Universidade do Minho
Escola de Engenharia

Emanuel José Oliveira Braga

Integração de informação de contexto em redes sociais online

Outubro de 2010

Emanuel José Oliveira Braga **Integração de informação de contexto em redes sociais online**

UMinho | 2010

Universidade do Minho
Escola de Engenharia

Emanuel José Oliveira Braga

Integração de informação de contexto em redes sociais online

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efectuado sob a orientação do
Professor Doutor António Nestor Ribeiro

Outubro de 2010

Declaração

Nome: Emanuel José Oliveira Braga

Endereço Electrónico: braga.emanuel@gmail.com

Telefone: 919068716

Bilhete de identidade: 13101754

Título da Tese: Integração de informação de contexto em redes sociais online

Orientador: Professor Doutor António Nestor Ribeiro

Ano de conclusão: 2010

Designação do Mestrado: Mestrado em Engenharia Informática

Universidade do Minho, Outubro de 2010

Assinatura: _____

Good judgment comes from experience. Experience comes from bad judgment.

Jim Horning

Agradecimentos

O período correspondente a esta dissertação foi sinónimo de exploração, descoberta e trabalho. Os resultados obtidos apenas foram conseguidos com muito empenho pessoal e com a ajuda das pessoas mais próximas. Visto isto, é importante referir aqueles que, de uma forma ou de outra, mais contribuíram para a conclusão e sucesso desta dissertação.

Primeiramente o meu orientador, Professor Doutor António Nestor Ribeiro, pela disponibilidade demonstrada ao longo de todo este ano, bem como por todas as críticas, ideias e sugestões que em muito melhoraram o presente trabalho.

Agradeço também aos amigos que comigo embarcaram nesta etapa de formação. Em especial ao Pedro Gomes, Matheus Almeida, Ricardo Gonçalves, Miguel Araújo, Carlos Silva, Henrique Castro, Pedro Silva e André Carvalho pelas muitas histórias trocadas e pelos muitos momentos bem passados.

Uma palavra especial para a minha namorada, Juliana, que sempre esteve ao meu lado com palavras de apoio e estímulo que ajudaram a ultrapassar os momentos em que as coisas corriam menos bem.

Por último, um agradecimento também aos meus pais e irmãos, pela paciência com que toleraram os momentos de mais trabalho e menor disponibilidade.

Sem a ajuda e incentivo de todos, esta dissertação não o seria.

Resumo

A área das redes sociais é, talvez, a área que melhor tem conseguido tirar partido da evolução das tecnologias Web, sendo que os serviços mais populares contam com milhões de utilizadores registados por todo o mundo. No entanto, a generalidade das suas funcionalidades limita-se a divulgar informações pessoais inseridas pelos próprios utilizadores.

Os serviços sociais estão constantemente sujeitos a carga computacional extremamente elevada, justificada com o elevado número de utilizadores activos em cada instante e pelas questões de concorrência entre os seus processos. Para permitir níveis de robustez mais elevados, foi incluída nesta dissertação a linguagem Erlang. Criada inicalmente para a área das aplicações telefónicas, é actualmente utilizada em diversos serviços críticos, incluindo as redes sociais online.

Ao longo desta dissertação é explorada a possibilidade de integração de mecanismos sensíveis ao contexto do utilizador nas redes sociais online, através das suas plataformas de interacção remota. Para tal, é apresentada uma solução arquitectural que permite o desenvolvimento de serviços sociais sensíveis ao contexto e com componentes desenvolvidos em Erlang. Esta solução é aplicada num caso de estudo, o sistema LocalChat, de forma a corroborar a sua aplicabilidade e a pertinência do próprio caso de estudo.

A arquitectura proposta segue um modelo cliente-servidor. O servidor é o componente crítico do sistema e foi alvo das maiores preocupações ao nível da escalabilidade e disponibilidade. Por sua vez, no caso do cliente as preocupações centraram-se ao nível da captação do contexto e interacção com o utilizador.

O caso de estudo atende a vários requisitos de modo a cativar o interesse do utilizador comum de redes sociais. A sua funcionalidade principal é baseada na localização do utilizador e permite que um utilizador possa pesquisar por outros dentro de um determinado raio e interagir directamente com eles.

Palávras-chave: informação contextual, redes sociais, Erlang, funcionalidades sociais, aplicações móveis.

Abstract

Online social networks have millions of users all over the world and are probably the services that better took advantage of the evolution of Web 2.0 technologies. However, their features are generally based on user generated content, being personal information, and changes to its status, the most popular content.

The amount of active users online on every moment, and the concurrency between processes, makes this kind of applicants very susceptible to performance and scaling problems. The Erlang programming language was included on this dissertation to minimize them and turn these services more reliable. From inception, Erlang was developed to solve this kind of problems on the telecommunications' sector, but today, it is used to build critical components of several areas, including online social networks.

Throughout this dissertation, it is explored the integration of context information on online social networks, through their remote interfaces. As a result, it is presented an architectural solution projected to enable the creation of context-aware social services, with scalability concerns and Erlang components. To assess the relevance of this architecture and of this kind of features, this dissertation also presents a case study, named LocalChat.

The architectural solution follows a client-server approach. The server is the critical component and it was planned with scalability and availability concerns. By its turn, the clientś concerns are related to the context gathering process and interactivity with users.

This system has the common social network user as target audience. Its main feature is based on location (context information) and it allows users to find other people, inside a given radius, and interact directly with them.

Keywords: context, online social networks, Erlang, social features, mobile applications.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Desde a sua criação até aos dias de hoje, a internet e os serviços Web disponibilizados têm sofrido uma grande evolução. Inicialmente estes serviços eram compostos por páginas completamente estáticas. No entanto, com a introdução do conceito de Web 2.0 assistiu-se a uma revolução na forma como o utilizador interage com os sistemas. Esta interacção deixou de ser estática e, o dinamismo que as novas tecnologias possibilitam levou, por um lado, à evolução dos serviços até então existentes e, por outro, à criação de muitos outros. Hoje existem disponíveis serviços para todos os gostos e com os mais variados propósitos, sendo provavelmente o fenómeno das redes sociais online o que está sujeito actualmente a uma maior popularidade. De tal forma que poderá ser difícil encontrar um internauta que não esteja registado em pelo menos um serviço deste género.

Nos dias que correm, as redes sociais mais populares possuem milhões de utilizadores por todo o mundo divididos por diversas faixas etárias, sendo usados como ponto de encontro a nível social e profissional. Cada utilizador é incentivado a publicar informações pessoais e conteúdos multimédia no seu perfil. Após a sua publicação, estas informações podem ser tornadas visíveis para os seus contactos ou para o público em geral. A frequência com que os utilizadores mais devotos actualizam os seus perfis é bastante assinalável, sendo considerado para alguns um acto de extrema importância.

O mundo das redes sociais vive essencialmente do chamado user generated content[Krumm et al., 2008], isto é, dos dados e da participação dos seus utilizadores. A evolução destes serviços sociais tem seguido uma via em que grande parte do processo criativo é delegado para os seus utilizadores. Para tal, a generalidade das redes sociais disponibiliza meios que permitem a interacção remota entre os seus sistemas e aplicações externas. Estes meios con-

sistem normalmente em APIs[1] remotas ou plataformas de desenvolvimento mais completas. Desta forma, qualquer utilizador que possua o mínimo de conhecimentos em desenvolvimento de software pode criar novas formas de interacção com as redes sociais ou mesmo estender as funcionalidades já existentes. Este tipo de evolução dos serviços confere um grande dinamismo às redes sociais, atraindo também desta forma novos utilizadores.

Paralelamente, assiste-se também a uma evolução nos dispositivos e nas tecnologias móveis. Esta evolução tem levado ao aparecimento de inúmeras aplicações nas mais variadas áreas, sendo a das redes sociais uma das mais exploradas. Esta aliança entre as redes sociais e as aplicações móveis permite aos seus utilizadores, a título de exemplo, a actualização dos seus perfis pessoais onde quer que se encontrem. Abre-se também a possibilidade de incluir, de forma automática, informações relativas ao contexto envolvente do utilizador nos conteúdos publicados, construindo desta forma um perfil pessoal mais rico.

1.1 Definição do problema

Devido à grande importância dada pelos seus utilizadores em actualizar os seus perfis, as redes sociais online concentram uma grande quantidade de informação pessoal. No entanto, esta informação está normalmente desprovida de contexto, isto é, nada diz acerca do ambiente em que o utilizador está inserido. As excepções são as aplicações em que o contexto é fornecido explicitamente pelo utilizador e os casos em que a própria rede social implementa já mecanismos nesse sentido, o que não é muito usual nos dias que correm.

Quando é referido o contexto do utilizador, podem ser pensados vários aspectos como por exemplo a sua localização geográfica, a sua agenda, as pessoas ou objectos que o rodeiam, etc. A lista de parâmetros que podem ser utilizados é extensa e, de entre todos, devem sempre ser escolhidos aqueles que são realmente relevantes para cada caso específico.

Já foi referido que as redes sociais online têm uma grande importância para os seus utilizadores. Visto isto, é comum assistir-se a casos em que o crescimento de popularidade é bastante acentuado em serviços sociais que lhes estejam associados. Os momentos em que estes casos acontecem podem ser considerados críticos, na medida em que caso não consigam responder positivamente ao aumento da procura, sejam preteridos por serviços análogos. Para precaver estas situações, os serviços com estas características devem sempre ser projectados com requisitos ao nível da escalabilidade e disponibilidade.

[1] Application Programming Interface

O conceito de escalabilidade pode ser definido como a forma em como uma solução particular resolve um determinado problema à medida que este aumenta [Schlossnagle, 2006]. Existem dois tipos de soluções a adoptar:

- **Escalar verticalmente:** aumentar o poder das máquinas que correm o sistema, diminuindo assim o tempo necessário para executar cada pedido;
- **Escalar horizontalmente:** distribuir o sistema por várias máquinas, para que este possa aumentar o número de pedidos processados.

A concorrência entre processos é outro ponto relevante nestes sistemas. À medida que o número de utilizadores activos em cada instante aumenta, o fluxo de dados tende também a variar no mesmo sentido. Além da inconveniência de os utilizadores poderem ver os seus dados pessoais trocados ou perdidos, um mau controlo de concorrência pode também influenciar o desempenho de todo o sistema.

Além da arquitectura, a escolha das tecnologias pode ter também um papel importante no que diz respeito ao esforço necessário para atingir os níveis de escalabilidade desejados. Desta forma, a escolha deve recair por uma tecnologia capaz de, por um lado, eliminar a concorrência no acesso aos dados e, por outro, de responder aos requisitos normais de aplicações distribuídas, escaláveis e disponíveis.

Com o presente trabalho pretende-se perceber qual o ponto de situação relativamente aos esforços das redes sociais em integrar informação contextual nas suas funcionalidades. Além disto, são também apresentados:

- uma arquitectura conceptual, com preocupações ao nível de requisitos não funcionais, como a escalabilidade, e que permita a criação de aplicações sociais sensíveis ao contexto;
- o sistema LocalChat (LC) que tira partido desta arquitectura e fornece um serviço que comprova a potencialidade das funcionalidades sociais dependentes contexto.

1.2 Motivações

Embora, na sua origem, as redes sociais não implementassem este tipo de funcionalidades, nota-se uma crescente preocupação por parte destas no sentido de utilizar informações do contexto. A informação mais tida em consideração

é a localização, sendo o Twitter[2] um dos pioneiros na sua integração. Esta integração começou por consistir na marcação das publicações com a posição geográfica dos utilizadores e no enriquecimento da sua API remota com novos métodos.

Esta crescente aposta na utilização do contexto, aliada aos mecanismos disponibilizados para a integração com aplicações externas, faz com que este seja um tópico interessante a explorar. Mais do que isso, resultado desta integração permite actualmente uma espécie de realidade aumentada, fornecendo novos tipos de experiencias aos seus utilizadores. Podem ser consideradas duas abordagens com diferentes objectivos:

- Estender as funcionalidades oferecidas pelas redes sociais, dando mais relevância aos aspectos do ambiente do utilizador;

- Utilizar as redes sociais como fonte de dados acerca do contexto do utilizador. Neste caso, as aplicações moldam-se sempre que detectam uma alteração no contexto com o intuito de oferecerem um serviço mais adequado às necessidades actuais.

Esta dissertação concretiza as duas alternativas. O sistema LC fornece um serviço social que promove a interacção entre utilizadores geograficamente próximos, através de um serviço de conversação, e que recorre a redes sociais para obter informações pessoais sobre eles. Este sistema é abordado de forma mais profunda nas secções 6 e 7.

Já foi referido que o tema das redes sociais é bastante popular nos dias que correm e que existe, por parte destas, uma vontade crescente da sua integração com informação de contexto. Um sistema deste tipo deve ser pensado de raiz para a possibilidade de ter um grande crescimento de popularidade num curto espaço de tempo. Deve, portanto, ser desenvolvido tendo como requisitos de disponibilidade, controlo da concorrência e facilidade em escalar, horizontal e verticalmente. A linguagem de programação Erlang [Armstrong, 2007], inicialmente desenvolvida para a área das telecomunicações, possui uma arquitectura que potencia todos estes requisitos. É uma linguagem que permite a instalação de novas versões sem a necessidade de parar o sistema, promovendo a disponibilidade e a escalabilidade. O seu funcionamento interno é baseado em processos bastante leves e eficientes. A comunicação entre estes processos é feita recorrendo a um sistema de mensagens assíncrono e não existe qualquer partilha de memória, eliminando desta forma problemas de concorrência.

[2] `http://www.twitter.com`

A popularidade desta linguagem não é comparável à de outras, como Java, C# ou PHP, mas tem ganho o seu espaço no desenvolvimento de componentes críticos, e de sistemas completos que necessitem de elevado desempenho e escalabilidade. Actualmente é possível verificar a sua presença em áreas desde a modelação 3D, bases de dados distribuídas e mesmo em redes sociais online, como é o caso do Facebook[3] e do Delicious[4].

Dadas as suas características e aplicações reais, a utilização da tecnologia Erlang figura-se extremamente apropriada para o desenvolvimento dos componentes mais críticos da arquitectura conceptual proposta. Desta forma, serão mais facilmente satisfeitos os requisitos não funcionais que tornam uma aplicação robusta.

1.3 Objectivos da dissertação

Tomando em consideração o que foi descrito até ao momento, podemos resumir os objectivos desta dissertação como sendo:

- Tomar conhecimento do estado da arte sobre aplicações sensíveis ao contexto e sobre as tendências em integrar estes sistemas com as redes sociais online;
- Perceber quais as possíveis vantagens da utilização da linguagem de programação Erlang no desenvolvimento de aplicações altamente escaláveis e concorrentes;
- Criação de uma arquitectura conceptual que promova a integração das redes sociais com informação de contexto. A estes serviços está normalmente associada uma elevada carga computacional, logo é imperativo o uso de uma tecnologia como o Erlang para que a escalabilidade e o desempenho do sistema sejam potenciados.
- Desenvolvimento do sistema LocalChat, capaz de tirar proveito da arquitectura proposta.
- Desenvolvimento de uma aplicação móvel capaz de comprovar, quer a utilidade do sistema LocalChat no quotidiano dos seus utilizadores, quer a importância da integração dos conceitos de contexto e redes sociais;

[3] `http://www.facebook.com/`
[4] `http://delicious.com/`

1.4 Considerações sobre o trabalho

O desenvolvimento de uma aplicação completa e funcional que integra os conceitos de informação contextual, redes sociais e mobilidade levanta alguns requisitos que não se enquadram no âmbito deste trabalho, nomeadamente a segurança dos dados. Visto isto, foi necessário definir algumas premissas para uma abordagem mais completa dos conceitos a desenvolver.

1.4.1 Segurança e privacidade

Os dados pessoais dos utilizadores, os seus conteúdos multimédia e a sua informação contextual constituem a base do software produzido no âmbito desta dissertação. Sendo este tipo de informação extremamente sensível, a segurança e privacidade dos utilizadores devem ser sempre acauteladas. No entanto, esta dissertação pretende evidenciar as mais-valias que é possível obter utilizando este tipo de informações, relegando estas questões para um segundo plano. Em todo o caso, foram implementados alguns mecanismos básicos para protecção dos dados e privacidade dos utilizadores. No entanto, existe a consciência que futuramente estes aspectos devem ser repensados e os mecanismos de segurança reforçados.

1.4.2 Plataformas móveis

Actualmente existe uma considerável diversidade de sistemas operativos destinados aos dispositivos móveis, tais como Google Android, Apple IOS, Windows Phone, Symbian, RIM Blackberry etc.

