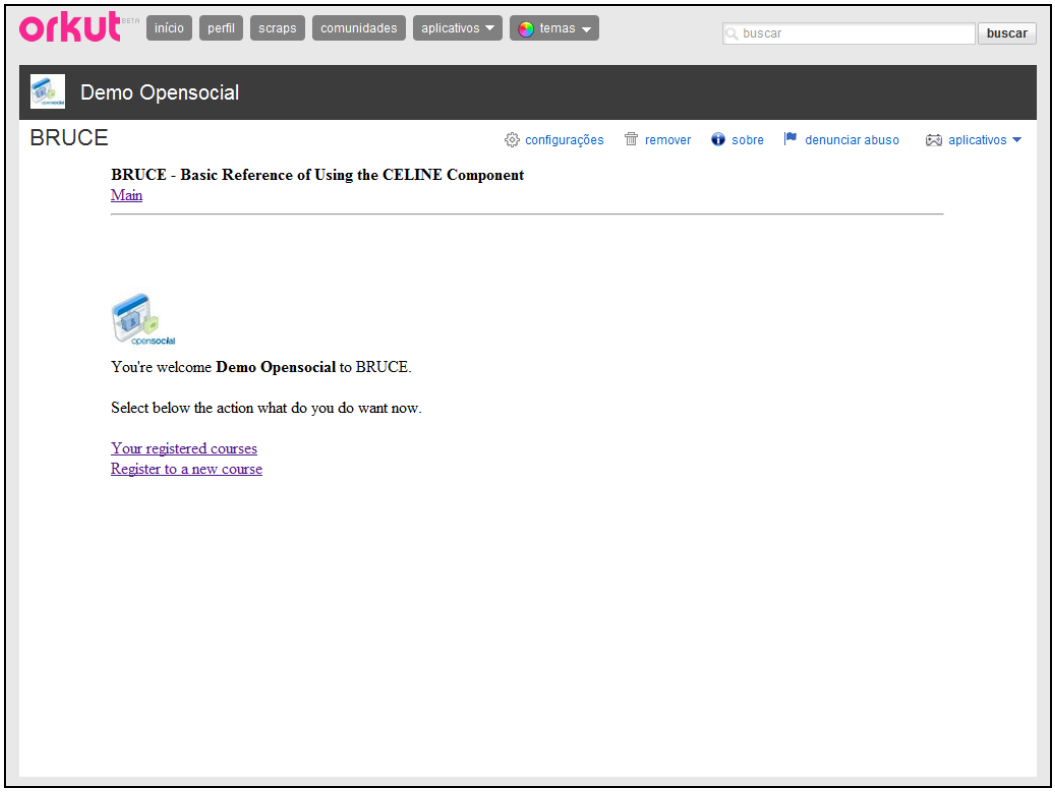


Figura 20 – Tela inicial do *gadget* de nível usuário

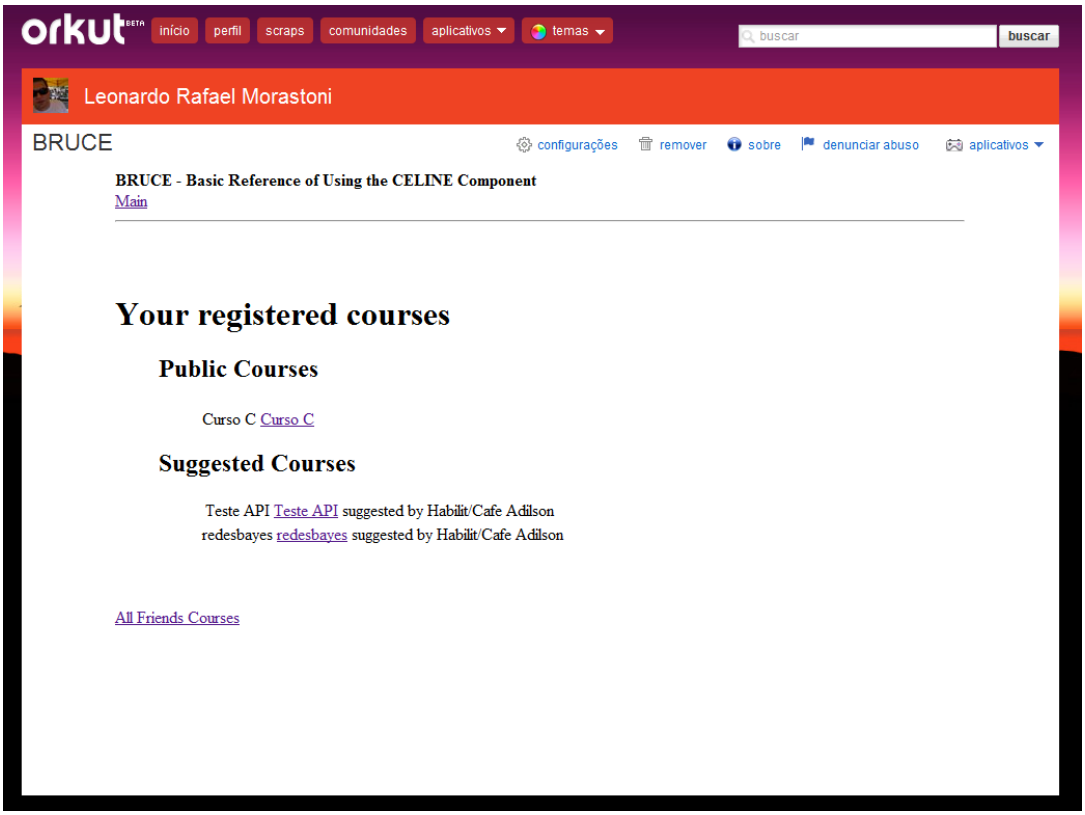


Figura 21 – Tela de cursos registrados

No item “Register to a new course” acessado pelos dois níveis (admin e usuário), o usuário pode realizar o registro e/ou cancelar o registro nos cursos públicos disponíveis (figura 22).

