Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Infraestrutura para o Desenvolvimento de Aplicações Pervasivas Cientes de Redes Sociais

Daniel Bruno Alves dos Santos

Campina Grande, Paraíba, Brasil
Maio – 2011

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Infraestrutura para o Desenvolvimento de Aplicações Pervasivas Cientes de Redes Sociais

Daniel Bruno Alves dos Santos

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Angelo Perkusich (Orientador)
Hyggo Oliveira de Almeida (Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S237 Santos, Daniel Bruno Alves dos.
Infraestrutura para o Desenvolvimento de Aplicações Pervasivas Cientes de Redes Sociais / Daniel Bruno Alves dos Santos. ─ Campina Grande, 2011.
104 f.: il. col.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientadores: Profº. Dr. Perkusich, Angelo e Profº. Dr. Almeida, Hyggo Oliveira de.
Referências.

1. Computação Pervasiva. 2. Redes Sociais Móveis. 3. Redes Sociais. 4. Desenvolvimento de Aplicações Móveis. I. Título.

CDU 004.4'2(043)

"INFRAESTRUTURA PARA O DESENVOLVIMENTO DE APLICAÇÕES PERVASIVAS CIENTES DE REDES SOCIAIS"

DANIEL BRUNO ALVES DOS SANTOS

DISSERTAÇÃO APROVADA EM 06.05.2011

PROF. ANGELO PERKUSICH, D.Sc
Orientador

PROF. HYGGO OLIVEIRA DE ALMEIDA, D.Sc
Orientador

PROFª MARIA DE FÁTIMA QUEIROZ VIEIRA, Ph.D
Examinadora

PROF. LEANDRO DIAS DA SILVA , D.Sc
Examinador

KYLLER COSTA GORGÔNIO, Dr.
Examinador

CAMPINA GRANDE – PB

Resumo

Os últimos anos têm sido caracterizados pela popularização do acesso à Internet, extraordinário avanço das tecnologias de comunicação sem fio e massificação dos dispositivos móveis no mercado consumidor. Esse cenário vem se mostrando propício para a viabilização do paradigma de Computação Pervasiva e o surgimento de aplicações para esse domínio. Ainda, esse avanço motivou o surgimento das Redes Sociais Móveis, resultantes da integração entre o paradigma de Computação Pervasiva com os serviços sociais da web 2.0.

As soluções existentes no desenvolvimento de aplicações para Redes Sociais Móveis não fornecem suporte a requisitos importantes desse domínio: (i) suporte à execução de serviços e acesso às informações de contexto social e de localização dos usuários; (ii) integração com as Redes Sociais Online para construção de Redes Sociais Móveis; (iii) conhecimento prévio sobre quem são os amigos diretos e indiretos dos usuários, e; (iv) conhecimento em tempo real da localização dos usuários.

Neste trabalho apresenta-se uma infraestrutura para o desenvolvimento de aplicações para Redes Sociais Móveis que disponibiliza o conjunto de funcionalidades anteriormente enumeradas. Como contribuição principal, também é disponibilizado um módulo para processamento distribuído das conexões sociais, representando os amigos diretos e indiretos dos usuários. A avaliação do trabalho constituiu-se do desenvolvimento de um estudo de caso e realização de simulações que demonstram o suporte da infraestrutura para o desenvolvimento de aplicações e a escalabilidade do módulo proposto.

Abstract

The past years has been characterized by widespread access to the Internet, extraordinary advances in wireless communications technology and mobile devices massification in consumer market. This scenario has proved to be conducive to the viability of Pervasive Computing paradigm and the emergence of applications for this domain. Moreover, this progress has motivated the emergence of Mobile Social Networks, that is the result from the integration of the Pervasive Computing paradigm with social services from Web 2.0.

The existing solutions for development of applications to Mobile Social Networks do not provide support for important requirements, such as: support to perform services and access to social and location context information of users, (ii) integration with Online Social Networks for building Mobile Social Networks. (iii) prior knowledge about who are the direct and indirect users' friends, and (iv) knowledge in real time of users location.

This work presents an infrastructure for developing applications to Mobile Social Networks, which provides the feature set listed above. As main contribution is also made available a mechanism for distributed processing of users' social connections, which represents the direct and indirect friendship connections of users. The evaluation of the work consisted of developing a case study and performing simulations that demonstrate the infrastructure utility and the scalability of the proposed module.

Agradecimentos

Agradeço primeiramente a Deus, por me proporcionar a vida, os desafios, as dificuldades e a força para superar mais um obstáculo.

Aos meus pais e irmã, Jeronimo, Neves e Jeane que me apoiam em todos os momentos.

A minha namorada, Wania, a quem expliquei tantas vezes o meu trabalho para que eu mesmo entendesse o que deveria fazer e mantivesse o foco. Obrigado pela paciência, compreensão e companheirismo de todos esses anos juntos.

Aos amigos do Residencial Flamingo, pela descontração de todos os momentos.

Também, aos amigos que fiz no mestrado, provenientes dos mais diversos lugares: Campina Grande, João Pessoa, Maceió e Teresina. Em especial aos amigos do Laboratório Embedded, com os quais desopilei em vários momentos e que contribuíram direta ou indiretamente com ideias valiosas para o desenvolvimento deste trabalho.

Aos professores e, principalmente, funcionárias da COPIN sempre tão solícitas.

Aos meus orientadores Angelo e Hyggo, pela contribuição substancial para a concretização deste trabalho e orientação recebida.

Ao CNPq, pelo apoio financeiro.

Conteúdo

Lista de Símbolos

3G - *3rd Generation*

A-GPS - *Assisted GPS*

API - *Application Programming Interface*

BFS - *Breadth-First Search*

CLDC - *Connected Limited Device Configuration*

DAO - *Data Access Object*

DFS - *Depth-First Search*

FOAF - *Friend of a Friend*

GPS - *Global Positioning System*

HTTP - *HyperText Transfer Protocol*

Java EE - *Java Enterprise Edition*

Java ME - *Java Micro Edition*

Java SE - *Java Standard Edition*

JAR - *Java ARchive*

JS - *JavaScript*

JSP - *Java Server Pages*

JSON - *JavaScript Object Notation*

JSR - *Java Specification Request*

LBS - *Location-based Services*

MIDP - *Mobile Information Device Profile*

MSCAs - *Mobile Social Computing Applications*

NFC - *Near Field Communication*

NFS - *Network File System*

OWL - *Web Ontology Language*

PR - *Page Ranking*

QoS - *Quality of Service*

RDF - *Resource Description Framework*

REST - *Representational State Transfer*

RF - *Requisito Funcional*

RMI - *Remote Method Invocation*

RPC - *Remote Procedure Call*

SAMOA - *Socially Aware and Mobile Architecture*

SCSI - *Small Computer System Interface*

SGBD - *Sistema de Gerenciamento de Banco de Dados*

SOA - *Service Oriented Architecture*

URL - *Uniform Resource Locator*

XML - *eXtensible Markup Language*

WLAN - *Wireless Local Area Network*

WS - *Web Service*

Wi-Fi - *Wireless Fidelity*

Wi-Max - *Worldwide Interoperability for Microwave Access*

WWW - *World Wide Web*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Nos últimos anos, tem-se observado a popularização do acesso à Internet e a massificação dos dispositivos móveis no mercado consumidor, principalmente, *notebooks*, *netbooks*, aparelhos celulares e *smartphones*. Os dois últimos, em especial, estão se tornando cada vez mais baratos, acessíveis e com uma grande gama de recursos disponíveis, tais como câmera digital, vídeo, GPS, bússola digital, acelerômetro, conexões *Wi-Fi*, 3G, *Bluetooth*, dentre outros. Consequentemente, tem-se tornado cada vez mais comum o fato das pessoas estarem sempre portando seus dispositivos móveis, assim como, passarem grande parte do tempo conectadas à Internet, em qualquer lugar, a qualquer momento.

Esse cenário vem se mostrando propício para a viabilização do paradigma de Computação Pervasiva [64], no qual dispositivos executam aplicações inteligentes e integram-se de forma transparente à vida das pessoas, auxiliando-as na realização de suas tarefas.

O avanço no paradigma de Computação Pervasiva, impulsionado pela popularização do acesso à Internet, motivou o surgimento de um domínio denominado Redes Sociais Móveis ou Redes Sociais Pervasivas [7, 48], que estuda a integração entre o paradigma de Computação Pervasiva com os serviços sociais da web 2.0 (e.g., [3, 13, 14, 20, 35, 66], entre outros). Essa integração possibilita aos usuários, que compartilham interesses em comum e frequentam os mesmos lugares no dia-a-dia, interagirem não apenas em ambientes virtuais [57], mas principalmente em ambientes reais para realizarem atividades de interesse comum, em qualquer lugar, a qualquer momento [15, 36].

Para que esse paradigma funcione plenamente é primordial que as aplicações sejam cientes de contexto [19], ou seja, possam recuperar as informações do ambiente em que se encon-

tram, com a finalidade de prover serviços e executar tarefas em favor do usuário [42, 58, 60].

Aplicações pervasivas para Redes Sociais Móveis possuem ciência de pelo menos duas informações de contexto relacionadas aos usuários: o contexto social do usuário (*social-awareness*) e o contexto de localização (*location-awareness*). O primeiro contém a informação sobre quem são os amigos do usuário e quais são as suas preferências, interesses e atividades. Já o último contém a informação sobre onde esses amigos estão localizados, a partir da aplicação de uma ou da combinação de várias técnicas de localização existentes [39, 56].

Uma vez de posse das informações de contexto social e de localização dos usuários, essas aplicações são capazes de perceber em tempo real a proximidade dos amigos, membros da comunidade ou de alguém cujo perfil possua um alto grau de similaridade ao do usuário. Com isso, as aplicações podem pró-ativamente alertar o usuário através de seu dispositivo móvel, ou ainda, recomendar o novo contato como potencial amigo, caso ele ainda não faça parte da lista de amigos do usuário.

Ainda, a integração das informações presentes nas redes sociais com o paradigma de Computação Pervasiva possibilita o desenvolvimento de novas aplicações, que tiram proveito dos recursos disponíveis nos dispositivos móveis. Alguns exemplos são os Serviços Baseados em Localização (LBS) [39] e os serviços que combinam as informações de localização com aquelas capturadas a partir da câmera do dispositivo móvel, adicionando outras informações ao que está sendo visto pelo usuário na tela, conceito conhecido como realidade aumentada [24].

Pode-se citar também aplicações que combinam informações de localização com aquelas provenientes dos perfis dos usuários das redes sociais, tais como: Google Latitude [29], Serendipity [21], WhozThat [5], dentre outras. Essas aplicações possibilitam aos usuários visualizarem, na tela dos seus dispositivos móveis, os amigos que estão próximos em tempo real. Assim, torna-se possível que eles entrem em contato uns com os outros para conversarem em um ambiente real e não mais virtual [10].

Com vistas a possibilitar o desenvolvimento de aplicações pervasivas cientes de contexto social são necessárias soluções que lidem com alguns desafios, dentre eles: heterogeneidade dos dispositivos móveis, tolerância a falhas nas redes, segurança e privacidade das informações dos usuários e outros [17, 58]. Ademais, essas soluções devem: abstrair o processo de comunicação com outras aplicações; permitir a aquisição, persistência e reutilização das in-

formações de contexto social e de localização armazenadas nos dispositivos dos usuários; e disponibilizar APIs para que os desenvolvedores possam acessar as informações de contexto e construir rapidamente novas aplicações sociais [10, 30, 62, 65].

1.1 Problemática

Com o propósito de dar suporte ao desenvolvimento rápido de aplicações para Redes Sociais Móveis, têm surgido diversas propostas de *middlewares* [8, 10, 30, 43, 48, 51, 63] que tentam capturar as informações de contexto armazenadas nos dispositivos móveis dos usuários com a finalidade de construir os perfis dos usuários e formar Redes Sociais Móveis. No entanto, as soluções de *middleware* existentes suportam parcialmente o desenvolvimento dessas aplicações, pois não disponibilizam todas as funcionalidades necessárias à sua construção. O cenário a seguir descreve uma aplicação para Redes Sociais Móveis que não é contemplada em sua totalidade pelas soluções de *middleware* existentes.

João está caminhando em direção à praça de alimentação do shopping de sua cidade na hora do almoço, quando decide iniciar uma aplicação instalada em seu smartphone para saber se algum dos seus amigos está próximo para almoçarem juntos. A aplicação utilizada por João envia, através da Internet, a informação sobre a localização dos usuários que a utilizam para um servidor externo que, além de receber e armazenar essas informações, também possui ciência das redes sociais online da qual os usuários fazem parte.

Utilizando as informações de contexto disponíveis, o servidor determina em tempo real, que não há nenhum amigo direto de João próximo da praça de alimentação. No entanto, também possui a informação de que João é amigo, no Orkut, de José, o qual também é membro do Facebook, onde é amigo de Pedro. Coincidentemente, Pedro é um amigo que João não vê desde os tempos de faculdade e está localizado a poucos metros de distância. Dessa forma, a aplicação informa a João que Pedro é amigo de José no Facebook e está próximo. João, por sua vez, observa a localização de Pedro no mapa gerado pela aplicação e decide ir ao seu encontro para conversarem e almoçarem juntos.

Além do cenário descrito anteriormente, outros exemplos de aplicações cientes de redes sociais que exploram as informações de contexto social e de localização são: propaganda direcionada aos usuários de uma mesma rede social móvel (e.g., "Envio de promoções sobre

produtos ou serviços de interesse do grupo, para os membros da rede social móvel"); serviços de recomendação (e.g., "Recomendação de amizade ou de lugares a serem visitados para usuários com interesses em comum e que frequentam os mesmos lugares no dia-a-dia"); serviços de micro-pagamento a partir dos dispositivos móveis (e.g., "Oferta dos recursos embarcados e/ou serviços disponíveis no dispositivo móvel dos usuários sob demanda e cobrança pela sua utilização").

Para desenvolver tais aplicações, faz-se necessário que exista uma abordagem de *middleware* para Redes Sociais Móveis que execute as seguintes atividades: (i) utilização das conexões sociais presentes nas múltiplas redes sociais online que os usuários façam parte para formar Redes Sociais Móveis; (ii) representação única e sem ambiguidades do perfil do usuário; (iii) persistência das informações de contexto social e de localização dos usuários; (iv) processamento e armazenamento dos níveis de relacionamento (diretos e indiretos) entre todos os usuários de uma rede social móvel; e (v) descoberta em tempo real da localização de todos os membros da rede social móvel.

As soluções de *middleware* existentes utilizam duas abordagens distintas para formação de Redes Sociais Móveis. A primeira é a formação de comunidades móveis através da descoberta *ad hoc* dos dispositivos móveis dos usuários [5,7,8,10,21,25,30,32,43,48,51,55,63], que compartilham interesses ou preferências em comum e/ou realizam atividades em um mesmo espaço físico. A segunda é a utilização das conexões sociais existentes nas redes sociais online [46,51].

Abordagens de *middleware* como [30, 51] apresentam modelos sintáticos para representação do perfil de pessoas e lugares. Especificamente, em [51] apresenta-se um modelo com base nas informações de perfil dos usuários do Facebook [66]. Outras soluções como [7,10,55,62] apresentam abordagens semânticas, baseadas em ontologias de domínio, FOAF [12] e RDF [6], para modelar os interesses dos usuários e suas atividades. O trabalho apresentado em [62] permite a modelagem das interações sociais envolvendo conteúdos digitais produzidos pelos usuários.

Também, soluções como [10,30,51] apresentam funcionalidades que extraem e persistem as informações de contexto social e de localização dos usuários do ambiente. Todavia, apenas em [30, 51] são propostas APIs para acesso aos dados de contexto social e de localização persistidos e suportam o desenvolvimento de aplicações sociais por terceiros.

Ainda, algumas soluções encontradas na literatura apresentam técnicas e algoritmos para extrair as informações de contexto social e de localização, com o objetivo de inferir relações sociais [41, 50] e analisar padrões de utilização dos dispositivos móveis [22, 23].

O serviço de *middleware* proposto em [30] disponibiliza um módulo de aprendizagem de contexto social que utiliza algoritmos para formação de grupos sociais e descoberta de padrões geo-sociais [31] previamente desconhecidos, tais como afinidades entre pessoas e entre pessoas e lugares. O *framework* desenvolvido em [54] utiliza registros das conexões *ad hoc*, ocorridas entre os dispositivos móveis dos usuários, em conjunto com algoritmos para prever novos relacionamentos de amizade, para construção de listas de recomendação. Todavia, os algoritmos utilizados pelas soluções anteriores não fornecem conhecimento sobre os níveis de relacionamentos (diretos e indiretos) entre os usuários. Por exemplo, não é possível determinar quais são os amigos dos amigos do usuário A (i.e. dois níveis de relacionamento) que fazem parte da lista de contatos do usuário B. Ou ainda, determinar se o usuário A está relacionado indiretamente ao usuário B, através de um relacionamento do tipo amigo do amigo do amigo (i.e. três níveis de relacionamento).

Por fim, iniciativas como [27, 28] disponibilizam um conjunto de APIs para que aplicações sociais acessem as informações dos perfis das redes sociais online e as conexões sociais presentes nos diferentes serviços sociais existentes na web, respectivamente.

Apesar de existir uma grande variedade de soluções que auxiliam no desenvolvimento de aplicações para Redes Sociais Móveis, elas ainda não fornecem suporte a requisitos importantes desse domínio. Especificamente, não há uma abordagem isolada que forneça as seguintes funcionalidades: (i) suporte à execução de serviços e acesso às informações de contexto social e de localização dos usuários; (ii) integração com as múltiplas redes sociais online dos usuários para formação das Redes Sociais Móveis; (iii) conhecimento prévio sobre quem são os amigos diretos e indiretos dos usuários em vários níveis de amizade (e.g., amigos, amigos dos amigos e assim por diante), e; (iv) conhecimento em tempo real da localização dos usuários.

1.2 Objetivos

Neste trabalho tem-se como objetivo desenvolver uma infraestrutura para a construção de aplicações pervasivas cientes de redes sociais, no contexto de Redes Sociais Móveis. Essa infraestrutura deve ser capaz de fornecer mecanismos para aquisição e persistência das informações de contexto social e de localização, provenientes das redes sociais online e dos dispositivos móveis dos usuários, assim como disponibilizar para as aplicações acesso às informações de contexto persistidas e aos serviços fornecidos pela infraestrutura.

Além disso, a infraestrutura deve disponibilizar um mecanismo para aumentar o conhecimento prévio sobre as informações de contexto social persistidas, disponibilizando as informações sobre os relacionamentos de amizade em vários níveis para todos os usuários de uma Rede Social Móvel.

Portanto, os objetivos específicos deste trabalho são: (i) adquirir as informações de contexto dos usuários das redes sociais online; (ii) disponibilizar a localização dos usuários em tempo real; (iii) desenvolver um módulo que determine os níveis de relacionamentos existentes entre todos os membros de uma rede social móvel; e (iv) disponibilizar uma API para que as aplicações executem os serviços da infraestrutura e acessem as informações de contexto social e de localização disponibilizadas.

A infraestrutura disponibilizará o Módulo Online para construção de Redes Sociais Móveis, que utiliza a abordagem baseada em redes sociais online para formação de Redes Sociais Móveis.

O Módulo Online funcionará a partir de um servidor central, que utiliza uma conexão direta com a Internet para recuperar as informações sociais (e.g., dados e preferências pessoais, lista de amigos, dentre outras informações) contidas nos perfis dos usuários nas redes sociais online. As informações recuperadas são armazenadas pela infraestrutura e auxiliam na criação do contexto social dos usuários e na formação de redes sociais móveis.

A infraestrutura também disponibilizará um módulo para aumentar o conhecimento prévio sobre as informações de contexto social persistidas pelo Módulo Online. Esse módulo determina os relacionamentos de amizade em vários níveis para todos os membros da Rede Social Móvel, a partir das informações de contexto social extraídas das redes sociais online. Dessa forma, a infraestrutura pode recomendar a formação de novos vínculos de amizade

entre usuários que compartilham interesses em comum, com isso, evoluindo as informações de contexto dos usuários. Ao final, essas informações são armazenadas para que possam ser re-utilizadas, posteriormente, por outras aplicações.

Como forma de validação da infraestrutura foram utilizados dois mecanismos de avaliação. O primeiro mecanismo constituiu-se no desenvolvimento de um estudo de caso para demonstrar o suporte da infraestrutura à criação de aplicações sociais cientes de contexto social e de localização. A aplicação desenvolvida foi um serviço social móvel baseado em localização, desenvolvido na plataforma Java ME, que apresenta através de lista de contatos e de um mapa quais são os amigos diretos e indiretos de um usuário que estão próximos. Já o segundo mecanismo, constituiu-se em uma avaliação de escalabilidade utilizando simulação sobre o Módulo de Expansão das Conexões Sociais, que determina os relacionamentos de amizade em vários níveis para todos os membros da Rede Social Móvel, com o objetivo de avaliar a escalabilidade do módulo proposto.

1.3 Relevância

O desenvolvimento deste trabalho possui como principal contribuição a criação de uma infraestrutura que facilita e torna transparente para o desenvolvedor o processo de formação e gerenciamento de Redes Sociais Móveis personalizadas. Desta forma, é possível, através da infraestrutura, explorar as informações de contexto social e de localização disponibilizadas e focar nas regras de negócio das aplicações, não necessitando preocupar-se com o processo de formação das Redes Sociais Móveis.

Além disso, a infraestrutura possibilita o desenvolvimento de novas aplicações pervasivas, cientes de contexto social e de localização dos usuários. Isso é possível, pois ela utiliza módulos que extraem as informações de contexto social das redes sociais online e que determinam os relacionamentos de amizade diretos e indiretos para todos os membros da Rede Social Móvel. Dessa forma, a infraestrutura pode disponibilizar, através de serviços web, as informações de contexto social e de localização dos usuários, para que outras aplicações instaladas no servidor ou nos dispositivos móveis dos usuários finais possam utilizá-las.

Por fim, este trabalho, se insere dentro do contexto do projeto PerComp, desenvolvido

no Laboratório de Sistemas Embarcados e Computação Pervasiva[1], da Universidade Federal de Campina Grande, contribuindo para o avanço da pesquisa em Computação Pervasiva.

1.4 Estrutura da Dissertação

O restante deste trabalho está organizado da seguinte forma:

- No *Capítulo 2* é apresentada uma fundamentação teórica sobre o paradigma de Computação Pervasiva, destacando seus princípios e os fatores necessários para sua viabilização. Em seguida, apresenta-se uma fundamentação sobre o domínio das Redes Sociais online. Por fim, caracteriza-se o domínio das Redes Sociais Móveis, resultante da integração entre o paradigma de Computação Pervasiva com o domínio das Redes Sociais online;
- No *Capítulo 3* são apresentados os trabalhos relacionados;
- No *Capítulo 4* é apresentado o projeto da infraestrutura desenvolvida, descrevendo detalhadamente os componentes da sua arquitetura, bem como, sua implementação. Destacando, principalmente, o Módulo de Expansão das Conexões Sociais e o seu funcionamento;
- No *Capítulo 5* é apresentado o estudo de caso desenvolvido a partir das informações de contexto e dos serviços disponibilizados pela infraestrutura;
- No *Capítulo 6* é descrita uma avaliação de escalabilidade utilizando simulação sobre o Módulo de Expansão das Conexões Sociais;
- Por fim, no *Capítulo 7* são apresentadas as conclusões do trabalho, suas contribuições e discussões sobre trabalhos futuros.

[1] `http://embedded.ufcg.edu.br/`

Capítulo 2

Fundamentação Teórica

Neste capítulo é apresentado o embasamento teórico que norteia os principais domínios de conhecimentos envolvidos neste trabalho. Inicialmente, será apresentada uma breve descrição sobre o paradigma de Computação Pervasiva, destacando seus princípios e os fatores necessários para sua viabilização. A segunda parte deste capítulo descreve o domínio das Redes Sociais online. Ao final, é apresentado o domínio das Redes Sociais Móveis, resultante da integração entre o paradigma de Computação Pervasiva com o domínio das Redes Sociais online.

2.1 Computação Pervasiva

Em 1991, Mark Weiser publicou seu trabalho contendo as bases para um novo paradigma, denominado Computação Pervasiva [64]. Nesse trabalho, Weiser vislumbrou um mundo no qual a computação e as aplicações estariam embutidas nos objetos do dia-a-dia, de forma integrada ao ambiente. Esses objetos seriam capazes de transparentemente interagir uns com os outros sem serem notados ou requererem atenção dos usuários. Nessa perspectiva, a computação seria a entrada para um repositório de recursos computacionais utilizados sem o conhecimento dos usuários, com o objetivo de realizar tarefas de acordo com as suas necessidades e preferências e tornar disponíveis informações relevantes para eles, no lugar certo e no momento certo [42].

Para que a visão de Weiser sobre o paradigma de Computação Pervasiva pudesse ser concretizada, três elementos principais deveriam estar largamente disponíveis, conforme desta-

cado pela Figura 2.1: *dispositivos baratos e com baixo consumo de energia*, *infraestrutura de rede sem fio* e *sistemas que implementassem aplicações pervasivas* [42]. No entanto, à época que Weiser lançou os fundamentos para o paradigma de Computação Pervasiva, esses três elementos não apresentavam um estágio de desenvolvimento avançado, nem a massificação e popularização necessárias para tornar realidade as primeiras aplicações pervasivas.

Figura 2.1: Visão geral dos princípios de Computação Pervasiva. Fonte: adaptado de [59].

Nos últimos anos, esse cenário sofreu uma mudança significativa com a popularização do acesso à Internet, em conjunto com o extraordinário avanço das tecnologias de comunicação sem fio (e.g., *Bluetooth*, Wi-Fi, 3G, Wi-Max, entre outros) e a massificação dos dispositivos móveis no mercado consumidor, principalmente, *notebooks*, *netbooks*, aparelhos celulares e *smartphones*. Os dois últimos, em especial, estão se tornando cada vez mais baratos, acessíveis e com uma grande gama de recursos disponíveis, tais como câmera digital, vídeo, GPS, bússola digital, acelerômetro, conexões *Wi-Fi*, 3G, *Bluetooth*, dentre outros. Consequentemente, tem-se tornado cada vez mais comum o fato das pessoas estarem sempre portando seus dispositivos móveis, assim como, passarem grande parte do tempo conectadas à Internet, em qualquer lugar, a qualquer momento. Esse novo cenário tem se mostrado propício para a viabilização do paradigma de Computação Pervasiva e para o desenvolvimento de aplicações que possam ser executadas em ambientes pervasivos.