No decorrer desta dissertação foi utilizado o Google Android. A escolha recaiu por este sistema operativo devido ao facto de o seu ambiente de desenvolvimento ser independente da plataforma. A versão utilizada foi a 2.2 – Froyo – por ser a mais recente e também porque é a mais completa a nível de funcionalidades oferecidas.

No que diz respeito a dispositivos de teste, apenas foi utilizado o emulador disponibilizado juntamente com a plataforma de desenvolvimento. Tendo isto em consideração, fará todo o sentido que o software desenvolvido seja testado num dispositivo real, tendo este acesso a um sensor GPS e conectividade com a internet.

1.4.3 O mundo não pára

Durante o último ano, período dedicado a esta dissertação, o universo que envolve as redes sociais sofreu uma constante evolução, quer ao nível das

aplicações externas, quer ao nível das próprias plataformas de desenvolvimento disponibilizadas. A integração das redes sociais com informação de contexto tem também vindo a ser algo explorada e surgiram sistemas análogos ao apresentado nesta dissertação. Alguns exemplos destes sistemas serão descritos no capítulo que aborda o estado da arte sobre a integração de aplicações sensíveis ao contexto em redes sociais.

Além destes novos serviços e funcionalidades entretanto lançadas, outras ideias semelhantes foram entretanto divulgadas nos blogues oficiais das redes sociais e em diversos blogues de cariz tecnológico.

1.5 Estrutura do documento

O teor da presente dissertação encontra-se dividido em vários capítulos. O primeiro introduz o âmbito e a motivação deste trabalho. No segundo capítulo são abordados os dois conceitos teóricos, e os sistemas existentes, que servem de base ao trabalho realizado. No capítulo 3 encontramos uma descrição sobre a linguagem de programação Erlang, evidenciando suas funcionalidades e casos em que é aplicada, justificando a sua inclusão nesta dissertação. O capítulo 4 explora outras tecnologias relevantes para o desenvolvimento do caso de estudo desenvolvido nesta dissertação. A sua apresentação é feita no capítulo 5, que além disto, apresenta também a arquitectura conceptual criada para o desenvolvimento de serviços sociais dependentes do contexto dos seus utilizadores. Os dois capítulos seguintes, capítulos 6 e 7, apresentam o servidor e um cliente do serviço LocalChat, que implementa a solução arquitectural proposta. Por último, o capítulo 8 apresenta as conclusões que resultam do trabalho desenvolvido ao longo desta dissertação.

Capítulo 2

Estado da arte

Este capítulo começa por apresentar os dois conceitos que servem de base a todo o trabalho desenvolvido nesta dissertação: as redes sociais online e a sensibilidade ao contexto do utilizador.

No caso específico das redes sociais, são explorados os serviços mais populares e as plataformas disponibilizadas por estes para integração de aplicações externas.

Relativamente ao contexto do utilizador, são também apresentados alguns exemplos reais onde o conceito é aplicado, bem como as fontes onde este tipo de informação pode ser capturado e como pode ser modelado.

Por fim, é feita a ponte entre estes dois conceitos. São mostrados vários exemplos concretos de serviços sociais dependentes do contexto, bem como os esforços que as redes sociais online mais populares estão a desenvolver para evoluir nesse sentido. Em destaque estará o caso da funcionalidade Places, recentemente lançado pela rede social Facebook. É ainda dada alguma atenção às questões relacionadas com a privacidade dos utilizadores.

2.1 Redes Sociais Online

Ao longo desta secção irá ser abordado o conceito de rede social online. Serão apresentados alguns dos serviços mais populares, bem como as plataformas que estes disponibilizam para o desenvolvimento e integração de aplicações externas.

2.1.1 O que são as redes sociais online

Tal como já foi referido, com o aparecimento da chamada Web 2.0, o conceito de rede social online tornou-se bastante popular. Hoje em dia, de-

pois de blogues, fóruns e salas de conversação, estes serviços podem mesmo ser considerados como a mais recente forma de comunicação sobre a internet [Joly et al., 2009]. De entre todos, os serviços mais populares actualmente são o Facebook, MySpace[1], LinkedIn[2] e Twitter, tendo cada um vários milhões de utilizadores registados.

Embora tenham diferentes objectivos e pretendam servir diferentes tipos de utilizadores, todos têm como base a partilha de informação pessoal e de conteúdos multimédia entre os seus utilizadores. Todas as redes sociais online permitem aos seus utilizadores [Boyd and Ellison, 2007]:

- Criar um espaço público ou semi-público contendo o seu perfil pessoal;
- Ter uma lista de contactos;
- Interagir com os seus contactos e com os contactos destes.

O conceito de rede social pode assim ser visto como uma extensão aos serviços de *instant messaging*, onde cada utilizador pode adicionar outros às suas comunidades e partilhar conteúdo multimédia. Alguns sistemas oferecem mesmo um serviço de conversação em tempo real, como é o caso do Facebook.

Embora todas as redes sociais permitam ao utilizador ter a sua comunidade, o nome que é dado a cada contacto pode variar. Por exemplo, enquanto um contacto no Facebook é normalmente denominado de *amigo*, no Twitter este mesmo contacto pode ser chamado de *seguidor*. O mesmo princípio pode ser aplicado a outros aspectos além dos contactos.

2.1.2 Exemplos

Um dos primeiros exemplos de redes sociais online foi o *classmates.com*, onde as comunidades eram compostas por pessoas que tinham andado numa mesma escola.

Mais tarde surgiu o MySpace. Neste serviço, os utilizadores possuem uma página pessoal personalizável chamada "espaço", como a apresentada na Figura 2.1. É um sistema bastante utilizado por artistas e bandas para publicar notícias, os seus trabalhos e para interacção com os seus admiradores. Este facto faz com que, no geral, as comunidades do MySpace não sejam compostas por pessoas que se conhecem pessoalmente, mas por desconhecidos que admiram o trabalho de alguém. Além do "espaço", o MySpace oferece

[1] `http://www.myspace.com/`
[2] `http://www.linkedin.com/`

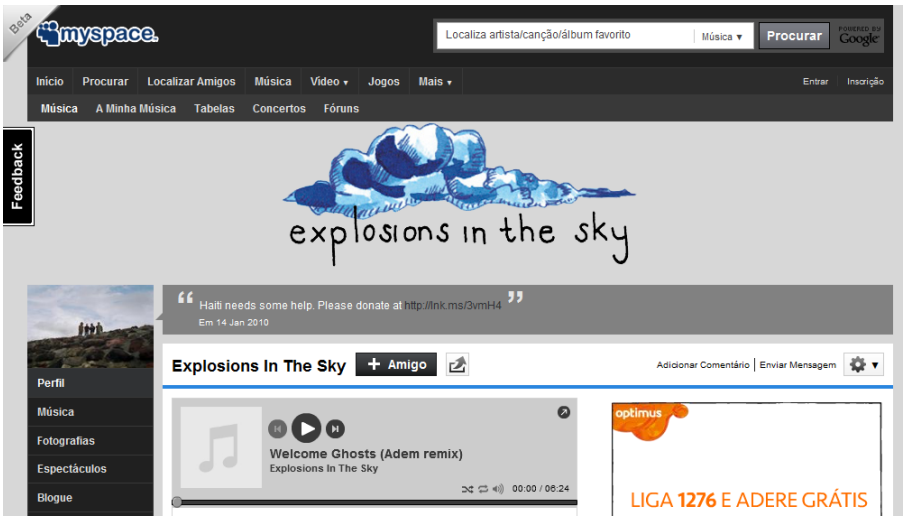

Figura 2.1: "Espaço"no MySpace da banda Explosions in the Sky.

várias funcionalidades aos seus utilizadores, tais como boletins que podem ser publicados e visíveis pelos seus contactos, a criação de grupos, um sistema de conversação, acessibilidade por dispositivos móveis, um calendário para marcação de eventos, entre outras.

Figura 2.2: Página da Coca-Cola no Facebook.

Mark Zuckerberg, ex-estudante da universidade de Harvard tinha como objectivo interligar todos os estudantes dessa universidade. Para tal desenvolveu um projecto, que foi mais tarde aberto a toda a comunidade e

chamado de Facebook. Actualmente, este é o mais directo concorrente do MySpace e tem uma noção de contacto mais próxima da vida real. Este pode ser um ponto importante na escolha de uma rede social de entre estas duas. Tendo em conta apenas este ponto, um utilizador escolherá o Facebook quando o seu objectivo for o de interagir com pessoas que ele já conhece da sua vida ("*social sharing*"), tal como parece ser o caso mais normal [Ellison et al., 2007, Lampe et al., 2006].

As funcionalidades oferecidas são outros pontos a ter em conta. Neste campo, o Facebook disponibiliza, além de uma área de partilha de informação e conteúdos multimédia (Figura 2.2), oferece também um serviço de conversação em tempo real, a incorporação de aplicações externas directamente nas páginas, etc. Estas funcionalidades, em conjunto com a possibilidade de integração de aplicações externas, são responsáveis por grande parte do sucesso do Facebook, que anunciou recentemente no seu blogue que atingiu a marca dos 500 milhões de utilizadores registados[3].

Figura 2.3: Página no LinkedIn de Linus Torvalds.

O serviço LinkedIn é análogo ao Facebook, no entanto está mais vocacionado para relações profissionais. Para tal, é permitido aos seus utilizadores fazer a distinção entre os seus contactos pessoais e os profissionais. Desta

[3] `http://blog.facebook.com/blog.php?post=409753352130`

forma, os utilizadores podem interagir com o mundo industrial, obter recomendações e aumentar a sua visibilidade, caso pretendam concorrer a um novo emprego. O LinkedIn é também uma boa opção para promover um novo negócio, concluindo assim que este é o serviço indicado quando o objectivo é estritamente profissional. Na Figura 2.3 podemos observar a página de Linus Torvalds no LinkedIn.

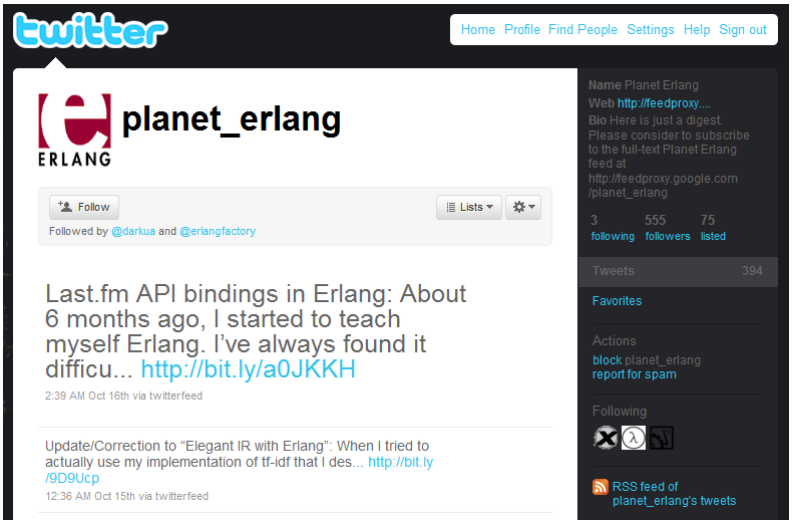

Figura 2.4: Página no Twitter do grupo Planet Erlang.

Um conceito diferente é o implementado pelo Twitter. Este é um serviço baseado em *micro-blogging*, onde os utilizadores são convidados a responder à questão "O que estás a fazer?", num máximo de 140 caracteres. Uma resposta a esta questão é chamada de *tweet* e os contactos do utilizador são denominados de *seguidores*. Quando um utilizador publica um *tweet*, a sua comunidade de seguidores é notificada em tempo real e é-lhes dada a possibilidade de responder ou reencaminhar essa informação. É também uma boa forma de uma entidade divulgar as suas iniciativas, como é o caso do grupo Planet Erlang[4] e cuja página pode ser visualizada na Figura 2.4.

2.1.3 Integração com aplicações externas - OpenSocial

Como forma de alargar o processo criativo até aos utilizadores, dando-lhes a possibilidade de criarem novas funcionalidades, a generalidade dos serviços de redes sociais online desenvolveram plataformas para integração de aplicações

[4] `http://www.planeterlang.org/`

externas. Estas plataformas estão normalmente sob a forma de API, tendo sido também desenvolvido um projecto com o objectivo de as uniformizar. Este projecto foi inicialmente desenvolvido pela Google chamado de OpenSocial [Mitchell-Wong et al., 2007]. É composto por um conjunto de especificações que ditam que tipo de informações as aplicações externas devem ter acesso e como esse acesso deve ser feito. As aplicações que estejam implementadas segundo estas especificações podem ser integradas em vários serviços, desde que estes também as sigam. O projecto OpenSocial, além das especificações, é também constituído por uma linguagem baseada em marcações, parecida com HTML [5], chamada OpenSocial Markup Language(OSML) e uma API remota. Esta linguagem é utilizada pelas aplicações externas para acesso a dados. A API está acessível pelo lado do cliente através de um ambiente de execução JavaScript denominado OpenSocial container. Pelo lado do servidor está acessível através de aplicações escritas nas mais variadas linguagens de programação. As redes sociais online mais populares que implementam o projecto OpenSocial são o LinkedIn, MySpace, hi5[6], NetLog[7] e o orkut[8].

Tendo em conta o caso específico do MySpace, as especificações do projecto OpenSocial fazem parte de uma plataforma de desenvolvimento própria, chamada de MySpace Open Plataform, ou MySpace Developer Plataform (MDP). Esta plataforma tenta seguir todas as especificações, no entanto existem algumas inconsistências, como é exemplo disso a interpretação do objecto *idspec*. Tem também algumas funcionalidades extra que não constam da especificação OpenSocial [Cole et al., 2010].

Embora não seguindo as especificações do projecto OpenSocial, as redes sociais Facebook e Twitter apostam também na integração de aplicações externas. Os recursos disponibilizados por estes sistemas, bem como os passos necessários para integrar as aplicações são descritos nas secções 2.1.6 e 2.1.4 respectivamente.

2.1.4 Integração de aplicações no Twitter

A rede social Twitter conta com mais de 105 milhões de utilizadores e cerca de 180 milhões de visitas diferentes por mês. Grande parte do seu tráfego, cerca de 75%, não é proveniente da sua página Web mas de aplicações exter-

[5] HyperText Markup Language
[6] `http://hi5.com/`
[7] `http://netlog.com/`
[8] `http://orkut.com/`

nas[9], processando as suas APIs remotas cerca de 3 mil milhões de pedidos diários. Estes números comprovam a dimensão deste serviço e a importância que as APIs remotas podem ter na popularidade de um serviço. Visto isto, justifica-se que estes serviços invistam não só em novas funcionalidades para a sua aplicação Web, mas também na possibilidade de essas e outras funcionalidades poderem ser acedidas através das suas APIs. Estas funcionalidades potenciam o desenvolvimento de novas aplicações externas capazes de as explorar de diferentes formas, chegando assim a mais utilizadores e aumentando a popularidade do serviço.

Tecnologias disponibilizadas

Contrariamente ao projecto OpenSocial, o Twitter não disponibiliza uma plataforma tão apetrechada de tecnologias aos programadores. Disponibiliza apenas APIs que podem ser invocadas remotamente e que permitem a interacção com o sistema. Estas APIs serão descritas com mais detalhe na secção 2.1.4.

Para facilitar a invocação dos métodos, foram entretanto desenvolvidas pela comunidade aplicações cliente em diversas linguagens de programação, tais como C++, C#, Erlang, Java, Perl, PHP e Ruby.

Requisitos

O Twitter aconselha todos os programadores que pretendam integrar as suas aplicações na plataforma Twitter a registarem-nas. Este passo não é obrigatório, excepto nos casos em que pretendam que as aplicações utilizem o protocolo OAuth[10] (apresentado num ponto mais avançado deste capítulo), visto ser nesta fase que são fornecidas os dados necessários ao processo de autenticação.

Caso os programadores não pretendam utilizar OAuth, podem aceder directamente aos métodos disponibilizados nas APIs da rede social.

APIs remotas

Contrariamenta ao Facebook que condensa os seus métodos em apenas uma API remota, o Twitter optou pela divisão em três APIs[11]: *Search API Methods*, *REST API Methods* e *Streaming API*. É possível ainda considerar uma divisão dos próprios métodos das APIs em grupos tendo em conta

[9] `http://www.huffingtonpost.com/2010/04/14/twitter-user-statistics-r_n_537992.html`

[10] `http://oauth.net/`

[11] `http://apiwiki.twitter.com/Twitter-API-Documentation`

a sua funcionalidade. A Figura 2.5 ilustra, de uma forma geral, as APIs disponibilizadas pelo Twitter.