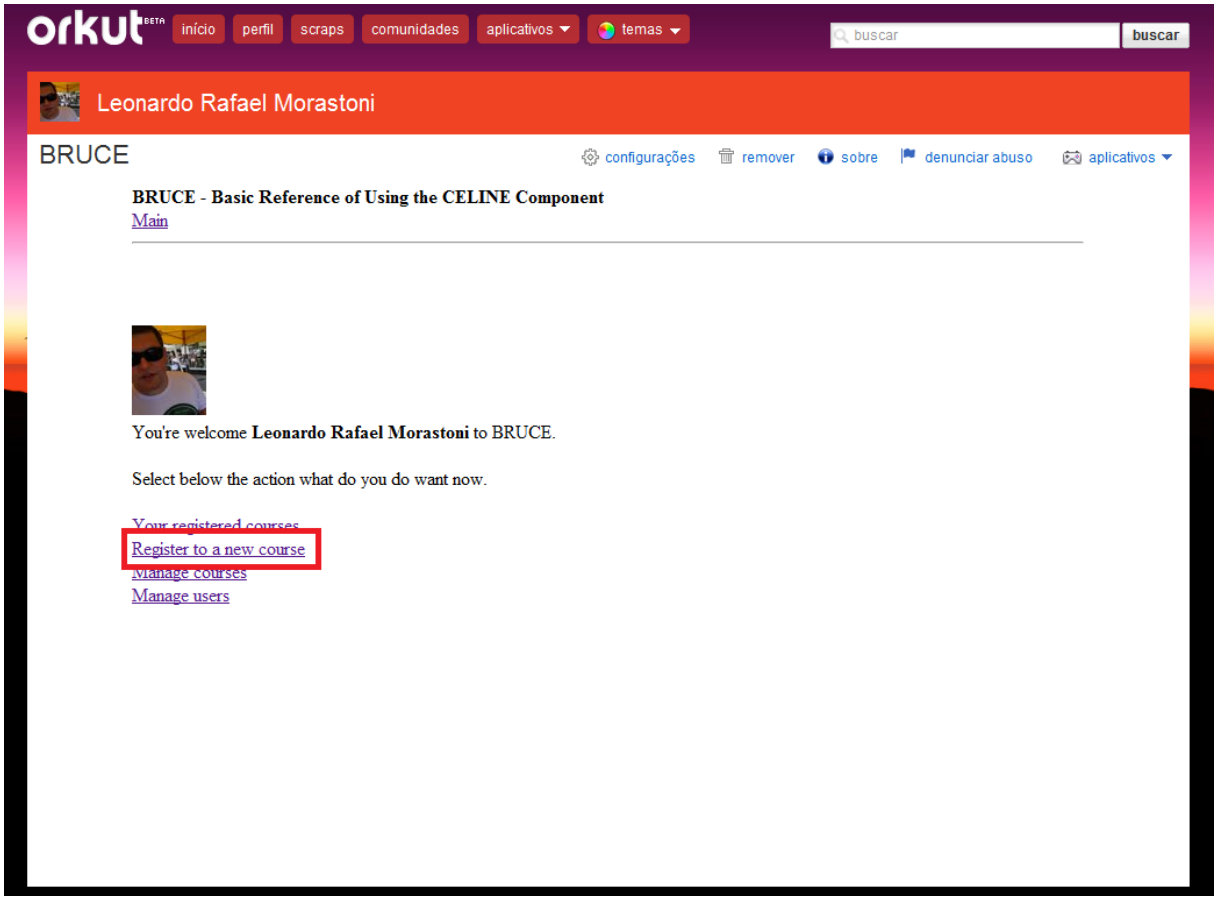


Figura 22 – Tela de registro de curso

Ao acessar o item “Manage Courses”, que requisita nível *admin* para ser acessado, o usuário irá receber a tela apresentada na figura 23, onde estão todos os curso disponíveis listados e onde ele poderá excluir esses mesmos cursos. Há também o item “Import Courses” (figura 24), onde o usuário pode inserir novos pacotes SCORM,habilitar se o curso é público ou se disponibilizará esse curso a um amigo. Para importar novos pacotes, o usuário deve utilizar um arquivo zip, que contenha um pacote de curso na especificação SCORM. Além de inserir o pacote, o usuário precisa adicionar um *id* e um *nome* para o curso. Nesta tela foi implementada uma alteração em um *tag* do CELINE para que fosse possível obter o id do curso e para inserir a *checkbox* “Público?”, recursos necessários para ser possível armazenar os dados capturados da rede social no LMS.