2.1.1 Ambientes Pervasivos

Ambientes pervasivos podem ser definidos como ambientes saturados com capacidade de computação e comunicação [60]. Dentre as suas principais características, destacam-se a heterogeneidade e a dinamicidade. A primeira característica está relacionada com a diversidade de: dispositivos com poder de processamento distintos que podem estar presentes no ambiente (e.g., celulares, *smartphones*, *netbooks*, *notebooks*, computadores pessoais, etc); interfaces de comunicação sem fio (e.g., *Bluetooth*, *Wi-Fi*, 3G, NFC, etc); Sistemas Operacionais; etc. Já a segunda característica está relacionada à: capacidade de mobilidade de tais dispositivos, permitindo que estes possam entrar/sair do ambiente a qualquer momento; capacidade de um dispositivo passar a fornecer ou deixar de fornecer algum serviço.

No nível de rede, a mobilidade e a descoberta de usuários são duas características importantes para ambientes pervasivos. A primeira permite que as aplicações movam-se entre as diversas redes existentes em um ambiente pervasivo, sem interromper a execução de um serviço. A segunda característica permite que os dispositivos descubram uns aos outros na rede e que possam trocar informações, compartilhar recursos, ou ainda, executar os serviços disponíveis na rede.

A ideia de dispositivos transparentemente integrados aos seres humanos aliada à necessidade de lidar com as características de ambientes pervasivos, requer que as aplicações se adaptem em tempo de execução às alterações no ambiente e às necessidades dos usuários. Dentro do escopo de Computação Pervasiva, esta adaptabilidade é guiada por dois elementos chave: as noções de contexto e ciência de contexto.

2.1.2 Contexto e Ciência de Contexto

Para que o paradigma de Computação Pervasiva funcione plenamente é primordial que as aplicações recuperem as informações do ambiente no qual os usuários se encontram, com a finalidade de prover serviços e executar tarefas em favor deles [42, 58, 60]. Para atingir esse objetivo, as aplicações devem possuir ciência de contexto. A definição que melhor representa o que vem a ser contexto foi proposta em [19]:

> "Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um lugar ou um

> objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação."

De acordo com Satyanarayanan, em [60], o uso eficaz da informação de contexto é um elemento fundamental na Computação Pervasiva, como um meio para alcançar o recurso de invisibilidade imaginado por Weiser. A fim de atingir esse objetivo, é necessário descobrir quais são as necessidades e interesses dos usuários e recuperar as informações de contexto disponíveis no ambiente no qual eles estão inseridos, bem como, dos seus dispositivos móveis.

Com base nas necessidades e interesses dos usuários, as aplicações podem definir que tipo de informação é relevante para os usuários. Essas informações podem ser obtidas dos ambientes nos quais os usuários se encontram, ou ainda dos dispositivos móveis que eles carregam consigo. Especificamente, essas informações podem ser provenientes de dados históricos (e.g., locais que o usuário costuma frequentar, compras de produtos, *feedbacks* sobre serviços, etc), ou ainda, podem ser fornecidas explicitamente pelos usuários.

Aplicações que utilizam as informações de contexto para determinar quais informações são úteis aos usuários, ou mesmo, quais serviços devem ser executados são denominadas *aplicações sensíveis ao contexto*, ou ainda, *aplicações cientes de contexto* [19]. A fim de habilitar a ciência de contexto nas aplicações pervasivas, faz-se necessário que essas aplicações executem três processos: *aquisição*, *representação* e *raciocínio* sobre as informações de contexto.

A aquisição de contexto estuda as formas como as informações contextuais podem ser obtidas do ambiente. De acordo com [49], as formas de aquisição de contexto podem ser classificadas em: sentida, derivada e explicitamente fornecida. Por exemplo, as informações de contexto podem ser: *sentidas* através de sensores físicos instalados no ambiente (e.g., sensores de proximidade, temperatura, ruído, entre outros); *derivadas* da execução de um recurso computacional sob demanda (e.g., cálculo da distância em metros entre dois usuários, com base nas coordenadas geográficas obtidas de um GPS); ou ainda, *explicitamente fornecida* pelo usuário (e.g., mediante o preenchimento de formulário eletrônico de uma página web).

Alguns exemplos de contexto, cujas informações podem ser adquiridas são: contexto do usuário; contexto de localização; contexto social; contexto do dispositivo; contexto do

ambiente; dentre outros [42].

Uma vez adquiridas, as informações de contexto devem ser disponibilizadas para as entidades interessadas. Isto implica que essas informações devem ser representadas em um formato padrão. Assim, quaisquer entidades, que estejam interessadas nas informações, poderão compreendê-las. Como já foi apontado em [34], a representação das informações de contexto, deverão ser estruturadas, intercambiáveis, combináveis/decomponíveis, uniformes, extensíveis e padronizadas.

Por fim, o processo de raciocínio interpreta as informações de contexto e, com base no resultado dessa interpretação, determina quais ações devem ser realizadas pelas aplicações.

2.2 Redes Sociais, Comunidades Virtuais e Redes Sociais Online

O estabelecimento de conexões ou relacionamentos sociais com outros indivíduos, seja através da formação de grupos ou comunidades, é uma característica intrínseca dos seres humanos. Desde a antiguidade até os dias atuais, as relações de amizade (e.g., através de afinidades pessoais e/ou interesses em comum), parentesco, profissional, crença, dentre outras, têm determinado as estruturas sociais e o papel das pessoas nessas estruturas. A essa rede de relações sociais, que constituem uma estrutura social composta por indivíduos e relacionamentos entre esses indivíduos, denomina-se Redes Sociais [15].

A popularização da Internet na década de 1990, aliada à já estabelecida utilização dos computadores pessoais, permitiu a criação de novos tipos de Redes Sociais, denominadas Comunidades Virtuais [57]. Nessas comunidades, os indivíduos podem se comunicar utilizando-se de ferramentas, tais como e-mails, fóruns eletrônicos, programas de mensagem instantânea e sessões de videoconferência. Com isso, a distância física entre os indivíduos deixou de ser uma limitação para a constituição de novas amizades.

Apesar das Comunidades Virtuais aproximarem pessoas geograficamente distantes, elas não eram centradas nos indivíduos, mas sim na interação destes a partir da comunidade. Com isso, elas foram vistas pelos usuários apenas com um meio pelo qual eles podiam interagir virtualmente com outras pessoas.

No final da década de 1990 e início dos anos 2000 começaram a surgir as primeiras redes

sociais online, centradas nos indivíduos e na expansão de suas relações sociais. Ao contrário das Comunidades Virtuais, as redes sociais online logo fizeram sucesso com os usuários. Uma vez que elas proporcionavam aos usuários, utilizando a própria plataforma da rede social online, não só estabelecerem novos vínculos de amizade entre si, mas principalmente reforçar as relações sociais com amigos de longa data, conhecidos, familiares, colegas de trabalho, etc.

De acordo com a definição proposta em [11], redes sociais online são serviços baseados na web que permitem aos indivíduos: construírem um perfil público ou semi-público dentro do sistema, gerenciar uma lista de contatos[1] com os quais eles podem interagir e visualizar e navegar em sua lista de contatos e na lista de contatos dos outros indivíduos pertencentes ao sistema. Alguns exemplos de serviços sociais da web, são: LinkedIn [35], MySpace [3], Orkut [13], Facebook [66], YouTube [14] e Twitter [20].

Nos serviços de redes sociais online existentes, os usuários mantém um perfil público contendo informações pessoais, tais como: nome, idade, interesses, preferências e atividades que gostam de realizar no dia-a-dia. Ademais, esses perfis disponibilizam uma lista de amigos ou de contatos, contendo outros usuários que fazem parte do sistema, com os quais o usuário está relacionado diretamente. Ao expor a estrutura das suas conexões sociais de forma explícita para outros membros da plataforma social, esses podem constituir novos vínculos de amizade com amigos de amigos, ou ainda com estranhos que compartilham interesses em comum.

Em relação à forma de representação das conexões sociais dos usuários, na literatura elas são frequentemente representadas com base em duas abordagens distintas: representação baseada na Teoria dos Grafos [44] e representação baseada em ontologias [33].

Na primeira abordagem, as conexões sociais dos usuários são modeladas através de grafos, onde os usuários são representados através de um conjunto de vértices e as relações sociais existentes entre eles são representadas por um conjunto de arestas. Nessa abordagem, os relacionamentos representados pelas arestas são explorados através de algoritmos de busca em grafos (e.g., *Breadth-First Search* - BFS e *Depth-First Search* - DFS) [16].

Já na segunda abordagem, as conexões sociais dos usuários são representadas através de ontologias. A ontologia mais comumente utilizada é a *Friend of a Friend* (FOAF) [12].

[1] Os contatos do usuário são outros membros da rede social online da qual o usuário faz parte.

Nesse tipo de representação, os usuários são representados através de classes/conceitos da ontologia (e.g., classe *Person*) e os relacionamentos sociais existentes entre eles são representados através de propriedades (e.g., a propriedade *Knows*). A fim de explorar os relacionamentos sociais dos usuários representados através de ontologias, faz-se necessária a utilização de *frameworks* para sua manipulação (e.g., Jena[2]) e de motores de inferência (e.g., Jess[3]) para explorar as propriedades existentes, ou mesmo, inferir novos conhecimentos.

Por fim, de acordo com o estudo realizado em [47], a estrutura das redes sociais online apresenta uma distribuição *power-law* e topologia *Scale-Free*. Nessa topologia, as arestas são adicionadas aos vértices existentes com base no modelo de ligação preferencial definido em [4]. Nesse modelo, os vértices contendo maior grau (i.e. maior quantidade de arestas) possuem uma probabilidade maior de receber as novas conexões [38]. Isto implica que, utilizando um grafo para representar as conexões sociais dos usuários de uma rede social online qualquer, haverá poucos usuários contendo muitas conexões com outros usuários (e.g., celebridades ou pessoas famosas) e milhares de usuários apresentando poucas conexões (e.g., pessoas comuns).

2.3 Redes Sociais Móveis

Redes Sociais Móveis, Comunidades Virtuais Móveis, ou ainda, Redes Sociais Pervasivas [4] [7, 48] são o resultado da integração entre o paradigma de Computação Pervasiva com os serviços sociais da web 2.0, tais como [3, 13, 14, 20, 35, 66]. Elas possibilitam que os usuários localizados simultaneamente em um mesmo local realizem e compartilhem atividades de interesse comum, em qualquer lugar, a qualquer momento [15, 36].

Para que esse paradigma funcione plenamente é primordial que as aplicações sejam cientes de contexto [19], ou seja, possam recuperar as informações do ambiente em que se encontram, com a finalidade de prover serviços e executar tarefas em favor do usuário [42, 58, 60].

Aplicações pervasivas para Redes Sociais Móveis possuem ciência de pelo menos duas informações de contexto relacionados aos usuários: (i) o contexto social do usuário (*social-awareness*); e (ii) o contexto de localização (*location-awareness*). O primeiro contém a

[2]Disponível em: `http://jena.sourceforge.net/`

[3]Disponível em: `http://www.jessrules.com/`

[4]Neste trabalho utiliza-se a nomenclatura de Redes Sociais Móveis

informação sobre quem são os amigos do usuário, quais são as suas preferências, interesses e atividades. Já o último contém a informação sobre onde esses amigos estão localizados, a partir da aplicação de uma ou da combinação de várias técnicas de localização existentes [39, 56].

2.3.1 Abordagens para Construção de Redes Sociais Móveis

Com base na literatura existente foram identificadas duas abordagens distintas para construção de Redes Sociais Móveis, sendo elas: abordagem oportunista baseada no estabelecimento de conexões *ad hoc* entre os dispositivos móveis dos usuários e abordagem baseada nas conexões sociais existentes nos serviços sociais da web 2.0.

Abordagem *Ad Hoc*

A formação de Redes Sociais Móveis utilizando uma abordagem *ad hoc* ocorre quando dois ou mais usuários, localizados simultaneamente em um mesmo local, aproximam-se fisicamente uns dos outros e seus dispositivos móveis, utilizando alguma conexão sem fio embarcada, trocam as informações dos perfis dos usuários entre si, com o objetivo de verificar a existência de similaridade entre as preferências dos usuários, em um processo chamado casamento de perfis (*profile matching*). Caso exista similaridade entre as informações trocadas, os dispositivos móveis estabelecerão uma conexão *ad hoc* entre si.

Na Figura 2.2 é ilustrado o processo descrito anteriormente. Esta abordagem não requer nenhuma infraestrutura previamente existente para acesso à Internet.

Como a formação das Redes Sociais Móveis nessa abordagem é realizada de forma oportunista, em um esquema descentralizado, as redes criadas não armazenam no decorrer do tempo a informação sobre quais foram as comunidades formadas, nem informações sobre os membros destas comunidades. Caso essas informações sejam armazenadas, esse processo ocorrerá nos dispositivos móveis dos usuários, os quais ainda apresentam restrições de processamento e armazenamento.

Além disso, essa abordagem limita a adição de novos membros à Rede Social Móvel, pois não permite a um usuário tornar-se amigo de alguém que não esteja localizado simultaneamente no mesmo espaço físico. Outros problemas decorrentes do uso desta abordagem

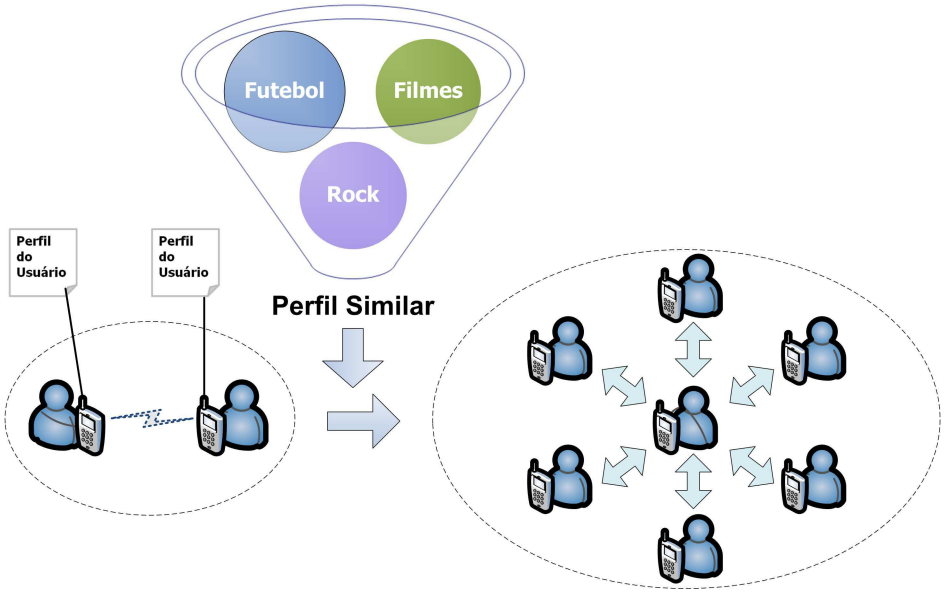


Figura 2.2: Formação de Redes Sociais Móveis utilizando uma abordagem oportunista

são: (i) dificuldade de recuperação das informações, em virtude da distribuição e descentralização das informações nas redes *ad hoc*; e (ii) os dispositivos, que outrora constituíram uma Rede Social Móvel, podem nunca mais estarem localizados simultaneamente no mesmo espaço físico, o que faz com que as informações armazenadas neles se tornem indisponíveis. Portanto, essa abordagem dificulta que a informação contextual possa ser reutilizada por outras aplicações presentes em um ambiente pervasivo.

Abordagem Baseada nas Redes Sociais Online

Nessa abordagem, um dispositivo móvel ou servidor central, de posse de uma conexão com a Internet, conecta-se a uma rede social online e reutiliza as informações sobre as conexões sociais existentes nos perfis dos usuários, para formação das Redes Sociais Móveis. A grande vantagem dessa abordagem é que todas as conexões sociais dos usuários já estão estabelecidas e são conhecidas *a priori*. Ou seja, independentemente da abordagem utilizada para representação das conexões sociais dos usuários, se cada conexão social for explorada sucessivamente, será possível determinar quais são os amigos dos usuários em vários níveis (e.g., amigos, amigos dos amigos, amigos dos amigos dos amigos e assim por diante). Ainda, essa abordagem pode ser utilizada para definir e/ou enriquecer a informação de contexto social dos usuários. Inclusive, pode ser utilizada como informação de entrada para mecanismos de

recomendação de usuários. Na Figura 2.3 é ilustrada a representação das conexões sociais dos usuários em vários níveis.

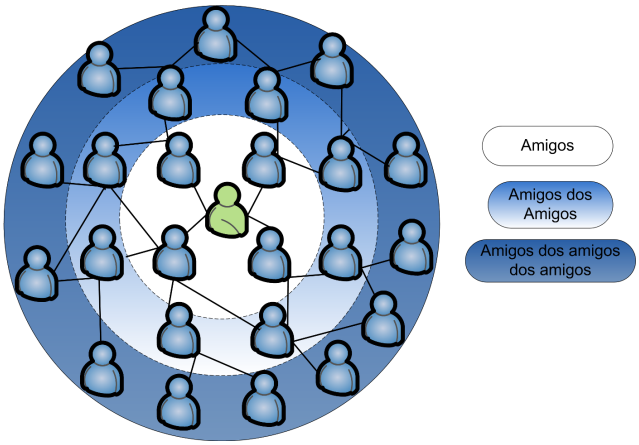


Figura 2.3: Formação de Redes Sociais Móveis utilizando as conexões sociais de uma Rede Social Online

No entanto, esta abordagem apresenta algumas limitações, tais como: (i) dificuldade/impossibilidade de recuperar as informações sobre as conexões dos usuários das redes sociais online; (ii) a impossibilidade de adicionar membros à Rede Social Móvel que não façam parte das redes sociais online; e (iii) dificuldade/impossibilidade de determinar a localização dos membros da comunidade em tempo real.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados os principais trabalhos relacionados à infraestrutura apresentada neste trabalho para desenvolvimento de aplicações para Redes Sociais Móveis. Com base na revisão de literatura realizada, os trabalhos relacionados foram separados em aplicações sociais e serviços de *middleware*, os quais são descritos e discutidos a seguir.

3.1 Aplicações Sociais Móveis

3.1.1 Serendipity

Serendipity [21] foi a primeira aplicação social móvel a utilizar informações semelhantes àquelas dos perfis dos usuários (e.g., descrição sobre as preferências dos usuários, lista de amigos, dentre outras) existentes nas redes sociais online e a armazená-las em um servidor central. Essa aplicação utiliza a conexão *Bluetooth* para descobrir outros dispositivos móveis fisicamente próximos, que estejam executando a mesma aplicação.

Além de armazenar algumas informações dos perfis dos usuários, o servidor central também possui uma entidade responsável por realizar o casamento das preferências (*profile matching*) presentes nos perfis dos usuários. O casamento de perfis é realizado com base em um sistema de pesos e de um limiar definidos previamente pelos usuários, que indica o grau de similaridade entre os perfis comparados. Uma vez identificado a ocorrência de casamento entre as preferências dos usuários, Serendipity envia mensagens para os dois dispositivos móveis, contendo: a foto do outro usuário, lista com interesses em comum e tópicos para

nortear a conversa entre eles.

Essa aplicação social está relacionada com a infraestrutura desenvolvida, pois utiliza: (i) informações de perfil semelhantes àquelas existentes em uma rede social online, embora, não possua mecanismo para extrair as informações de perfil das redes sociais online; (ii) servidor central para armazenar as informações do perfil dos usuários; e (iii) mecanismo de casamento de perfis para determinar a similaridade entre os perfis dos usuários.

3.1.2 WhozThat

WhozThat [5] é um sistema que utiliza as informações das redes sociais online para enriquecer o contexto social e auxiliar os usuários para que eles descubram informações *a priori* sobre as pessoas que estão próximas e que possuam interesses em comum. Para atingir esse objetivo, os autores apresentam um protocolo composto das seguintes fases: (1) troca de identificadores das redes sociais online que os usuários fazem parte; e (2) recuperação do perfil do usuário.

O processo se inicia quando um usuário de posse de um dispositivo móvel, contendo a aplicação WhozThat instalada, entra em um ambiente físico contendo outros usuários possuindo a mesma aplicação instalada em seus dispositivos. Os dispositivos móveis, através de buscas na rede utilizando a conexão *Bluetooth*, identificam outros dispositivos contendo a aplicação e iniciam a troca de identificadores das redes sociais online. Uma vez trocados os identificadores, os dispositivos iniciam uma conexão com a Internet (e.g., 3G, Wi-Fi, etc) para recuperar as informações de perfil dos usuários das respectivas redes sociais online, a partir dos identificadores. Por fim, os usuários visualizam na tela dos dispositivos os perfis recuperados e decidem se desejam interagir uns com os outros no ambiente em que se encontram.

Os autores descrevem dois cenários onde o protocolo proposto é utilizado e melhorado. No primeiro cenário, o protocolo é evoluído através do acréscimo de um *gateway* (i.e. um computador com maior poder computacional) na rede, que realiza computações complexas em favor dos dispositivos móveis. Já no segundo cenário, os autores acrescentam a possibilidade dos dispositivos móveis, que não possuem conexão com a Internet, encaminharem os identificadores dos usuários para outros dispositivos na rede, para que eles acessem as informações de perfil dos usuários das redes sociais online e retornem essas informações,

tão logo estejam disponíveis.

Apesar desse sistema compartilhar a ideia de utilizar as informações das redes sociais online para enriquecer o contexto social dos usuários, ele possui algumas limitações e problemas. Primeiro, não lida com a possibilidade dos usuários possuírem vários perfis em redes sociais diferentes, que podem conter informações incompletas e contraditórias entre si. Segundo, sempre necessita de conexão com a Internet para recuperar os dados sobre os perfis dos usuários e confia na disponibilidade dos servidores web onde estão localizados. Por fim, ao utilizar-se de outros dispositivos para recuperar o perfil do usuário, recai sobre problemas de privacidade das informações trocadas e interceptação dos dados, seguida de utilização indevida por parte de entidades maliciosas.

Em relação às funcionalidades providas pela infraestrutura desenvolvida, WhozThat não disponibiliza um mecanismo para acessar às informações de contexto social armazenadas pelo servidor central.

3.1.3 CenceMe

CenceMe [46] é uma aplicação para *iPhone* e *iPod Touch*, resultante do projeto MetroSense da Universidade de DartMouth, EUA, cujo objetivo é transformar os recursos disponíveis nos *smartphones* (e.g., acelerômetro, microfone, câmera, GPS e *Bluetooth*) em redes de sensores móveis. As informações capturadas pelos sensores são combinadas para tentar determinar quais atividades os usuários estão realizando (e.g., correndo, participando de uma reunião ou de uma festa), assim como, sua localização. Uma vez adquiridas, essas informações são compartilhadas nas redes sociais online dos usuários (e.g., [3, 66]) e em serviços de *micro-blog* (e.g., [20]) para que os amigos dos usuários possam estar cientes do que eles estão fazendo.

A arquitetura do CenceMe está dividida em dois módulos, executados nos dispositivos móveis dos usuários e em um servidor externo, respectivamente. Sendo o último utilizado para realizar processamentos computacionalmente complexos e, consequentemente, economizar a bateria do dispositivo móvel.

O módulo que funciona dentro do dispositivo móvel do usuário, possui a responsabilidade de: (i) recuperar as informações de baixo nível de cada um dos sensores dos dispositivos móveis; (ii) armazenar essas informações; e (iii) transformá-las, através de classificadores

(e.g., áudio e acelerômetro), em dados de mais alto nível, denominados de primitivas. As primitivas são enviadas periodicamente ao servidor, através de um módulo específico para posterior processamento.

Já o módulo executado no servidor possui os seguintes objetivos: (i) recuperar, armazenar e processar as primitivas, transformando-as em fatos; (ii) disponibilizar essas informações para outras aplicações; e (iii) publicá-las nas redes sociais online.

Apesar de o CenceMe tentar inferir informações sobre o contexto do usuário utilizando os recursos disponíveis nos dispositivos móveis, as informações produzidas pelos classificadores demonstraram não serem confiáveis durante avaliação realizada pelos autores do projeto, apresentando baixa precisão e alta taxa de falsos positivos, em alguns casos. Ainda, o uso intenso dos sensores diminuiu a carga da bateria dos dispositivos para menos de 1 hora.

Diferentemente dos objetivos apresentados pela infraestrutura desenvolvida e de outras soluções anteriormente mencionadas, CenceMe não utiliza as informações de contexto social e de localização dos usuários que utilizam a aplicação (obtidas através do GPS e do *Bluetooth*), para aumentar as conexões sociais existentes, por exemplo, recomendando usuários com interesses em comum.

3.2 Soluções de *Middlware*

3.2.1 Infraestrutura de Software Baseada em Componentes para a Construção de aplicações para Comunidades Virtuais Móveis

Essa infraestrutura é o resultado de um trabalho de dissertação [25] desenvolvido no Laboratório de Sistemas Embarcados e Computação Pervasiva, da Universidade Federal de Campina Grande. A infraestrutura de software foi desenvolvida com base no modelo de componentes COMPOR [1] e oferece suporte à construção de aplicações para comunidades móveis, disponibilizando as seguintes funcionalidades: (i) formação de comunidades móveis entre os usuários, (ii) notificação de proximidade física entre usuários, (iii) notificação de proximidade dos dispositivos móveis, (iv) suporte a definição de preferências e interesses dos usuários, (v) identificação de usuários com interesses em comum, (vi) compartilhamento de dados entre os membros da comunidade e (vii) autorização de acesso aos dados fornecidos

pela infraestrutura.

O autor modela sua infraestrutura com base em cinco componentes principais, que contemplam as funcionalidades enumeradas anteriormente, são eles: *UserSearcher* e *UserManager*, que abrangem as funcionalidades (i), (ii) e (iv) e gerenciam o acesso a essas informações, respectivamente. O componente *DeviceSearcher*, responsável por realizar a busca por dispositivos próximos ao usuário, abrangendo a funcionalidade (iii). O componente *Similarity*, que determina a similaridade entre as preferências dos usuários com base em um valor de limiar fornecido, abrangendo a funcionalidade (v) e *ContentSharing*, responsável pelo compartilhamento e acesso aos dados, abrangendo as funcionalidades (vi) e (vii).