Figura 2.5: APIs do Twitter e divisão dos métodos da *REST API Methods* em grupos.

A API *Search API Methods* permite às aplicações externas pesquisarem *tweets* com determinadas características. Permite também aceder aos temas mais mencionados nos *tweets*de um determinado dia, semana ou mês.

Por sua vez, a API *REST API Methods* é composta por um número bastante superior de métodos e, por essa razão, estes encontram-se divididos em grupos de acordo com a sua finalidade:

- ***Timeline Methods*:** Métodos de acesso aos mais recentes tweets feitos pelos utilizadores;
- ***Status Methods*:** Permite ao utilizador publicar o seu novo estado, ver os seus estados anteriores e reencaminhar um estado publicado por outro utilizador;

- ***User Methods*:** Métodos relacionados com o utilizador;
- ***List Methods*:** Métodos para criação, actualização e visualização de listas, entre outras operações;
- ***List Members Methods*:** Métodos que permitem o acesso a informações acerca dos membros de uma lista;
- ***List Subscribers Methods*:** Métodos que permitem o acesso a informações acerca dos assinantes de uma lista;
- ***Direct Message Methods*:** Métodos para criação, visualização envio e destruição de mensagens directas;
- ***Friendship Methods*:** Métodos para gestão dos amigos;
- ***Social Graph Methods*:** Permitem o acesso a um grafo com os amigos e os seguidores de um utilizador;
- ***Account Methods*:** Métodos relacionados com a conta do utilizador;
- ***Favorite Methods*:** Métodos que permitem a gestão dos favoritos;
- ***Notification Methods*:** Permite ao utilizador autenticado activar ou desactivar as notificações de um outro utilizador;
- ***Block Methods*:** Métodos para gestão dos utilizadores bloqueados;
- ***Spam Reporting Methods*:** Método para reportar um dado utilizador como *spammer*;
- ***Saved Searches Methods*:** Métodos para gestão das procuras gravadas efectuadas pelo utilizador;
- ***OAuth Methods*:** Métodos relacionados com a autenticação no sistema utilizando o protocolo OAuth (tema abordado ma secção 2.1.5);
- ***Local Trends Methods*:** Método para acesso aos temas mais abordados numa dada localização. Possui também um outro método que devolve a lista de localizações que é possível passar como parâmetro no método anterior.
- ***Geo Methods*:** Métodos que permitem a procura de locais que podem ser utilizados como referência nos tweets marcados geograficamente;

- ***Help Methods*:** Método de teste que pode ser utilizado para confirmar o estado do servidor;

Por último, a API *Streaming API* permite que uma aplicação externa possa aceder quase em tempo real aos *tweets* públicos, filtrados por palavra-chave ou utilizador.

Os dados devolvidos pelos métodos as três APIs podem vir codificados em XML[12], JSON[13], RSS[14] e Atom, sendo os dois primeiros os mais usuais. Não é no entanto garantido que todos os métodos implementem os quatro tipos de codificação.

2.1.5 OAuth

O processo de autenticação implementado nas APIs do Twitter é algo rudimentar, não existindo qualquer mecanismo de protecção ou encriptação dos dados no seu transporte. Este facto constitui uma falha de segurança grave. Sempre que uma aplicação invoque um método de uma API do Twitter que necessite de autenticação, os seus dados são expostos.

Muitas das aplicações integradas com a plataforma do Twitter têm como objectivo estender as suas funcionalidades. Para tal, é uma pré-condição que os seus utilizadores estejam registados na rede social. Não faz grande sentido que o uso destas aplicações obrigue a mais um registo por parte dos seus utilizadores. Este processo é trabalhoso, quer para os utilizadores, quer para a própria equipa de desenvolvimento das aplicações, que teriam o trabalho extra de implementarem serviços de armazenamento e protecção destes dados.

Para evitar esta replicação desnecessária de dados e colmatar a exposição dos dados utilizados no processo de autenticação, surgiu o protocolo OAuth. Inicialmente projectado para o caso do Twitter, mas que pode ser aplicado em qualquer sistema semelhante. O seu funcionamento passa por delegar o processo de autenticação ao próprio Twitter. Em vez de o utilizador ser obrigado a inserir os seus dados na própria aplicação, é-lhe apresentada uma hiperligação (normalmente com o aspecto da Figura 2.6) para uma página do Twitter e é aí que o utilizador se autentica. Depois de processada a autenticação, a aplicação é informada do sucesso desta operação e é-lhe passada uma chave que lhe permite utilizar as APIs de forma segura.

Este sistema permite ainda que, através do Twitter, o utilizador possa gerir quais as aplicações a que pretende dar acesso.

[12]Extensible Markup Language
[13]JavaScript Object Notation
[14]Really Simple Syndication

Figura 2.6: Aspecto normal da hiperligação para autenticação via OAuth no Twitter.

Tal como foi anteriormente referido, todos os serviços que pretendam estar integrados com o Twitter devem estar previamente registados na sua plataforma de aplicações. Caso contrário, não terão acesso às chaves de acesso que terão de utilizar para comunicar com a plataforma do Twitter.

2.1.6 Plataforma de aplicações do Facebook

A rede social Facebook é bastante popular, em grande parte devido à sua plataforma de aplicações. O grande número e variedade de aplicações existentes têm cativado os utilizadores. No entanto, a diversidade destas aplicações apenas é possível porque a plataforma disponibilizada pela rede social possui uma grande variedade de soluções e evolui a par das necessidades para a criação de novas funcionalidades.

A plataforma aplicacional do Facebook foi actualizada recentemente, passando a contemplar as mais recentes funcionalidades como o serviço de *chat* e os *plug-ins* sociais.

As tecnologias e funcionalidades disponibilizadas por esta nova versão da plataforma estão agrupadas em três grupos: Core APIs, Facebook SDKs e APIs Avançadas, tal como mostra a Figura 2.7.

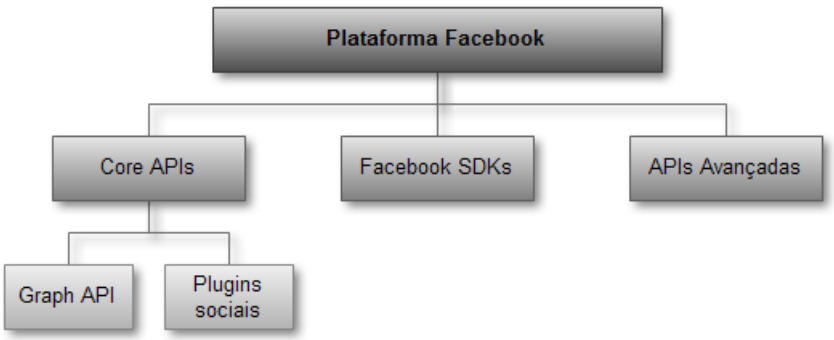

Figura 2.7: Componentes da plataforma de desenvolvimento do Facebook.

O grupo *Core APIs* contempla as principais APIs da plataforma. Neste

grupo é possível encontrar a *Graph API*, que contém todos os métodos responsáveis pelo acesso remoto às funcionalidades base da rede social, tais como informações pessoais e conteúdos multimédia. É ainda possível encontrar neste grupo a API capaz de processar os pedidos relacionados com os *plugins* sociais. Diversos SDKs[15] oficiais, conjuntos de ferramentas que permitem a criação de aplicações, foram também lançados nesta versão da plataforma, ficando agrupados no grupo *Facebook SDKs*. No grupo *APIs Avançadas* podemos encontrar diversas APIs, umas que provêm da versão anterior da plataforma e outras com objectivos mais específicos.

Após uma breve descrição geral sobre a plataforma, será abordado cada componente em particular e de forma mais aprofundada.

Graph API

A *Graph API* surge como uma forma de simplificar o acesso aos dados do Facebook. A forma como foi desenhada mostra de forma simples e consistente o grafo da rede social, representando uniformemente todos os seus objectos (pessoas, fotos) e as relações entre estes (relações de amizade, marcações nas fotos).

Os objectos que constituem o grafo social do Facebook são agrupados em: utilizadores, páginas, eventos, grupos, aplicações, mensagens, fotos, álbuns, imagens de perfil, vídeos, notas, ligações partilhadas e *posts*.

Cada objecto possui um identificador único, sendo desta forma fácil o acesso aos dados a si associados. Para tal basta aceder ao endereço *https://graph.facebook.com/ID*, sendo o ID o identificador do objecto em questão. Tomando como caso prático a página oficial da plataforma do Facebook, sendo *19292868552* o seu identificador, é possível aceder à sua informação pública através do endereço *https://graph.facebook.com/19292868552*. Além do identificador, cada objecto pode ter também um nome único associado e definido pelo utilizador responsável por manter o objecto. Este nome pode também ser utilizado nas invocações remotas em alternativa ao identificador.

O resultado de qualquer invocação remota direccionada à *Graph API*, é um objecto JSON. Neste caso em concreto é possível obter os seguintes dados:

```
{
  "name": "Facebook Platform",
  "type": "page",
  "website": "http://developers.facebook.com",
```

[15] Software Development Kit

```
   "username": "platform",
   "founded": "May 2007",
   "company_overview":
        "Facebook Platform enables anyone to build...",
   "mission": "To make the web more open and social.",
   "products":
        "Facebook Application Programming Interface (API)...",
   "fan_count": 449921,
   "id": 19292868552,
   "category": "Technology"
}
```


No que diz respeito às relações entre objectos, podemos encontrar no grafo social vários grupos de relações. A título de exemplo, algumas das relações suportadas entre um utilizador e uma página incluem: amigos, notas, marcações em fotos, *feed* pessoal, entre outras. O acesso remoto a estas relações é análogo ao descrito anteriormente para o caso dos objectos com a pequena diferença que é necessário acrescentar ao endereço o tipo de relação. Assim sendo o endereço genérico para aceder informações acerca de uma relação é *https://graph.facebook.com/ID/CONNECTION*, sendo *ID* o identificador do objecto e *CONNECTION* o tipo de relação.

É possível conhecer todas as relações que um objecto pode estabelecer através do processo de introspecção. Este processo é bastante útil na medida em que não é necessário ter conhecimento prévio acerca do tipo de objecto. Para tal, basta adicionar a variável "metadata=1" ao endereço do objecto na *Graph API*. Na prática e tendo em consideração o objecto que traduz o grupo dos utilizadores de Emacs no Facebook, temos que o um pedido direccionado ao endereço *https://graph.facebook.com/cocacola?metadata=1* tem como resposta o seguinte objecto JSON:


```
{
   "id": "2204501798",
   "name": "Emacs users",
   "description": "People who use the greatest,
        most efficient editor around the world.",
   "link": "http://www.gnu.org/software/emacs/",
   "privacy": "OPEN",
   "updated_time": "2006-07-17T10:23:21+0000",
   "metadata": {
      "connections": {
         "feed": "https://graph.facebook.com/2204501798/feed",
```

```
        "members":
           "https://graph.facebook.com/2204501798/members",
        "picture":
           "https://graph.facebook.com/2204501798/picture"
     }
  },
  "type": "group"
}
```

A partilha da localização geográfica está também presente na Graph API, quer nos objectos, quer nas relações entre eles. Desta forma é possível aceder aos utilizadores que partilharam a sua localização num dado local e aos locais partilhados por um dado utilizador, respectivamente. Estes métodos foram introduzidos na GraphAPI em conjunto com a funcionalidade Facebook Places, abordada na secção 2.3.1.

Plugins sociais

Os plugins sociais são uma funcionalidade recentemente introduzida pelo Facebook, tendo como objectivo estender as funcionalidades das redes sociais a outros sites Web. A título de exemplo, estas extensões permitem aos utilizadores visualizarem que sites é que os seus amigos gostaram e comentaram sem ter a necessidade de aceder ao Facebook. Para tal, os utilizadores apenas têm de adicionar no seu site uma linha de código HTML fornecida pelo Facebook. Um exemplo deste código pode ser observado algures.

```
<iframe
   src="http://www.facebook.com/plugins/like.php?href=..."
   scrolling="no"
   frameborder="0"
   style="border:none;
   overflow:hidden;
   width:450px;
   height:80px;"
   allowTransparency="true">
</iframe>
```

O resultado desta linha de código será semelhante ao encontrado na Figura 2.8.

Figura 2.8: Botão *gostar* - Plugin Social do Facebook

SDKs Facebook

Tal como já foi referido, esta nova versão da plataforma aplicacional do Facebook é também composta por uma lista de SDKs oficiais. Estes SDKs podem ser um auxílio precioso no processo de desenvolvimento, minimizando o esforço necessário por parte dos programadores na invocação remota das funcionalidades disponibilizadas pelas APIs. Os SDKs oficiais lançados cobrem as seguintes linguagens de programação: PHP, Python e Javascript e ainda as plataformas móveis iPhone e Android.

Na linguagem de programação Erlang não foi, até ao momento da escrita desta dissertação, possível encontrar nenhum SDK completo capaz de implementar as especificações da *Graph API*. Para colmatar tal falha foi desenvolvido o erlFBGraph[16], que será abordado na secção 4.1. Além destas linguagens, é possível encontrar SDKs desenvolvidos em muitas outras linguagens. No entanto e, tal como acontecia com os exemplos em Erlang encontrados, não é garantido que implementem a *Graph API*.

APIs avançadas

O grupo de APIs avançadas é composto por várias APIs, tal como mostra a Figura 2.9.

Tal como o projecto OpenSocial, já anteriormente descrito, a plataforma aplicacional do Facebook possui uma linguagem de acesso a dados baseada na linguagem SQL[17] e que é chamada de *Facebook Query Language*. Esta tecnologia é útil na medida em que permite interrogar a *Graph API* de uma forma mais avançada, isto é, permite por exemplo incluir num único pedido vários dados a que se pretende aceder. Os pedidos utilizando a linguagem FQL são enviados para o seguinte endereço: `https://api.facebook.com/method/fql.query?query=QUERY`, sendo que *QUERY* diz respeito à interrogação que se pretende processada. Num caso prático, *QUERY* poderia ser substituído por:

```
SELECT name FROM user WHERE uid = me()
```

[16] `http://github.com/ejbraga/erlFBGraph`
[17] Structured Query Language

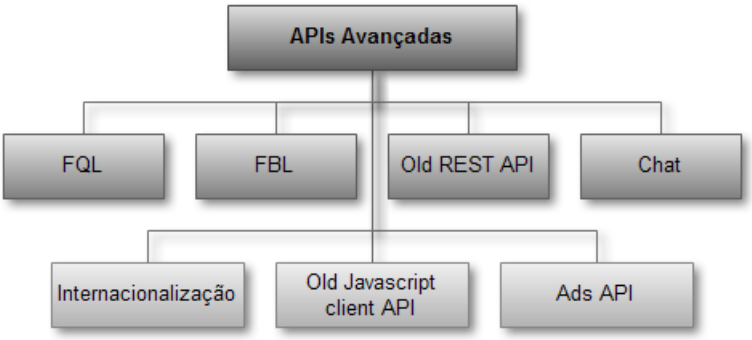

Figura 2.9: APIs Avançadas do Facebook

Esta *query* devolve o nome do utilizador que tem o valor do identificador igual ao do utilizador que se encontra autenticado no sistema, isto é, devolveria o nome do próprio utilizador. A resposta devolvida pela plataforma aplicacional do Facebook pode vir formatada em XML ou através de objectos JSON, sendo o programador o responsável por identificar qual o tipo de formatação que prefere.

Além desta linguagem, o grupo de APIs avançadas disponibiliza ainda uma outra, a *Facebook Markup Language* (FBML), que permite a integração de aplicações de uma forma facilitada. No entanto, esta tecnologia é desaconselhada visto que faz parte da versão anterior da plataforma aplicacional. Em contrapartida, o Facebook aconselha o uso do SDK desenvolvido em Javascript que já foi apresentado no ponto anterior. Este SDK substitui também o *Old Javascript client API*.

Tal como as tecnologias FBML e *Old Javascript client API*, a *Old REST API* é também desaconselhada. Esta API consiste num Web service do tipo REST[18] e foi, até ao aparecimento da *Graph API*, a interface disponibilizada pelo Facebook para interacção com aplicações externas. No entanto, tal como as outras, continua a ser suportada por questões de compatibilidade com aplicações já existentes. Embora seja desaconselhada, é feita uma descrição mais detalhada no ponto 2.1.6, uma vez que continua a ser bastante utilizada.