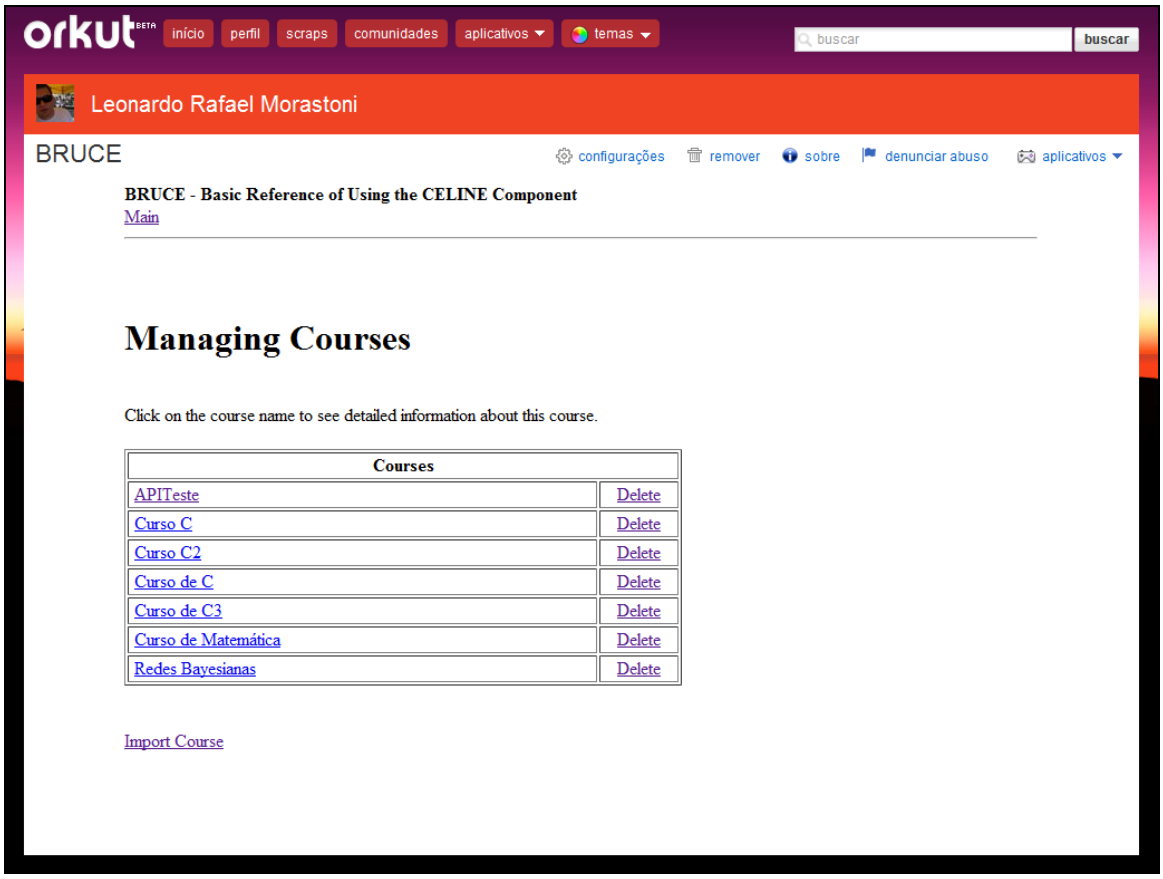


Figura 23 – Tela de manutenção de curso

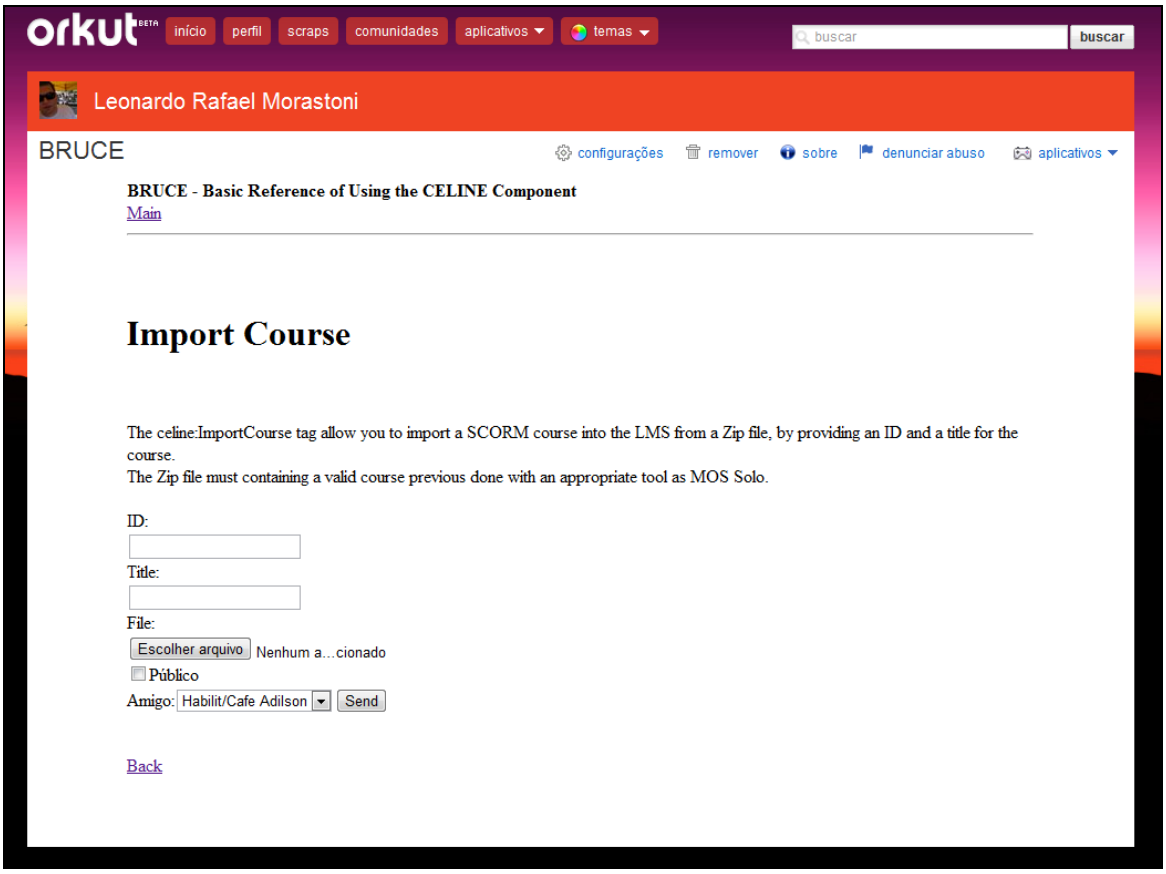


Figura 24 – Tela de importação de curso

Para a administração de usuários,o usuário deve acessar o item “Manage users”, este