As principais diferenças entre a infraestrutura baseada em componentes e a infraestrutura desenvolvida neste trabalho, são: (i) na primeira abordagem, as informações de contexto são definidas pelos usuários a partir das aplicações, enquanto que na segunda abordagem pretende-se utilizar as informações dos perfis dos usuários existentes nas redes sociais online para determinar o contexto social dos usuários, o que torna a última mais escalável em relação à quantidade de usuários; (ii) a primeira abordagem utiliza as conexões *ad hoc* entre os dispositivos móveis para formar comunidades móveis, enquanto que a infraestrutura desenvolvida utiliza as conexões sociais dos usuários presentes nos perfis das redes sociais online para construir as comunidades móveis; e (iii) a infraestrutura desenvolvida, de posse das informações sobre as conexões sociais dos usuários, determina os amigos dos usuários em vários níveis.

3.2.2 Google Latitude

O Google Latitude [29] é um serviço de *middleware* fornecido pelo Google, que provê funcionalidades para mostrar a localização dos usuários, através de seu dispositivo móvel ou pela web. Seu objetivo é indicar a localização do usuário para seus contatos, no momento em que este escreve uma mensagem (Gmail), inicia uma sessão de bate papo (Google Talk), ou simplesmente, define a melhor rota para uma determinada localidade (Google Maps). O serviço também integra-se com a rede social Orkut, onde é possível que um usuário realize o acompanhamento da localização de seus amigos em tempo real.

Esse serviço apresenta as seguintes funcionalidades: histórico de localização, alertas de proximidade e publicação de informações de localização em redes sociais.

A partir da funcionalidade de alerta de proximidade, o usuário pode ser notificado sempre que um de seus amigos estiver próximo. Essa funcionalidade, em conjunto com a de histórico de localizações, permite que o usuário receba notificações de proximidade apenas quando estiver em lugares não usuais, evitando, por exemplo, notificações enquanto estiver em casa ou no ambiente de trabalho. Ainda, o histórico de localizações permite que o usuário revisite os trajetos percorridos em um determinado período de tempo. A integração com o serviço Google Talk permite que contatos no serviço de mensagens instantâneas visualizem a localização atual de um usuário em diferentes níveis (e.g., país, estado, cidade ou bairro).

Ademais, o serviço de *middleware* para dispositivos móveis pode determinar a localização dos usuários a partir de três fontes de informação, sendo elas: o GPS do dispositivo, triangulação das antenas da rede móvel celular e triangulação das antenas de pontos de acesso *Wi-Fi*. Dessa forma, a solução garante cobertura mesmo em locais onde o sinal de satélite não está disponível, como em locais fechados. Logo, a precisão das informações depende da fonte de dados disponível em um determinado momento.

Por ser uma solução proprietária, não são fornecidos detalhes sobre a arquitetura de implantação da solução, entretanto, especula-se que toda a troca de informações entre os usuário ocorra através de um serviço de computação em nuvem, onde são armazenados o perfil dos usuários e suas informações de localização.

Apesar do Google Latitude realizar a integração com algumas Redes Sociais Online, ele não disponibiliza mecanismos de acesso às informações de contexto social e de localização armazenadas pelo serviço de *middleware*, assim como, não disponibiliza suporte ao desenvolvimento de aplicações. Além disso, esse serviço não fornece mecanismo para expandir as informações de contexto social armazenadas, determinando as informações de amizade em vários níveis para todos os usuários. Por sua vez, a infraestrutura desenvolvida não apresenta tais limitações.

3.2.3 iGroups

iGroups [43] é uma patente da Apple, que especifica a formação de grupos sociais móveis a partir da troca de tokens entre os dispositivos móveis (Apple iPhones) localizados em um mesmo local. Ainda, a patente especifica a existência de um serviço centralizado confiável, que gera os *tokens* trocados pelos *iPhones* para formação de grupos e os recebe posterior-

mente, armazenando-os em uma base de dados. Os *tokens* compartilhados pelos dispositivos móveis utilizam algum mecanismo de criptografia e são enviados para o serviço central.

O serviço funciona da seguinte forma: usuários localizados em um mesmo ambiente físico, com as conexões *Bluetooth* ou *Wi-Fi* dos seus dispositivos móveis ligadas, contendo a aplicação *iGroups* instalada e dentro de uma distância suficiente para que ocorra transmissão, recebem uma informação de *token* gerada pelo serviço centralizado. O *token* recebido identificará o futuro grupo, sendo criptografado e trocado pelos dispositivos móveis, que constituem o mesmo grupo.

Ademais, a patente destaca os seguintes pontos: (i) não há um limite para quantidade de membros participantes de um grupo; (ii) em algum momento, as informações sobre o grupo criado são sincronizadas com o serviço centralizado, através da Internet ou rede local; e (iii) caso o dispositivo móvel não contenha conexão sem fio para enviar as informações ao serviço centralizado, estas informações podem ser enviadas através de outros dispositivos pertencentes ao mesmo grupo, dotados com esse recurso.

O servidor centralizado armazena algumas informações sobre os dispositivos móveis que constituem os grupos, tais como: quem são os membros do grupo; quando o grupo foi criado; qual a localização dos membros do grupo; dentre outras. A informação de localização dos membros do grupo que não possuam recursos em seus dispositivos para determinar a própria localização (e.g., GPS), pode ser determinada, por exemplo, pela triangulação de antenas dos dispositivos constituintes do mesmo grupo, que possuem GPS disponível.

Esse serviço mantém uma base histórica, que pode ser utilizada para determinar, o padrão de mobilidade entre os membros do grupo e recuperar informações sobre os grupos formados anteriormente. A patente ainda torna possível a integração do servidor com provedores de conteúdo (e.g., redes sociais online) para recuperar as informações de perfil dos usuários e utilizá-las para enriquecer as informações de contexto do usuário.

Esse trabalho está relacionado com a solução desenvolvida, pois oferece: (i) utilização de serviço central confiável para armazenar as informações sobre as redes sociais móveis constituídas, assim como, a informação de localização dos seus membros; e (ii) integração com provedores de serviço (e.g., redes sociais online) para enriquecer as informações de contexto do usuário.

No entanto, a patente não especifica a utilização da abordagem baseada nas conexões

sociais dos usuários para formação de grupos. Ainda, não há no serviço central um módulo para calcular os níveis de relacionamento entre os usuários da Rede Social Móvel. Ademais, a restrição da patente para dispositivos *Apple iPhone* limita sua abrangência e utilização em outras plataformas móveis. Por fim, não apresenta uma API para acessar as informações armazenadas no serviço centralizado e para desenvolvimento de aplicações. Essas limitações não estão presentes na infraestrutura desenvolvida neste trabalho, que pode ser utilizada para construir aplicações sociais móveis para qualquer plataforma ou dispositivo móvel.

3.2.4 FriendSensing

FriendSensing é um *framework* proposto em [54] que permite aos novos membros das redes sociais online descobrirem seus amigos automaticamente [53]. Esse *framework* permite ainda que os usuários expandam as conexões sociais das redes sociais online que fazem parte, com base na abordagem de encontros *ad hoc* entre os usuários, utilizando seus dispositivos móveis. O *framework* armazena as informações sobre os encontros dos usuários ao longo do tempo, quando estes estão localizados simultaneamente no mesmo local. Essas informações são processadas por algoritmos que exploram informações sobre os encontros dos usuários, tais como a frequência com a qual os usuários se encontram ao longo do tempo; em que lugares o encontro ocorreu; a duração do encontro; dentre outras. Essas informações são utilizadas para construção de listas de recomendação de amigos, com objetivo de expandir as conexões sociais dos usuários nas redes sociais online.

Em seu trabalho, os autores realizaram estudos para demonstrar que a abordagem proposta pode ajudar os usuários a melhorarem e/ou reforçarem as suas relações sociais existentes no mundo real. Dentre os resultados obtidos, durante realização de experimentos para escolher a melhor abordagem para construção de listas de recomendação, os autores verificaram que utilizar uma *abordagem de caminho mais curto* é mais relevante para construção de listas de recomendação. Isto é possível, pois essa abordagem tira proveito das relações de amizade já existentes entre os usuários das redes sociais online (e.g., relações de amizade entre amigos, amigos dos amigos e assim por diante), o que aumenta a relevância do processo de recomendação de novos amigos.

FriendSensing assemelha-se ao trabalho proposto, pois utiliza as informações sobre os encontros *ad hoc* entre os usuários, em conjunto com as informações das relações sociais

existentes nas redes sociais online, para expandir as conexões sociais dos usuários. Além disso, os resultados obtidos a partir dos experimentos realizados pelos autores, reforçam a viabilidade de uma abordagem baseada no caminho mais curto para expandir as conexões sociais dos usuários. Uma abordagem para expansão das conexões sociais dos usuários, com base no caminho mais curto, é utilizada por um dos módulos da infraestrutura, o qual será apresentado no *Capítulo 4*.

3.2.5 Serviço de *Middleware* para Redes Sociais Pervasivas

O trabalho desenvolvido em [48] tem como objetivos construir Redes Sociais Móveis e recomendar novos amigos aos membros das Redes Sociais Móveis. Para isso, o *middleware* proposto combina as informações sobre as relações sociais dos usuários com as informações de localização dos dispositivos móveis, quando os usuários estão simultaneamente localizados no mesmo espaço físico.

Nesse trabalho os autores apresentam uma abordagem de *middleware* semi-distribuída, composta de dois componentes lógicos principais, denominados clientes e *brokers*.

Os clientes são nós representando os usuários (a relação é de um nó por dispositivo móvel do usuário), que possuem como objetivo armazenar localmente as informações sobre os encontros com outros clientes e as atividades realizadas entre eles. Além disso, os clientes também armazenam as informações sobre os clientes com quem estão frequentemente em contato. As informações armazenadas nos clientes são repassadas para os *brokers*, no instante em que eles estão fisicamente próximos entre si, dentro da distância de transmissão.

Por sua vez, os *brokers* têm como objetivo coletar as informações que foram enviadas pelos clientes, processá-las por meio de componentes de aprendizagem e propagá-las para outros *brokers*. Quando dois *brokers* se encontram, eles trocam os conhecimentos que possuem sobre os clientes conhecidos e propagam localmente as informações sobre as novas ligações sociais obtidas. Por fim, eles raciocinam sobre o conhecimento obtido, que será utilizado para recomendação de usuários, notificando os clientes durante o próximo encontro físico que tiverem.

O processo de eleição dos *brokers* é baseado no algoritmo de *Page Ranking (PR)*, no qual dispositivos com maior popularidade são escolhidos (i.e. aqueles que possuem maior quantidade de conexões com outros dispositivos móveis).

Os autores desse trabalho discutem uma abordagem interessante para recomendação de usuários, com base nas informações sociais e de localização obtidas dos dispositivos móveis dos usuários. Todavia, não são disponibilizados mecanismos (e.g., APIs) para que essas informações possam ser reutilizadas por outros módulos de um serviço de *middleware*, ou mesmo para construção de outras aplicações. Além disso, essa abordagem não apresenta integração com os serviços sociais da web 2.0, com o propósito de aumentar o conhecimento sobre as informações de contexto social dos usuários.

3.2.6 *Middleware* para Redes Oportunistas

Nesse trabalho os autores propõem uma solução de *middleware*, construída como um módulo integrado à arquitetura de referência do projeto Haggle[1], para aquisição, armazenamento e distribuição das informações de contexto social, para o domínio de Redes Oportunistas.

De acordo com Boldrini, em [8], Redes Oportunistas são caracterizadas pela mobilidade e conectividade intermitente dos nós, que pode levar ao seu particionamento em várias sub-redes. Em virtude dessas características, nessas redes há uma ausência de informações globais sobre os dispositivos presentes na Rede. Esse domínio está relacionado a problemas, tais como aqueles relacionados a protocolos de roteamento, troca de conteúdos, *Quality of Service* (QoS), etc.

Os autores demonstraram a utilidade do *middleware* proposto, a partir da construção de um serviço de compartilhamento de conteúdos, que explora as informações de contexto social dos usuários (obtidas dos encontros *ad hoc* entre dispositivos móveis dos usuários e armazenadas pelo middleware Haggle), para maximizar a distribuição dos dados entre os nós e aumentar a probabilidade de entregar informações mais relevantes a eles [8].

O serviço de compartilhamento de conteúdo explora a mobilidade dos nós na Rede Oportunista, estabelecendo redes *ad hoc* entre os dispositivos dos usuários. Os nós carregam informações sobre as comunidades que os usuários fazem parte, utilizando as informações de contexto social de nós intermediários, denominados *travellers* (i.e. nós que estão se movendo através da rede), para distribuí-las a nós terminais. Dessa forma, estabelecendo a troca de conteúdo entre os nós pertencentes a comunidades distintas.

Os autores demonstraram através de suítes de testes e simulações de mobilidade dos

[1] `http://www.haggleproject.org`

usuários que, ao se utilizar de políticas contendo informações sobre o contexto social dos usuários, o *middleware* é bastante efetivo para solucionar os problemas relacionados à troca de conteúdos entre os nós, presentes no domínio das Redes Oportunistas.

Por fim, esse trabalho não provê uma API para: (i) acesso as informações de contexto social, provenientes das interações das Redes Oportunistas e (ii) desenvolvimento de aplicações. Já a infraestrutura apresentada neste trabalho, não apresenta essas limitações.

3.2.7 SAMOA

Nesse trabalho os autores propõem um serviço de *middleware*, denominado SAMOA (*Socially Aware and Mobile Architecture* [10]), ciente de contexto social e de localização para formação de redes sociais em qualquer lugar, a qualquer momento. SAMOA tem como objetivo construir Redes Sociais Móveis, a partir de conexões *ad hoc* entre os dispositivos móveis dos usuários presentes simultaneamente no mesmo espaço físico. Além disso, disponibiliza essas informações para aplicações sociais construídas por terceiros que utilizem a infraestrutura criada. Para isso, o *middleware* define: (i) três papéis para os usuários (gerente, cliente e membro); (ii) três tipos de perfis (usuários, lugar e descoberta), representados através de ontologias de domínio, descrevendo atividades e preferências; e (iii) dois algoritmos de casamento semântico de perfis. Ainda, (iv) disponibiliza uma visão global sobre todas as comunidades formadas pelo gerente durante o tempo e (v) informações sobre os membros da comunidade móvel consultada.

Essa solução é dita centrada no usuário. Isto é, existe um usuário, denominado gerente, que deseja formar uma comunidade móvel pessoal com outros usuários, denominados clientes, presentes em um mesmo espaço físico, que compartilham interesses em comum determinados pelo gerente da comunidade. Ainda, SAMOA permite que usuários sejam gerentes e membros simultaneamente de Redes Sociais Móveis distintas.

Para construir as Redes Sociais Móveis, SAMOA utiliza o espaço físico (denominado lugar) em que o dispositivo móvel se encontra (e.g., videolocadora) para delimitar o espaço de descoberta dos membros da Rede Social Móvel. Para cada lugar, o gerente especifica um perfil contendo atividades que caracterizam a finalidade do lugar (e.g., usuários interessados em filmes de terror, suspense, etc). Essas informações são utilizadas para filtrar os clientes, a partir do casamento semântico das atividades descritas em seus perfis com as atividades do

lugar (primeiro algoritmo de casamento semântico). Dessa forma, apenas alguns clientes poderão se tornar membros da Rede Social Móvel. O segundo algoritmo realiza o casamento semântico das preferências descritas nos perfis de descoberta (e.g., interesse em filmes de terror) com as descritas nos perfis dos clientes resultantes do primeiro algoritmo de casamento. Nesse momento, os clientes cujas preferências sejam semanticamente compatíveis tornam-se membros da Rede Social Móvel configurada pelo gerente.

SAMOA difere da infraestrutura apresentada neste trabalho, pois não utiliza as informações dos perfis dos usuários presentes nas redes sociais online para construir o contexto social dos usuários. Assim, os proprietários dos dispositivos móveis devem definir e disponibilizar as informações dos seus perfis (e.g., atividades, preferências, entre outras), a partir dos próprios dispositivos móveis.

Por se tratar de uma solução descentralizada, SAMOA não possui conhecimento global sobre as conexões sociais de todos os usuários. Ainda, essa solução não possui a funcionalidade de expansão das conexões sociais dos usuários, o que a torna dependente da interação entre os dispositivos móveis em redes *ad hoc* para evoluir as informações de contexto social. A infraestrutura apresentada neste trabalho não possui essas limitações.

3.2.8 MobiSoC

Os autores desse trabalho propõem uma solução de *middleware* para computação social móvel, denominada MobiSoC [9, 30], que permite o desenvolvimento de aplicações sociais denominadas MSCAs (*Mobile Social Computing Applications*) através de API. Além disso, essa solução provê uma plataforma que captura, gerencia e compartilha as informações sobre o estado social das comunidades físicas com as aplicações. As informações que constituem o estado social são: (i) perfis de pessoas e lugares; e (ii) afinidades pessoa-pessoa e pessoa-lugares. Essas informações evoluem com o passar do tempo à medida que novos perfis de usuários, conexões sociais, informações sobre lugares e eventos são adicionados à plataforma. Ainda, MobiSoC utiliza algoritmos de aprendizagem para descobrir padrões geo-sociais [31] previamente desconhecidos, tais como afinidades entre pessoas e entre pessoas e lugares.

MobiSoC foi implementado com base em uma arquitetura centralizada e orientada a serviços. Essa arquitetura é dividida em módulos, que por sua vez são divididos em sub-

módulos, e as informações presentes nesses últimos são armazenadas em um banco de dados. Os principais módulos do sistema são: módulo de coleção de dados e aprendizagem do contexto social.

O módulo de coleção de dados possui os seguintes sub-módulos:

- Pessoas: permite às aplicações coletarem, armazenarem e modificarem as informações dos perfis dos usuários (e.g., interesses, preferências, entre outras). Ainda, permite adicionar novos grupos, assim como, contatos sociais;

- Lugares: armazena informações de dados geográficos sobre construções e mapas. Ainda, provê mecanismos para adicionar informações sobre eventos relacionados com o lugar;

- Localização: recebe e armazena as atualizações das informações de localização provenientes dos dispositivos móveis dos usuários.

O módulo de aprendizagem de contexto social é responsável por inferir novas informações sobre o relacionamento entre pessoas e entre pessoas e lugares. Este módulo é dividido nos seguintes sub-módulos:

- Perfil do usuário: provê informações sobre os perfis dos usuários, grafo das conexões sociais do usuário e grupos sociais que este faz parte;

- Perfil do lugar: compartilha informações armazenadas sobre o lugar;

- Aprendizagem de afinidade Pessoa-Pessoa: responsável por computar afinidades sociais entre pares de usuários, com base nos interesses pessoais, amigos em comum, ou ainda, lugares em comum que costumam frequentar;

- Aprendizagem de afinidade Pessoa-Lugar: tenta descobrir quais são os lugares de interesse dos usuários, através da análise histórica dos dados de localização dos usuários.

O *middleware* ainda disponibiliza outros dois módulos: (i) um gerenciador de eventos, que registra as configurações de eventos criados pelas aplicações em funcionamento nos dispositivos móveis dos usuários e realiza a notificação desses eventos quando ocorre mudança

no contexto social, utilizando um modelo do tipo *pull*; e (ii) um gerenciador de privacidade que armazena as regras de privacidade criadas pelas aplicações e assegura que as mesmas serão executadas.

MobiSoC difere da infraestrutura apresentada em dois pontos importantes, os quais são: (i) não se integra às redes sociais online para explorar as informações de contexto social dos usuários; e (ii) na computação do contexto social dos usuários. A infraestrutura apresentada neste trabalho disponibiliza um módulo que utiliza o grafo das conexões sociais dos usuários, construído com base nos relacionamentos previamente existentes nas redes sociais online que eles fazem parte, para determinar os relacionamentos de amizade em vários níveis entre todos os usuários existentes em uma Rede Social Móvel.

3.2.9 MobiClique

MobiClique [51] é um serviço de *middleware* para Redes Sociais Móveis que explora as conexões *ad hoc* entre os dispositivos móveis dos usuários, com o objetivo de estender os relacionamentos de amizades das redes sociais online que eles fazem parte. Essa solução oferece uma API para que as aplicações possam registrar-se, recuperar eventos e dados dos perfis dos usuários.

Para atingir os objetivos enumerados anteriormente, o *middleware* recupera as informações de perfil (e.g., lista de amigos) dos usuários das redes sociais online[2] e utiliza essas informações como base para formação do contexto social do usuário.

MobiClique não utiliza qualquer infraestrutura pré-estabelecida, confiando apenas nas conexões *ad hoc* entre os dispositivos móveis para construir a Rede Social Móvel. Portanto, não se trata de uma solução centralizada que mantém uma conexão com a Internet para recuperar as informações das redes sociais online. A recuperação das informações dos perfis é realizada através das conexões *Wi-Fi* dos próprios *smartphones* com as redes sociais online, ou ainda, utilizando algum dispositivo ou computador que possua conexão com a Internet.

Uma vez de posse das informações de perfil social dos usuários, o *middleware* encoraja a formação de novos relacionamentos entre os usuários, à medida que estes se encontram fisicamente no mesmo local e seus dispositivos móveis estabelecem conexões *ad hoc* entre si, utilizando *Bluetooth*. Ao ser estabelecida uma conexão entre os dispositivos, as informações

[2]Informações provenientes do Facebook [66] foram utilizadas para demonstrar a viabilidade da solução.

compartilhadas nos perfis dos usuários tais como a existência de amizade entre os usuários ou interesses em comum são comparadas e, caso ocorra um casamento entre essas informações, o *middleware* dispara eventos e avisa os usuários através de mensagens. Nesse momento, os usuários podem trocar mensagens de texto entre si, assim como adicionar uns aos outros em suas listas de contatos nas redes sociais online.

Uma das limitações encontradas nessa solução, refere-se a falta de escalabilidade na quantidade de conexões realizadas através do *Bluetooth* entre os dispositivos móveis. Os testes realizados pelos autores apresentaram um limite de apenas três conexões simultâneas por dispositivo. Dessa forma, perde-se a oportunidade de criar ou manter mais laços de amizade simultaneamente. Além disso, a solução apresenta problemas em relação à privacidade das informações dos usuários. Uma vez que, caso o dispositivo móvel não possua conexão com a Internet, este dependerá de outro dispositivo ou computador pessoal que possua tal recurso para obter ou atualizar as informações do perfil do usuário.

Esse serviço de *middleware* difere da infraestrutura apresentada, pois não fornece um mecanismo para expandir as conexões sociais dos usuários. Além disso, em virtude da distribuição das informações entre os dispositivos presentes na rede móvel, o *middleware* não possui o conhecimento global sobre a localização dos usuários em tempo real. Dessa forma, não é possível determinar a localização dos demais usuários que não estejam no raio de cobertura do *Bluetooth* ou *Wi-Fi*.

3.3 Considerações sobre os Trabalhos Relacionados

Apesar de existir uma grande variedade de soluções que auxiliam no desenvolvimento de aplicações para Redes Sociais Móveis, elas ainda não fornecem suporte a requisitos importantes desse domínio. Especificamente, não há uma abordagem isolada que forneça as seguintes funcionalidades: (i) suporte à execução de serviços e acesso às informações de contexto social e de localização dos usuários; (ii) integração com múltiplas redes sociais online para formação das Redes Sociais Móveis; (iii) conhecimento prévio sobre quem são os amigos diretos e indiretos dos usuários (em vários níveis de amizade), e; (iv) conhecimento em tempo real da localização dos usuários.

Na Tabela 3.1 são sumarizadas as funcionalidades apresentadas por cada uma das solu-

ções de *middleware* analisadas anteriormente.

Tabela 3.1: Análise dos Trabalhos Relacionados

Trabalho	**Abordagem**	**API**	**Expansão Conexões Sociais**	**Integração Múltiplas Redes Sociais**
Infraestrutura de Software Baseada em Componentes para a Construção de aplicações para Comunidades Virtuais Móveis [25]	*Ad Hoc*	Não	Não	Não
Google Latitude [29]	Redes Sociais Online	Não	Não	Não
iGroups [43]	*Ad Hoc*	Não	Não	Não
FriendSensing [54]	*Ad Hoc* e Redes Sociais Online	Não	Sim	Não
Serviço de *Middleware* para Redes Sociais Pervasivas [48]	*Ad Hoc*	Não	Não	Não
Middleware para Redes Oportunistas [8]	*Ad Hoc*	Não	Não	Não
SAMOA [10]	*Ad Hoc*	Não	Não	Não
MobiSoc [30]	*Ad Hoc*	Sim	Sim	Não
Mobiclique [51]	*Ad Hoc* e Redes Sociais Online	Sim	Não	Não

Capítulo 4

Infraestrutura para o Desenvolvimento de Aplicações Pervasivas Cientes de Redes Sociais

Neste capítulo apresenta-se o projeto de uma infraestrutura que permite o desenvolvimento de aplicações pervasivas cientes de Redes Sociais. Inicialmente, são descritos os requisitos funcionais da infraestrutura para Redes Sociais Móveis, que compartilha as informações de contexto social e de localização dos usuários. Em seguida é apresentada sua arquitetura, destacando seus principais componentes, com as respectivas responsabilidades, e como eles interagem entre si. Ao final do capítulo, apresenta-se a modelagem da arquitetura da infraestrutura e descreve-se como cada módulo foi implementado.

4.1 Requisitos da Infraestrutura

A fim de possibilitar o desenvolvimento de aplicações pervasivas cientes de Redes Sociais, uma infraestrutura para Redes Sociais Móveis deve permitir a aquisição e a persistência das informações de contexto social e de localização dos usuários. A combinação dessas duas informações de contexto constitui o alicerce para a formação de Redes Sociais Móveis e consequentemente a construção de aplicações para esse domínio.

Ainda, conforme descrito no *Capítulo 2*, as duas principais abordagens para a formação de Redes Sociais Móveis são aquelas provenientes da utilização das interações *ad hoc* entre

os dispositivos móveis dos usuários e aquelas provenientes da utilização das conexões sociais existentes nas redes sociais online.

O suporte à primeira abordagem não faz parte do escopo deste trabalho. Para esse propósito, algumas soluções (discutidas no *Capítulo 3*) como aquelas propostas em [25], que foi desenvolvida previamente dentro do contexto do Projeto PerComp, no qual este trabalho está inserido, ou ainda em [8] podem ser utilizadas para prover a integração com essa abordagem. Ainda, o acoplamento dessas soluções ao projeto da infraestrutura pode ser habilitado em trabalhos futuros. Portanto, a infraestrutura proposta tem como foco a integração com as redes sociais online.