Ainda no grupo das APIs avançadas podemos encontrar uma API que permite a integração de aplicações Web, *desktop* ou móveis com o serviço de conversação do Facebook. Esta integração é feita via protocolo XMPP/Jabber [Saint-Andre, 2005a], embora com algumas limitações (por exemplo ao nível da presença e não permite que sejam adicionados novos contactos).

[18]Representational State Transfer

Já foi referido que a rede social Facebook tem milhões de utilizadores espalhados por todo o mundo. Para que a língua não seja um entrave à utilização dos seus serviços, o Facebook está disponível em mais de 70 idiomas[19]. Este feito foi conseguido através de uma plataforma que permite que seja a própria comunidade a traduzir os textos. Esta plataforma surge sob a forma de *plugin* social e é possível integra-lo em qualquer site, passando este também a usufruir dos seus benefícios.

Por último, a *Ads API*. Esta API encontra-se ainda em fase beta mas permite a gestão dos anúncios publicitários do Facebook. É indicada para utilizadores que representem empresas de publicidade e utilizadores detentores de vários anúncios. Utilizando esta API é possível criarem aplicações personalizadas que tornem mais fácil a gestão de todos os seus anúncios na rede social Facebook.

Old REST API

A *Old REST API* era provavelmente, antes do lançamento da *Graph API*, a funcionalidade com maior relevo disponibilizada pela plataforma Facebook. No entanto, e tal como foi já referido, esta API apenas é ainda disponibilizada por questões de compatibilidade para com as aplicações já existentes e que estão programadas para interagir com esta interface.

Os métodos que compõem esta API estão divididos em oito grupos, tal como ilustra a Figura 2.10, de acordo com a sua finalidade:

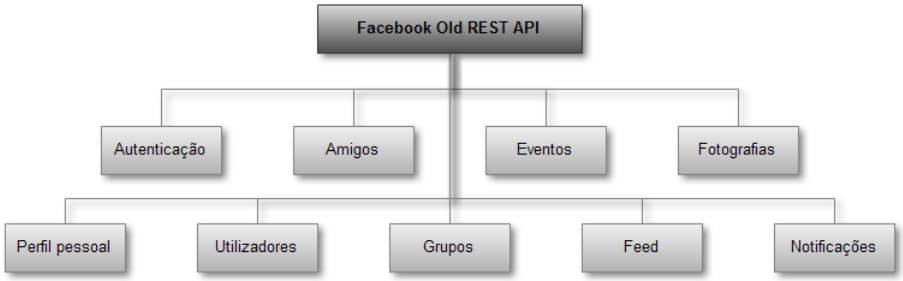

Figura 2.10: Divisão da Old REST API do Facebook por funcionalidades.

- ***facebook.auth*:** Oferece métodos de autenticação básicos para utilizadores do Facebook;

[19] `http://www.facebook.com/translations/FacebookLocales.xml`

- ***facebook.feed*:** Permite aos utilizadores publicarem novidades na feed do Facebook;
- ***facebook.friends*:** Disponibiliza métodos para controlo dos amigos de um utilizador;
- ***facebook.notifications*:** Permite o envio de mensagens para outros utilizadores;
- ***facebook.profile*:** Permite a inclusão de FBML no perfil do utilizador;
- ***facebook.users*:** Disponibiliza métodos para acesso a informações acerca dos utilizadores, por exemplo o conteúdo do seu perfil;
- ***facebook.events*:** Permite o acesso aos eventos dos utilizadores;
- ***facebook.groups*:** Disponibiliza métodos para acesso a informações acerca de grupos;
- ***facebook.photos*:** Permite a interacção com as fotos dos utilizadores.

Na sua essência, estes métodos traduzem-se em interacções FQL mais sofisticadas na plataforma Facebook, no entanto, esta subida no nível de abstracção permite aos programadores agilizarem a criação de aplicações. Para facilitar ainda mais o processo de desenvolvimento, esta API conta com uma série de aplicações cliente escritas nas mais variadas linguagens de programação, tais como, ActionScript, ASP.NET, C++, C#, Erlang, Java, PHP, Python, Ruby e VB.NET.

Requisitos

A plataforma do Facebook permite a integração de aplicações Web e *desktop*. Para este último caso, é apenas necessária uma conta Facebook válida. Para o caso de uma aplicação Web é também necessário acesso a um servidor Web. Após a criação da conta Facebook (preferencialmente do tipo *developer*), deve-se proceder ao registo da nova aplicação na plataforma Facebook, indicando o seu tipo, identificação e outros parâmetros relevantes.

Nos dois casos, com o intuito de facilitar o processo de desenvolvimento deve-se recorrer a uma aplicação cliente que implemente os métodos remotos da API do Facebook.

Parâmetros da aplicação

No acto de registo de uma nova aplicação Web devem ser preenchidos os seguintes parâmetros:

- ***Callback URL*:** URL da aplicação. Caso a aplicação esteja alojada em `http://dominido.falso.pt/facebook_app`, esta é a URL que deve ser inserida neste parâmetro;
- ***Canvas Page URL*:** URL disponibilizada para os utilizadores do Facebook acederem à aplicação;
- **Support *E-mail*:** Email de contacto para suporte da aplicação;
- ***Application Type*:** Tipo de aplicação, isto é, aplicação Web ou desktop;
- ***IP Addresses of Servers Making Requests*:** Lista de ips, separados por vírgulas, que podem aceder à aplicação;
- ***Can your application be added on Facebook?*:** Responder afirmativamente caso se pretenda permitir aos utilizadores adicionarem a aplicação aos seus perfis, ou negativamente caso não seja o caso;
- ***TOS URL*:** URL com os termos de utilização da aplicação. Caso este parâmetro seja configurado, os utilizadores terão de aceitar previamente os termos de utilização antes que possam usufruir da aplicação;
- ***Developers*:** Nome das pessoas que desenvolveram o projecto.

Outros parâmetros podem ser configurados, no entanto são estes os mais relevantes.

Integração de aplicações *desktop*

O processo de integração de aplicações *desktop* com a plataforma Facebook é relativamente simples e está visualmente descrito na Figura 2.11. Este processo é constituído por dois intervenientes: o servidor do Facebook e a aplicação cliente. A interacção é sempre iniciada pelo cliente que interroga a API (1). Por sua vez, o servidor processa o pedido e retorna o resultado como um objecto JSON (2).

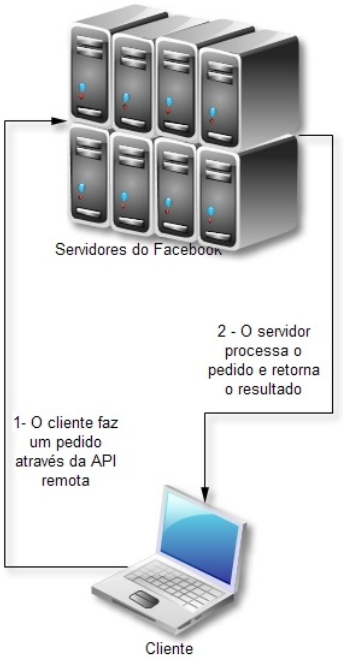

Figura 2.11: Integração de aplicações *desktop* na plataforma Facebook.

Integração de aplicações Web

No que diz respeito às aplicações Web, o processo é mais complexo havendo, não dois mas três intervenientes: utilizador da aplicação (cliente), o servidor do Facebook e ainda a própria aplicação Web, alojada num servidor Web.

O endereço da aplicação é configurado no parâmetro *Callback URL* e é o conteúdo deste endereço que o Facebook irá importar para a região *canvas page* da aplicação. A região *canvas page* está indicada na Figura 2.12.

A interacção, apresentada visualmente na Figura 2.13, é iniciada quando o utilizador despoleta uma acção (1). Esta acção é traduzida num pedido à plataforma do Facebook que o reencaminha para a aplicação Web (2). Caso haja essa necessidade, a aplicação recorre aos métodos remotos da API (3), que lhe retornam os resultados (4). Após ter processado o pedido, a aplicação constrói uma página Web, recorrendo ou não à linguagem FBML e envia-a para o servidor do Facebook (5). Este servidor por sua vez processa a página recebida e envia-a em formato HTML para o utilizador (6).

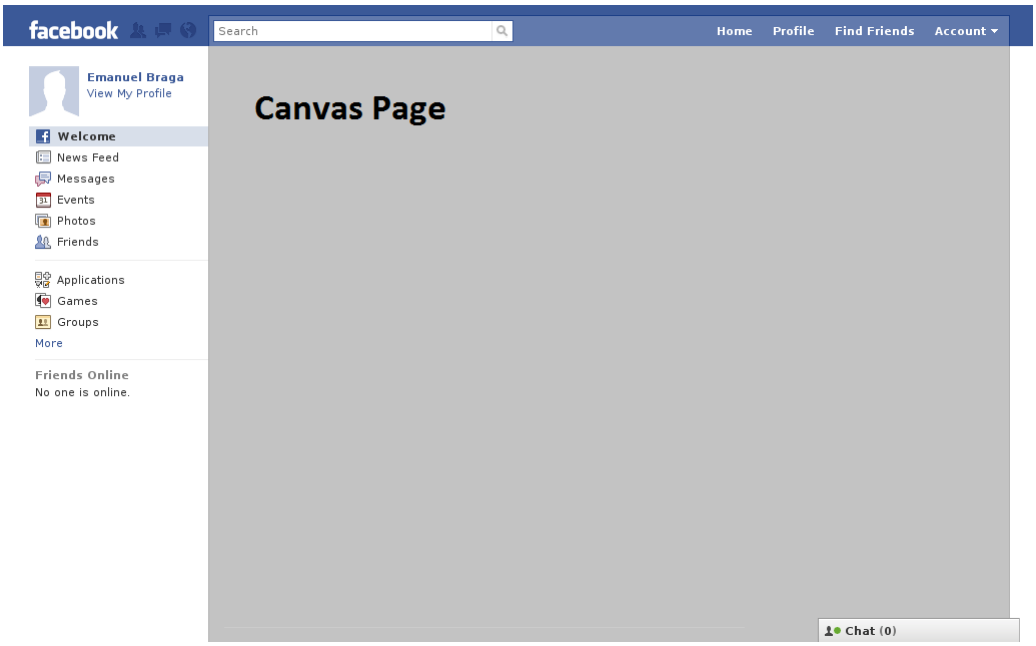

Figura 2.12: Aspecto genérico de uma página do Facebook.

FBML vs Iframe

Ainda que actualmente o uso da linguagem FBML seja desaconselhada em detrimento da utilização de *iframes*, é efectuada neste ponto da dissertação uma comparação entre as duas tecnologias.

A linguagem FBML permite uma série de abstracções ao desenvolvedor, no entanto obriga-o a estudar mais uma linguagem, o que pode atrasar o processo de desenvolvimento. Em alternativa, estas páginas podem ser integradas através de iframe, podendo o desenvolvedor utilizar as linguagens em que estiver mais à vontade.

Além do que já foi referido, existem outras diferenças que podem fazer pender a balança na altura de escolher entre FBML e iframes[20]. Podemos considerar os seguintes factores favoráveis à linguagem FBML:

- Permite o desenvolvimento rápido de aplicações a partir do nada, bom para iniciantes;
- Permite um tempo mais rápido no carregamento das páginas pela primeira vez;

[20] `http://wiki.developers.facebook.com/index.php/Choosing_between_an_FBML_or_IFrame_Application`

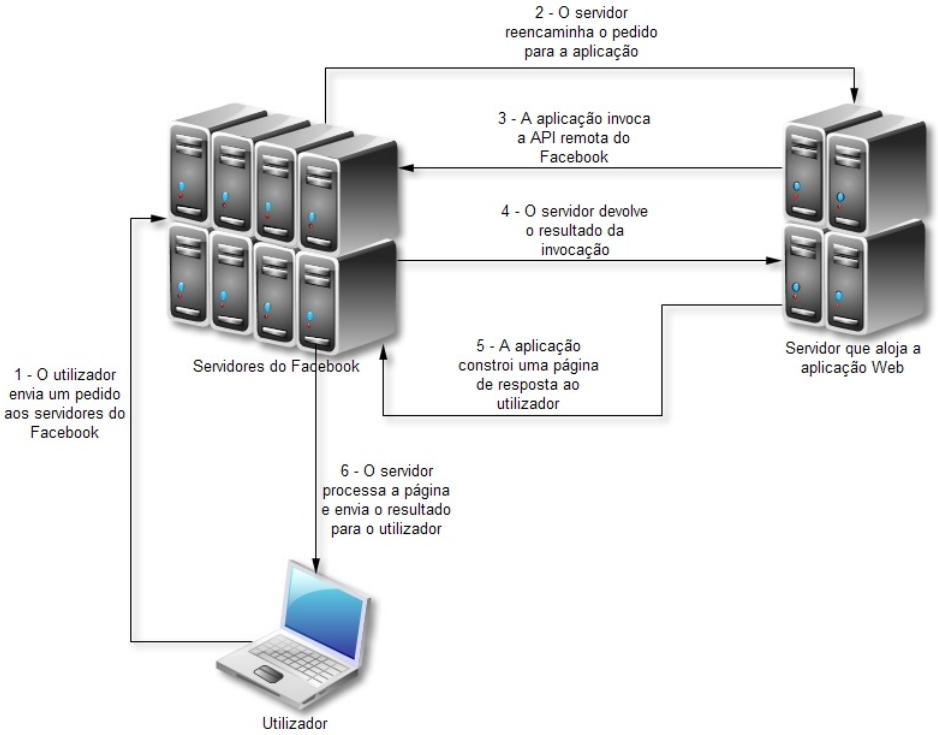

Figura 2.13: Integração de aplicações Web na plataforma Facebook.

- O paradigma de desenvolvimento está próximo do da Web normal;
- Permite o acesso de forma facilitada a diversas funcionalidades do Facebook;
- Permite uma mais fácil gestão das URLs das páginas que compõe a aplicação;
- Possui um mecanismo sensível para controlo de acessos;

Em relação às iframes podemos apontar os seguintes aspectos:

- Permite um desenvolvimento mais rápido nos casos em que se pretenda adaptar uma aplicação já existente;
- São susceptíveis a conduzir o utilizador a uma melhor usabilidade a longo prazo;

- Permite a utilização de Javascript, HTML e CSS[21];
- Permite uma maior fluidez na usabilidade das páginas caso se utilize muito AJAX[22], dado que estes pedidos não têm de passar pelo proxy do Facebook;
- A depuração de código HTML e Javascript é mais simples do que de código FBML e FBJS;

Tendo em conta estes factores, é possível concluir que a linguagem FBML seria, caso ainda fosse aconselhada, a eleita quando se pretende desenvolver uma aplicação de raiz e com muitas funcionalidades relacionadas com o próprio Facebook. Visto ser desaconselhada, é proposto aos programadores de novas aplicações a utilização de *iframes* em conjunto com a nova versão do Javascript SDK.

Autorização

Estando em causa o acesso a dados e conteúdos pessoais, os processos de autorização e autenticação merecem especial atenção por parte da plataforma aplicacional do Facebook. O processo de autorização tem início quando o utilizador da rede social manifesta interesse em utilizar uma aplicação. Nesse momento, o utilizador é redireccionado para uma página, semelhante à que é possível observar na Figura 2.14 em que lhe é pedida autorização para que os seus dados sejam tornados públicos. Importa salientar que ao autorizar este pedido, o utilizador não disponibiliza os dados para todas as aplicações, apenas para a aplicação que manifestou interesse em utilizar.

Por defeito e depois de ter sido pedida autorização ao utilizador, uma aplicação pode aceder às informações contidas no seu perfil pessoal, como sendo o seu nome, fotografia, sexo e lista de amigos. Para casos em que as aplicações necessitem de estender as suas permissões para outro tipo de informações privadas, estas devem indicar especificamente quais. A lista completa das permissões pode ser consultada no página oficial da documentação da plataforma[23]. Este sistema de autorização permite ainda que o utilizador possa gerir as aplicações que utiliza e a que dados têm acesso.