item também solicita autenticação de um usuário de nível *admin*. Ao acessar a tela de gerenciamento de usuários (figura 25), são listados todos os usuários criados no *gadget*, com a opção de editar e excluir os mesmos.

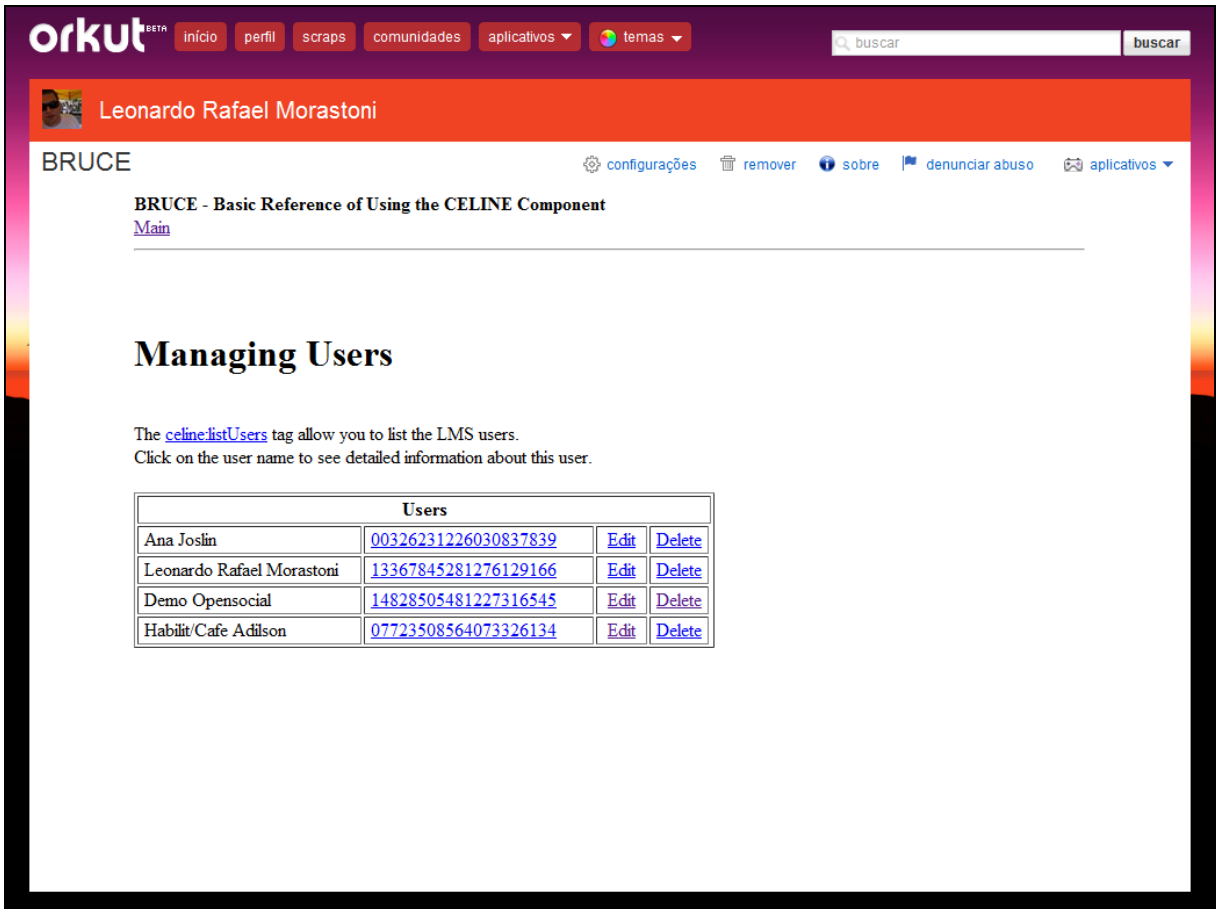


Figura 25 – Tela de gerenciamento de usuários

3.4 RESULTADOS E DISCUSSÃO

O quadro 22 demonstra o comparativo entre as funções executadas pelos *gadget* Udututeach, UdutuLearn da rede social Facebook e o *gadget* desenvolvido nesse trabalho.