Também é importante destacar a necessidade de uma funcionalidade que realize a expansão das conexões sociais dos usuários. Tal funcionalidade possibilita a evolução da informação de contexto social dos usuários, podendo ser utilizada, por exemplo, no processo de recomendação de novos vínculos de amizade entre os usuários constituintes de uma mesma Rede Social Móvel.

Além disso, para que os desenvolvedores de aplicações pervasivas possam criar suas aplicações a partir de uma infraestrutura para Redes Sociais Móveis, eles devem poder acessar e utilizar as informações de contexto social e de localização dos usuários. A infraestrutura deve disponibilizar serviços que maximizem a produtividade dos desenvolvedores e consequentemente abstraiam o processamento interno realizado. De uma maneira geral, deve-se fornecer funcionalidades que respondam a questionamentos, tais como:

- Quem são os membros da Rede Social Móvel?
- Quais são os interesses e preferências pessoais desses usuários?
- Quem são os amigos diretos e indiretos dos membros da Rede Social Móvel?
- Quais são as Redes Sociais Móveis das quais estes usuários fazem parte?
- Onde os membros da Rede Social Móvel estão localizados?
- Quais membros da Rede Social estão próximos uns dos outros em um determinado instante de tempo?

Na Tabela 4.1 são sumarizados os requisitos funcionais descritos anteriormente. A implementação desses requisitos constitui o projeto da infraestrutura apresentada neste trabalho.

Tabela 4.1: Requisitos Funcionais da Infraestrutura.

Código	Requisito Funcional
RF1	Adquirir e armazenar as informações de contexto social e de localização dos usuários
RF2	Utilizar as conexões sociais das redes sociais para construção de Redes Sociais Móveis
RF3	Expandir as conexões sociais dos usuários
RF4	Disponibilizar acesso às informações de contexto
RF5	Disponibilizar suporte para o desenvolvimento de aplicações

4.2 Arquitetura

A infraestrutura proposta baseia-se na premissa de que as tecnologias de acesso sem fio são pervasivas, ou seja, estão presentes em qualquer lugar, a qualquer momento e estão embarcadas nos dispositivos móveis disponíveis no mercado consumidor. Dessa forma, os usuários possuem acesso constante à Internet através de seus dispositivos móveis (e.g., através das conexões *Wi-Fi* e 3G disponíveis no dispositivo móvel).

Dessa forma, a infraestrutura para o desenvolvimento de aplicações pervasivas cientes de Redes Sociais está implantada em um Servidor de Aplicação Web (e.g., Apache Tomcat[1]), contendo conexão permanente com a Internet. Ademais, a infraestrutura é composta por componentes que contemplam os requisitos funcionais descritos na Tabela 4.1.

A infraestrutura foi construída a partir de uma Arquitetura Orientada a Serviços (SOA), fornecendo abstração sobre a implementação dos serviços disponibilizados, bem como, heterogeneidade e interoperabilidade entre as diversas plataformas de desenvolvimento disponíveis.

Na Figura 4.1 é apresentada uma visão geral da arquitetura da solução, a qual é constituída por cinco componentes, são eles: *Web Service* (WS); *Web Service Manager*; Módulos Online e de Expansão das Conexões Sociais; e o Banco de Dados da infraestrutura.

A comunicação entre as aplicações contidas nos dispositivos móveis dos usuários e a

[1]Disponível em: `http://tomcat.apache.org/download-60.cgi`

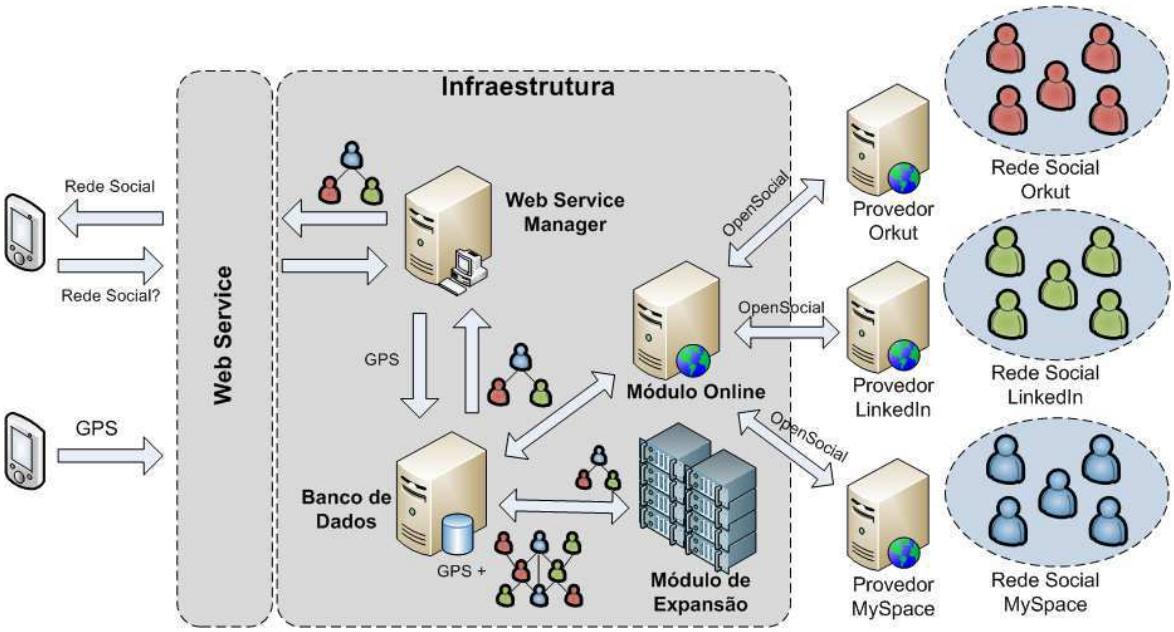


Figura 4.1: Visão Geral da Infraestrutura

infraestrutura ocorre através do *Web Service*, o qual constitui a API de serviços disponibilizados pela infraestrutura e funciona como ponto único de acesso às informações de contexto social e de localização dos usuários que utilizam a infraestrutura. Quando as aplicações instaladas nos dispositivos móveis utilizam o *Web Service*, elas podem simplesmente enviar atualizações sobre as informações de localização dos usuários (e.g., latitude e longitude obtidas do GPS presente nos dispositivos móveis), ou ainda, utilizar os serviços fornecidos para prover aos usuários funcionalidades que utilizem as suas informações de contexto social e de localização, como por exemplo, informar quais são os amigos que estão próximos.

As requisições das aplicações, realizadas através do *Web Service*, aos serviços da infraestrutura são interceptadas pelo componente *Web Service Manager*, o qual é responsável por validar as informações fornecidas pelas aplicações, acessar os recursos disponíveis e coordenar o fluxo de execução das atividades necessárias para atender a requisição do serviço. Por exemplo, se a aplicação no dispositivo móvel requisitar um serviço do *Web Service* que retorne as informações sobre os amigos dos amigos de um usuário (i.e. informações sobre o segundo nível de amizade), esse componente irá acessar as informações de contexto desse usuário contidas no Banco de Dados da infraestrutura, que foram extraídas das redes sociais online pelo Módulo Online e processadas pelo componente de expansão, retornando-as ao dispositivo móvel.

O Módulo Online utiliza as conexões sociais presentes nas redes sociais online para enriquecer as informações de contexto social dos usuários. Esse componente monitora periodicamente as redes sociais online (e.g., Orkut, LinkedIn e MySpace), em busca de atualizações ocorridas sobre as conexões sociais dos usuários cadastrados na infraestrutura, com o propósito de responder a mudanças nessas informações em tempo real. Para isso, esse componente utiliza a conexão com a Internet presente na infraestrutura para comunicar-se via API *OpenSocial* [27] com cada um dos provedores das redes sociais online. Durante essa comunicação, o módulo utiliza um modelo de autorização baseado na instalação de *plugins*[2] para obter permissão de acesso às informações contidas nos perfis dos usuários nas redes sociais online.

Após obter autorização de cada provedor das redes sociais online, ocorre a extração das informações de contexto social dos usuários (principalmente da lista de amigos dos usuários) em cada rede social online. Uma vez de posse dessas informações, esse componente as armazena no Banco de Dados para que elas sejam processadas pelo componente de expansão.

Por sua vez, o Módulo de Expansão das Conexões Sociais recupera as informações de contexto social dos usuários do Banco de Dados e constrói uma representação dessas informações através de grafos [44], onde os vértices representam os usuários e as arestas os relacionamentos existentes entre eles. Após a construção do grafo, o Módulo de Expansão das Conexões Sociais realiza um processamento distribuído sobre o grafo utilizando o algoritmo *Breadth-First Search (BFS)* [16], que determina no máximo seis níveis de amizade para todos os usuários ao explorar a menor quantidade de arestas existente no caminho entre cada par de vértices contidos no grafo. As informações processadas por esse componente são novamente armazenadas no Banco de Dados, aumentando o conhecimento da infraestrutura sobre as informações de contexto social dos usuários. Essas informações são recuperadas através da requisição dos serviços disponibilizados pela API da infraestrutura.

Nas próximas seções descreve-se detalhadamente cada um dos componentes envolvidos na arquitetura da infraestrutura.

[2]No contexto deste trabalho, um *plugin* é uma aplicação social desenvolvida via OpenSocial e disponibilizada pela infraestrutura que, ao ser instalado pelo usuário em seu perfil em uma rede social online qualquer, fornece autorização a uma entidade externa (e.g. uma outra aplicação ou servidor web) para que essa recupere um conjunto de informações públicas disponíveis nos perfis dos usuários, tais como: lista de amigos, preferências pessoais, atualizações, dentre outras.

4.2.1 Web Service e Web Service Manager

O *Web Service*[3] provê acesso às informações de contexto persistidas e aos serviços fornecidos pela infraestrutura. As requisições realizadas aos serviços do *Web Service* são interceptadas pelo *Web Service Manager*, o qual coordena as atividades a serem realizadas internamente pela infraestrutura com o intuito de responder a requisição do serviço.

Na Tabela 4.2 são sumarizados os serviços disponibilizados pelo *Web Service* da infraestrutura. Detalhes sobre os parâmetros de entrada e o retorno de cada serviço estão disponíveis em: `http://embedded.ufcg.edu.br/~danielbruno/infraestrutura/API/`.

Tabela 4.2: API da Infraestrutura.

Serviço	Descrição
addFriendship	Adiciona um novo contato à lista de amigos do usuário
doLogin	Realiza a autenticação do usuário na Infraestrutura
registerUser	Registra um novo usuário
registerLocation	Registra a localização do usuário
getMembersFrom	Lista os membros da Rede Social Móvel
getUserInfo	Recupera as informações sobre o usuário
getUserPreferences	Lista os interesses e/ou preferências do usuário
getFriendsFrom	Lista os amigos do usuário
getFriendsAtLevel	Lista os amigos do usuário para o nível especificado
getSocialNetworksFrom	Lista as redes sociais que o usuário faz parte
getCurrentLocation	Recupera a informação sobre a última localização do usuário
getNearByUsers	Lista os usuários que estão próximos

A API de serviços disponibilizada pelo *Web Service* é utilizada pelas aplicações instaladas nos dispositivos móveis dos usuários, as quais provêem funcionalidades que utilizam as informações de contexto social e de localização dos usuários.

[3]Desenvolvido através da implementação de referência da especificação JSR 224 (JAX-WS - *Java API for XML Web Services*).

4.2.2 Módulo Online

O Módulo Online tem como objetivo principal realizar a integração com as redes sociais online, a fim de utilizar as informações existentes nessas redes para enriquecer as informações de contexto social dos usuários cadastrados na infraestrutura.

Para que a infraestrutura possa extrair as informações dos usuários das redes sociais online, ela precisa ter ciência de quem são os usuários, quais são as redes sociais que eles fazem parte, além de possuir autorização para acessar essas informações. Uma solução para o primeiro problema pode ser obtida através de um processo de cadastramento dos usuários na infraestrutura, no qual é atribuído um identificador único para cada usuário. Por sua vez, a segunda solução consiste em realizar o mapeamento entre os identificadores dos usuários das redes sociais online com aqueles cadastrados na infraestrutura. Por fim, uma solução para o terceiro problema consiste em utilizar a API OpenSocial e um dos seus mecanismos de autenticação e autorização.

A API OpenSocial [27], desenvolvida pelo Google, fornece um conjunto de bibliotecas padronizadas do lado do cliente, baseadas em JavaScript, para construção de aplicações sociais para redes sociais online. Uma vez que essas aplicações estão instaladas no perfil dos usuários, elas fornecem um mecanismo de acesso às informações de perfil dos usuários. Também são fornecidas bibliotecas padronizadas do lado do servidor[4], baseadas em REST (*Representational State Transfer*) e RPC (*Remote Procedure Call*), para interação com a aplicação instalada no perfil do usuário.

É importante destacar que os *plugins* utilizados pelo Módulo Online são implementados via API OpenSocial. Dessa forma, o mesmo *plugin* pode ser instalado em todas as redes sociais suportadas por essa API, sem necessidade de realizar alterações em sua programação.

Ademais, OpenSocial também provê dois mecanismos distintos para autenticação e autorização junto aos provedores das redes sociais online. A principal diferença entre esses mecanismos está no nível de interação exigido com o usuário.

No primeiro mecanismo, quando um servidor ou aplicação tenta acessar as informações do usuário em uma rede social online, ele é automaticamente redirecionado para a página de autenticação do serviço social, para que sejam fornecidas as credenciais do usuário (e.g., *login* e senha) para acesso ao serviço. Após a autenticação bem-sucedida no serviço social,

[4]Disponíveis em: `http://wiki.opensocial.org/index.php?title=Client_Libraries`

é concedida autorização, através de um *token* recebido, para acessar e/ou extrair as informações do perfil do usuário. Assim, sempre que o *token* expirar o usuário necessitará realizar o processo de autenticação novamente. Esse mecanismo de autenticação e autorização é denominado *3-legged OAuth*.

Já no segundo mecanismo, o servidor ou aplicação mantém armazenadas as credenciais dos usuários (i.e. identificador OpenSocial específico para cada rede social online) e as credenciais de uma aplicação instalada no perfil dos usuários (i.e. um identificador único para cada aplicação), as quais são utilizadas para acessar as conexões sociais dos usuários nas redes sociais online. Isso é possível, pois o próprio usuário, mediante instalação de uma aplicação social disponibilizada pelo servidor, autoriza o acesso aos dados de seu perfil na rede social online. Assim, o servidor ou aplicação necessita da interação dos usuários com o serviço social uma única vez (durante a instalação da aplicação), para extrair os dados dos seus perfis. Esse mecanismo de autenticação e autorização é denominado *2-legged OAuth*.

Na Figura 4.2 é exemplificado como ocorre a interação entre o Módulo Online e o provedor de acesso da rede social online, utilizando o mecanismo de autenticação e autorização *2-legged OAuth*, para recuperação das informações dos usuários das redes sociais online.

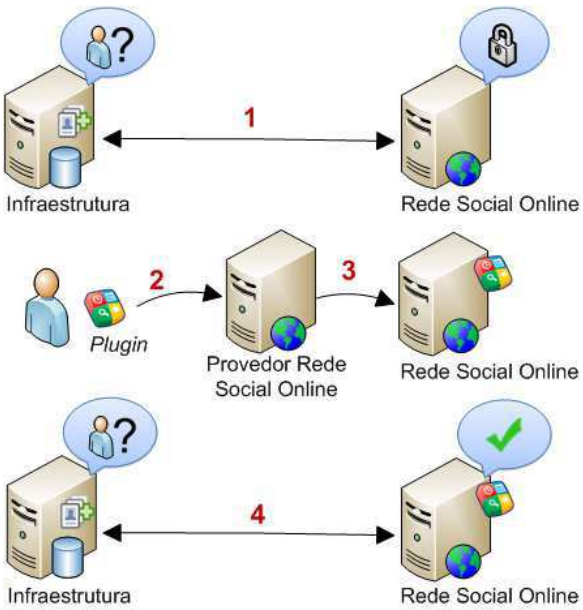


Figura 4.2: Autenticação *OAuth 2-legged*

Inicialmente, a infraestrutura tenta acessar as informações dos usuários através do provedor da rede social online. Contudo, o acesso a essas informações é negado, pois a infraestrutura não possui autorização do usuário para acessar as suas informações pessoais *(1)*. Para que essas informações possam ser acessadas, o usuário utiliza o diretório de aplicações da sua rede social online para proceder com a instalação do *plugin* disponibilizado pela infraestrutura *(2)*. Após essa instalação, o usuário configura as permissões de acesso do *plugin* a partir do seu perfil *(3)*. Uma vez que o *plugin* esteja instalado e configurado, a infraestrutura pode então fornecer novamente as credenciais do usuário e acessar as informações públicas desse perfil na rede social online, tais como lista de amigos, interesses, alguns dados pessoais do usuário, entre outras *(4)*.

As etapas a seguir descrevem o processo de configuração necessário para que a infraestrutura torne-se ciente das redes sociais dos usuários:

1. Geração de identificador único do usuário na infraestrutura;
2. Instalação do *plugin* disponibilizado pela infraestrutura em cada rede social online que os usuários façam parte;
3. Execução do *plugin* instalado, para realização do mapeamento entre o identificador único do usuário na infraestrutura com o seu identificador da rede social online.

Na primeira etapa, o usuário deve efetuar seu cadastro através do *website* disponibilizado pela infraestrutura. Conforme ilustrado na Figura 4.3, o usuário deve fornecer algumas informações pessoais, tais como: primeiro e último nome, *login* e senha. Essas informações são enviadas à infraestrutura através da sua API, sendo atribuído o *login* fornecido pelo usuário como sendo seu identificador único na infraestrutura. Este processo também pode ser realizado via aplicação móvel, utilizando a API de WS.

Após efetuação do cadastro, o usuário deve realizar a instalação do *plugin* disponibilizado pela infraestrutura nas redes sociais online das quais seja membro. Na Figura 4.4 é mostrada a tela de um *plugin* instalado na rede social Orkut.

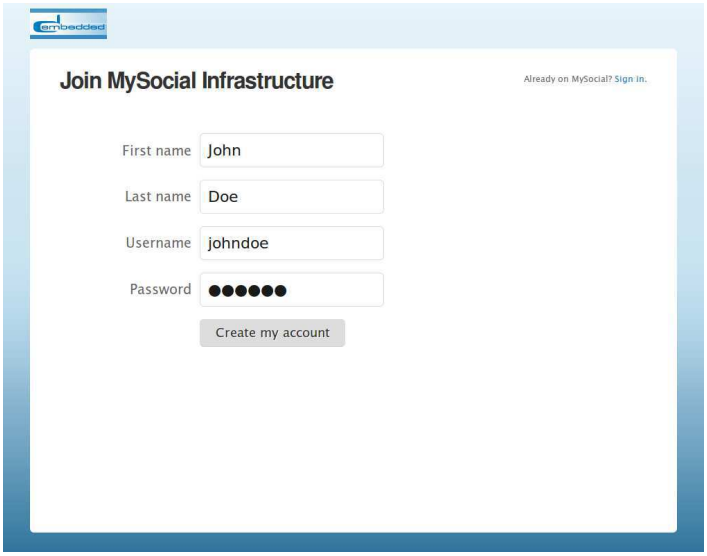


Figura 4.3: Exemplo da tela de cadastro de usuários

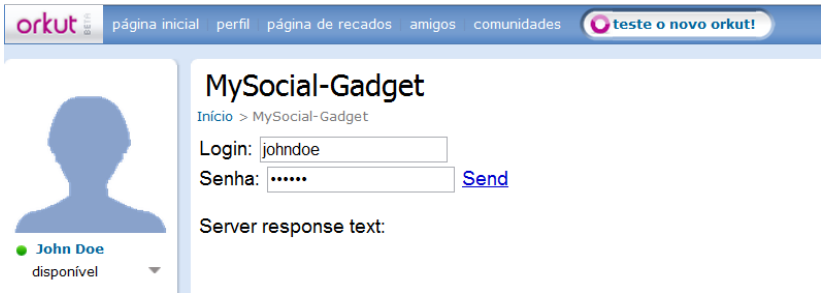


Figura 4.4: *Plugin* MySocial instalado na Rede Social Orkut

Por fim, após instalação do *plugin* em seu perfil da rede social online, o usuário deve executá-lo e autenticar-se na infraestrutura, fornecendo os valores para o *login* e senha cadastrados através do *website* da infraestrutura. Esse último passo envia à infraestrutura as informações necessárias para realização do mapeamento entre o *login* do usuário e seu identificador social (OpenSocial ID). Um exemplo desse mapeamento é apresentado através da Tabela 4.3.

A partir desse momento, o processo de autorização está completo e o usuário pode utilizar qualquer aplicação móvel instalada em seu dispositivo móvel que utilize a infraestrutura.

A modelagem do Módulo Online está disponível na Seção A.1.2. Já o Código Fonte completo do *plugin* desenvolvido está disponível no *Apêndice B*.

Tabela 4.3: Mapeamento do Identificador OpenSocial do usuário para o Identificador único da Infraestrutura.

Login	**OpenSocialId**	**Rede Social**
johndoe	16402456860419996289	Orkut
johndoe	530420627	MySpace
fulanosousa	05504131713099562528	Orkut
fulanosousa	531009501	MySpace
sicranasantos	05347336523688684229	Orkut
beltranosilva	531010418	MySpace

4.2.3 Módulo de Expansão das Conexões Sociais

O Módulo de Expansão das Conexões Sociais possui como objetivo calcular os níveis de relacionamento diretos e indiretos entre todos os membros de uma Rede Social Móvel, utilizando as informações de contexto social (advindas do processamento prévio do Módulo Online) disponíveis na infraestrutura, e armazenar essas novas informações em um mecanismo de armazenamento persistente para que elas sejam utilizadas pelos outros módulos da infraestrutura.

O desenvolvimento desse módulo teve como motivação os resultados obtidos em [54], os quais demonstraram que há uma maior probabilidade de um usuário vir a formar um novo vínculo de amizade com alguém próximo ao seu ciclo de amizades (e.g., com um amigo de um amigo em comum), do que com um usuário completamente desconhecido (e.g., usuário recomendado a partir de um algoritmo de casamento de perfis ou recomendação).

O módulo proposto modela os relacionamentos sociais dos usuários através de uma representação de grafos [44]. Nesse modelo, os vértices representam os usuários e as arestas os relacionamentos existentes entre eles. Ainda, as arestas são não-direcionadas, uma vez que o relacionamento de amizade entre dois usuários é considerado mútuo ou bidirecional pela infraestrutura. Além disso, as arestas do grafo podem ser ponderadas com peso igual a 1, ou ainda, podem ser representadas sem a utilização de pesos.

Mais especificamente, o Módulo de Expansão das Conexões Sociais utiliza um algo-

ritmo que soluciona um problema clássico da Teoria dos Grafos, denominado *problema de caminhos mais curtos de todos os pares*, que consiste em determinar a menor distância entre cada par de vértice de um grafo. De acordo com Cormen, em [16], dentre as possíveis soluções para este problema destacam-se: os algoritmos de *Johnson*, *Floyd-Warshall* e da *elevação ao quadrado repetida em matrizes*, com complexidade de $O(V^2 lgV + VE)$, $\Theta(V^3)$ e $\Theta(V^3 lgV)$, respectivamente. Outra alternativa possível é aplicar um algoritmo que soluciona o *problema de caminhos mais curtos de única origem* para todos os vértices do grafo. Algumas soluções para esse problema são os algoritmos de *Bellman-Ford* $O(V^2E)$ [16] e *Dijkstra* $O(V^2 lgV + VE)$ [16].

Entretanto, uma vez que na representação escolhida as arestas podem ser representadas sem peso, ou com peso igual a 1, o algoritmo *Breadth-First Search (BFS)* [16] para travessia em largura em grafos, com complexidade $O(V + E)$, mostra-se uma boa alternativa para solucionar o problema. Essa solução apresenta uma complexidade assintoticamente melhor do que os algoritmos de *Dijkstra* e *Bellman-Ford*, que são utilizados em problemas onde os pesos podem ser positivos ou negativos entre as arestas do grafo. Além disso, diferentemente das outras soluções, esse algoritmo pode ser facilmente adaptado para que a travessia no grafo seja limitada pela quantidade de níveis a serem percorridos, conforme demonstrado através do Código Fonte 4.1.

Código Fonte 4.1: Algoritmo BFS limitado por nível

```
1  BFS(G, s, max)
2  para cada vértice u ∈ V[G] − {s}
3          do cor[u] ← BRANCA
4                  distancia[u] ← ∞
5                  antecessor[u] ← NIL
6  cor[s] ← CINZA
7  distancia[s] ← 0
8  antecessor[s] ← NIL
9  Q ← 0
10 ENQUEUE(Q, s)
11 while Q ≠ 0
12   do u ← DEQUEUE(Q)
13     para cada vértice v ← listaAdjacencia[u]
14       do if cor[v] = BRANCA
```

```
15          then cor[v] = CINZA
16               distancia[v] = distancia[u] + 1
17               antecessor[v] = u
18               do if distancia[v] < max
19                 then ENQUEUE(Q,v)
20    cor[u] = PRETA
```

O procedimento BFS limitado por nível funciona da seguinte forma. Nas linhas 2 e 3 é atribuído para todos os vértices do Grafo *G* a cor *branca* (i.e. vértice não descoberto), em seguida são definidos a distância para todo vértice *u* como sendo infinita (linha 4) e o antecessor de todo vértice como sendo NIL (i.e. desconhecido - linha 5). Na linha 6 é redefinida a cor do vértice *s* (vértice de origem) como sendo cinza, pois apesar do vértice ter sido descoberto as suas arestas ainda precisam ser exploradas. Na linha 7 é atribuído NIL como valor do antecessor do vértice *s* (i.e. o vértice de origem não possui antecessor). Nas linhas 8 e 9, a Fila *Q* é inicializada e o vértice *s* é inserido na posição inicial, respectivamente.

No escopo do *Loop while*, linhas 11 a 20, é verificado se a Fila *Q* não está vazia (linha 11), iniciando cada nova iteração com a remoção do próximo vértice *u* (sempre de cor *cinza*) da Fila *Q* (linha 12). Ainda, para cada vértice contido na lista de adjacência do vértice *u* (linha 13), são realizados os seguintes procedimentos: verificação da cor do vértice (linha 14), caso esteja atribuída a cor *branca*, será atribuída a cor *cinza* (linha 15), atribuição do valor da distância para o vértice descoberto como sendo igual a distância do vértice *u* mais 1 (linha 16), finalizando com a atribuição do vértice *u* como antecessor do vértice *v* (linha 17). Em seguida, na linha 18 é verificado se a distância do vértice *v* é menor do que o valor contido na variável *max* (i.e. o nível máximo a ser descoberto), em caso afirmativo, os vértices adjacentes ao vértice *v* serão inseridos ao final da Fila *Q* (linha 19). Quando todos os vértices contidos na lista de adjacência do vértice *u* forem explorados, a cor do vértice *u* é modificada para *preta* (linha 20).