Autenticação

No que diz respeito ao processo de autenticação, o Facebook, tal como o Twitter, optou por implementar o protocolo OAuth 2.0 nesta nova versão da

[21] Cascading Style Sheets
[22] Asynchronous Javascript And XML
[23] `http://developers.facebook.com/docs/authentication/permissions`

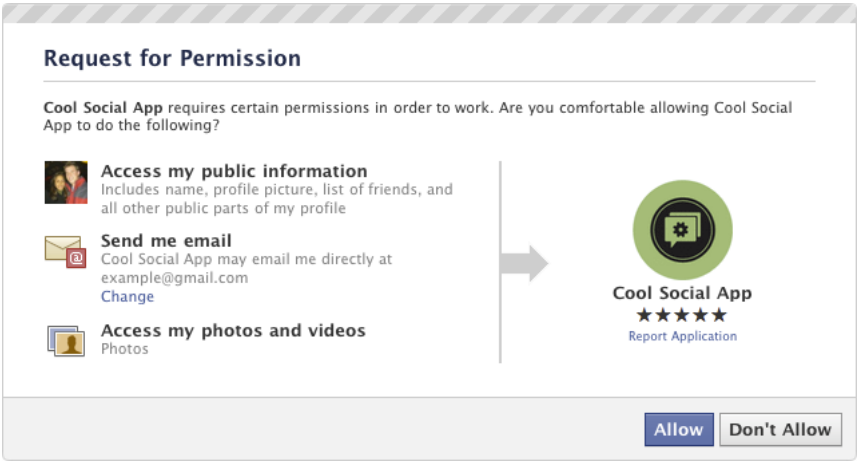

Figura 2.14: Exemplo de uma página de autorização de aplicações do Facebook

plataforma. Este sistema permite a autenticação Web, através de Javascript, *desktop* e mesmo para aplicações moveis, através dos SDKs para iPhone e Android.

Este processo é diferente quando se trata da autenticação de um utilizador ou de uma aplicação. No caso de se tratar de um utilizador, existe ainda a distinção quando se trata de uma aplicação Web. Neste caso, a autenticação do utilizador é feita directamente numa página do Facebook, sendo apenas necessário à aplicação Web redireccionar o utilizador para o seguinte endereço:

```
https://graph.facebook.com/oauth/access_token?
    client_id=...&
    redirect_uri=http://www.example.com/oauth_redirect&
    client_secret=...&
    code=...
```

As variáveis *client_id*, *client_secret* e *code* são as credenciais da aplicação e *redirect_uri* é a página para onde o utilizador deve ser redireccionado após o processo de autenticação.

O processo de autenticação de utilizadores através de aplicações *desktop* implica, normalmente, a integração de um *browser*. Esta integração é necessária para que o utilizador possa inserir o seu *username* e *password*. Este processo é assim semelhante ao registado para o caso das aplicaçõies Web.

Nos casos em que as aplicações não são Web, o processo de autenticação é efectuado enviando um pedido análogo ao seguinte:

```
curl -F type=client_cred
    -F client_id=...
    -F client_secret=...
    https://graph.facebook.com/oauth/access_token
```

Mais uma vez, as variáveis *client_id* e *client_secret* correspondem às credenciais da aplicação. Neste caso, a chave de acesso indentifica a própria aplicação, não existindo um utilizador associado à sessão. Esta limitação implica que as aplicações apenas tenham acesso a informações básicas, independentemente das permissões dadas pelo utilizador.

O resultado de qualquer processo de autenticação bem sucedido é um *token* temporário, que terá de ser sempre passado como parâmetro em cada interrogação à plataforma aplicacional. O transporte deste token, tal como da restante informação trocada entre as aplicações e a plataforma do Facebook não é sujeito a qualquer tipo de encriptação. Os dados necessários para que uma aplicação se possa autenticar são fornecidos aquando do seu registo.

2.2 Sensibilidade ao contexto

Ao longo desta secção são dadas definições ao conceito de contexto e de aplicação sensível ao contexto. É também explorada uma estratégia de uniformização do contexto, sendo que pode ser composto por informações bastante variadas. Por fim, são apresentadas várias aplicações deste conceito, estando algumas delas integradas com alguns serviços de redes sociais.

2.2.1 O que é o contexto

O termo contexto tem uma das suas melhores definições associadas a Dey [Dey, 2001]. É então descrito como *qualquer informação que possa ser usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, um espaço ou qualquer objecto que pode ser considerado relevante para a interacção entre o utilizador e a aplicação, incluindo o próprio utilizador e aplicação.* A captura deste tipo de informação, a que chamamos de contexto, está actualmente muito associada a dispositivos móveis. Nos últimos anos, estes dispositivos têm sofrido uma grande evolução ao nível do *software* e *hardware*, sendo possível que sejam embebidos mais e melhores sensores, capturando de forma mais eficiente o contexto do utilizador. Como são dispositivos essenciais no quotidiano da generalidade das pessoas, os telemóveis possuem também um estatuto privilegiado estando sempre junto do utilizador. Por consequência podem capturar informação sempre actualizada, sem que a pessoa sinta a necessidade de transportar dispositivos adicionais.

As informações de contexto mais comummente utilizadas pelas aplicações são a localização geográfica, a actividade actual e futuras, o tempo e a identidade do próprio utilizador e daqueles que lhe estão na sua proximidade [Gregory D. Abowd, 1999]. O conjunto destas informações que são captados directamente podem ser catalogados como informações primárias. Este tipo de informação pode posteriormente ser processado e, como consequência, podem ser inferidos outro tipo de informações, informações secundárias. Desta forma podemos, por exemplo, saber se é possível comparecer em determinado evento, tendo em conta a agenda do utilizador e a sua localização geográfica. Esta divisão do tipo de informação contextual como sendo primária e secundária potencia uma gestão mais eficaz de todos os dados.

2.2.2 O que são aplicações sensíveis ao contexto

As aplicações que utilizam informações do contexto do utilizador são normalmente denominadas de aplicações sensíveis ao contexto[24]. A sua definição original pertence a Schilit [Schilit et al., 1994] que definiu este tipo de serviços como aqueles que são capazes de se moldar dinamicamente de acordo com a localização onde são usados, o conjunto de pessoas e objectos próximos, bem como as alterações destes dados ao longo do tempo. Desta forma, este tipo de aplicações pode oferecer serviços mais adequados, tendo em conta as condições em que o utilizador se encontra e as suas actividades e futuras.

No entanto, esta definição de Schilit exclui as aplicações que apenas captam a informação do contexto do utilizador e não se adaptam de acordo com ele. A ideia de que estas aplicações também devem ser designadas por sensíveis ao contexto é defendido em [Gregory D. Abowd, 1999] e, é dada uma definição alternativa para este conceito. A definição refere que um sistema sensível ao contexto tem a capacidade de utilizar o contexto, com o objectivo de fornecer informações ou serviços relevantes ao utilizador. A relevância de cada informação ou serviço é relativa a cada utilizador.

Os mesmos autores que defendem esta visão acerca das aplicações sensíveis ao contexto, defendem também três aspectos que este tipo de sistemas devem suportar:

- Apresentação de informação de contexto ao utilizador;
- Execução automática de um serviço;
- Marcação da informação do contexto para utilização futura.

[24] ou "context-aware applications" na língua inglesa

A interacção entre o utilizador e as aplicações sensíveis ao contexto pode seguir várias estratégias. Chen e Kotz [Chen and Kotz, 2000] propuseram a classificação das aplicações tendo em conta esta interacção como sendo activas e passivas. Nas aplicações activas, é o próprio sistema que decide que acções tomar tendo em conta as alterações no contexto do utilizador. Em contraposição, nas aplicações passivas a escolha é feita pelo utilizador, através de uma lista de possíveis acções. Numa classe intermédia podemos ter as aplicações personalizáveis. Estas dão a liberdade aos utilizadores de especificarem as suas preferências sobre que comportamento as aplicações devem adoptar em dadas situações. Louise Barkhuus e Anind Dey [Barkhuus and Dey, 2003] realizaram um estudo sobre qual a preferência dos utilizadores que mostrou que os utilizadores sentem uma falta de controlo em relação às aplicações mais activas. No entanto estas aplicações, em conjunto com as passivas, reúnem mais popularidade em relação às personalizáveis. Os autores concluíram desta forma que os utilizadores preferem perder parte do controlo das aplicações, desde que a utilidade destas assim o justifique.

2.2.3 Modelação da informação de contexto

Como é possível constatar pela sua definição, o conceito de contexto pode englobar uma imensa variedade de informações. Normalmente, as aplicações sensíveis ao contexto utilizam apenas uma pequena parte do que pode ser considerado o contexto do utilizador, especificando essa informação da forma mais conveniente para o caso específico. No entanto, caso se pretenda um sistema genérico e capaz de interagir com outros sistemas, a uniformização da especificação dada a cada tipo de informação pode ser uma boa prática a adoptar.

No sentido de uniformizar as informações relativas ao contexto, foi proposto em [Jang et al.,] um paradigma de modelação centrado no utilizador passível de ser partilhado entre serviços. Neste estudo, o contexto é dividido em quatro subtipos de contexto, tal como mostra a Figura 2.15, tendo em conta a origem e o objectivo da informação.

O tipo de informação que é inserido em cada subtipo é definido da seguinte forma:

- **Contexto preliminar:** informação factual acerca dos utilizadores retirada directamente dos sensores;

- **Contexto integrado:** informação mais precisa e obtida pela fusão de vários contextos preliminares;

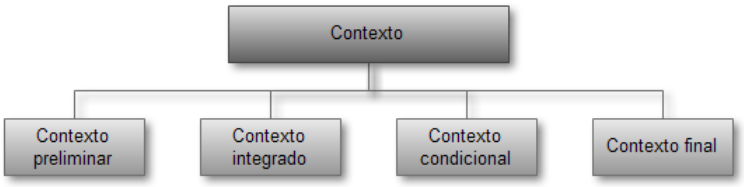

Figura 2.15: Divisão do contexto em subtipos.

- **Contexto condicional:** informações especificadas pelos próprios utilizadores sobre as acções que devem ser espoletadas quando forem detectadas certas condições no seu ambiente;
- **Contexto final:** despoleta uma acção sempre que existir uma ocorrência entre o contexto condicional e o contexto integrado do utilizador.

Existe também uma distribuição das informações, que por vezes podem ser bastante complexas, em seis categorias [Jang et al.,]: quem, o quê, onde, quando, como e porquê[25]. Cada categoria possui uma especificação concreta acerca de quais os parâmetros e a forma em como estes devem ser integrados.

Em jeito de resumo, a utilização de um paradigma de modelação como este cria uma abstracção sobre o tipo de informações e sobre a forma em como estas foram captadas pelos sensores. Com isto, é possível a partilha do contexto entre serviços e também alterar os próprios sensores ou a forma como eles funcionam, sem que haja necessidade de alterar os próprios serviços.

2.2.4 Fontes de contexto

A captação do contexto do utilizador é o ponto que despoleta todas as restantes acções das aplicações sensíveis ao contexto. Esta informação pode ser captada das mais variadas formas, podendo a mais adequada variar de situação para situação. Tal como o tipo de fontes, a precisão que lhes é associada é também um factor importante. Para aumentar a exactidão de uma informação é aconselhado o cruzamento de informações provenientes de vários sensores.

Pela sua portabilidade, pela constante evolução e por normalmente estarem sempre junto dos utilizadores, os telemóveis são, hoje em dia, provavelmente a maior fonte de informação contextual. No entanto esta não é a única fonte e, tendo em consideração a quantidade de informação pessoal

[25]na terminologia inglesa: Who, Where, When, What, How e Why

que possuem e a sua recente abertura em relação ao contexto, fazem dos serviços de redes sociais online uma grande fonte de informação que pode ser considerada como contexto.

2.2.5 Mobile Context Framework

A plataforma Mobile Context Framework (MCF) [Oliveira, 2009] é um tipo diferente de fonte de informação contextual.

Após a sua instalação num dispositivo móvel, a acção da MCF passa pela obtenção do contexto do utilizador e pela sua disponibilização a aplicações Web. Permite assim que as aplicações Web que o utilizador esteja a aceder, através do seu dispositivo móvel, tenham acesso a estas informações. Desta forma, estas podem oferecer um serviço adequado ao seu contexto.

As aplicações Web podem utilizar a plataforma MCF de duas formas distintas:

- através de um servidor remoto, actualizado por uma aplicação MCF;
- através de um servidor Web instalado no próprio dispositivo móvel.

Utilizando esta plataforma, é possível a criação de aplicações Web sensíveis ao contexto sem que haja a preocupação da obtenção directa do contexto do utilizador.

O protótipo apresentado juntamente com a plataforma MCF, o *Cinema Mobile Tickets* consiste numa aplicação Web que permite aos utilizadores que, a partir dos seus dispositivos, possam comprar bilhetes de cinema. O contexto utilizado neste protótipo passa por consultar a agenda do utilizador, a sua lista de contactos e a sua localização geográfica. O primeiro parâmetro é relevante para indicar a sua disponibilidade nos horários em que o filme é exibido. Após a compra do bilhete, a sua agenda é também actualizada com o evento. A integração da sua lista de contactos permite que possa comprar bilhetes não só para si, como também para os seus amigos. Por último, a localização geográfica é utilizada para determinar quais as salas de cinema mais próximas e que passam o filme escolhido pelo utilizador.

2.2.6 Exemplos de aplicações existentes sensíveis ao contexto

Desde o seu aparecimento, o conceito de sensibilidade ao contexto foi aplicado em diversos sistemas de software de várias áreas.

Lembranças

A área das lembranças foi uma das primeiras a ser explorada através do Forget-me-not [Lamming and Flynn, 1994], em 1994. O objectivo principal deste sistema é ajudar os seus utilizadores a não esquecerem as suas tarefas do quotidiano, tais como pagar contas ou saber as coordenadas do lugar de estacionamento. Esta área foi também explorada com a aplicação ComMotion [Marmasse and Schmandt, 2000] que é capaz de lembrar os utilizadores de coisas como a sua lista de compras sempre que estes estejam próximos de uma loja ou supermercado. Os aspectos a serem lembrados podem ser adicionados manualmente, mas a grande vantagem deste sistema é a possibilidade de subscrição de conteúdos Web acerca desses locais. Mais tarde, em 2005, surgiu um sistema chamado Place-its [Phones et al., 2005]. Tal como os sistemas apresentados anteriormente, este sistema funciona sobre dispositivos móveis. No entanto é um sistema mais completo, visto que possui mecanismos capazes de prever actividades dos utilizadores através da análise das actividades passadas. Estes sistemas mostram a sua utilidade quando o utilizador pretende ser lembrado oportunamente sobre os mais variados aspectos do seu quotidiano, podendo substituir itens como blocos de notas ou listas de tarefas.

Turismo

Os sistemas C-MAP [Sumi et al., 1998] e George Square [Brown et al., 2005] são exemplos de como a sensibilidade ao contexto pode também ser um tema interessante quando aplicado na área do turismo. A primeira aplicação permite aos turistas serem notificados com informação histórica acerca tendo em conta a sua localização e interesses pessoais. Por sua vez, a segunda aplicação possibilita aos seus utilizadores a partilha de experiências pessoais e conteúdos multimédia acerca de monumentos, através de Tablet PCs.

Localização geográfica

A localização geográfica das pessoas é outra das áreas bastante exploradas. Um dos sistemas pioneiros foi o Active Badges [Hopper et al., 1993] que foi desenvolvido com o objectivo de localizar pessoas no interior de edifícios. Mais tarde foi apresentado o Reno [Smith et al., 2005], uma aplicação que corre sobre dispositivos móveis que permite aos seus utilizadores partilharem a sua localização. Posteriormente foi lançada uma aplicação análoga chamada Whereabouts Clock [Brown et al., 2007]. O seu objectivo é permitir a localização dos elementos da família, sabendo assim o utilizador se eles estão em casa, na escola ou no trabalho. Actualmente existem aplicações

comerciais com o objectivo comum da partilha da localização. Os sistemas mais populares são o Loopt[26], o Mologogo[27] e o Disney Family Locator[28], que apenas opera dentro da DisneyLand.

Pesquisa de locais e serviços

É também possível encontrar aplicações sensíveis ao contexto que permitam a pesquisa de locais ou serviços tendo como base a localização geográfica dos utilizadores. O sistema AroundMe[29] é gratuito e permite aos seus utilizadores encontrar serviços, desde cinemas a supermercados, dentro de uma determinada distância. No entanto, este serviço tem a desvantagem de estar apenas disponível para iPhone. A dependência de plataforma é uma desvantagem não só desta aplicação mas da generalidade das aplicações móveis. O serviço Flixster[30] é também específico da plataforma iPhone e pode ser considerado um bom complemento para o AroundMe. Este é um sistema direccionado para a área dos cinemas, capaz de informar os utilizadores acerca de quais os filmes que estão em exibição na sua região. Esta informação vem acompanhada das sinopses, classificações, entre outras informações. Para a plataforma Android existe um serviço análogo chamado Movie Finder[31]. O Google integrou na sua página Web móvel um serviço com funcionalidades análogas, sendo este projecto denominado de Google Near me Now[32].