Foram utilizados os *gadgets* UdutuTeach e UdutuLearn pois também gerenciam pacotes SCORM e também estão inseridos em uma rede social.

Os requisitos abaixo foram elencados por se tratarem de operações comuns executadas por um LMS que administra pacotes SCORM. Também foram utilizados quesitos que demonstram a interação do LMS com a rede social em que o mesmo esta embutido.

Funcionalidade	BRUCE	UdutuTeach	UdutuLearn
Visualizar cursos	✔	✖	✔
Gerenciar Cursos	✔	✔	✖
Gerenciar Usuários	✔	✔	✔
Convidar amigos	✖	✔	✔
Interface amigável	✖	✔	✔
Iniciar curso	✔	✖	✔
Gerar atualização no perfil	✔	✖	✖
Liberar curso para amigo	✔	✖	✖
Procura de cursos	✖	✖	✔
Gerar certificado de curso	✖	✖	✔

Quadro 22 – Características do BRUCE e dos trabalhos correlatos

Os requisitos acima foram elencados por se tratarem de operações comuns executadas por um LMS que administra pacotes SCORM. Também foram utilizados quesitos que demonstram a interação do LMS com a rede social em que o mesmo esta embutido.

A análise foi realizada a partir dos seguintes aspectos:

a) visualizar cursos: opção que lista todos os cursos disponíveis para usuário iniciar ou se registrar. Esta opção está somente habilitada para o BRUCE e UdutuLearn, pois no aplicativo UdutuTeach somente é possível adicionar novos cursos ;

b) gerenciar cursos: concede a permissão ao usuário de adicionar, editar ou remover cursos. Esta capacidade só é concedida ao UdutuTeach e BRUCE, pois so é possível executar cursos no UdutuLearn;

c) gerenciar usuários: concede permissão de alterar as permissões dos usuários e/ou excluí-los. Aplicável a todos os aplicativos;

d) convidar amigos: função quer permite ao gadget adicionar amigos do usuário. Esta opção não foi implementada no BRUCE;

e) interface amigável: interface de fácil entendimento e organizada. A interface do UdutuTeach e UdutuLearn por ter suas funções divididas é de compreensão mais fácil;

f) iniciar curso: permite ao usuário iniciar um curso onde o mesmo esta registrado. Na implementação UdutuTeach não existem acessos a cursos;

g) gerar atualização no perfil: opção que permite avisar a todos os usuários do gadget que um novo curso público está disponível para registro. Essa opção somente esta habilitada para o BRUCE pois os outros dois aplicativos estão hospedados no Facebook;

h) liberar curso para amigo: função que permite indicar um curso em que o usuário esta registrado para um de seus amigos já registrados. Esta funcionalidade não é possível ser utilizada no UdutuTeach, pois este aplicativo somente permite adicionar novos cursos;

i) procurar curso: permite ao usuário procurar cursos de seu interesse. Esta função não foi implementada no BRUCE e não faz parte das funções do UdutuTeach;

j) gerar certificado de curso: ao concluir um curso o usuário pode gerar o certificado de conclusão desse curso. Esta opção não foi implementada no *gadget* BRUCE.

4 CONCLUSÕES

As atuais condições do aprendizado levam os usuários a desfrutar de vários ambientes de estudo, sejam eles *online* ou não. Com o ensino à distância, verificou-se que poucos ambientes encontrados disponibilizavam compartilhamento de informações entre seus usuários e poucos destes ambientes seguiam uma padronização capaz de gerenciar e avaliar o rendimento dos seus utilizadores.

Os ambientes de redes sociais têm crescido nos últimos anos, mostrando uma tendência de uma nova forma de relacionamento. Alguns destes ambientes tem a sua disposição APIs para integração com outras aplicações. Observando que os maiores meios de interação atuais são os *sites* de relacionamento, surgiu a idéia de migrar um ambiente de aprendizagem gerenciado para um *site* como estes utilizando, a API implementada em redes sociais.