O módulo proposto executa a versão modificada do algoritmo BFS para todos os vértices do grafo, objetivando determinar no máximo seis níveis de amizade para cada vértice. De acordo com [16], o tempo de execução do algoritmo BFS para resolução do *problema de caminhos mais curtos de todos os pares* possui complexidade $O(V^2 + VE)$, ou $O(VE)$, assumindo que o grafo está conectado.

A escolha de processar no máximo seis níveis foi realizada com base na *Teoria dos Seis Graus de Separação* [37, 45], que afirma que são necessários em média seis laços de amizade para que duas pessoas quaisquer no mundo estejam conectadas entre si. Ainda, alguns estudos como aqueles realizados em [38] e [40] demonstraram que essa teoria pode ser aplicada para as Redes Sociais Online[5].

O resultado obtido como saída da execução do algoritmo, pode ser utilizado para indicar a menor distância no grafo entre dois pares de vértices. Ou ainda, na modelagem utilizada, o nível de relacionamento existente entre dois usuários. Dessa forma, o nível 1 (i.e. distância igual a 1 entre dois pares de vértices) indica que os usuários são amigos diretos e os demais níveis (i.e. distâncias maiores ou iguais a 2 entre dois pares de vértices) indicam que os usuários estão relacionados indiretamente (e.g., através de amigos em comum na passagem de um nível para outro).

Por fim, as informações calculadas pelo módulo são persistidas na base de dados da infraestrutura para que possam ser reutilizadas, posteriormente, por outras aplicações.

O Módulo de Expansão das Conexões Sociais foi implementado através de uma arquitetura Cliente/Servidor ou Produtor/Consumidor, sendo executado em um conjunto de computadores organizados em *cluster*. A comunicação entre o servidor e o(s) cliente(s) é realizada através de Java RMI (*Remote Method Invocation*).

A modelagem do módulo está disponível na Seção A.1.3. No *Capítulo 6* é realizada uma avaliação de escalabilidade utilizando simulação sobre a implementação realizada do Módulo de Expansão das Conexões Sociais.

4.2.4 Banco de Dados

A comunicação com o mecanismo de persistência da infraestrutura é realizada através de uma Camada de Dados que é responsável por fornecer serviços para armazenamento e recuperação das informações de contexto social e de localização dos usuários, utilizando o mecanismo de persistência da infraestrutura, aos Módulos Online e de Expansão das Conexões Sociais.

A infraestrutura desenvolvida utiliza uma implementação para o mecanismo de persis-

[5]Nos estudos realizados, os autores encontraram um valor médio para o caminho mínimo entre 5 e 7, para quaisquer pares de vértices distintos pertencentes às amostras dos grafos das redes sociais analisadas.

tência baseada no Sistema de Gerenciamento de Banco de Dados (SGBD) Relacional PostgreSQL[6], sendo realizada a comunicação entre a Camada de Dados e o Banco de Dados PostgreSQL através de JDBC (*Java Data Base Connectivity)*.

A modelagem deste módulo está disponível na Seção A.1.4. Adicionalmente, no *Apêndice C* estão disponíveis o modelo de Entidade-Relacionamento e o *script* para geração do Banco de Dados utilizado pela infraestrutura.

4.3 Conclusões do Capítulo

Neste capítulo foram discutidos os principais aspectos relacionados ao projeto da Infraestrutura para Desenvolvimento de Aplicações Pervasivas Cientes de Redes Sociais. Inicialmente, foram apresentados os requisitos funcionais da infraestrutura. Com base nesses requisitos, foi especificada e depois instanciada a arquitetura da infraestrutura. Em seguida, foram descritos cada um dos módulos contidos na arquitetura e como eles interagem entre si para prover as funcionalidades especificadas. Destaque especial para a especificação e implementação do Módulo de Expansão das Conexões Sociais, o qual constitui o diferencial entre a infraestrutura desenvolvida e as soluções correlatas discutidas no *Capítulo 3*. A avaliação da infraestrutura foi realizada através do desenvolvimento de estudo de caso e avaliação de escalabilidade do Módulo de Expansão das Conexões Sociais utilizando simulação, discutidas nos *Capítulos 5 e 6*, respectivamente.

[6]Disponível em: `http://www.postgresql.org/`

Capítulo 5

Estudo de Caso

Neste capítulo é apresentado o desenvolvimento de um estudo de caso para demonstrar o suporte da infraestrutura à criação de aplicações sociais cientes de redes sociais. A aplicação desenvolvida no estudo de caso apresenta funcionalidades que utilizam as informações de contexto social e de localização, bem como, os serviços disponibilizados pela API da infraestrutura. O capítulo está organizado da seguinte forma. Inicialmente, é realizada uma breve descrição da aplicação, apresentando as suas principais funcionalidades e descrevendo como elas interagem com a infraestrutura. Ao final, é realizada uma simulação da aplicação em execução.

5.1 Descrição da Aplicação

MySocial é uma aplicação social móvel baseada em localização, que possui como objetivos indicar a localização dos usuários para seus contatos das redes sociais online e possibilitar que os usuários realizem o acompanhamento da localização dos contatos que estão próximos em tempo real. Essa aplicação pode ser utilizada, por exemplo, em cenários de Computação Pervasiva como o descrito a seguir:

João está caminhando em direção à praça de alimentação do shopping de sua cidade na hora do almoço, quando decide iniciar uma aplicação instalada em seu smartphone para saber se algum dos seus amigos está próximo para almoçarem juntos. A aplicação utilizada por João envia, através da Internet, a informação sobre a localização dos usuários que a utilizam para um servidor externo que, além de receber e armazenar essas informações,

também possui ciência das redes sociais online da qual os usuários fazem parte.

Utilizando as informações de contexto disponíveis, o servidor determina em tempo real, que não há nenhum amigo direto de João próximo da praça de alimentação. No entanto, também possui a informação de que João é amigo, no Orkut, de José, o qual também é membro do Facebook, onde é amigo de Pedro. Coincidentemente, Pedro é um amigo que João não vê desde os tempos de faculdade e está localizado a poucos metros de distância. Dessa forma, a aplicação informa a João que Pedro é amigo de José no Facebook e está próximo. João, por sua vez, observa a localização de Pedro no mapa gerado pela aplicação e decide ir ao seu encontro para conversarem e almoçarem juntos.

Especificamente, a aplicação MySocial utiliza as informações de contexto social e de localização dos usuários, obtidas a partir da API de *Web Service* da infraestrutura, para apresentar através de listas de contatos os amigos diretos e indiretos das redes sociais online que os usuários participam e informar aos usuários quando esses contatos estiverem próximos.

Na Tabela 5.1 está sumarizado o conjunto de funcionalidades providas pela aplicação MySocial.

Tabela 5.1: Funcionalidades da aplicação MySocial

Código	**Funcionalidade**
F1	Gerenciar o acesso do usuário à aplicação
F2	Listar os contatos do usuário por redes sociais online
F3	Disponibilizar informações sobre os contatos
F4	Exibir o nível de relacionamento existente entre o usuário e seus contatos
F5	Enviar a informação sobre a localização dos usuários
F6	Recuperar a informação sobre a localização dos usuários que estão próximos
F7	Exibir a localização geográfica dos usuários através de mapa

5.2 Desenvolvimento da Aplicação

Nesta seção é descrito como foi realizado o desenvolvimento da aplicação MySocial, a partir da implementação das funcionalidades descritas na Tabela 5.1.

A aplicação MySocial foi desenvolvida para plataforma Java ME (*Java Micro Edition*), requerendo como configuração mínima recomendada CLDC (*Connected Limited Device Configuration*) 1.1 e MIDP (*Mobile Information Device Profile*) 2.0, para instalação e execução da aplicação no dispositivo móvel do usuário. Além disso, o dispositivo móvel deve suportar as seguintes APIs: *Location API* (JSR 179) e *J2ME Web Services* (JSR 172). Com base nas especificações anteriores, foi utilizado um dispositivo móvel Nokia E71 para instalação e execução da aplicação, o qual contém as características presentes na Tabela 5.2.

Tabela 5.2: Especificação do dispositivo Nokia E71

Característica	**Descrição**
Sistema Operacional	Symbian OS 9.2, Series 60 v3.1 UI
Processador	ARM 11 de 369 MHz
Memória	128 MB RAM
Armazenamento	110 MB de armazenamento
3G	HSDPA, 3.6 Mbps
WLAN	Wi-Fi 802.11 b/g
GPS	Suporte a A-GPS

5.2.1 Implementação das Funcionalidades da Aplicação

Nesta seção é descrito como as funcionalidades da aplicação MySocial foram implementadas a partir da interação com os serviços disponibilizados pela API de *Web Service* da infraestrutura.

Gerenciar o acesso do usuário à aplicação

Essa funcionalidade é responsável por prover um mecanismo de autenticação visual, através do qual os usuários obtém acesso as funcionalidades da aplicação MySocial. Tal funcionalidade é implementada através de um formulário onde o usuário fornece o *login* e senha cadastrados na infraestrutura para obter acesso à aplicação.

O serviço disponibilizado pela API de *Web Service* utilizado para implementar essa funcionalidade é o *doLogin*.

Listar os contatos do usuário por redes sociais online

Essa funcionalidade é responsável por disponibilizar listas de contatos ao usuário, para que ele possa visualizar as informações sobre os contatos de cada rede social online armazenadas pela infraestrutura. A implementação realizada é responsável por criar uma visão dos contatos dos usuários para cada rede social online separadas por abas. Por exemplo, se o usuário instalar o *plugin* MySocial nas redes sociais Orkut e MySpace, a aplicação apresentará as informações sobre os contatos do usuário através de três abas distintas, são elas: uma aba contendo uma lista de contatos do Orkut, outra contendo uma lista de contatos do MySpace e a terceira contendo o resultado da união entre as duas listas de contatos utilizadas (i.e. resultante da união entre os grafos das duas redes sociais online).

O serviço disponibilizado pela API de *Web Service* utilizado para implementar essa funcionalidade é o *getFriendsFrom*, que recupera as informações sobre os contatos do usuário de cada rede social online, armazenadas pela infraestrutura.

Exibir o nível de relacionamento existente entre o usuário e seus contatos

Essa funcionalidade é responsável por adicionar, na lista de contatos de cada rede social online, a informação sobre o nível de amizade entre o usuário e os contatos. Dessa forma, essa funcionalidade é alcançada através da combinação dos resultados obtidos pela invocação dos serviços *getFriendsFrom* e *getFriendsAtLevel* do *Web Service* da infraestrutura.

Disponibilizar informações sobre os contatos

Essa funcionalidade permite que o usuário visualize na tela do dispositivo móvel informações sobre um contato selecionado na lista de contatos. Essa funcionalidade é implementada através de um formulário e utiliza a combinação dos resultados obtidos da invocação dos serviços *getUserInfo* e *getUserPreferences* da API de *Web Service* fornecida pela infraestrutura.

Enviar a informação sobre a localização dos usuários

Essa funcionalidade envia periodicamente à infraestrutura atualizações sobre a informação de localização do usuário. Para isso, é utilizada a API de localização (JSR 179), disponível na plataforma Java ME. Especificamente, após a configuração do provedor de localização (e.g., GPS do dispositivo, triangulação das antenas da rede móvel celular ou triangulação das antenas de pontos de acesso *Wi-Fi*), as informações sobre a localização do usuário (latitude e longitude obtidas a partir do GPS do dispositivo móvel) são atualizadas a cada minuto e enviadas através do *Web Service* da infraestrutura, a partir da invocação do serviço *registerLocation*, para serem persistidas no banco de dados da infraestrutura, mantendo um histórico sobre a localização dos usuários.

Recuperar a informação sobre a localização dos usuários que estão próximos

Essa funcionalidade permite que a aplicação MySocial informe visualmente os usuários sobre os contatos que estão próximos. Para isso, a aplicação comunica-se com o *Web Service* através da requisição do serviço *getNearByUsers*. Esse serviço retorna a distância em metros entre o usuário e os contatos em cada rede social online. A informação sobre a localização dos usuários que estão próximos é adicionada a cada uma das listas de contatos do usuário.

Exibir a localização geográfica dos usuários através de mapa

Essa funcionalidade permite ao usuário visualizar na tela do dispositivo móvel a sua distância em metros em relação a um contato presente nas listas de contatos de cada rede social. Para disponibilizar essa funcionalidade, a aplicação MySocial comunica-se com o serviço *Google Maps* a partir da API estática desse serviço, enviando as informações de localização (latitude e longitude) sobre os pontos de interesse que devem ser mostrados no mapa. Como resultado, a API do serviço *Google Maps* retorna uma imagem à aplicação contendo a localização dos usuários, para que essa seja renderizada na tela do dispositivo móvel.

5.3 Configuração da Infraestrutura

5.3.1 Instalação dos *Plugins* nas Redes Sociais

Para que a infraestrutura possa extrair as informações de contexto social e de localização das redes sociais, o usuário deve instalar e configurar os *plugins* disponibilizados pela infraestrutura em cada rede social online da qual faça parte. A realização desse processo, conforme discutido na Seção 4.2.2, é requerida para que a infraestrutura torne-se ciente das redes sociais online dos usuários e obtenha autorização de cada provedor da rede social online para acessar e extrair as informações de contexto dos usuários dessas redes.

5.3.2 Simulação do Grafo das Conexões Sociais dos Usuários

Com o objetivo de simular as conexões sociais dos usuários das redes sociais online e manipular estruturas de dados representadas por grafos, foi utilizada a biblioteca em Java JGraphT[1]. Essa biblioteca disponibiliza classes que constroem grafos de acordo com a topologia escolhida pelo usuário, a quantidade de vértices e de arestas passadas como parâmetros. Para o estudo de caso utilizado foi gerado um grafo, a partir da biblioteca JGraphT, representando a topologia *Scale-Free* [47], que caracteriza as redes sociais online. O grafo gerado constitui-se de 100 mil vértices e aproximadamente 158 mil arestas[2], representando os usuários do sistema e os relacionamentos existentes entre eles, respectivamente.

5.3.3 Processamento do Grafo das Conexões Sociais dos Usuários

O grafo gerado na seção anterior é processado pelo Módulo de Expansão das Conexões Sociais presente na Infraestrutura. Conforme discutido na Seção 4.2.3, esse módulo tem como objetivo executar um algoritmo que soluciona o *problema de caminhos mais curtos de todos os pares*. Assim, após execução do algoritmo, espera-se obter como resultado, para cada usuário presente no grafo, uma tabela *hash* [16]. Onde, as chaves (valores inteiros entre 1 e 6) representam a menor distância entre o vértice inicial e os demais vértices no grafo.

No contexto deste trabalho, o resultado obtido pode ser interpretado da seguinte maneira:

[1]Disponível em: `http://www.jgrapht.org/`

[2]JGraphT não permite que a quantidade de arestas seja determinada *a priori* para a topologia *Scale-Free*.

para cada Tabela *hash* gerada, suas chaves armazenam os níveis de relacionamento existente entre o usuário inicial e os demais usuários do sistema. Onde, o nível 1 contém os amigos diretos do usuário (i.e. os vizinhos do vértice inicial) e os demais níveis contém os amigos indiretos do usuário (i.e. os vértices alcançáveis a partir do vértice inicial cuja distância seja maior ou igual a dois).

5.4 Simulação do Funcionamento da Aplicação MySocial

Nesta seção é demonstrado o funcionamento da aplicação MySocial através de sua interação com a infraestrutura. A fim de atingir esse objetivo, realizou-se uma simulação de vários usuários utilizando a aplicação, os quais enviam simultaneamente atualizações de sua localização à infraestrutura.

5.4.1 Obtenção dos dados para Simulação da Mobilidade dos Usuários

O objetivo desta etapa foi o de obter um conjunto de coordenadas geográficas para simular a mobilidade dos usuários no sistema. Para atingir este objetivo foram escolhidas as coordenadas de latitude e longitude referentes à localização do Laboratório Embedded, para compor as coordenadas geográficas que servem de base para obtenção das demais.

Utilizando o par de coordenadas escolhido, aplicou-se uma variação equivalente a cinco metros no valor da latitude, fixando-se o valor da longitude, até o raio máximo de 1500 metros da distância inicial. Esse procedimento foi realizado novamente, dessa vez fixando-se o valor da latitude e variando-se o valor da longitude, utilizando novamente um raio de 1500 metros. Como resultado foram obtidas duas retas perpendiculares com raio de 1500 metros (sentidos norte, sul, leste e oste), contendo o par de coordenadas de latitude e longitude iniciais como ponto de intersecção entre elas, cujos pontos são equidistantes entre si, com uma distância de ±5 metros entre o ponto anterior e o posterior.

5.4.2 Simulação da Mobilidade dos Usuários

Com base nas coordenadas geográficas obtidas anteriormente e de uma lista contendo os usuários cadastrados no sistema, aplicou-se o seguinte procedimento para simular a mobili-

dade dos usuários.

Inicialmente, foram selecionados de forma aleatória, um usuário da lista e um par de coordenadas de latitude e longitude (i.e. um ponto presente nas retas geradas na etapa anterior) a cada segundo. Durante o processo de escolha das novas coordenadas, o procedimento verifica a última posição em que o usuário se encontrava, a fim de não produzir uma variação de movimento não permitida (e.g., o usuário em um instante de tempo estar localizado a uma distância de 200 metros das coordenadas iniciais e 1 segundo após estar a 1000 metros da distância anterior). Caso as coordenadas selecionadas sejam inválidas (i.e. produzindo deslocamento não permitido), o procedimento selecionará novas coordenadas, reiniciando o processo de verificação anterior.

Por fim, de posse das informações selecionadas, o procedimento utilizado registra através da API da infraestrutura o novo par de coordenadas geográficas para o usuário selecionado. Esse procedimento é repetido para os demais usuários do sistema indefinidamente, enquanto não tiver sua execução finalizada.

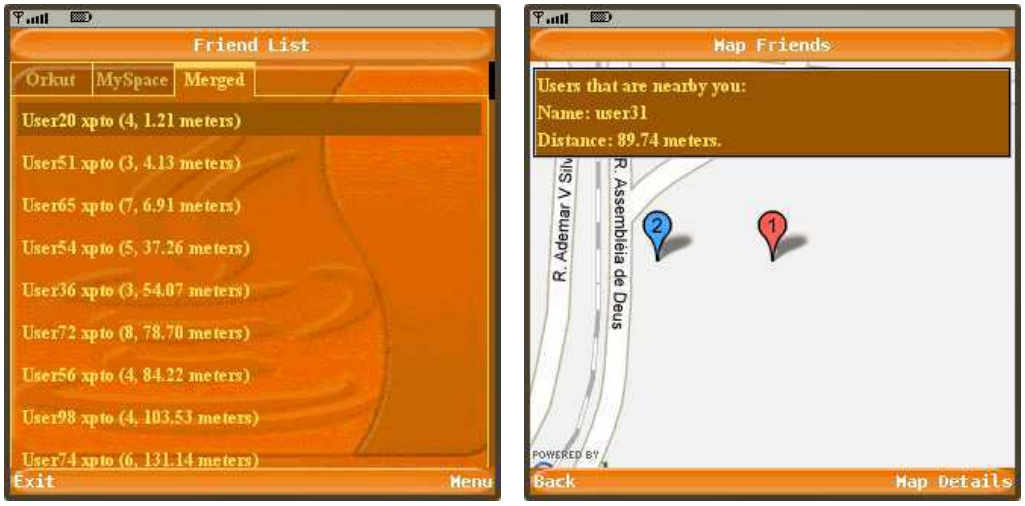
5.5 Resultados

No lado da infraestrutura, as informações de localização dos usuários enviadas pelo simulador são recebidas através do *Web Service* e armazenadas no Banco de Dados, sendo persistidas na tabela que armazena o histórico da localização dos usuários. Quando a aplicação MySocial requisita o serviço *getNearByUsers*, através da API da infraestrutura, esta recupera as informações sobre os amigos diretos e indiretos do usuário (amigos de nível 1 e entre os níveis 2 e 6, respectivamente) em tempo real. Uma vez que essas informações já foram processadas previamente pelo Módulo de Expansão das Conexões Sociais, em conjunto com as informações sobre a localização desses usuários armazenadas na base de dados.

Do ponto de vista do usuário, que está observando a aplicação MySocial sendo executada em seu *smartphone*, a cada minuto as informações da sua lista de contatos são atualizadas, com base nas informações obtidas a partir da API da infraestrutura. A *interface* gráfica do dispositivo móvel do usuário apresentará as seguintes informações: nome do contato, o nível de amizade e a distância em metros entre os dois usuários. Caso o usuário deseje visualizar a localização do contato, a aplicação apresentará através de um mapa a localização dos dois

usuários, indicando a distância entre eles.

Os resultados obtidos são apresentados através da Figura 5.1. Na Figura 5.1(a), o usuário pode visualizar os amigos diretos e indiretos que estão próximos, através de listas de contatos separadas pelas redes sociais online que o usuário faz parte. Já na Figura 5.1(b), é ilustrada a tela da aplicação MySocial que mostra a partir de um mapa a distância em metros do usuário que está utilizando a aplicação para outro usuário presente na sua lista de contatos.



(a) Lista de contatos

(b) Visualização da localização dos usuários através de um mapa

Figura 5.1: Telas da aplicação MySocial

5.6 Conclusões do Capítulo

Neste capítulo foi apresentado o primeiro mecanismo de validação da infraestrutura desenvolvida. Esse mecanismo como estudo de caso, o desenvolvimento da aplicação MySocial, construída a partir das funcionalidades disponibilizadas pela infraestrutura. O objetivo desse estudo de caso foi demonstrar como a infraestrutura pode ser utilizada para auxiliar no desenvolvimento de aplicações pervasivas cientes de redes sociais. No *Capítulo 6*, é apresentado o segundo mecanismo de validação da infraestrutura, o qual tem como objetivo avaliar a escalabilidade do Módulo de Expansão das Conexões Sociais.

Capítulo 6

Avaliação do Módulo de Expansão das Conexões Sociais

Neste capítulo é realizada uma avaliação do Módulo de Expansão das Conexões Sociais fornecido pela infraestrutura utilizando simulação. Esta análise tem como objetivo específico avaliar a escalabilidade do módulo desenvolvido.

6.1 Avaliação

No decorrer desta seção são descritas as etapas realizadas para avaliação da escalabilidade do Módulo de Expansão das Conexões Sociais disponibilizado pela infraestrutura.

6.1.1 Configuração do ambiente de execução

Para execução das simulações foi utilizado o *cluster* disponível no Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded) da Universidade Federal de Campina Grande.

O *cluster* utilizado apresenta a configuração descrita a seguir. São 9 máquinas conectadas através de um *switch gigabit ethernet*, cada uma contendo: 2 processadores Intel (R) Xeon (R) x64 de 2,33 GHz, 4 GB de memória DDR2 800 MHz, 2 HDs SATA 7500 RPM de 250 GB e 2 placas de rede *Gigabit Ethernet*. Além disso, cada um dos nós do *cluster* executa de forma independente o Sistema Operacional *Ubuntu Server*, versão 10.10, para arquitetura

Intel de 64 bits e mantém um compartilhamento de arquivos na rede através de NFS (*Network File System*).

Na configuração utilizada, um dos nós desempenha o papel de servidor e os oito nós restantes desempenham o papel de clientes. O servidor possui as seguintes atribuições: (1) construir grafos com topologia *Scale-Free* (i.e. topologia análoga a das Redes Sociais Online) e com a quantidade de vértices requerida; (2) gerenciar as conexões com os clientes (e.g., abertura e fechamento de conexões, etc); (3) dividir a quantidade de vértices total de forma proporcional à quantidade de clientes disponíveis, de modo que eles processem cargas de trabalho com complexidade computacional aproximadamente equivalente[1]. Para isso, o servidor utiliza o algoritmo de *Round-robin* para divisão do conjunto de vértices entre as cargas de trabalho dos clientes; (4) gerenciar as estatísticas dos processamentos enviadas pelos clientes, após o término da execução de sua carga de trabalho; e (5) persistir os dados processados na base de dados.

Cada um dos clientes recebe o grafo criado pelo servidor, em conjunto com uma lista de trabalho (i.e. um conjunto de vértices distintos a serem processados), e são responsáveis por determinar no máximo seis níveis de relacionamento para cada vértice pertencente ao conjunto de vértices sob sua responsabilidade. De posse do grafo gerado e da lista de vértices a serem processados, os clientes iniciam o processamento distribuído da sua carga de trabalho. Nessa etapa são gerados arquivos contendo o resultado do processamento dos clientes. Esses arquivos são armazenados no diretório de rede compartilhado por todos os nós do *cluster* e persistidos na base de dados pelo servidor. Após o término do processamento de todos os vértices de suas respectivas listas de trabalho, cada cliente envia para o servidor as estatísticas referentes ao processamento realizado.

No *Apêndice D* estão disponíveis os *scripts bash* para execução do servidor e clientes no *cluster*.

[1]Embora cada cliente possua as mesmas configurações de *hardware* e receba cargas de trabalho com a mesma quantidade de vértices, essas últimas podem requerer esforços computacionais bastante distintos para serem processadas. Por exemplo, considerando oito clientes disponíveis, ao dividir o conjunto de vértices proporcionalmente entre eles, aqueles vértices com maior quantidade de conexões podem ser concentrados em determinadas cargas de trabalho, fazendo com que alguns clientes trabalhem muito mais do que outros.

6.1.2 Metodologia

O objetivo da simulação realizada foi avaliar o tempo de resposta necessário para manter consistente as informações de amizade calculadas (máximo de 6 níveis) para todos os usuários após alterações no grafo, utilizando o Módulo de Expansão das Conexões Sociais. Para isso, foram analisados o tempo médio de processamento e a frequência de atualização das informações de cada nível, por unidade de tempo (e.g., minuto, hora e dia), em grafos contendo as configurações dispostas na Tabela 6.1.