Além da desvantagem de não serem multi-plataforma, algumas destas aplicações têm também a contra-partida de estarem apenas disponíveis para certas regiões ou países. Um exemplo é o sistema UrbanSpoon[33], que permite aos seus utilizadores pesquisarem sobre os restaurantes mais próximos de si, mas que está apenas disponível nos Estados Unidos da América.

2.3 Integração do contexto em redes sociais

A integração de aplicações sensíveis ao contexto com os serviços de redes sociais online tem sido uma área bastante explorada.

[26] http://loopt.com
[27] http://www.mologogo.com/
[28] http://disneymobile.go.com
[29] http://www.tweakersoft.com/mobile/aroundme.html
[30] http://www.flixster.com/
[31] http://www.ikamobile.com/moviefinder/
[32] http://googlemobile.blogspot.com/2010/01/finding-places\
\-near-me-now-is-easier.html
[33] http://www.urbanspoon.com

Social Serendipity

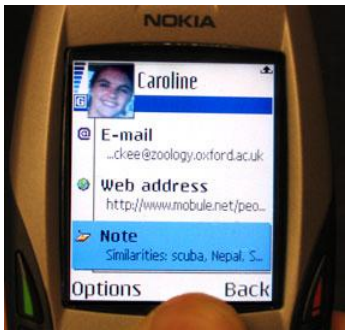

Figura 2.16: Aplicação Social Serendipity num dispositivo móvel.

Em 2005, foi apresentado o Social Serendipity [Eagle and Pentland, 2005] que é por si só uma rede social sensível ao contexto. Tem a capacidade de calcular um coeficiente de afinidade entre duas pessoas que estejam geograficamente próximas, tendo como base os seus perfis. Sempre que este coeficiente atinja um dado valor, as pessoas visadas são alertadas para a presença de alguém possivelmente interessante. A localização dos utilizadores, bem como a interacção com o sistema é feito através de uma aplicação móvel, cujo aspecto pode ser observado na Figura 2.16.

CenceMe

A partilha do contexto entre utilizadores de redes sociais é o objectivo do CenceMe [Cen, 2007], disponível actulmente para a plataforma iPhone (Figura 2.17). Com este sistema é possível ao utilizador partilhar as actividades que está a realizar, o seu sentido de humor, os seus hábitos e qualquer variável do ambiente que o rodeia.

O conceito do CenceMe é baseado em dois outros sistemas: o iCAMS [Nakanishi et al., 2002] e o WatchMe [Marmasse et al., 2004] mas pretende ser mais evoluído, não estando possuindo requisitos ao de hardware específico e principalmente pela forma como captura o contexto. Esta captura é feita através de sensores físicos e pela capacidade de cálculo de padrões nas actividades diárias dos utilizadores.

WhozThat

Em 2008, foi desenvolvido o WhozThat [Beach et al., 2008], cujo objectivo principal é ajudar o utilizador na questão “Quem é aquele?”, em relação a

Figura 2.17: Aplicação CenceMe para iPhone.

um outro que esteja próximo. A resposta é encontrada através da troca de identificadores via Bluetooth, que posteriormente são utilizados para pesquisa de informações pessoais relevantes em redes sociais como Facebook, MySpace ou LinkedIn. Este sistema pretende disponibilizar um verdadeiro ecossistema para a criação de aplicações sensíveis ao contexto. Estas têm a capacidade de reagir de acordo com os perfis das pessoas presentes num dado local. Existe já uma aplicação criada sobre este sistema e chamada de WZPlaylistGen [Beach et al., 2008]. Pode ser instalada, por exemplo, num bar e permite a geração de listas de músicas tendo em conta as preferências dos clientes presentes no estabelecimento. Estes clientes apenas têm de instalar a aplicação nos seus dispositivos móveis e configurar o seu perfil, por exemplo no Facebook. A informação sobre as músicas e artistas é pesquisada em serviços Web como MusicBrainz[34], freeDB[35] e Last.fm[36], tal como as próprias músicas descarregadas de AudioScrobbler[37].

[34] http://musicbrainz.org/
[35] http://www.freedb.org/
[36] http://www.last.fm/
[37] http://www.audioscrobbler.net/data/webservices/

PersonalTV

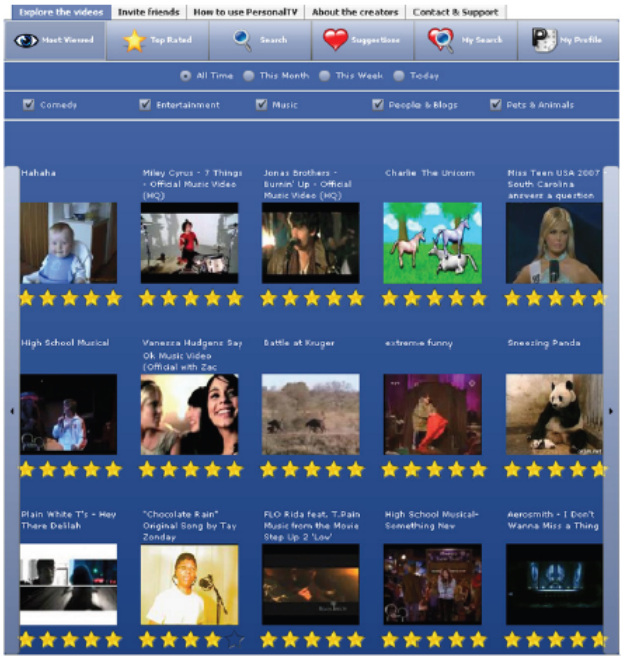

Figura 2.18: Aplicação PersonalTV inserido no Facebook.

Em [Pessemier et al., 2009] é apresentado, como caso de estudo, o PersonalTV, um serviço capaz de recomendar vídeos tendo em conta as classificações dadas aos vistos anteriormente. Para tal, encontra-se integrado com o serviço de partilha de vídeos Youtube[38]. Além das classificações anteriormente atribuídas, o algoritmo subjacente a este serviço é também sensível a outros aspectos do contexto do utilizador, tais como a localização o horário a que um vídeo foi visto, e ainda em sugestões de outros utilizadores. O PersonalTV está também integrado na plataforma do Facebook, com o aspecto da Figura 2.18. Além de permitir a interacção com o serviço sem que o utilizador tenha de sair da rede social, permite também que os resultados obtidos do algoritmo de recomendações possam ser filtrados tendo em consideração as suas informações pessoais. Por fim, esta integração permite ainda o envio e sugestões a outros utilizadores da lista de contactos do utilizador.

[38] `http://www.youtube.com`

SocialFusion

O SocialFusion [Beach et al., 2010] consiste numa solução capaz de recolher informação contextual de três fontes distintas: dispositivos móveis, redes sociais e ainda de sensores próprios (por exemplo uma webcam no caso do vídeo). Após a sua captura, os dados são processados para que o sistema possa concluir qual a acção apropriada, considerando o contexto capturado. É portanto uma solução sensível ao contexto. Pode ser utilizada para um único utilizador, para calcular por exemplo o seu estilo musical preferido, ou para um grupo de utilizadores, calculando neste caso o nível de afinidade entre eles.

Segundo a referência bibliográfica, esta solução ainda não se encontra totalmente desenvolvida. No entanto, é apresentado como caso de estudo o SocialFlicks [Beach et al., 2010], desenvolvido com recurso ao SocialFusion. O seu objectivo passa pela recomendação de vídeos atendendo aos que foram visualizados no passado. Tal como o SocialFusion, as recomendações podem ser sensíveis a um único ou a um grupo de utilizadores. A utilidade deste serviço pode ser comprovada com a sua instalação numa loja de aluguer de vídeos, onde dá sugestões aos clientes sobre novos filmes que estejam de acordo com os seus interesses.

Brightkite

Já foi abordado o sistema Social Serendipity como sendo, por si só, uma rede social sensível ao contexto. Pelas descrições dos sistemas que têm sido expostos, podemos observar também que o factor localização é o mais utilizado, estando presente em todos. O sistema Brightkite[39], apresentado pela Figura 2.19, é considerada uma rede social sensível à localização e funciona sobre aplicações móveis. Permite aos seus utilizadores visualizarem a localização dos seus contactos em tempo real e também adicionar outros que frequentem os mesmos espaços. Este sistema tem a vantagem de estar disponíveis para várias plataformas: iPhone, Android, BlackBarry e Nokia. Um sistema análogo é o Google Latitude[40], que tem a desvantagem de não funcionar exactamente como uma rede social e utiliza os contactos do Gmail.

Yelp

A área da pesquisa de locais e serviços baseada na localização do utilizador já foi anteriormente abordada. No entanto é importante referir a existência

[39] `http://www.brightkite.com/`

[40] `http://www.google.com/latitude/`

Figura 2.19: Imagem de apresentação do serviço Brightkite.

de uma aplicação semelhante integrada com redes sociais. A aplicação tem como nome Yelp[41] e, além da componente de pesquisa, tem também uma componente de partilha de opiniões. Um utilizador pode, por exemplo, pesquisar sobre restaurantes na sua área e, no final da refeição, informar os seus contactos do Facebook acerca do local e da qualidade das refeições. Este sistema tem também um serviço de publicidade baseado na localização do utilizador. Encontra-se disponível para várias plataformas móveis, entre as quais para o iPhone (Figura 2.20).

Lokast

A aplicação Lokast[42], disponível para as plataformas iPhone (Figura 2.21) e Android, permite a troca de informações entre utilizadores que estejam próximos. Este serviço vasculha o telemóvel onde estão instaladas em busca dos seus contactos, fotos e outras informações de contexto. Estas informações são depois apresentadas aos utilizadores para que estes decidam quais podem ser reunidas num perfil público, que fica visível para todos os outros utilizado-

[41] `http://www.yelp.com/`
[42] `http://www.nearverse.com/lokast`

Figura 2.20: Imagem da versão iPhone do serviço Yelp.

res geograficamente muito próximos (distância inferior a 100 metros). Além de partilharem os perfis pessoais, os utilizadores podem partilhar conteúdos multimédia entre si. A sua utilização tem sido testada em concertos, possibilitando a partilha de fotografias e vídeos entre os assistentes.

2.3.1 Contexto no Twitter, Facebook e MySpace

Os sistemas de publicidade baseados em localização não são uma novidade, isto é, estamos desde sempre habituados a ter cartazes ou placas de sinalização a indicar um estabelecimento ou serviço próximo. No entanto, com a capacidade de os dispositivos móveis captarem a localização do utilizador, surge a oportunidade de os integrar com este conceito de publicidade, criando o conceito de anúncios móveis e sensíveis à localização. Este conceito é descrito em [Bruner and Kumar, 2007] e permite que o utilizador possa pesquisar ou receber notificações publicitárias acerca de serviços próximos. Este pode ser um meio das empresas de serviços focarem a sua publicidade em áreas mais reduzidas e em públicos realmente interessados.

Várias empresas têm visto este tipo de publicidade como uma boa oportunidade de negócio e entre elas encontram-se algumas redes sociais online. Um dos casos mais evidentes é o do Facebook, que através do serviço Facebook Places permite essa possibilidade. Este serviço será descrito de forma

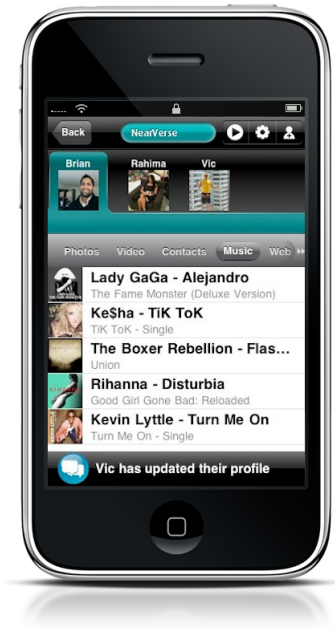

Figura 2.21: Imagem da versão iPhone do serviço Lokast.

mais profunda na secção 2.3.1.

A rede social Twitter foi a primeira a integrar o factor localização. Os seus esforços começaram por se concretizar na possibilidade de marcar os tweets dos utilizadores, vindos de clientes móveis pela sua API remota, com a sua posição geográfica. Posteriormente, esta API foi reforçada com métodos específicos para interacções e pesquisas com base na localização. Estas funcionalidades são já implementadas por uma série de aplicações móveis. É o caso do Birdfeed[43], Seesmic Web[44], Foursquare[45], Gowalla[46], Twidroid[47] entre outros. Tal como foi referido, os esforços do Twitter começaram por ser inicialmente visíveis na sua API remota, no entanto estes esforços já chegaram à sua interface Web. Actualmente, esta encontra-se integrada com o serviço Google Maps[48] e permite aos utilizadores ver um pequeno mapa com a localização geográfica de onde o tweet foi publicado. Para o futuro, estão assim criadas condições para a criação de uma plataforma de publicidade

[43] `http://birdfeedapp.com/`
[44] `http://seesmic.com/app/`
[45] `http://foursquare.com/`
[46] `http://gowalla.com/`
[47] `http://twidroid.com/`
[48] `http://maps.google.pt/`

e notícias locais, permitindo aos seus utilizadores pesquisarem por serviços próximos e serem alertados para novidades da sua região, por exemplo.

Figura 2.22: Imagem da versão Android da aplicação Foursquare.

Por serem algo mais do que clientes móveis do Twitter, as aplicações móveis Foursquare (Figura 2.22) e Gowalla merecem uma atenção especial. O Foursquare, disponível para várias plataformas, é também uma rede social sensível à localização e permite a partilha de conteúdos multimédia entre utilizadores. Esta aplicação tem também um jogo embebido, baseado na localização. Cada vez que o utilizador entra no sistema e partilha a sua localização ganha pontos e, em determinadas circunstâncias, pode mesmo coleccionar selos e desbloquear novas funcionalidades. Um princípio bastante análogo é adoptado pelo Gowalla.

No que diz respeito à rede social MySpace, a integração com o contexto dos utilizadores não é tão marcante como nos casos das outras redes sociais. Já foi referido que este serviço oferecia aos seus utilizadores uma funcionalidade de calendário. Este calendário, aliado ao facto de esta ser uma rede social mais vocacionada para o mundo artístico, serve essencialmente para que os artistas exponham as datas e outras informações dos seus eventos. A intenção do MySpace é marcar geograficamente estes conteúdos, permitindo desta forma aos utilizadores efectuarem pesquisas com base na sua localização, ou então ser automaticamente alertado de futuros eventos.

Facebook Places

Muito recentemente, a rede social Facebook lançou uma nova funcionalidade, o Facebook Places[49]. Inicialmente disponível apenas em território dos EUA, está aos poucos a ser estendido para os países da Europa e Ásia.

Este serviço é análogo aos já descritos Foursquare e Gowalla, onde os utilizadores são encorajados a partilhar a sua localização geográfica. Com esta informação permite que os utilizadores tenham conhecimento sobre quais dos seus amigos estão a usar a rede social perto de si. Por defeito, além da localização geográfica é ainda possível partilhar comentários com esses amigos.

É ainda possível editar as preferências relativas à privacidade do serviço e utilizar o modo "People here now". Neste modo, os utilizadores partilham a sua localização não apenas com os seus amigos mas com todos os utilizadores. É, portanto, uma funcionalidade aconselhada para os casos em que se pretende conhecer novas pessoas e não apenas interagir com os amigos já existentes[50].

O modo "People here now" é também útil para promover instituições e serviços. Os seus representantes podem adicionar uma localização às páginas do Facebook que utilizam para promover o seu negócio. Desta forma, as páginas irão aparecer nos mapas dos utilizadores que estão próximos, constituindo uma forma de publicidade local[51]. No entanto, esta pode também ser uma oportunidade para falsa publicidade, onde os serviços não existem ou não estão onde dizem estar. Para combater tal possibilidade, cada página é sujeita a um processo de validação efectuado pela própria comunidade.

Além das funcionalidades fornecidas, um dos pontos a favor do Facebook Places é a integração directa com serviços análogos populares, como o FourSquare, Yelp e Gowalla[52]. Desta forma, evita-se que os utilizadores tenham a necessidade de usar mais do que uma aplicação e, à partida, é aumentado o número de utilizadores encontrados na proximidade.

O serviço está disponível através da versão 3.2 da aplicação móvel oficial da rede Facebook, mas apenas para a plataforma iPhone. Neste momento, as restantes plataformas (Android, BlackBerry, etc) podem igualmente usufruir do serviço mas apenas através de uma versão em HTML5 alojada em *touch.facebook.com*.