Dentro das redes sociais a palavra chave é colaboração.As redes são um espaço social e, como todo o espaço social, é também um de educação e aprendizado.Cabe explorar essa potencialidade com criatividade, procurando entender como as ferramentas são utilizadas e a partir desse uso, inserir esse novo espaço no processo de aprendizado.Um fator motivador para essa mudança é o grande número de usuários que estão contidos em cada rede social, pois com quanto mais pessoas tem um interesse em comum é possível transformar a colaboração do conhecimento em algo de escala superior, abrangendo cada vez um número maior de participantes. O grande desafio da educação nas redes sociais é manter o foco no conhecimento e na colaboração não deixando espaços para demais atividades que possam ser mais prazerosas ou motivadoras.

Este trabalho descreveu uma forma de utilizar redes sociais através da API OpenSocial permitindo o desenvolvimento de objetos de aprendizagem no padrão SCORM e o seu compartilhamento pelos usuários da rede Orkut.

Este recurso permite que os amigos indiquem cursos que lhes parece interessante a outros amigos, promovendo uma forma alternativa de troca de conhecimento.

Foi realizada uma análise comparativa com os *gadgets* UdutuTeach e UdutuLearn demonstrando a usabilidade do projeto. Nessa comparação foram elencados os processos mais relevantes de cada um dos *gadgets* e confrontados com o LMS desenvolvido.

4.1 EXTENSÕES

Para trabalhos futuros, há a necessidade de adequar o LMS a outras redes sociais que também tem compatibilidade com a OpenSocial.

Outro ponto a elencar é o aumento de interação dos usuários com o ambiente, permitindo que o LMS utilize mais recursos das redes sociais além da redefinição de um nova interface com usuário. Uma forma de realizarmos esse aumento é verificar os requisitos verificados na Quadro 22, onde estão detalhados as funções que o BRUCE não abrange.

Como fator de análise, o *gadget* deverá permitir ao usuário adicionar comentários sobre os cursos e também atrelar uma nota.

REFERÊNCIAS BIBLIOGRÁFICAS

[ADL. **SCORM® 2004 3rd edition content aggregation model (CAM) version 1.0**. ADL: Virginia, 2006.

AGUIAR, Sonia. **Redes sociais na internet:** desafio à pesquisa., Intercom – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação XXX Congresso Brasileiro de Ciências da Comunicação – Santos – 29 de agosto a 2 de setembro de 2007 Disponível em: <http://www.sitedaescola.com/downloads/portal_aluno/Maio/Redes%20sociais%20na%20internet-%20desafios%20%E0%20pesquisa.pdf>. Acesso em: 11 nov. 2010.

ARANHA, Elzo. **Critérios ergonômicos e a concepção de sistema de gerenciamento de aprendizagem virtual (LMS)**. Florianópolis, 2004. Disponível em: <http://www.abepro.org.br/biblioteca/ENEGEP2004_Enegep0402_1871.pdf>. Acesso em: 15 jan. 2011

BAILEY, Warwick.; CURRIER, Sam. **What is ADL SCORM?** Reino Unido, 2005. Disponível em: <http://www.cetis.ac.uk/lib/media/WhatIsSCORM_web.pdf>. Acesso em: 11 nov. 2010.

BOYD, Dannah.; ELLISON, Nicolle. **Social network sites:** Definition, History, and Scholarship. [S.l.], 2007. Disponível em: <http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html>. Acesso: 11 nov. 2010.

FACEBOOK. **Seja bem vindo ao Facebook**. [S.l.], 2009. Disponível em: <http://www.facebook.com/>. Acesso em: 30 abr. 2011.

FOLHAONLINE. **Leia dicionário da web e aprenda a falar "internetês".** [S.l.], 2009. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u462348.shtml>. Acesso em: 06 nov. 2010.

GOOGLE. **A web fica melhor com os relacionamentos**. [S.l.], 2009a. Disponível em: <http://code.google.com/intl/pt-BR/apis/OpenSocial/>. Acesso em: 11 ago. 2010.