Tabela 6.1: Configurações dos grafos avaliados

Configuração	**# Vértices**	**# Arestas**	**Topologia**
1	10.000	~15,8 mil	*Scale-Free*
2	25.000	~39,6 mil	*Scale-Free*
3	50.000	~79 mil	*Scale-Free*
4	75.000	~118,7 mil	*Scale-Free*
5	100.000	~158,8 mil	*Scale-Free*

Mediante utilização da biblioteca JGraphT, foram realizadas simulações das conexões sociais dos usuários, a partir da geração de grafos contendo topologia análoga a das Redes Sociais Online. Durante cada simulação, o Módulo de Expansão das Conexões Sociais determinou o tempo necessário para processar cada nível de amizade para todos os usuários presentes nos grafos, contendo as configurações anteriores. Para cada uma das cinco configurações avaliadas, os nós-clientes processaram dez grafos distintos, sendo armazenados os piores tempos de processamento (registrados em milissegundos) dentre os resultados enviados pelos clientes, em cada simulação. Dessa forma, a análise levou em consideração a evolução dos grafos (i.e. o aumento da quantidade de vértices e arestas de uma configuração para outra) no decorrer do tempo e o correspondente impacto dessa evolução no processamento do módulo desenvolvido.

Também foi avaliada a influência da quantidade de nós-clientes utilizados pelo *cluster* sobre o tempo de resposta necessário para manter consistente as informações de amizade de todos os usuários. Para atingir esse propósito, variou-se a quantidade de nós-clientes

utilizados no processamento de grafos contendo 10 mil e 100 mil usuários. Dessa forma, a quantidade total de nós-clientes utilizados nas simulações foram: 1 nó, 2 nós, 4 nós e os 8 nós-clientes disponíveis. Novamente, as métricas utilizadas foram as análises do tempo médio de processamento e da frequência de atualização das informações de cada nível, por unidade de tempo (e.g., minuto, hora e dia), para todos os usuários.

A Equação 6.1 foi utilizada para calcular o valor do Tempo Médio de processamento para cada nível de amizade, nos grafos processados pelo *cluster* (total de 10 repetições). Já as Equações 6.2, 6.3 e 6.4 foram utilizadas para calcular a frequência por unidade de tempo (minuto, hora e dia, respectivamente) de atualização dessas informações.

$$TempoMedio = \frac{\sum_{n=1}^{10} a_n}{n} \tag{6.1}$$

Onde,

- *TempoMedio* é o tempo médio de processamento em segundos para determinar cada nível dos grafos processados;
- a_n representa o tempo de processamento em segundos para cada repetição;
- *n* representa a quantidade total de repetições de cada simulação.

$$Freq/min = \frac{60}{TempoMedio} \tag{6.2}$$

Onde,

- *Freq/min* representa a quantidade máxima de atualizações que podem ser realizadas no intervalo de 1 minuto sobre as informações de cada nível dos grafos processados.

$$Freq/h = \frac{3.600}{TempoMedio} \tag{6.3}$$

Onde,

- *Freq/h* representa a quantidade máxima de atualizações que podem ser realizadas no intervalo de 1 hora sobre as informações de cada nível dos grafos processados.

$$Freq/dia = \frac{86.400}{TempoMedio} \tag{6.4}$$

Onde,

- *Freq/dia* representa a quantidade máxima de atualizações que podem ser realizadas no intervalo de 1 dia sobre as informações de cada nível dos grafos processados.

6.1.3 Apresentação dos Resultados

Os resultados das simulações estão sumarizados através das Tabelas 6.2, 6.3 e 6.4.

Tabela 6.2: Resultados do Processamento dos grafos, utilizando os 8 nós do *cluster*

# Usuários	Nível	Tempo Médio (s)	Freq/min	Freq/h	Freq/dia
10.000	1	2,29	26,26	1.575,49	37.811,82
	2	4,15	14,47	868,43	20.842,38
	3	11,12	5,39	323,64	7.767,27
	4	21,43	2,80	167,97	4.031,17
	5	46,37	1,29	77,63	1.863,17
	6	71,91	0,83	50,06	1.201,50
25.000	1	5,04	11,91	714,58	17.150
	2	10,51	5,71	342,47	8.219,18
	3	19,17	3,13	187,77	4.506,53
	4	66,43	0,90	54,19	1.300,57
	5	212,60	0,28	16,93	406,40
	6	400,29	0,15	8,99	215,84
50.000	1	9,77	6,14	368,62	8.846,93
	2	15,88	3,78	226,64	5.439,44
	3	34,59	1,73	104,09	2.498,06
	4	184,78	0,32	19,48	467,58

Continua na próxima página

Tabela 6.2 – continuação da página anterior

# Usuários	Nível	Tempo Médio (s)	Freq/min	Freq/h	Freq/dia
50.000	5	679,86	0,09	5,30	127,08
	6	1.453,13	0,04	2,48	59,46
75.000	1	12,73	4,71	282,76	6.786,26
	2	20,21	2,97	178,11	4.274,65
	3	51,24	1,17	70,26	1.686,28
	4	308,50	0,19	11,67	280,06
	5	1.294,97	0,05	2,78	66,72
	6	3.028,14	0,02	1,19	28,53
100.000	1	17,26	3,48	208,60	5.006,29
	2	25,70	2,33	140,06	3.361,48
	3	72,82	0,82	49,43	1.186,41
	4	466,33	0,13	7,72	185,28
	5	2.066,26	0,03	1,74	41,81
	6	5.110,02	0,01	0,70	16,91

Tabela 6.3: Análise da Escalabilidade do *cluster* para grafos contendo 10 mil usuários

# Máquinas	Nível	Tempo Médio (s)	Freq/min	Freq/h	Freq/dia
1	1	13,23	4,53	272,09	6.530,17
	2	19,54	3,07	184,26	4.422,24
	3	38,94	1,54	92,45	2.218,86
	4	131,58	0,46	27,36	656,64
	5	325,30	0,18	11,07	265.60
	6	525,20	0,11	6,85	164,51
2	1	7,73	7,76	465,63	11.175,06
	2	13,37	4,49	269,30	6.463,24
Continua na próxima página					

Tabela 6.3 – continuação da página anterior

# Máquinas	Nível	Tempo Médio (s)	Freq/min	Freq/h	Freq/dia
2	3	21,72	2,76	165,75	3.977,27
	4	71,09	0,84	50,64	1.215,41
	5	167,10	0,36	21,54	517,05
	6	265,34	0,23	13,57	325,62
4	1	4,08	14,69	881,66	21.159,87
	2	7,88	7,61	456,59	10.958,07
	3	15,75	3,81	228,52	5.484,57
	4	35,80	1,68	100,55	2.413,23
	5	88,25	0,68	40,79	979,03
	6	135,46	0,43	26,58	637,83
8	1	2,29	26,26	1.575,49	37.811,82
	2	4,15	14,47	868,43	20.842,38
	3	11,12	5,39	323,64	7.767,27
	4	20,52	2,92	175,40	4.209,59
	5	46,37	1,29	77,63	1.863,17
	6	71,91	0,83	50,06	1.201,50

Tabela 6.4: Análise da Escalabilidade do *cluster* para grafos contendo 100 mil usuários

# Máquinas	Nível	Tempo Médio (s)	Freq/min	Freq/h	Freq/dia
1	1	314,85	0,19	11,43	274,42
	2	325,35	0,18	11,06	265,56
	3	631,19	0,10	5,70	136,88
	4	3.695,09	0,02	0,97	23,38
	5	15.763,90	0,00	0,23	5,48
	6	42.290,79	0,00	0,09	2,04

Continua na próxima página

Tabela 6.4 – continuação da página anterior

# Máquinas	Nível	Tempo Médio (s)	Freq/min	Freq/h	Freq/dia
2	1	108,53	0,55	33,17	796,10
	2	128,03	0,47	28,12	674,82
	3	297,40	0,20	12,10	290,52
	4	1.790,57	0,03	2,01	48,25
	5	7.687,73	0,01	0,47	11,24
	6	20.029,05	0,00	0,18	4,31
4	1	37,30	1,61	96,52	2.316,49
	2	49,66	1,21	72,50	1.739,95
	3	139,97	0,43	25,72	617,30
	4	979,76	0,06	3,67	88,18
	5	4.178,72	0,01	0,86	20,68
	6	10.254,88	0,01	0,35	8,43
8	1	17,26	3,48	208,60	5.006,29
	2	25,70	2,33	140,06	3.361,48
	3	72,82	0,82	49,43	1.186,41
	4	466,33	0,13	7,72	185,28
	5	2.066,26	0,03	1,74	41,81
	6	5.110,02	0,01	0,70	16,91

6.1.4 Análise dos Resultados

Analisando os resultados da Tabela 6.2, observa-se que ao evoluir os grafos representando as conexões sociais dos usuários, da configuração inicial (i.e. grafos contendo características semelhantes àquelas da Configuração 1) até a configuração final (i.e. grafos contendo características semelhantes àquelas da Configuração 5), fez com que a frequência de atualização sobre as informações de cada nível diminuísse. Além disso, em todas as configurações analisadas, a frequência de atualização também diminuiu com o aumento da quantidade de níveis a serem processados, apresentando valor máximo para o primeiro nível e mínimo para

o sexto nível.

O resultado anterior, é decorrente do crescimento exponencial do tempo de processamento nos grafos quando aumenta-se a quantidade de níveis a serem processados, em todas as configurações. Esse resultado está fortemente relacionado à topologia *Scale-Free*, uma vez que grafos contendo essa topologia são caracterizados como esparsos, contendo uma pequena quantidade de arestas em relação a quantidade de vértices total. Portanto, quanto maior a quantidade de níveis a serem processados, mais tempo será necessário para processar grafos com essa topologia.

Por outro lado, os resultados obtidos sobre a variação da quantidade de nós do *cluster*, demonstram que à medida que são adicionados nós clientes, às configurações contendo 10 mil e 100 mil usuários, para realização do processamento distribuído das informações sociais dos usuários, a frequência de atualizações que podem ser realizadas sobre cada nível também aumenta.

Especificamente, os resultados presentes nas Tabelas 6.3 e 6.4 fornecem indícios de haver uma relação inversamente proporcional entre a quantidade de nós e o tempo de processamento. Ou seja, cada vez que a infraestrutura dobra a quantidade de nós utilizados pelo *cluster*, o tempo de processamento diminui na mesma proporção. Consequentemente, como a frequência de atualização das informações de cada nível, depende inversamente do tempo de processamento, uma diminuição no valor do último faz com que a frequência de atualização cresça na mesma proporção, conforme pode ser observado através da Figura 6.1. Esse fato demonstra que o algoritmo BFS, utilizado pelo Módulo de Expansão das Conexões Sociais, é paralelizável e que a carga de trabalho total está sendo corretamente distribuída entre os nós clientes.

De forma sucinta, ao aumentar a quantidade de vértices e arestas do grafo, sem variação da topologia, o tempo de processamento sofre um crescimento exponencial. Todavia, esse crescimento pode ser minimizado quando aumenta-se a quantidade de máquinas para o processamento do grafo. Por exemplo, nas simulações realizadas, o processamento de seis níveis em grafos representando 10 mil e 100 mil usuários, respectivamente, utilizando um *cluster* com oito nós, foi cerca de oito vezes mais rápido do que o tempo de processamento de apenas um nó para as mesmas configurações.

Os resultados obtidos podem ser analisados sobre diferentes cenários. Por exemplo, con-

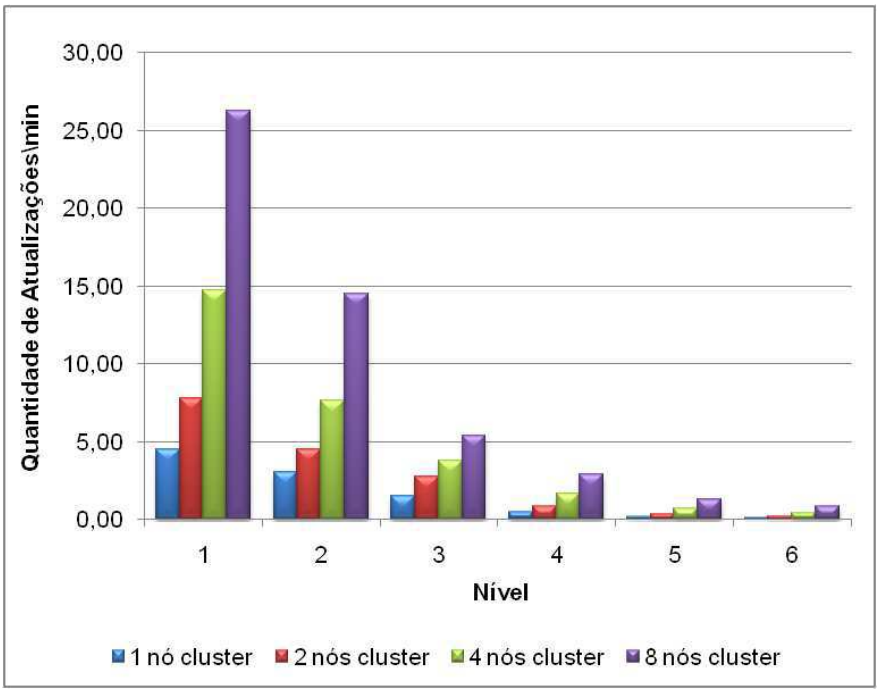


Figura 6.1: Frequência de atualização para grafos contendo 10 mil usuários.

sidere a utilização da aplicação MySocial (descrita na Seção 5.1) por parte dos usuários que utilizam a aplicação. Do ponto de vista desses usuários, a aplicação deve prover resposta em tempo real sobre as informações de localização dos contatos e amigos dos contatos. Possivelmente, se for utilizada uma escala graduando a importância por nível das informações de contexto social, os usuários indicarão que as informações mais relevantes são aquelas contidas nos primeiros níveis (i.e. informações sobre os amigos e amigos dos amigos), sendo as informações dos demais níveis consideradas menos importantes. Logo, é desejável que as informações dos primeiros níveis sejam atualizadas pela infraestrutura com uma frequência maior (e.g., a cada 1 minuto) do que as informações dos demais níveis.

Com base nos resultados apresentados anteriormente, observa-se que para atualizar as informações dos dois primeiros níveis em um intervalo de tempo de 1 minuto, para grafos contendo 10 mil usuários, é necessário utilizar apenas um dos nós do *cluster*. Todavia, à medida em que forem sendo adicionados novos usuários ao grafo e, consequentemente, novos contatos, até atingir a configuração contendo 100 mil usuários, passa a ser necessária a utilização de, pelo menos, 4 nós do *cluster* para processar essas informações.

Caso os usuários também desejem receber as atualizações de localização dos contatos de nível 3, a infraestrutura deve escalar a quantidade de nós clientes para a configuração de 8

nós do *cluster*, a fim de manter a frequência de atualização o mais próximo possível de uma atualização em tempo real.

Além disso, as informações dos demais níveis podem ser atualizadas a cada 1 hora e 30 minutos, 3 horas, 6 horas ou 12 horas (i.e. valores correspondentes a utilização de 8 nós, 4 nós, 2 nós e 1 nó, respectivamente), dependendo da frequência de atualização escolhida. Para a aplicação MySocial, uma boa estratégia pode ser privilegiar o processamento dos três primeiros níveis, atualizando as suas informações com uma frequência maior (e.g., utilizando 4 ou 8 nós do *cluster*) e processar os demais níveis com uma frequência menor (e.g., duas ou quatro vezes por dia). Dessa forma, aumentando a disponibilidade do *cluster*.

A estratégia de atualização descrita anteriormente, está alinhada com os resultados apresentados em [38]. Em seu trabalho, os autores analisaram a estrutura das redes sociais online e observaram que com o passar do tempo, essa estrutura sofre uma taxa de alteração bem menor do que aquela encontrada quando a rede social online está em expansão. De acordo com esse resultado, pode-se concluir que não é necessário recalcular as informações sociais dos usuários com uma frequência alta, uma vez que essas informações não sofrerão uma alteração significativa. Logo, é aceitável atualizar as informações sobre as conexões sociais de todos os usuários utilizando um intervalo de tempo de no máximo 24 horas.

Em outro cenário, onde as informações de contexto social são utilizadas, por exemplo, como entrada para algoritmos de recomendação, as informações dos três primeiros níveis de amizade calculadas previamente pela infraestrutura, não podem ser recalculadas em um intervalo de 1 minuto. Analisando os dados da Tabela 6.2, observa-se que, utilizando todo o poder computacional do *cluster* disponível, é possível atualizar as informações de recomendação dos três primeiros níveis para todos os usuários, entre 1 minuto e 30 segundos a 2 minutos. Caso o algoritmo de recomendação, utilize as informações dos seis níveis de amizade de todos os usuários, é possível realizar esse processamento a cada 1 hora e 30 minutos.

Por fim, durante a realização das simulações foi observado um alto consumo de memória por parte dos nós-clientes, conforme sumarizado na Tabela 6.5. Esse resultado está diretamente relacionado com as características do algoritmo BFS, o qual utiliza uma estratégia gulosa (*Greedy*) para determinar cada nível do grafo. Isto é, durante o processamento do vértice corrente, o algoritmo armazena em memória o conjunto de vértices de sua vizi-

nhança, bem como, os vizinhos dos vizinhos do vértice corrente. Contribui ainda para esse resultado, o fato de os clientes receberem uma cópia do grafo completo para realizarem o processamento da sua sub-lista de trabalho. Destaque para o processamento de seis níveis realizado em grafos contendo 100 mil vértices e aproximadamente 158 mil arestas. Para cada um desses grafos, o consumo de memória atingiu uma média de 2,6 GB e máximo de 3,6 GB para alguns nós do *cluster*. Esse último resultado impossibilitou que fossem processados seis níveis para grafos contendo quantidade de vértices e arestas superiores àquelas da Configuração 5, uma vez que cada nó do *cluster* dispõe de no máximo 4 GB de memória física.

Tabela 6.5: Utilização de Memória pelos nós clientes do *cluster*

# Vértices	**# Arestas**	**Nível**	**Consumo Médio**	**Consumo Máximo**
10.000	~15,8 mil	1	500 MB	1,0 GB
10.000	~15,8 mil	6	700 MB	1,2 GB
100.000	~158 mil	1	1,6 GB	2,1 GB
100.000	~158 mil	6	2,6 GB	3,6 GB

6.2 Conclusões do Capítulo

Neste capítulo foi realizada uma avaliação da escalabilidade do Módulo de Expansão das Conexões Sociais utilizando simulação. Essa análise teve como objetivo específico avaliar o tempo de resposta necessário para que o Módulo de Expansão das Conexões Sociais calculasse e mantivesse consistente as informações de cada nível de amizade (máximo de 6 níveis) para todos os usuários, após alterações no grafo de amizades. Essa análise também levou em consideração a influência da quantidade de nós-clientes utilizados pelo *cluster*.

Capítulo 7

Considerações Finais

Nos últimos anos, têm ocorrido diversos avanços nas tecnologias necessárias para viabilização do paradigma de Computação Pervasiva. Como consequência desse fato, tem sido presenciado o surgimento das primeiras aplicações e cenários de Computação Pervasiva. Especificamente, para o domínio de Redes Sociais Móveis, várias soluções têm sido propostas para auxiliar os desenvolvedores na construção de aplicações para esse domínio.

Apesar dessas soluções fornecerem mecanismos para acessar as informações de contexto social e de localização dos usuários e executar um conjunto de serviços comuns às aplicações, não eram fornecidas funcionalidades que aumentassem o conhecimento prévio sobre as informações de contexto social dos usuários. Como por exemplo, conhecimento sobre os amigos diretos e indiretos dos usuários, em vários níveis de amizade.

Neste trabalho foi apresentada uma infraestrutura para o desenvolvimento de aplicações pervasivas cientes de Redes Sociais. A infraestrutura foi desenvolvida com base em uma Arquitetura Orientada a Serviços, sendo implantada através de um servidor centralizado, integrando-se através de *plugins* com as Redes Sociais Online, das quais os usuários que utilizam a infraestrutura fazem parte. Além disso, foi desenvolvida uma API baseada em *Web Service* que fornece acesso às informações de contexto social e de localização dos usuários, assim como, aos serviços disponibilizados pela infraestrutura. Ademais, como diferencial em relação às soluções encontradas na literatura, foi definido (Seção 4.2.3) e implementado (Seção A.1.3) um módulo que realiza o processamento distribuído sobre as conexões sociais dos usuários extraídas das redes sociais online, com o propósito de expandir o conhecimento prévio da infraestrutura sobre as informações de contexto social persistidas.

Como formas de validação da infraestrutura, foi desenvolvido um estudo de caso utilizando a API e os serviços disponibilizados pela infraestrutura, assim como, realizou-se uma avaliação do Módulo de Expansão das Conexões Sociais utilizando simulação.

Como estudo de caso, foi construída uma aplicação baseada em localização denominada MySocial, que se integra através de *plugins* às Redes Sociais Online Orkut e MySpace. Com isso, foi demonstrado o suporte da infraestrutura para o desenvolvimento de aplicações para Redes Sociais Móveis.

Na validação do Módulo de Expansão das Conexões Sociais disponibilizado pela infraestrutura, foi avaliada a sua escalabilidade. Para isso, foram realizadas simulações de grafos contendo cinco configurações distintas (Seção 6.1.2), com topologia análoga àquela presente nas Redes Sociais Online, com o propósito de avaliar o tempo de resposta necessário para que o módulo calculasse e mantivesse consistente as informações de cada nível de amizade (máximo de 6 níveis) para todos os usuários, após alterações no grafo de amizades. A análise também levou em consideração a influência da quantidade de nós-clientes utilizados pelo *cluster*.

Especificamente, observou-se que ao aumentar a quantidade de vértices e arestas do grafo, sem variação da topologia, o tempo de processamento sofre um crescimento exponencial e a frequência de atualização das informações presentes em cada nível diminui. Contudo, os resultados obtidos sobre a influência da quantidade de nós-clientes utilizados pelo *cluster* fornecem indícios de haver uma relação inversamente proporcional entre a quantidade de nós e o tempo de processamento. Dessa forma, tanto o crescimento exponencial pode ser minimizado, quanto a frequência de atualização das informações de cada nível pode ser mantida/aumentada com o acréscimo de novos nós ao *cluster* para processamento do grafo.

Portanto, para as configurações avaliadas, o Módulo de Expansão das Conexões Sociais pode ser utilizado de forma satisfatória para determinar até seis níveis de amizade para todos os usuários de uma Rede Social Móvel.

7.1 Contribuições

As principais contribuições deste trabalho são enumeradas a seguir:

- Especificação de uma infraestrutura que se integra através de *plugins* com as Redes

Sociais Online, armazenando as informações de contexto social e de localização dos usuários;

- Especificação e implementação de um Módulo de Expansão das Conexões Sociais dos usuários, que utiliza uma abordagem baseada em grafos, em conjunto com o algoritmo BFS, para solucionar o *problema de caminhos mais curtos de todos os pares*. Ademais, o módulo desenvolvido permite que novas aplicações para Redes Sociais Móveis possam ser construídas, uma vez que as informações sobre as conexões sociais (no máximo seis níveis de amizade) de todos os usuários são conhecidas *a priori* e podem ser reutilizadas, por exemplo, como entrada para serviços de Recomendação de amizade;

- Desenvolvimento e disponibilização de API baseada em *Web Service* para acessar as informações de contexto social e de localização dos usuários, bem como, para executar as funcionalidades providas pela infraestrutura;

- A partir da utilização dos serviços fornecidos pela infraestrutura desenvolvida neste trabalho, o processo de desenvolvimento de aplicações pervasivas para Redes Sociais Móveis tornar-se-á menos complexo e tornará as aplicações menos propensas a erros durante a fase de implementação. Uma vez que os desenvolvedores não mais necessitarão re-implementar funcionalidades comuns a maioria das aplicações para esse domínio.

7.2 Limitações e Trabalhos Futuros

Algumas questões que não foram foco deste trabalho, mas que devem ser consideradas em trabalhos futuros, são a privacidade e a segurança das informações de contexto dos usuários armazenadas pela infraestrutura. Uma vez que essas informações são muito sensíveis e são consideradas o *core business* de qualquer solução de *middleware* para Redes Sociais Móveis, faz-se necessário empregar mecanismos, técnicas e/ou políticas de segurança que garantam a proteção e o sigilo dessas informações. Tanto daquelas armazenadas pela infraestrutura, quanto daquelas que forem trafegadas pela rede (e.g., durante a comunicação com a aplicação móvel e/ou *plugin* instalado na Rede Social Online) e que possam ser acessadas sem

autorização dos usuários, ou mesmo, interceptadas por entidades maliciosas para identificar e/ou causar quaisquer danos aos usuários.

Ainda, faz-se necessário estudar mecanismos para remover ambiguidades, ou mesmo, conflitos sobre as informações de contexto dos usuários. Uma vez que essas informações podem estar espalhadas entre as diversas Redes Sociais Online das quais os usuários sejam membros, ou ainda, podem apresentar-se conflitantes entre si. Uma abordagem promissora para resolução desse problema é a utilização de ontologias de domínios, em conjunto com técnicas para reconciliação ontológica [18, 33].

Como trabalhos futuros, pretende-se realizar uma análise estatística sobre os resultados obtidos no *Capítulo 6*, resultantes do processamento do Módulo de Expansão das Conexões Sociais. Os objetivos dessa análise serão avaliar o impacto do conjunto de variáveis e fatores envolvidos para resolução do *problema de caminhos mais curtos de todos os pares* e construir um modelo estatístico representativo para esse problema. Dessa forma, a partir desse modelo espera-se generalizar os resultados obtidos para grafos contendo uma quantidade de usuários semelhante àquelas encontradas nas Redes Sociais Online.

Além disso, pode-se estudar o acoplamento à infraestrutura de um módulo que realize a integração com as informações de contexto social advindas da abordagem *ad hoc*. Algumas soluções como aquelas apresentadas em [25], ou ainda, em [8] podem ser utilizadas para prover tal integração.

Por fim, pode-se avaliar a utilização de técnicas de Particionamento de Grafos [52, 61], com o objetivo de minimizar a quantidade de dados transferidos pela rede e otimizar a quantidade de memória utilizada no processamento dos nós clientes. Uma vez que, na implementação realizada, o servidor instancia o grafo e o transfere através da rede para cada nó cliente. Dessa forma, fazendo com que os clientes possuam conhecimento do grafo completo. Adicionalmente, também pode-se avaliar a utilização do algoritmo de *Johnson* no Módulo de Expansão das Conexões Sociais, pois, de acordo com [16], esse algoritmo mostra-se mais eficiente para o processamento de grafos esparsos.

Bibliografia

[1] Hyggo Oliveira de Almeida. COMPOR - Desenvolvimento de Software para Sistemas Multiagentes. Dissertação de mestrado, Universidade Federal de Campina Grande, Campina Grande, Paraíba - Brasil, Março 2004.