---

[49] `http://www.facebook.com/places/`

[50] `http://www.engadget.com/2010/08/19/facebook-3-2-\`
`\for-iphone-adds-places-location-check-in-with-fours/`

[51] `http://mashable.com/2010/08/19/facebook-places-api/`

[52] `http://socialightmedia.com/social-networks/facebook-places\`
`\-future-locationbased-services/`

É ainda possível integrar aplicações de terceiros com este serviço através da *Graph API*. No entanto, estas aplicações estão limitadas aos métodos de leitura, estando os restantes (escrita e procura) apenas disponíveis para os serviços já mencionados como estando directamente integrados com o Facebook Places.

Além de o serviço ainda não ter chegado a todos os países, um ponto que é criticado pelos utilizadores é a não existência da possibilidade de referir a saída de um local[53]. A partir do momento em que um utilizador partilha a sua localização não tem a possibilidade de a esconder sem sair do serviço ou voltar a entrar noutro local.

Não existe também informação sobre se o algoritmo de pesquisa de utilizadores é baseado apenas na localização, ou se é dada relevância a outras variáveis do contexto do utilizador. A título de exemplo, poderiam ser utilizados os dados do perfil do utilizador para pesquisas mais personalizadas, filtrando os resultados por sexo, idade, preferências culturais, etc.

face2face

O projecto face2face[54] foi recentemente apresentado ao público e encontra-se integrado com as redes sociais mais populares (Facebook, Twitter, MySpace e LinkedIn). O seu funcionamento passa por aplicar o conceito de proximidade entre os contactos estabelecidos nas redes sociais. Desta forma, um utilizador é notificado sempre que os seus contactos nas redes sociais, que utilizem também este sistema, estejam geograficamente próximos. Além destes alertas, a aplicação face2face promove também formas de contacto entre os utilizadores, tais como o envio de mensagens escritas e conversações em *instant messaging*. Este sistema permite ainda a interacção com a lista de contactos dos contactos do utilizador, possibilitando a descoberta de novas pessoas, sempre tendo em consideração a localização geográfica. Duas das mais-valias deste sistema são a sua independência de plataforma (existem versões para Android, iPhone, BlackBerry (Figura 2.23) e está ainda prevista em J2ME) e as preocupações com a privacidade, visto que a localização exacta dos contactos nunca é mostrada. Infelizmente, o face2face não possui integração com os sistemas Foursquare e Gowalla, que aumentaria o número de contactos encontrados na proximidade do utilizador. Desta forma, a pesquisa está apenas limitada aos utilizadores desta aplicação.

[53] `http://forum.developers.facebook.net/viewtopic.php?id=74043`
[54] `http://face2face.ws/`

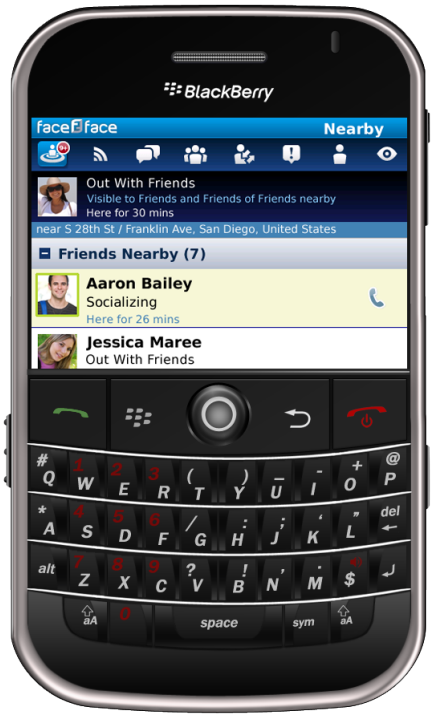

Figura 2.23: Ecrã da aplicação face2face para a plataforma BlackBarry.

2.3.2 Privacidade dos utilizadores

A partilha de informações relativas ao contexto do utilizador, em particular a sua localização geográfica, tem levantado uma série de questões relacionadas com a privacidade. As informações de contexto partilhadas podem servir de base para estudos acerca do quotidiano dos utilizadores, encontrando vulnerabilidades. A título de exemplo, quando um utilizador publica uma informação no Twitter marcada geograficamente como estando num centro comercial, uma pessoa mal intencionada pode imediatamente concluir que o utilizador não está em casa e aproveitar o momento para um assalto. Este mesmo exemplo é o tema de um projecto chamado Please Rob Me[55] que analisa as últimas publicações no Twitter marcadas geograficamente e dá relevância àqueles que mostram que determinada pessoa saiu de casa. Ironicamente calcula inclusive o número de oportunidades de assalto.

Os serviços sensíveis ao contexto têm contornado estas questões através dos seus termos de utilização, incluindo cláusulas de não responsabilização pelo conteúdo da informação publicada pelos próprios utilizadores. Em

[55] `http://pleaserobme.com/`

relação à localização geográfica algumas chegam mesmo a implementar medidas concretas. A mais utilizada é a inclusão de níveis de precisão, isto é, é dada a opção ao utilizador de publicar as coordenadas concretas da sua posição geográfica, ou então publicar apenas o nome da rua, da cidade, o código postal, etc. Desta forma, os utilizadores podem usufruir dos serviços sem a necessidade de dar uma informação muito específica. No entanto, a qualidade do serviço pode também com isso sofrer algum decréscimo.

2.4 Resumo

Neste capítulo foram apresentados os dois conceitos que fundamentam todo o trabalho desta dissertação. Foram apresentados as redes sociais mais populares e o tipo de evolução que pretendem seguir relativamente à criação de funcionalidades sensíveis ao contexto.

A rede social Facebook, através do lançamento da sua funcionalidade Places, mostra uma forte aposta na integração dos mundos das redes sociais e da sensibilidade ao contexto. Tal como foi possível constar, este tipo de serviços, mais do que um novo tipo de interacção entre os utilizadores e as redes sociais, potenciam também novos modelos de negócio para as empresas. Em concreto, foi dado o exemplo dos serviços de publicidade móvel sensível à localização.

Capítulo 3

Erlang

Desde a sua génese, a linguagem de programação Erlang tem como objectivo o desenvolvimento de aplicações com várias actividades concorrentes e tolerantes a falhas. Estes são requisitos normais também das grandes aplicações que lidam com um número elevado de utilizadores activos em cada momento. O desempenho, a arquitectura e funcionalidades tornam Erlang numa tecnologia adequada para o desenvolvimento de aplicações distribuídas e escaláveis que solucionem os problemas actuais.

O caso de estudo proposto nesta dissertação enquadra-se nesta situação. Existe a possibilidade de muitos utilizadores estarem activos ao mesmo tempo e a aceder aos mesmos dados. Como consequência, o elevado número de processos em execução será claramente um problema grave, pelo que uma tecnologia de alto desempenho como o Erlang e o seu suporte a actividades concorrentes será de extrema utilidade.

O Erlang não possui a popularidade de linguagens como Java, C# ou PHP. Todavia, dada a sua propensão para o desenvolvimento de aplicações escaláveis e com requisitos de controlo de concorrência, é natural encontrar-se sistemas completos, ou componentes, desenvolvidos nesta linguagem. O elevado número de utilizadores activos em cada instante nas redes sociais faz com que também estes serviços tenham recorrido ao Erlang para a implementação de algumas funcionalidades, como é o caso do serviço de conversação do Facebook. Este serviço será abordado posteriormente com mais detalhe.

Neste capítulo irá ser apresentada uma versão sucinta da história da linguagem Erlang. Serão também abordadas as suas funcionalidades mais relevantes, bem como alguns dos casos práticos onde esta tecnologia é aplicada.

A soma de todos estes dados justificará a escolha desta linguagem para o desenvolvimento do componente mais crítico do sistema LocalChat, proposto nesta dissertação.

3.1 História da linguagem

Em 1985, os engenheiros do laboratório de ciências computacionais da Ericsson procuravam novas e mais eficientes de desenvolver aplicações robustas na área telefónica. Inicialmente, tentaram várias linguagens e não tinham qualquer intenção de criarem uma nova. Os requisitos prendiam-se essencialmente com o suporte a actividades concorrentes e tolerância a faltas, ao nível de software e hardware. As aplicações pretendiam-se de elevada escalabilidade, distribuídas e altíssima disponibilidade, isto é, que fossem capazes de estar vários anos a executar sem interrupções. Para tal era também necessário garantir a possibilidade de a manutenção destes sistemas ter de ser feita sem os desligar. Visto não terem encontrado nenhuma linguagem de programação que solucionasse todos os seus problemas, Joe Armstrong, Robert Virding, Claes Wikstrom e Mike Williams acabaram por decidir criar a sua. Foi então desenvolvida a linguagem Erlang.

Na sua primeira versão, esta era uma linguagem declarativa com mecanismos de controlo de concorrência. Em 1995, a linguagem Erlang era já funcional e, dez anos mais tarde, era ela própria concorrente, através que componentes que comunicam entre si. Actualmente, a linguagem é multiplataforma e continua funcional e concorrente, sendo esta uma das características mais apreciadas [Armstrong, 2007].

3.2 Principais características

A concorrência do Erlang assenta em processos bastante leves, mais leves do que as próprias threads. Estes processos são comummente denominados de microprocessos, pertencem à própria linguagem (não ao sistema operativo) e podem estar em qualquer máquina (transparência da localização). Não possuem qualquer memória partilhada entre si e a comunicação entre processos é feita através de um sistema de entrega de mensagens assíncrono.

Cada um destes microprocessos é monitorizado por um outro, que é alertado sempre que este lance uma excepção. Ao receber esta excepção, o processo monitor tem a possibilidade de tratar a excepção e recuperar o processo. Este é um modelo que faz uma separação entre o processo que executa uma acção e o seu supervisor. Desta forma, os programadores são encorajados a adoptar um processo de desenvolvimento orientado à concorrência e não defensivo, aceitando o facto que os processos podem falhar e pensando apenas na melhor forma de recuperar dessas falhas.

O núcleo da linguagem Erlang é tipado, simples e dinâmico. Permite a troca de código em tempo de execução, extensões que permitem a persistência

dos dados e um grande número de bibliotecas, chamado de Open Telecom Plataform [Torstendahl, 1997] (OTP). A relação entre a OTP e o Erlang pode ser comparada à relação entre a linguagem de programação C e a plataforma Unix [Armstrong, 2007]. As funcionalidades do OTP serão abordadas com mais detalhe no ponto 3.4.

3.3 Desenvolvimento em Erlang

A linguagem de programação Erlang conta também com um conjunto de frameworks para desenvolvimento Web [Bryson and Vinoski, 2009]. Entre elas podemos destacar as seguintes: Erlyweb[1], Erlang Web[2], Webmachine[3], Nitrogen[4] and BeepBeep[5]. Possui também servidores Web, entre eles destacam-se o MochiWeb[6] e o Yaws[7]. Este último demonstrou um excelente desempenho em comparação com o servidor Apache[8]. Possui ainda uma base de dados distribuída chamada Mnesia [Mattsson et al., 1999] extremamente rápida e capaz de persistir qualquer estrutura definida em Erlang.

Esta linguagem permite a interacção com programas externos e escritos em linguagens diferentes, por exemplo C. Esta interacção é feita através de comunicação por portas (*sockets*) ou através de *drivers* embebidos. Este último método é o mais eficiente, no entanto é também o mais perigoso, visto que cada erro ocorrido no *driver* provoca um erro no sistema Erlang. É ainda possível executar aplicações desenvolvidas em Erlang na Java Virtual Machine (JVM) utilizando o projecto Erjang[9]. Este projecto carrega ficheiros binários Erlang (*.beam*) e converte-os em ficheiros binários Java (*.class*).

Esta linguagem possui também uma framework de testes unitários denominada de EUnit [Carlsson and Rémond, 2006], bem como uma série de outras ferramentas úteis [Nagy and Nagyné Víg, 2008].

[1] `http://erlyweb.org`
[2] `www.erlang-web.org`
[3] `http://bitbucket.org/justin/webmachine`
[4] `http://nitrogenproject.com/`
[5] `http://github.com/davebryson/beepbeep/tree/master`
[6] `http://code.google.com/p/mochiweb/`
[7] `http://yaws.hyber.org/`
[8] `http://www.sics.se/~joe/apachevsyaws.html`
[9] `http://wiki.github.com/krestenkrab/erjang`

3.4 OTP - Open Telecom Plataform

Devido ao seu desempenho e funcionalidades, a linguagem de programação Erlang é tida em conta como apropriada para o desenvolvimento de aplicações distribuídas, altamente escaláveis e tolerantes a falhas. No entanto, o processo de desenvolvimento deste tipo de aplicações pode ser substancialmente agilizado com a utilização da plataforma OTP.

Tal como a linguagem Erlang, plataforma OTP foi criada nos laboratórios da Erickson para facilitar o desenvolvimento de soluções para a área das telecomunicações. No entanto, as suas características mostraram-se igualmente úteis no desenvolvimento de qualquer tipo de aplicações com os mesmos requisitos [Armstrong, 2003]. É composta por, entre outras coisas, um servidor aplicacional, a base de dados Mnesia e por um grande número de bibliotecas. As grandes vantagens apontadas à sua utilização passam por [LOGAN et al., 2010]:

- **Produtividade:** utilizando as funcionalidades disponibilizadas pela plataforma, é possível desenvolver sistemas de forma rápida;
- **Estabilidade:** existem funcionalidades genéricas que podem e devem ser aplicadas a todas aplicações. A plataforma OTP permite ao programador utilizar essas funcionalidades, o que leva a que possa dedicar mais atenção à lógica da própria aplicação;
- **Supervisão:** a estrutura das aplicações potenciada pela plataforma OTP facilita a supervisão dos sistemas de uma forma automática ou através de interfaces gráficas;
- **Actualização:** a plataforma oferece funcionalidades que permitem a actualização dos sistemas sem que haja necessidade de os desligar;
- **Confiança:** o código que implementa as funcionalidades da plataforma OTP foi intensivamente testado.

Um programador, ao utilizar a plataforma OTP, ganha a capacidade de aplicar determinados comportamentos ao seu código. A aplicação destes comportamentos obriga algumas vezes à definição de algumas funções extra, no entanto são eles que conferem a facilidade em atingir os níveis de escalabilidade e tolerância a faltas desejados. Existem vários tipos de comportamentos e a criação de um sistema ideal passa pela combinação de vários [AB, 2010].

- ***Server:*** comportamento para implementação de um servidor para sistemas que funcionem segundo a lógica cliente-servidor. Um processo

que implemente este comportamento disponibiliza uma interface com as funções que podem ser invocadas no servidor e funcionalidades de detecção e tratamento de erros, tal como os comportamentos Finit State Machine e Event apresentados de seguida;

- ***Finite State Machine (FSM):*** comportamento utilizado em processos que implementem máquinas de estado finitas;
- ***Event:*** este comportamento é importante quando se pretende implementar uma funcionalidade capaz de processar eventos. Consiste num gestor de eventos genérico capaz de criar e eliminar dinamicamente processos que processes esses eventos;
- ***Supervisor:*** um processo que implemente este comportamento tem a capacidade de iniciar, parar e monitorizar todos os seus processos filho. A principal função de um processo supervisor é manter os seus processos filho activos enquanto são necessários e reinicia-los sempre que ocorram erros inesperados.

O comportamento *Supervisor* é de extrema importância na criação de aplicações tolerantes a faltas e com capacidade de recuperar de erros ocorridos. Utilizando este comportamento, podemos associar os processos trabalhadores a um processo supervisor e sempre que ocorram erros inesperados, os processos trabalhadores são reiniciados. Nestes casos, existem duas políticas distintas que podem ser adoptadas pelo processo supervisor em relação aos trabalhadores:

- ***one-for-one*:** apenas o processo que falhou é reiniciado, para os casos em que não existe dependência entre este processo e os restantes.
- ***one-for-all*:** em caso de falha de um dos processos, todos são reiniciados. Política adequada para quando existe dependencia entre os processos trabalhadores.

A designação de processo trabalhador é dada àqueles que têm como função processar os pedidos que lhes chegam, em oposição aos processos supervisores que apenas monitorizam os processos trabalhadores. Para aumentar a tolerância a faltas destes sistemas, é possível interligar vários processos supervisores de forma a criar uma árvore de supervisão. Desta forma é feita uma protecção não só aos processos trabalhadores como também aos processos supervisores [AB, 2010]. Pode ser observado na Figura 3.1 um exemplo de uma árvore de supervisão, em que os processos supervisores aparecem sob a forma de quadrados e os trabalhadores sob a forma circular. Neste

exemplo em concreto é possível observar que todos os processos trabalhadores estão a ser monitorizados por um processo supervisor, que por sua vez é monitorizado por um supervisor geral.