_____. **Visão geral da API**. [S.l.], 2009b. Disponível em: <http://code.google.com/intl/pt-BR/apis/gadgets/docs/overview.html>. Acesso em: 15 set. 2010.

_____. **Orkut**. [S.l.], 2009d. Disponível em: <http://www.orkut.com>. Acesso em: 11 set. 2010.

_____. **Google verify**. [S.l.], 2011. Disponível em: <https://www.google.com/gadgets/directory/verify>. Acesso em: 10 abr. 2011.

HILL, Newman. **e-Learning provider udutu unveils first monetized business application to work directly within facebook**. [S.l.], 2008. Disponível em: <http://www.udutu.com/udututeach_pr.pdf>. Acesso em: 15 set. 2009

LIVINGSOCIAL. ReadingSocial gadget. [S.l.], 2009. Disponivel em: <http://orkut.readers.livingsocial.com/orkut/gadget.xml>. Acesso em: 10 set. 2009.

MACHADO, Joicemegue; TIJIBOY, Ana. **Redes sociais virtuais:** um espaço para efetivação da aprendizagem cooperativa. [S.l.], 2005. Disponível em: <http://www.cinted.ufrgs.br/renote/maio2005/artigos/a37_redessociaisvirtuais.pdf>. Acesso em: 14 set. 2009.

OPENSOCIAL. **Articles & tutorials**. [S.l.],2010. Disponível em: <http://wiki.OpenSocial.org/index.php?title=Articles_%26_Tutorials>. Acesso em: 15 nov. 2010.

RECUERO, Raquel. **Redes sociais na internet**. Porto Alegre: Sulina, 2009.

RECUERO, Raquel da C. **Trabalho enviado para o Núcleo de Pesquisa (NP-08) de tecnologias da comunicação e informação do IV encontro dos núcleos de pesquisa da XXVII INTERCOM**, setembro de 2004, em Porto Alegre/RS. Disponível em:< http://galaxy.intercom.org.br:8180/dspace/bitstream/1904/17792/1/R0625-1.pdf >. Acesso em: 11-11-2010.

SANTAROSA, L. **Criação de ambientes de aprendizagem colaborativa.** Curitiba, 1999. Disponível em: <http://penta.ufrgs.br/pgie/sbie99/acac.html>. Acesso em: 2 set. 2009.

SCHLEMMER, Eliane. **AVA:** um ambiente virtual de convivência interacionista sistêmico para comunidades virtuais na cultura da aprendizagem. 2002. 370 f. Disssertação (Mestrado em Educação) - Programa de Pós-Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

SCHLEMMER, Eliane; SACCOL, Amaronilda; GARRIDO, Susane. **Avaliação de ambientes virtuais de aprendizagem na perspectiva da complexidade**. [S.l.], 2006. Disponível em: <https://www.sbc.org.br/bibliotecadigital/download.php?paper=766>. Acesso em: 1 set. 2009.

SITZE, Amy. **Six pieces of advice on how to evalvate a learning management system**. Canadá, 2002. Disponível em: <https://www.amherst.edu/people/facstaff/asitze>. Acesso em: 1 set. 2009.

UDUTUTEACH. **UdutuTeach**. [S.l.], 2009. Disponível em: <http://apps.facebook.com/udututeach/>. Acesso em: 11 set. 2009.

TAKAMOTO, Tiago; NAVARRO, Rafael; VAHL, José. **OpenSocial ,** a tradução da torre de babel nas redes sociais. Java Magazine, [S.1.], ed. 68, 54-65. fev. 2011.

TANENBAUM, Andrew S. **Redes de Computadores**. Tradução Vanderberg D. de Souza. Rio de Janeiro: Elselvier, 2003.

VAHLDICK, Adilson. **Celine:** um modelo para utilização e adaptação de conteúdo SCORM em ambientes inteligentes de aprendizagem. 2008. 140 f. Dissertação (Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura) - Programa de Mestrado Acadêmico em Computação Aplicada, Universidade do Vale do Itajaí, São José, 2008.

VAHLDICK, Adilson; RAABE, Andre. L. A. **Adaptação de conteúdo SCORM em ambientes inteligentes de aprendizagem**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 19., 2008, Fortaleza. **Anais...** Fortaleza: Sociedade Brasileira de Computação, 2008. p. 53 - 68. 1 CD-ROM.