[2] Deepak Alur, Dan Malks, and John Crupi. *Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition)*. Prentice Hall, 2 edition, 2003.

[3] Tom Anderson. MySpace. `http://www.myspace.com/`, August 2003. Último acesso Novembro 2010.

[4] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286(5439):509–512, October 1999.

[5] Aaron Beach, Mike Gartrell, Sirisha Akkala, Jack Elston, John Kelley, Keisuke Nishimoto, Baishakhi Ray, Sergei Razgulin, Karthik Sundaresan, Bonnie Surendar, Michael Terada, and Richard Han. Whozthat? evolving an ecosystem for context-aware mobile social networks. *IEEE Network*, 22(4):50–55, 2008.

[6] Dave Beckett. RDF/XML Syntax Specification (revised) W3C Recommendation 10 february 2004. `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`, February 2004. Último acesso Novembro 2010.

[7] Sonia Ben Mokhtar and Licia Capra. From pervasive to social computing: algorithms and deployments. In *ICPS '09: Proceedings of the 2009 international conference on Pervasive services*, pages 169–178, New York, NY, USA, 2009. ACM.

[8] C. Boldrini, M. Conti, F. Delmastro, and A. Passarella. Context- and social-aware middleware for opportunistic networks. *Journal of Network and Computer Applications*, 33(5):525–541, 2010. Middleware Trends for Network Applications.

[9] Cristian Borcea, Ankur Gupta, Achir Kalra, Quentin Jones, and Liviu Iftode. The mobisoc middleware for mobile social computing: challenges, design, and early experiences. In *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[10] Dario Bottazzi, Rebecca Montanari, and Alessandra Toninelli. Context-Aware Middleware for Anytime, Anywhere Social Networks. *IEEE Intelligent Systems*, 22(5):23–32, 2007.

[11] Danah Boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1-2), November 2007.

[12] Dan Brickley and Libby Miller. FOAF Vocabulary Specification 0.98. `http://xmlns.com/foaf/spec/20100809.html`, August 2010. Último acesso Novembro 2010.

[13] Orkut Buyukkokten. Orkut. `http://www.orkut.com/`, January 2004. Último acesso Novembro 2010.

[14] Steve Chen, Chad Hurley, and Jawed Karim. YouTube. `http://www.youtube.com/`, February 2005. Último acesso Novembro 2010.

[15] Elizabeth F. Churchill and Christine A. Halverson. Guest Editors' Introduction: Social Networks and Social Networking. *IEEE Internet Computing*, 9(5):14–19, 2005.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[17] Cristiano Andre da Costa, Adenauer Correa Yamin, and Claudio Fernando Resin Geyer. Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, 7(1):64–73, 2008.

[18] Patrício de Alencar Silva. PERSONÆ: Um Modelo de Mediação Semântica para Arquiteturas Orientadas a Serviços. Dissertação de mestrado, Universidade Federal de Campina Grande, Campina Grande, Paraíba - Brasil, Maio 2007.

[19] Anind K. Dey. Understanding and Using Context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[20] Jack Dorsey, Evan Williams, and Biz Stone. Twitter. `http://www.twitter.com`, July 2006. Último acesso Novembro 2010.

[21] Nathan Eagle and Alex Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.

[22] Nathan Eagle, Alex S. Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, August 2009.

[23] Nathan Eagle and Alex (Sandy) Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268, 2006.

[24] Steven K. Feiner. Augmented reality: A new way of seeing. *Scientific American*, 286(4):48–55, 2002.

[25] Glauber V. Ferreira. Infra-estrutura de Software Baseada em Componentes para a Construção de Aplicações para Comunidades Virtuais Móveis. Dissertação de mestrado, Universidade Federal de Campina Grande, Campina Grande, Paraíba - Brasil, Outubro 2006.

[26] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[27] Google. Open Social. `http://code.google.com/apis/opensocial/`, November 2007. Último acesso Janeiro 2010.

[28] Google. Social Graph API. `http://code.google.com/intl/pt-BR/apis/socialgraph/`, February 2008. Último acesso Agosto 2010.

[29] Google. Google Latitude. `http://www.google.com/latitude`, February 2009. Último acesso Janeiro 2010.

[30] Ankur Gupta, Achir Kalra, Daniel Boston, and Cristian Borcea. Mobisoc: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14(1):35–52, 2009.

[31] Ankur Gupta, Sanil Paul, Quentin Jones, and Cristian Borcea. Automatic identification of informal social groups and places for geo-social recommendations. *International Journal of Mobile Network Design and Innovation*, 2(3/4):159–171, 2007.

[32] Josef Hallberg, Mia Backlund Norberg, Johan Kristiansson, Kåre Synnes, and Chris Nugent. Creating dynamic groups using context-awareness. In *MUM '07: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 42–49, New York, NY, USA, 2007. ACM.

[33] Adil Hameed, Alun D. Preece, and Derek H. Sleeman. Ontology reconciliation. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 231–250. Springer, 2004.

[34] Albert Held, Sven Buchholz, and Alexander Schill. Modeling of context information for pervasive computing applications. In *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002), Orlando, Florida*, 2002.

[35] Reid Hoffman. LinkedIn. `http://www.linkedin.com/`, May 2003. Último acesso Novembro 2010.

[36] Q. Jones and S.A. Grandhi. P3 systems: putting the place back into social networks. *Internet Computing, IEEE*, 9(5):38 – 46, sept.-oct. 2005.

[37] Jon Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 163–170, New York, NY, USA, 2000. ACM.

[38] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference*

on Knowledge discovery and data mining, KDD '06, pages 611–617, New York, NY, USA, 2006. ACM.

[39] Axel Kupper. *Location-based Services: Fundamentals and Operation*. John Wiley & Sons, 2005.

[40] Nan Li and Guanling Chen. Analysis of a location-based social network. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, pages 263–270, Washington, DC, USA, 2009. IEEE Computer Society.

[41] Nan Li and Guanling Chen. Multi-layered friendship modeling for location-based mobile social networks. In *Mobile and Ubiquitous Systems: Networking Services, MobiQuitous, 2009. MobiQuitous '09. 6th Annual International*, pages 1 –10, 13-16 2009.

[42] Emerson Loureiro, Glauber Ferreira, Hyggo Almeida, and Angelo Perkusich. Pervasive Computing: What is it Anyway? In M. Lytras and A. Naeve, editors, *Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to their full potential*, chapter 1, pages 9–36. Idea Group Inc, Hershey, PA, USA, 2007.

[43] Daryl Mun-kid Low, Ronald Keryuan Huang, Puneet Mishra, Gaurav Jain, Jason Gosnell, and Jeff Bush. Group Formation Using Anonymous Broadcast Information. http://www.freepatentsonline.com/y2010/0070758.html, March 2010. United States Patent Application.

[44] Udi Manber. *Introduction to Algorithms: A Creative Approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[45] Stanley Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[46] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350, New York, NY, USA, 2008. ACM.

[47] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[48] Sonia Ben Mokhtar, Liam McNamara, and Licia Capra. A middleware service for pervasive social networking. In *M-PAC '09: Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, pages 1–6, New York, NY, USA, 2009. ACM.

[49] Ghita Kouadri Mostéfaoui, Jacques Pasquier-Rocha, and Patrick Brézillon. Context-aware computing: A guide for the pervasive computing community. *Pervasive Services, IEEE/ACS International Conference on*, 0:39–48, 2004.

[50] Anastasios Noulas, Mirco Musolesi, Massimiliano Pontil, and Cecilia Mascolo. Inferring Interests from Mobility and Social Interactions. In *Proceedings of the NIPS Workshop on Analyzing Networks and Learning with Graphs*, Whistler, Canada, December 2009.

[51] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. Mobiclique: middleware for mobile social networking. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 49–54, New York, NY, USA, 2009. ACM.

[52] Josep M. Pujol, Vijay Erramilli, Georgos Siganos, Xiaoyuan Yang, Nikos Laoutaris, Parminder Chhabra, and Pablo Rodriguez. The little engine(s) that could: scaling online social networks. *SIGCOMM Comput. Commun. Rev.*, 40:375–386, August 2010.

[53] Daniele Quercia and Licia Capra. Friendsensing: recommending friends using mobile phones. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 273–276, New York, NY, USA, 2009. ACM.

[54] Daniele Quercia, Jonathan Ellis, and Licia Capra. Using mobile phones to nurture social networks. *IEEE Pervasive Computing*, 9(3):12–20, July 2010.

[55] Juwel Rana, Johan Kristiansson, Josef Hallberg, and Kåre Synnes. An Architecture for Mobile Social Networking Applications. In *CICSYN '09: Proceedings of the 2009 First International Conference on Computational Intelligence, Communication Systems and Networks*, pages 241–246, Washington, DC, USA, 2009. IEEE Computer Society.

[56] Bharat Rao and Louis Minakakis. Evolution of mobile location-based services. *Communications of the ACM*, 46(12):61–65, 2003.

[57] Howard Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier*. MIT Press, Cambridge, MA, USA, 2000.

[58] Debashis Saha and Amitava Mukherjee. Pervasive Computing: A Paradigm for the 21st Century. *Computer*, 36(3):25–31, 2003.

[59] Thiago Bruno Melo de Sales. Especificação baseada no padrão UPnP para autenticação e autorização de usuários em ambientes de computação pervasiva. Dissertação de mestrado, Universidade Federal de Campina Grande, Campina Grande, Paraíba - Brasil, Março 2010.

[60] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8:10–17, 2001.

[61] Kirk Schloegel, George Karypis, and Vipin Kumar. Sourcebook of parallel computing. chapter Graph partitioning for high-performance scientific simulations, pages 491–541. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[62] Alessandra Toninelli, Animesh Pathak, Amir Seyedi, Roberto Speicys Cardoso, and Valerie Issarny. Middleware Support for Mobile Social Ecosystems. In IEEE Computer Society Press, editor, *Second IEEE International Workshop on Middleware Engineering (COMPSAC Workshops)*, Seoul Korea, Republic Of, 05 2010.

[63] Marco von Arb, Matthias Bader, Michael Kuhn, and Roger Wattenhofer. VENETA: Serverless Friend-of-Friend Detection in Mobile Social Networking. In *WIMOB '08: Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication*, pages 184–189, Washington, DC, USA, 2008. IEEE Computer Society.

[64] Mark Weiser. The Computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991.

[65] Kin Yeung Wong. NAN: Near-me Area Network. *IEEE Internet Computing*, 99(Pre-Prints), 2010.

[66] Mark Zuckerberg, Chris Hughes, Dustin Moskovitz, and Eduardo Saverin. Facebook. `http://www.facebook.com/`, February 2004. Último acesso Novembro 2010.

Apêndice A

Modelagem da Infraestrutura

A.1 Arquitetura

Nessa seção apresenta-se a modelagem dos módulos da infraestrutura. Especificamente, descrevem-se as funcionalidades providas pelas classes presentes em cada módulo.

A.1.1 Web Service

- *ProfileServiceWS*: classe responsável pela comunicação entre as aplicações desenvolvidas com a infraestrutura. Seus métodos disponibilizam o acesso aos serviços fornecidos pela API da infraestrutura . Os métodos disponibilizados são: *addFriendship*, *registerUser*, *doLogin*, *registerLocation*, *getNearByUsers*, *getCurrentLocation*, *getFriendsFrom*, *getFriendsAtLevel*, *getMembersFrom*, *getSocialNetworksFrom*, *getUserInfo* e *getUserPreferences*. Ver Tabela 4.2 para descrição dos serviços.

A.1.2 Módulo Online

- *FriendManager*: classe responsável por gerenciar as informações sobre as conexões sociais entre os usuários. Especificamente, essa classe se conecta com as Redes Sociais Online e extrai as informações sobre os usuários e seus amigos. Os principais métodos são: *addFriendship*, *fetchFriends*, *fetchPerson*, *getFriends* e *getFriendsAtLevel*. O método *addFriendship* relaciona dois usuários como sendo amigos para uma Rede Social. Já o método *fetchFriends* recupera a informação sobre a lista de amigos dos usuários

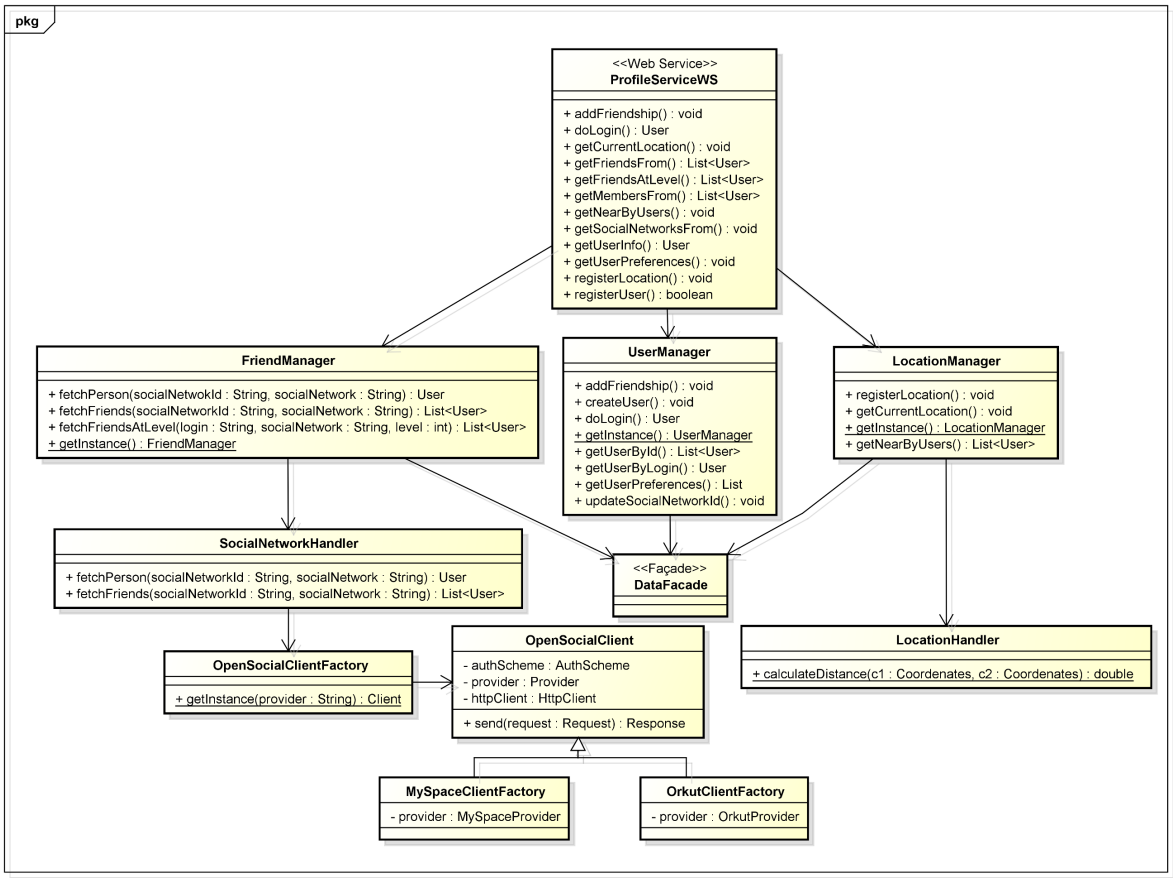


Figura A.1: Diagrama de classes da camada de serviços

das Redes Sociais Online e as armazena no mecanismo de armazenamento persistente da infraestrutura. Essas informações podem ser recuperadas de forma detalhada (i.e. contendo os dados preenchidos pelos usuários em seu perfil das Redes Sociais), ou contendo apenas os dados obrigatórios (e.g., primeiro nome, último nome, identificador OpenSocial, entre outros). O método *fetchPerson* extrai as informações sobre um único usuário diretamente das Redes Sociais e armazena essa informação na base de dados da infraestrutura. Por fim, os métodos *getFriends* e *getFriendsAtLevel* recuperam da camada de dados todos os amigos do usuário (amigos diretos e indiretos) e os amigos do usuário do nível especificado, respectivamente;

- *UserManager*: classe responsável por gerenciar as informações sobre os usuários da Rede Social Móvel. Os seus principais métodos são: *createUser*, *doLogin*, *getUserByLogin*, *getUserById* e *getUserPreferences*. O método *createUser)* cadastra um novo usuário na infraestrutura. Já o método *doLogin* realiza o processo de autenticação do

usuário na infraestrutura. Os métodos *getUserByLogin* e *getUserById* recuperam as informações do usuário da camada de dados utilizando a informação de *login* do usuário e *OpenSocialId*, respectivamente. Por fim, *getUserPreferences* retorna uma lista contendo as preferências do usuário;

- *LocationManager*: classe responsável por gerenciar as informações sobre a localização dos usuários da Rede Social Móvel. Os seus principais métodos são: *getCurrentLocation*, *getNearByUsers*, *registerLocation*. O método *getCurrentLocation* recupera da camada de dados a informação sobre a última localização do usuário registrada pela infraestrutura. Já *getNearByUsers* recupera a informação sobre os usuários que estão dentro do raio de distância em metros, no intervalo de tempo inicial e final passados como parâmetro (e.g., considerando os últimos 5 minutos, quais usuários estão a no máximo 900 metros de distância da posição corrente do usuário). Por fim, o método *registerLocation* armazena a localização geográfica corrente do usuário (e.g., valores de latitude e longitude);

- *SocialNetworkHandler*: classe responsável por recuperar das Redes Sociais Online as informações sobre os usuários e seus amigos. Os principais métodos são: *fetchPerson* e *fetchFriends*, que recuperam as informações sobre um usuário e os amigos dos usuários, através do identificador *OpenSocial*, respectivamente;

- *LocationHandler*: classe responsável pelo calculo da distância entre duas coordenadas geográficas. Seu principal método é o *calculateDistance* que recebe como parâmetros duas instâncias da classe *Coordenates*, representando a coordenada de origem e a destino, respectivamente. O retorno da execução do método é a distância em metros entre as duas coordenadas;

- *OpenSocialClientFactory*: classe responsável pela criação de instâncias específicas da classe *OpenSocialClient*, que acessa as informações do usuário das Redes Sociais Online através de requisições utilizando o método *send*. As classes *MySpaceClientProvider* e *OrkutClientProvider* disponibilizam clientes específicos, para acessar as informações das Redes Sociais MySpace [3] e Orkut [13], respectivamente. *OpenSocialClientFactory* implementa os padrões de projetos *Strategy* [26] e *Singleton* [26]

para criar instâncias específicas das classes *MySpaceClientProvider* e *OrkutClientProvider*, que por sua vez implementam o padrão de projeto *Factory Method* [26].

A.1.3 Módulo de Expansão das Conexões Sociais

Nesta seção são apresentadas as principais classes envolvidas no Módulo de Expansão das Conexões Sociais dos usuários, disponibilizado pela infraestrutura. O diagrama de classes do módulo está representado através da Figura A.2. A seguir são descritas cada uma das classes contidas no diagrama e suas respectivas responsabilidades.

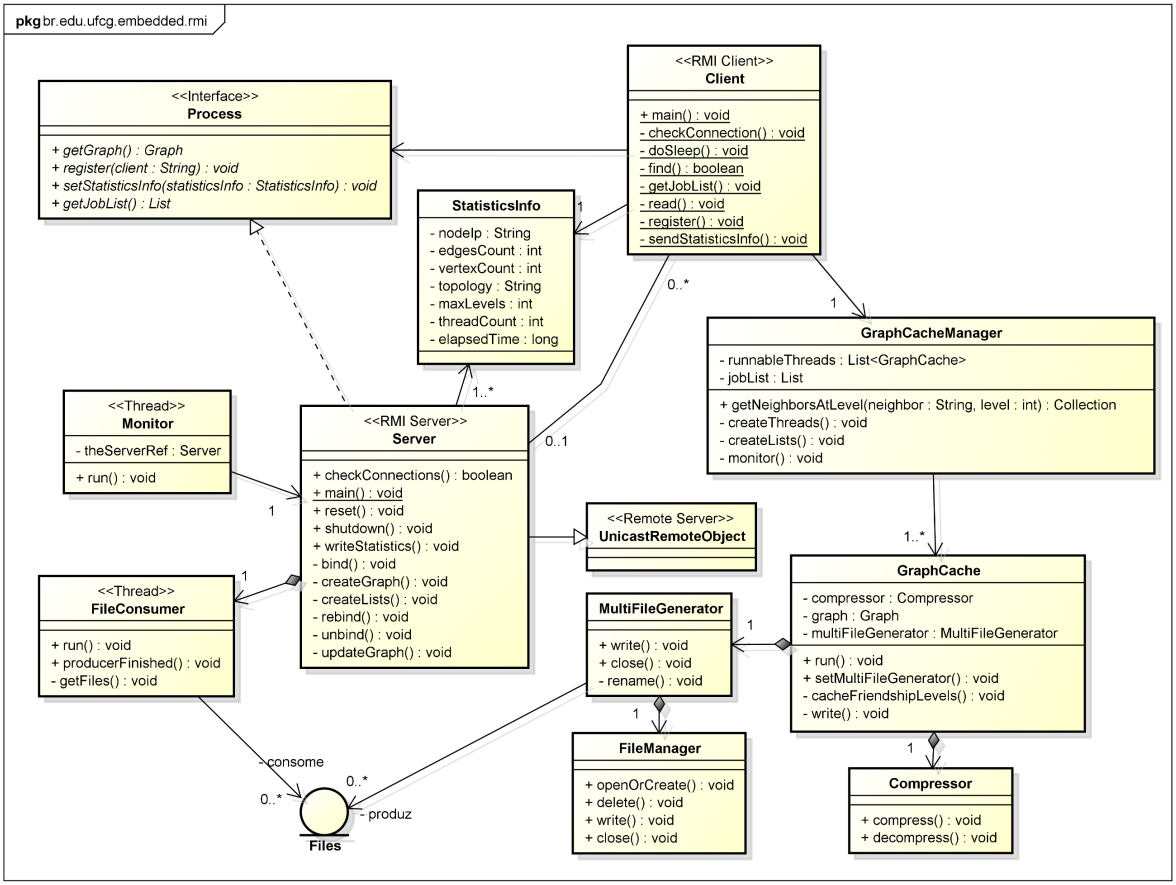


Figura A.2: Módulo de Expansão das Conexões Sociais

- *Process*: *interface* que define os métodos remotos representando as funcionalidades disponibilizadas aos clientes, que devem ser implementadas pela classe representando o servidor. Dentre os métodos disponibilizados, destacam-se: *getGraph*, *getJobList*, *register* e *setStatisticsInfo*. O método *register* registra um novo cliente junto ao servidor. Já o método *getGraph* retorna para o(s) cliente(s) o grafo completo representando

as conexões sociais existentes entre todos os usuários da Rede Social Móvel, armazenadas no mecanismo de armazenamento persistente da Infraestrutura. Por sua vez, o método *getJobList* retorna para o(s) cliente(s) um conjunto de vértices[1] existentes no grafo das conexões sociais, que devem ser processados por ele(s). Por fim, o método *setStatisticsInfo* recebe um objeto contendo o resultado do processamento realizado pelo(s) cliente(s);

- *Server*: classe que representa o servidor responsável por: construir e disponibilizar o grafo representando as conexões sociais dos usuários, gerenciar a comunicação com os clientes, dividindo a carga de trabalho de forma proporcional à quantidade de clientes conectados. Essa classe implementa os métodos da interface *Process*, disponibilizando aos clientes as funcionalidades ali definidas e explicadas anteriormente. Adicionalmente, a classe *Server* implementa outros métodos, dentre eles destacam-se: *checkConnections*, *shutdown* e *writeStatistics*. O método *checkConnections* é utilizado pela classe *Monitor* para que esta informe ao servidor sobre o término do processamento dos clientes. Já o método *shutdown* é utilizado para finalizar o servidor, encerrando todas as conexões abertas com os clientes. Por fim, o método *writeStatistics* é utilizado para escrever em arquivo os resultados dos processamentos realizados por cada cliente;

- *Client*: classe que representa um cliente, cujo objetivo é receber o grafo disponibilizado pelo servidor, em conjunto com a lista de trabalho a ser processada, e gerenciar o processamento dessa lista de trabalho. Por fim, esta classe deve enviar o resultado do processamento realizado ao servidor. Os métodos *read*, *getJobList*, *register* e *sendStatisticsInfo* possuem a mesma semântica dos métodos disponibilizados pela *interface Server*, os quais foram explicados anteriormente. Já os métodos *doSleep* e *checkConnection* são utilizados para realizar a sincronização entre o(s) cliente(s) e o servidor. Uma vez que as classes *Server* e *Client* são executadas em processos e/ou máquinas distintas. Assim, o método *doSleep* faz com que o cliente execute em espera ocupada, até que o servidor tenha disponibilizado o grafo, em conjunto com a(s) lista(s) de trabalho, para o(s) cliente(s). Já o método *checkConnection* verifica se o servidor está

[1] Cada vértice do grafo representa um usuário do sistema

executando e pronto para receber uma nova conexão;

- *GraphCacheManager*: essa classe é responsável por gerenciar o processamento concorrente da lista de trabalho recebida pela classe *Client*. Seus principais métodos são: *createLists*, *createThreads* e *monitor*. O método *createThreads* instância uma lista de objetos da classe *GraphCache*, os quais processarão a lista de trabalho de forma concorrente. Já o método *createLists* divide a lista de trabalho em sub-listas, que serão processadas pela lista de objetos da classe *GraphCache*. Por fim, o método *monitor* verifica em espera ocupada o término da execução do processamento realizado pelas *threads*;

- *GraphCache:* essa classe é responsável por processar todos os vértices contidos na sub-lista de trabalho sob sua responsabilidade. O processamento ocorre da seguinte forma, dentro do escopo do método *cacheFriendshipLevels*, para cada vértice da sub-lista de trabalho, é executado uma instância do algoritmo BFS (implementado pela biblioteca JGraphT), onde o vértice de origem é o vértice corrente e o comprimento do caminho a ser percorrido é limitado, com base em parâmetro definido. Após o término de cada processamento, o resultado obtido é transformado em uma representação XML ou JSON (e.g., transformação realizada através da biblioteca XStream[2]), sendo compactado (método *compress* da classe *Compressor*) para otimizar a quantidade de dados utilizados, e escrito em um ou mais arquivos (método *write* das instâncias de *MultiFileGenerator* e *FileManager*) para serem persistidos pela *Camada de Dados* da infraestrutura;

- *Monitor*: essa classe é responsável por monitorar as conexões estabelecidas com os clientes, informando ao servidor quando eles tiverem finalizado o processamento de suas cargas de trabalho. O comportamento descrito anteriormente é implementado pelo método *run*, que executa em espera ocupada o método *checkConnections*, implementado pelo servidor;

- *FileConsumer*: essa classe é responsável por consumir os arquivos produzidos como resultado do processamento realizado pelo(s) cliente(s). Seus principais métodos são:

[2]Disponível em: `http://xstream.codehaus.org/download.html`

run, *getFiles* e *producerFinished*. O método *run* executa em espera ocupada o método *getFiles*, que recupera os arquivos produzidos durante o processamento realizado pelos clientes e os consome. O processo de consumir os arquivos, armazenando-os em um mecanismo persistente, continua até que o método *producerFinished* seja executado pelo Servidor. Isto ocorre após a instância da classe *Monitor* informar à instância de *Server* o término do processamento do(s) cliente(s).

A.1.4 Camada de Dados

Nesta seção são apresentadas as principais classes presentes no projeto da Camada de Dados. Essas classes implementam o padrão de projeto DAO [2] para realizar as operações sob sua responsabilidade sobre o mecanismo de persistência. O diagrama de classes do módulo está representado pela Figura A.3. A seguir são descritas cada uma das classes contidas no diagrama e suas respectivas responsabilidades.

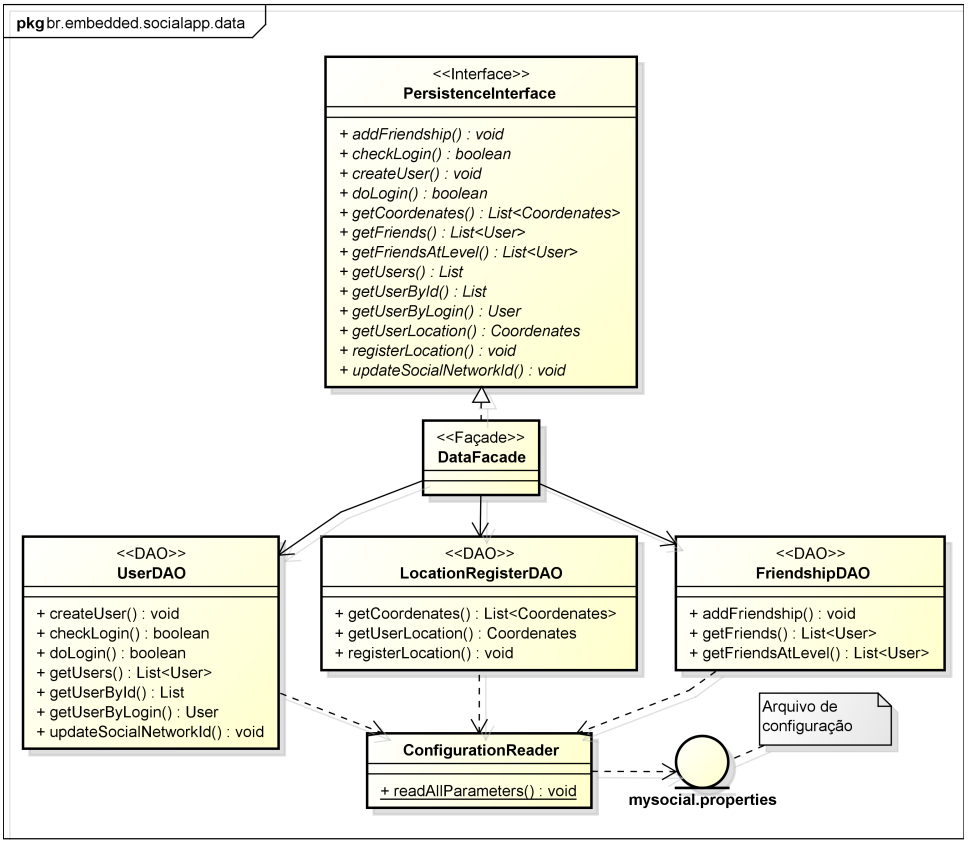


Figura A.3: Diagrama de classes da camada de dados

- *PersistenceInterface*: interface que define os métodos da Camada de Dados. Os principais métodos definidos são: *addFriendship*, *checkLogin*, *createUser*, *doLogin*, *getCoordenates*, *getFriends*, *getFriendsAtLevel*, *getUsers*, *getUserById*, *getUserByLogin*, *getUserLocation*, *registerLocation*, *updateSocialNetworkId*;

- *DataFacade*: classe que implementa o padrão de projeto *Façade* [26]. Essa classe disponibiliza às camadas superiores acesso transparente aos métodos da Camada de Dados;

- *UserDAO*: classe que implementa o padrão de projeto DAO (*Data Access Object*) [2] e abstrai as operações sobre os usuários no mecanismo de persistência da infraestrutura. Os principais métodos disponibilizados são: *checkLogin*, *createUser*, *doLogin*, *getUsers*, *getUserById*, *getUserByLogin* e *updateSocialNetworkId*. O método *checkLogin* verifica a existência de um usuário no sistema com o identificador fornecido. Já o método *createUser* adiciona um novo usuário à infraestrutura. O método *doLogin* autentica o usuário utilizando login e senha fornecidos. *getUsers* recupera todos os usuários registrados na infraestrutura. Por sua vez, os métodos *getUserById*, *getUserByLogin* recuperam as informações sobre um usuário através do identificador OpenSocial e login, respectivamente. Já o método *updateSocialNetworkId* adiciona/atualiza a informação sobre o identificador OpenSocial do usuário;

- *LocationRegisterDAO*: classe que implementa o padrão de projeto DAO e abstrai as operações sobre a localização dos usuários no mecanismo de persistência da infraestrutura. Os principais métodos disponibilizados são: *getCoordenates*, *getUserLocation* e *registerLocation*. O método *getCoordenates* recupera as informações sobre a localização dos usuários passados como parâmetro, com base na data inicial e final. Já o método *getUserLocation* recupera a informação sobre a última localização disponível sobre o usuário. Por fim, o método *registerLocation* registra a localização atual do usuário no mecanismo de armazenamento persistente da infraestrutura;

- *FriendshipDAO*: classe que abstrai as operações sobre o grafo representando as conexões sociais dos usuários. Os principais métodos disponíveis são: *addFriendship*, *getFriends* e *getFriendsAtLevel*. O método *addFriendship* adiciona a informação sobre a

amizade, representada de forma bidirecional, entre dois usuários da infraestrutura para uma determinada Rede Social. Já o método *getFriends* recupera as informações sobre todos os contatos (amigos diretos e indiretos) do usuário para a Rede Social solicitada. Por fim, o método *getFriendsAtLevel* recupera as informações sobre os contatos dos usuários do nível solicitado;

- *ConfigurationReader*: classe responsável por recuperar as informações necessárias para conexão do Módulo de Dados com o mecanismo de armazenamento persistente. O principal método é o *readAllParameters*, que utiliza o padrão de projeto *Singleton* [26] para manter uma única instância das informações do arquivo de configuração.

Apêndice B

Código Fonte do *Plugin* MySocial

No Código Fonte B.1 está exemplificado o código contendo a programação do *plugin* MySocial, desenvolvido a partir da API OpenSocial.

Código Fonte B.1: Código Fonte do MySocial-Gadget

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Module>
3    <ModulePrefs title="MySocial-Gadget">
4      <Require feature="opensocial-0.8" />
5    </ModulePrefs>
6    <Content type="html">
7      <![CDATA[
8  <!--ALm6fM3EMBLxgi8p/RvkTycnJxfQd/6awhLrL55OU9iENew
9  0ptRLNmx5qTk84E3chg74QZjjWBYWgJwqKc0MNX2gRECHucYCq9I
10 113tvLzQvgxTOQGk9aDzCE4A0S3sb63oft87K+qahFw3E+qr3VURxL
11 vO1WSv34HQGdXiqew+MeRCbrezj/kMOamW5eNuL7pegk9tG/qGR-->
12       <script>
13   var openSocialId = "";
14   var userlogin = "";
15   var pwd = "";
16   var socialNetwork = "";
17     // Sends a signed makeRequest to a specified remote server for
         testing purposes
18     function sendAuthRequest()
19     {
20       userlogin = document.getElementById('login').value;
```

```
21        pwd = document.getElementById('pwd').value;
22        socialNetwork = opensocial.getEnvironment().getDomain();
23      var params = {};
24      var postdata = { login : userlogin, passwd : pwd };
25      params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.
            ContentType.TEXT;
26      params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.
            POST;
27          params[gadgets.io.RequestParameters.POST_DATA] = gadgets.io.
                encodeValues(postdata);
28          var urlAuth = "http://tomcat.embedded.ufcg.edu.br/SocialNetsApp/
                servlet/AuthServlet";
29      gadgets.io.makeRequest(urlAuth, getAuthResponse, params);
30      };
31    // Sends a signed makeRequest to a specified remote server for testing
          purposes
32      function sendInfoRequest() {
33      var url = "http://tomcat.embedded.ufcg.edu.br/SocialNetsApp/servlet/
            RegisterServlet";
34          var params = {};
35      var postdata = { login : userlogin, "id" : openSocialId, network :
            socialNetwork };
36      params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.
            ContentType.TEXT;
37          params[gadgets.io.RequestParameters.METHOD] = gadgets.io.
                MethodType.POST;
38          params[gadgets.io.RequestParameters.POST_DATA] = gadgets.io.
                encodeValues(postdata);
39          gadgets.io.makeRequest(url, getResponseId, params);
40      };
41      // Displays the response data
42      function getAuthResponse(response) {
43      if (response.text.search("<body>OK</body>") > -1)
44        sendInfoRequest();
45      else
46        alert('Authentication Failed');
47      };
```

```
48    // Displays the response data
49      function getResponseId(response)
50      {
51      if (response.text.search("<body>OK</body>") > -1)
52        {
53        var someToken = {'hash' : {'user' : userlogin, 'id' : openSocialId,
              'socialNetwork' : socialNetwork} };
54              var json = gadgets.json.stringify(someToken);
55              var req = opensocial.newDataRequest();
56              req.add(req.newUpdatePersonAppDataRequest("VIEWER", 'token',
                    json));
57        req.send(function(data) {});
58        document.getElementById('response').innerHTML = response.text;
59        }
60      };
61    // Loads the VIEWER ID and saves a token to appData
62      function init()
63      {
64        var req = opensocial.newDataRequest();
65        req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.OWNER)
              , "me");
66        req.send(function(data) {
67        openSocialId = data.get("me").getData().getId();
68              });
69      }
70      gadgets.util.registerOnLoadHandler(init);
71    </script>
72    Login: <input type="text" name="login" id="login" /> <br />
73      Senha: <input type="password" name="senha" id="pwd" />
74      <a href='javascript:void(0);' onclick='sendAuthRequest();'>Send</a>
75      <p>Server response text:</p>
76      <p id='response'></p>
77      ]]>
78    </Content>
79  </Module>
```

Apêndice C

Modelagem do Banco de Dados

C.1 Modelo de Entidade-Relacionamento

O Modelo de Entidade-Relacionamento especificado é ilustrado através da Figura C.1. Ainda, na seção C.2 está disponível o *script* para geração do Banco de Dados utilizado pela infraestrutura. A seguir são descritas as entidades que compõem o modelo:

- *Users*: armazena as informações sobre os usuários cadastrados na infraestrutura, tais como: identificador único do usuário (*user_login*), senha (*passwd*), primeiro e último nome do usuário (*first_name* e *last_name*, respectivamente);

- *Preferences*: armazena as informações sobre as preferências/interesses dos usuários. Os principais campos são: identificador do usuário (*user_login*) e o valor para a preferência (*preference*);

- *Friendship*: armazena as informações sobre as conexões sociais dos usuários. Essas informações são originadas de duas fontes de dados distintas, são elas: (i) das listas de contatos dos usuários, extraídas das Redes Sociais Online e (ii) do resultado das conexões *ad hoc* realizadas pelos dispositivos móveis dos usuários. As principais informações armazenadas são: identificador do primeiro usuário (*user_login1*), identificador do segundo usuário (*user_login2*) e o identificador da Rede Social (*social_network*) em que os dois usuários estão relacionados. É importante destacar que todos os relacionamentos armazenados nessa tabela são considerados bidirecionais.

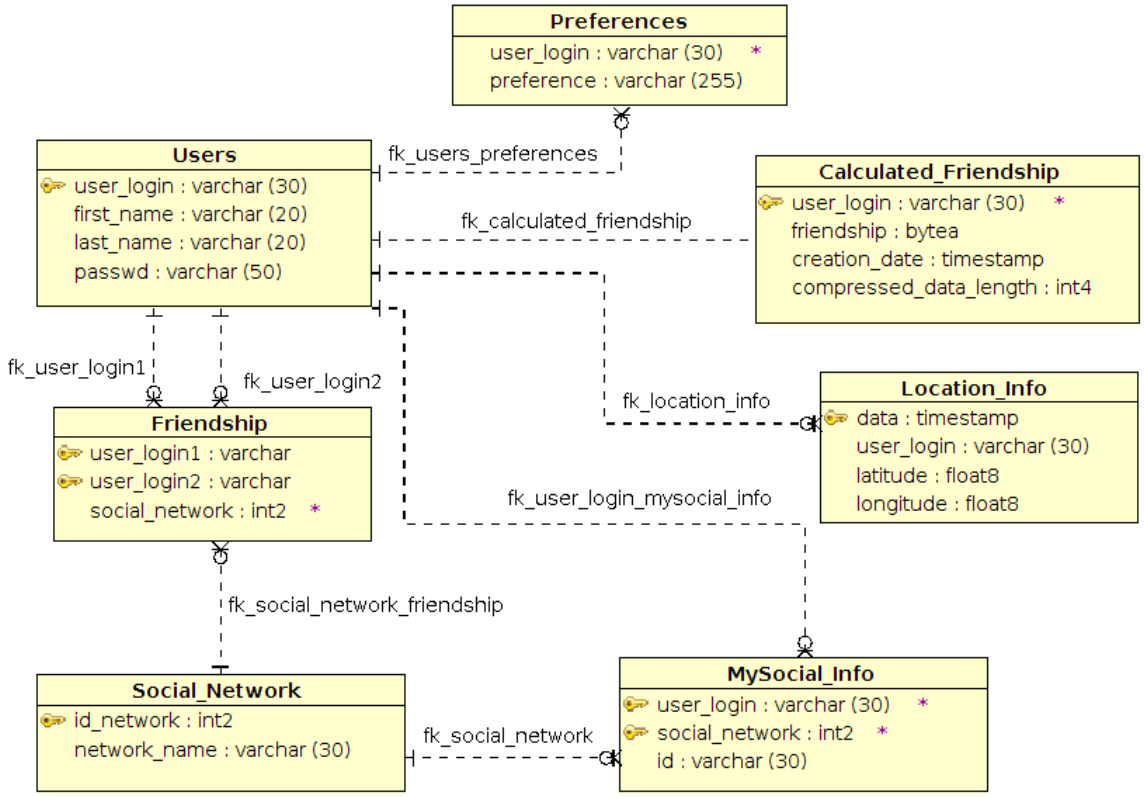


Figura C.1: Diagrama de Entidade-Relacionamento da Camada de Dados

Portanto, quando a infraestrutura constrói o grafo representando as conexões sociais dos usuários a partir da Base de Dados, *se o usuário A for amigo do usuário B, então, o usuário B também é amigo do usuário A*[1];

- *Calculated_Friendship*: armazena o resultado do processamento realizado pelo Módulo de Expansão das Conexões Sociais sobre o grafo representando os relacionamentos sociais dos usuários, armazenados na tabela *Friendship*. Os principais campos são: identificador do usuário (*user_login*), data de criação (*creation_date*), relacionamentos calculados (*friendship*) e quantidade de *bytes* utilizados (*compressed_data_length*). É importante destacar que para diminuir o espaço necessário para armazenar o conteúdo do campo *friendship*, representado através de uma Tabela *hash* [16] serializada no formato XML (*eXtensible Markup Language*), foi utilizado um processo de compactação sobre os dados. Sendo, o resultado desse último processamento armazenado no Banco de Dados como um *array* de *bytes*;

[1]Em alguns serviços sociais como, por exemplo, o Twitter [20] o relacionamento é unidirecional.

- *Location_Info*: armazena as informações sobre a localização dos usuários. Especificamente, essa tabela armazena as seguintes informações: data (no formato: ano-mês-dia hora:minuto:segundo), identificador do usuário (*user_login*), latitude e longitude. Onde, os dois últimos utilizam uma representação decimal para o valor da coordenada correspondente.

- *MySocial_Info*: realiza o mapeamento entre o identificador do usuário com o identificador único para a Rede Social Online, obtido a partir da API OpenSocial. Portanto, os principais campos dessa tabela são: identificador do usuário na infraestrutura (*user_login*), identificador OpenSocial (*id*) e identificador da Rede Social Online (*social_network*), cadastrado na tabela *Social_Network*;

- *Social_Network*: armazena as informações que identificam as Redes Sociais Online. Os principais campos são: identificador da Rede Social (*id_network*) e nome da Rede Social Online (*network_name*).

C.2 *Script* para criação do Banco de Dados

No Código Fonte C.1 está exemplificado o *script* para criação do Banco de Dados utilizado pela infraestrutura.

Código Fonte C.1: *Script* para criação do Banco de Dados MySocial

```
1  CREATE DATABASE mysocial WITH TEMPLATE = template0
2  ENCODING = 'UTF8' LC_COLLATE = 'pt_BR.UTF-8' LC_CTYPE = 'pt_BR.UTF-8';
3  ALTER DATABASE mysocial OWNER TO postgres;
4  ALTER TABLE "MySocial_Info"
5          DROP CONSTRAINT "fk_social_network" CASCADE ;
6  ALTER TABLE "Friendship"
7          DROP CONSTRAINT "fk_social_network_friendship" CASCADE ;
8  ALTER TABLE "Calculated_Friendship"
9          DROP CONSTRAINT "fk_calculated_friendship" CASCADE ;
10 ALTER TABLE "Location_Info"
11         DROP CONSTRAINT "fk_location_info" CASCADE ;
12 ALTER TABLE "Friendship"
13         DROP CONSTRAINT "fk_user_login2" CASCADE ;
```

```
14 ALTER TABLE "MySocial_Info"
15         DROP CONSTRAINT "fk_user_login_mysocial_info" CASCADE ;
16 ALTER TABLE "Friendship"
17         DROP CONSTRAINT "fk_user_login1" CASCADE ;
18 ALTER TABLE "Preferences"
19         DROP CONSTRAINT "fk_users_preferences" CASCADE ;
20 DROP TABLE "Calculated_Friendship";
21 DROP TABLE "Friendship";
22 DROP TABLE "Location_Info";
23 DROP TABLE "MySocial_Info";
24 DROP TABLE "Preferences";
25 DROP TABLE "Social_Network";
26 DROP TABLE "Users";
27 CREATE TABLE "Calculated_Friendship"  (
28         "user_login"                    varchar(30) NOT NULL,
29         "friendship"                    bytea NOT NULL,
30         "creation_date"                 timestamp NOT NULL,
31         "compressed_data_length"        int4 NOT NULL,
32         PRIMARY KEY("user_login") );
33 CREATE TABLE "Friendship"  (
34         "user_login1"           varchar(30) NOT NULL,
35         "user_login2"           varchar(30) NOT NULL,
36         "social_network"        int2 NOT NULL,
37         PRIMARY KEY("user_login1","user_login2") );
38 CREATE TABLE "Location_Info"  (
39         "data"          timestamp NOT NULL,
40         "user_login"    varchar(30) NOT NULL,
41         "latitude"      float8 NOT NULL,
42         "longitude"     float8 NOT NULL,
43         PRIMARY KEY("data") );
44 CREATE TABLE "MySocial_Info"  (
45         "user_login"            varchar(30) NOT NULL,
46         "social_network"        int2 NOT NULL,
47         "id"                    varchar(30) NOT NULL,
48         PRIMARY KEY("user_login","social_network") );
49 CREATE TABLE "Preferences"  (
50         "user_login"    varchar(30) NOT NULL,
```

```
51          "preference"    varchar(255) NOT NULL
52          );
53  CREATE TABLE "Social_Network"  (
54          "id_network"    int2 NOT NULL,
55          "network_name"  varchar(30) NOT NULL,
56          PRIMARY KEY("id_network") );
57  CREATE TABLE "Users"  (
58          "user_login"    varchar(30) NOT NULL,
59          "first_name"    varchar(20) NOT NULL,
60          "last_name"     varchar(20) NOT NULL,
61          "passwd"        varchar(50) NOT NULL,
62          PRIMARY KEY("user_login") );
63  ALTER TABLE "MySocial_Info"
64          ADD CONSTRAINT "fk_social_network"
65          FOREIGN KEY("social_network")
66          REFERENCES "Social_Network"("id_network");
67  ALTER TABLE "Friendship"
68          ADD CONSTRAINT "fk_social_network_friendship"
69          FOREIGN KEY("social_network")
70          REFERENCES "Social_Network"("id_network");
71  ALTER TABLE "Calculated_Friendship"
72          ADD CONSTRAINT "fk_calculated_friendship"
73          FOREIGN KEY("user_login")
74          REFERENCES "Users"("user_login");
75  ALTER TABLE "Location_Info"
76          ADD CONSTRAINT "fk_location_info"
77          FOREIGN KEY("user_login")
78          REFERENCES "Users"("user_login");
79  ALTER TABLE "Friendship"
80          ADD CONSTRAINT "fk_user_login2"
81          FOREIGN KEY("user_login2")
82          REFERENCES "Users"("user_login");
83  ALTER TABLE "MySocial_Info"
84          ADD CONSTRAINT "fk_user_login_mysocial_info"
85          FOREIGN KEY("user_login")
86          REFERENCES "Users"("user_login");
87  ALTER TABLE "Friendship"
```

```
88          ADD CONSTRAINT "fk_user_login1"
89          FOREIGN KEY("user_login1")
90          REFERENCES "Users"("user_login");
91  ALTER TABLE "Preferences"
92          ADD CONSTRAINT "fk_users_preferences"
93          FOREIGN KEY("user_login")
94          REFERENCES "Users"("user_login");
```

Apêndice D

Scripts para Execução do Módulo de Expansão das Conexões Sociais

Nos Códigos Fontes D.1 e D.2 estão exemplificados os *scripts bash* utilizados para execução do Servidor e Cliente RMI, respectivamente.

Código Fonte D.1: *Script* para execução do Servidor RMI

```
1  #!/bin/bash
2
3  count=0
4  j=1
5  vertex_degree=200
6  hosts=8
7  max=120
8  update=0.1
9
10 while [ $count -lt $max ]
11 do
12
13 if [ $count -lt 60 ]; then
14 vertex=10000
15 else
16 vertex=100000
17 fi
18
19 java -Xms256M -Xmx3678M -Djava.rmi.server.hostname=192.168.1.31 -Djava.
```

```
    rmi.server.codebase="file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/bin/ file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/Geodesy.jar file:/home/clusteruser/daniel/
    SocialNsAppClientRMI/lib/jgraph.jar file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/jgrapht-jdk1.6.jar file:/home/clusteruser/
    daniel/SocialNetsAppClientRMI/lib/xstream-1.3.1.jar file:/home/
    clusteruser/daniel/SocialNetsAppClientRMI/lib/xpp3_min-1.1.4c.jar file
    :/home/clusteruser/daniel/SocialNetsAppClientRMI/lib/postgresql
    -8.4-701.jdbc4.jar file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/jettison-1.2.jar" -jar ServerRMI_fat.jar
    rmi://192.168.1.31:1099/ProcessService$j $hosts $vertex $vertex_degree
     ScaleFree /home/clusteruser/daniel/RMI/files/ $update

j=`expr $j + 1`
count=`expr $count + 1`
done
```

Código Fonte D.2: *Script* para execução do Cliente RMI

```
#!/bin/bash

count=0
j=1
levels=6
current_level=1
repetitions_count=10
node="clusternode2"

while [ $count -lt $levels ]
do

repetitions=0

while [ $repetitions -lt $repetitions_count ]
do

java -Xms256M -Xmx3678M -Djava.rmi.server.hostname=192.168.1.31 -Djava.
    rmi.server.codebase="file:/home/clusteruser/daniel/
```

```
    SocialNetsAppClientRMI/bin/ file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/Geodesy.jar file:/home/clusteruser/daniel/
    SocialNsAppClientRMI/lib/jgraph.jar file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/jgrapht-jdk1.6.jar file:/home/clusteruser/
    daniel/SocialNetsAppClientRMI/lib/xstream-1.3.1.jar file:/home/
    clusteruser/daniel/SocialNetsAppClientRMI/lib/xpp3_min-1.1.4c.jar file
    :/home/clusteruser/daniel/SocialNetsAppClientRMI/lib/postgresql
    -8.4-701.jdbc4.jar file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/jettison-1.2.jar" -jar ClientRMI_fat.jar
    rmi://192.168.1.31:1099/ProcessService$j 10 1000 $current_level /home/
    clusteruser/daniel/RMI/files/ $node
19
20  j=`expr $j + 1`
21  repetitions=`expr $repetitions + 1`
22  done
23
24  current_level=`expr $current_level + 1`
25  count=`expr $count + 1`
26  done
27
28  count=0
29  current_level=1
30
31  while [ $count -lt $levels ]
32  do
33
34  repetitions=0
35
36  while [ $repetitions -lt $repetitions_count ]
37  do
38
39  java -Xms256M -Xmx3678M -Djava.rmi.server.hostname=192.168.1.31 -Djava.
    rmi.server.codebase="file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/bin/ file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/Geodesy.jar file:/home/clusteruser/daniel/
    SocialNsAppClientRMI/lib/jgraph.jar file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/jgrapht-jdk1.6.jar file:/home/clusteruser/
```

```
    daniel/SocialNetsAppClientRMI/lib/xstream-1.3.1.jar file:/home/
    clusteruser/daniel/SocialNetsAppClientRMI/lib/xpp3_min-1.1.4c.jar file
    :/home/clusteruser/daniel/SocialNetsAppClientRMI/lib/postgresql
    -8.4-701.jdbc4.jar file:/home/clusteruser/daniel/
    SocialNetsAppClientRMI/lib/jettison-1.2.jar" -jar ClientRMI_fat.jar
    rmi://192.168.1.31:1099/ProcessService$j 10 1000 $current_level /home/
    clusteruser/daniel/RMI/files/ $node
40
41  j=`expr $j + 1`
42  repetitions=`expr $repetitions + 1`
43  done
44
45  current_level=`expr $current_level + 1`
46  count=`expr $count + 1`
47  done
```