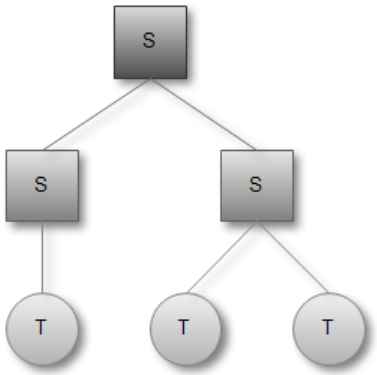

Figura 3.1: Árvore de supervisão. Processos supervisores com forma quadrada e trabalhadores com forma circular.

Normalmente, um sistema deve funcionar como uma unidade, isto é, deve ser possível iniciar e terminar todos os seus componentes em simultâneo. Utilizando Erlang e as funcionalidades da plataforma OTP, este procedimento é definido através de uma aplicação. Nas propriedades da aplicação são definidos, entre outros, os módulos que compõem o sistema e o nome do módulo que processa os pedidos.

```
{application, teste, [
 {description, "Aplicação de teste"},
 {vsn, "0.1"},
 {modules, [teste, teste_supervisor, teste_trabalhador]},
 {registered, [teste]},
 {included_applications, [outro_teste]},
 {applications, [kernel, stdlib, sasl]},
 {mod, {teste, []}},
 {start_phases, []}
]}.
```

No exemplo acima mostrado, é possível observar o nome da aplicação ("teste") na primeira linha. Nas linhas seguintes e por esta ordem, encontra-se uma descrição da aplicação, a versão actual, uma lista com os módulos que a compõem, o nome com que a aplicação é registada na plataforma Erlang, as

aplicações incluidas. Por último, o nome do ficheiro onde o comportamento da aplicação é definido e alguns parâmetros relativos à forma como esta deve ser iniciada.

Uma propriedade interessante das aplicações é que podem incluir e ser incluídas por outras. Desta forma é possível reutilizar funcionalidades sob a forma de aplicações, podendo testar cada uma de forma individual. Uma aplicação que não seja incluída por nenhuma outra, é designada de aplicação primária. A Figura 3.2 ilustra a dependência estabelecida através da inclusão de aplicações.

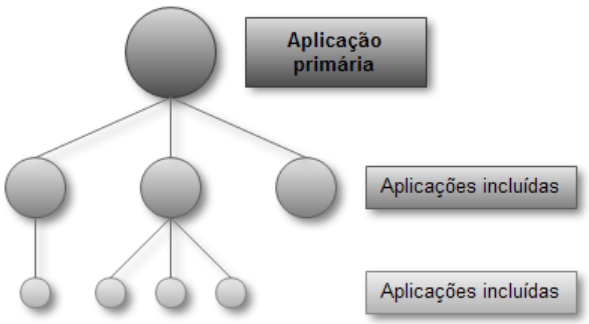

Figura 3.2: Cadeia de inclusão de aplicações.

Uma aplicação, quando é iniciada, carrega todos os controladores das aplicações por si incluídas. No entanto, estas só entram em execução quando o supervisor da aplicação primária as ordenar, funcionando assim a árvore de supervisão para tolerância a faltas.

Convém ainda referir que as aplicações incluídas directamente por uma aplicação primária e, as aplicações que, por sua vez, estas incluem são todas consideradas como pertencentes à aplicação primária [AB, 2010].

Um dos pontos mais aclamados da linguagem Erlang é a facilidade com que é possível desenvolver aplicações distribuídas. Quando se pretende uma aplicação deste género, convém também que a monitorização seja feita de forma distribuída. Em termos práticos temos por exemplo que, quando ocorre um erro numa das máquinas afectando serviços, a árvore de supervisão deve reiniciar os mesmos serviços numa máquina activa. Para facilitar este processo de monitorização, a plataforma OTP permite a definição de aplicações distribuídas e a personalização de que máquinas fazem parte do sistema e o tipo de controlo que deve ser aplicado a cada uma [AB, 2010].

3.5 Erlang em aplicações reais

A evolução da linguagem Erlang tornou-a uma linguagem interessante não só para desenvolver aplicações da área das telecomunicações, mas para uma vasta lista de áreas industriais com requisitos de resposta em tempo real [Armstrong et al., 1997].

3.5.1 Telecomunicações

Na área telefónica, o Erlang continua a ser utilizado na Erickson no seu sistema de GPRS[10] e nas suas redes 3G por todo o mundo. A empresa Nortel também utiliza Erlang em alguns dos seus produtos. Por sua vez, a T-Mobile utiliza esta linguagem nos seus sistemas de SMS[11] e autenticação. Por último, importa ainda referir a presença desta linguagem também na Motorola, mais precisamente em componentes de mobilidade de dados e processamento de chamadas.

3.5.2 Amazon Elastic Compute Cloud

Fora da área telefónica, a linguagem Erlang está presente em alguns dos mais importantes serviços de grandes empresas mundiais. É o caso da Amazon e do SimpleDB[12], o serviço de base de dados incorporado no seu sistema Amazon Elastic Compute Cloud (EC2). Este é um sistema de base de dados não relacional caracterizado como sendo rápido, altamente disponível e capaz de armazenar grandes conjuntos de dados. Estando integrado no serviço EC2, este sistema tem também a vantagem de poder escalar tendo em conta as suas necessidades reais[13].

3.5.3 Delicious

A Yahoo! também utiliza Erlang no seu sistema Delicious, que é provavelmente o sistema mais utilizado para sincronismo e partilha de favoritos (*bookmarks*) entre vários computadores e utilizadores. Esta linguagem foi importante no desenvolvimento da segunda versão do serviço, em que este foi totalmente reescrito. Componentes com tarefas bastante morosas e com requisitos de controlo de concorrência, que antes estavam desenvolvidos com scripts em Perl, foram codificados em Erlang obtendo ganhos

[10] General Packet Radio Service
[11] Short Message Service
[12] `http://aws.amazon.com/simpledb/`
[13] `http://www.satine.org/archives/2007/12/13/amazon-simpledb/`

no seu desempenho. Foram os casos dos componentes relacionados com migração de dados, tratamento de spam e pesquisas. Em alguns casos, a Yahoo! recorreu também à base de dados Mnesia e às interfaces disponibilizadas pela plataforma OTP para comunicação com outras linguagens [Nick Gerakines and Goffinet, 2008].

3.5.4 CouchDB

A linguagem Erlang é também utilizada em projectos não comerciais, como é exemplo o sistema de base de dados CouchDB[14]. Este é um sistema orientado a documentos e fomenta a escalabilidade entre clusters com vários núcleos e servidores. Os documentos são guardados com formato JSON e o sistema é acessível através de uma interface REST. Este sistema foi inicialmente desenvolvido em C++ mas, à medida que os problemas de concorrência surgiram, o programador Damien Katz decidiu começar a desenvolver, primeiro apenas alguns dos componentes e posteriormente todo o sistema em Erlang. Os ganhos no desempenho foram a principal razão para esta mudança de tecnologia [Cesarini and Thompson, 2009].

3.5.5 ejabberd

O protocolo XMPP[15] [Saint-Andre, 2005b] é uma alternativa de código aberto para sistemas de mensagens instantâneas e possui, entre outras, uma implementação em Erlang. Esta implementação tem o nome de ejabberd[16] e consiste numa plataforma distribuida, tolerante a falhas e capaz de processar elevados números de utilizadores activos em simultâneo em cada nó. A tecnologia ejabberd faz parte de várias aplicações de grande escala como é o caso do Facebook, cujo applicação específica pode ser observada no ponto 3.5.7.

O protocolo Jabber/XMPP irá abordado na secção 4.3.

3.5.6 Tsung

Os testes de carga e performance são também áreas que exigem que as suas ferramentas tenham grande desempenho, com o objectivo de levar as aplicações testadas ao seu limite. A linguagem Erlang marca também presença neste âmbito através do Tsung[17]. Este projecto, de código aberto, pode

[14] `http://couchdb.apache.org/`
[15] Extensible Messaging and Presence Protocol
[16] `http://www.ejabberd.im/`
[17] `http://tsung.erlang-projects.org/`

ser utilizado para testes de carga de serviços HTTP, SOAP[18], PostgreSQL, MySQL, LDAP[19], XMPP e WebDAV [Whitehead and Wiggins, 1998]. O sistema Tsung pode ser distribuído por várias máquinas e simula a interacção de vários utilizadores com as aplicações do tipo cliente-servidor acessíveis via IP.

3.5.7 Facebook

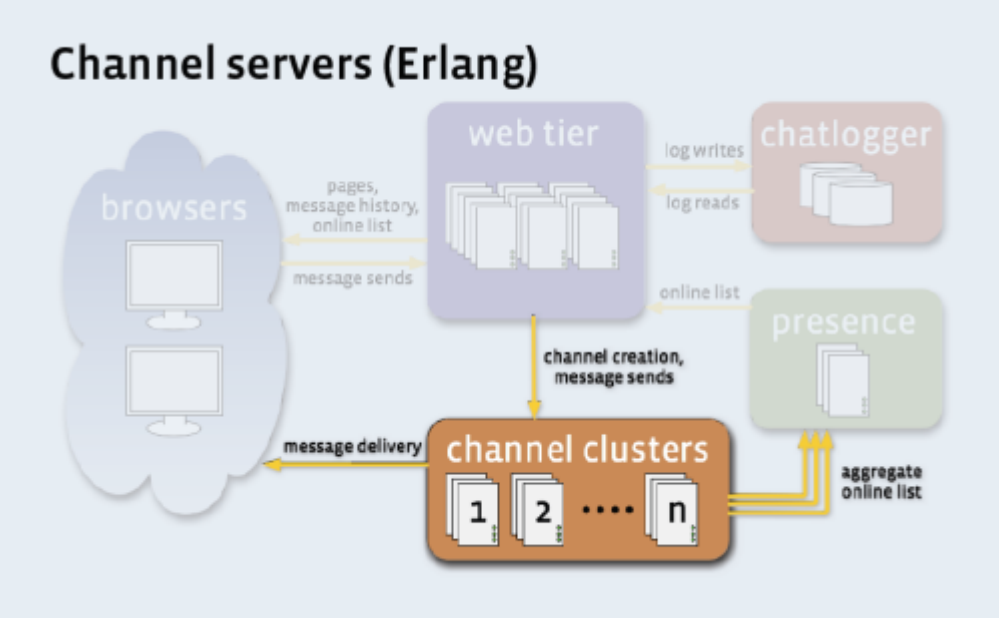

Figura 3.3: Erlang no Facebook [Letuchy, 2009].

Na Figura 3.3 está descrita a arquitectura do serviço de conversação em tempo real do Facebook, tendo particular destaque o componente desenvolvido em Erlang, *channel clusters*. A sua função passa por criar um canal dedicado a cada conversação e gerir o trajecto das mensagens até ao seu destino. Sempre que uma nova mensagem é enviada, este componente tem também a função de actualizar o estado do utilizador que a enviou, isto é, confirma que o utilizador se encontra activo.

As mensagens são entregues no seu destino como resposta a pedidos HTTP[20].

[18]Simple Object Access Protocol
[19]Lightweight Directory Access Protocol
[20]Hypertext Transfer Protocol

Tendo em conta os índices de utilização deste serviço[21], pode-se concluir que o processo de entrega das mensagens é crítico, exigente ao nível da escalabilidade, performance e concorrência. No entanto, até ao momento, não são conhecidas falhas graves na sua actividade.

3.5.8 Outros

O Rab-bitMQ[22] é um sistema empresarial baseado no protocolo *Advanced Message Queuing Protocol* (AMQP) e também utiliza Erlang. Tal como a aplicação Wings3d, usada para modelar redes de polígonos e muito utilizada na área da aeronáutica.

3.6 Resumo

Ao longo deste capítulo foi abordada a tecnologia Erlang, começando com uma breve exposição da sua origem, evolução e principais características.

Desde a sua génese, o Erlang foi projectado para resolver os problemas de escalabilidade e disponibilidade de aplicações que se queriam robustas. Está presente em sistemas críticos com estes requisitos nas mais variadas áreas, desde motores de bases de dados a serviços de redes sociais.

As características desta tecnologia, em conjunto com as funcionalidades oferecidas pela plataforma OTP e os casos reais em que é aplicada com sucesso, justificam a sua inclusão nesta dissertação. Em especial, os padrões oferecidos pela plataforma OTP, principalmente os de supervisor e supervisor, e a possibilidade de actualização de código sem desligar o sistema, serão de extrema importância para garantir a disponibilidade de um serviço como o LocalChat. Os microprocessos e o sistema de comunicação assíncrono com passagem de mensagens eliminam os problemas relacionados com o desempenho e concorrência no acesso aos dados associados a sistemas com elevados números de utilizadores, como é o caso do sistema proposto nesta dissertação.

O Erlang permitirá desta forma que os requisitos não funcionais associados à arquitectura conceptual, apresentada nesta dissertação, possam mais facilmente ser atingidos e com um nível de desempenho mais elevado.

[21] `http://www.insidefacebook.com/2009/02/23/facebook-chat\`
`\-latest-stats-whats-coming-next/`

[22] `http://www.rabbitmq.com/`

Capítulo 4

Tecnologias

O capítulo anterior apresentou a tecnologia Erlang e exemplos reais onde é aplicada, justificando assim a sua inclusão nesta dissertação. Ao longo deste, serão abordadas outras tecnologias utilizadas no decorrer desta dissertação. Primeiramente será apresentado o projecto erlFBGraph, uma biblioteca desenvolvida em Erlang para interacção com a nova versão da plataforma do Facebook, a *Graph API*. A plataforma Android é utilizada no desenvolvimento do cliente LC para potenciar uma interacção mais rica entre o utilizador e o serviço. Nesta secção são abordadas essencialmente a plataforma de desenvolvimento e questões que é importante ter em atenção no processo de desenvolvimento. Por último é feita uma descrição superficial sobre o protocolo XMPP, utilizado no serviço de conversação embutido no sistema LC, evidenciando servidores, clientes e bibliotecas.

4.1 Biblioteca Erlang para interacção com Facebook

Tal como foi descrito na secção 2.1.6, até ao momento não foi possível encontrar nenhuma aplicação cliente, desenvolvida em Erlang, que seguisse as especificações da *Graph API* e que fosse suficientemente completa a ponto de agilizar o trabalho a desenvolver. Visto isto, optou-se pela criação de um projecto capaz de responder às necessidades desta dissertação e que fosse suficientemente genérico a fim de ser utilizado como elo de ligação com o Facebook noutros projectos. Para tal é essencial a capacidade de interrogação da plataforma do Facebook sobre qualquer objecto ou relação do seu grafo social. Desta forma nasceu o projecto erlFBGraph[1].

[1] `http://github.com/ejbraga/erlFBGraph`

O erlFBGraph disponibiliza cinco métodos, sendo dois deles responsáveis por diferentes processo de autenticação, outros dois com a função de interrogar a API acerca de objectos e relações. Por último, disponibiliza também um método específico capaz de obter o endereço da imagem do perfil de um dado objecto.

A aplicação cliente erlFBGraph suporta os dois tipos de autenticação: autenticação como aplicação e como utilizador. O primeiro caso é processado pelo método *get_token_application/2*, recebendo como parâmetros a identificação e a chave da aplicação, geradas pela plataforma do Facebook.

```
(erlFBGraph@node)1>
  {ok, Token} =
       erlFBGraph:get_token_application(ID_App, Chave_App).
"128584520491486%7C2.ok4n7ERw_N0o2Y1vxa28ow__.3600.1275307200-
 100000373587590%7CnVe3bRTnvYs99z9_vApPVbX5ioc."
```

Por outro lado, a autenticação de utilizadores é processada de forma diferente e mais complexa. O método de autenticação, *get_token/3*, recebe como parâmetros o *username*, a *password* e o identificador da aplicação, devolvendo o respectivo *token* no caso do processo ser concluído com sucesso.

```
(erlFBGraph@node)2>
  {ok, Token} =
       erlFBGraph:get_token(Email, Password, ID_App).
"119486491427669|ER-SjTfA-aMTQC3CFkiMRI3f9jc"
```

O método *get_token* está implementado segundo as especificações anteriormente descritas para o caso da autenticação de utilizadores a partir de uma aplicação *desktop*. No entanto, foi possível eliminar a dependência do *browser*. Isto foi conseguindo simulando a interacção HTTP com a plataforma do Facebook, através de código e mascarando os pacotes com o acrescentar do *header* HTTP *User-Agent*. Neste caso em concreto foi-lhe atribuído o valor *Mozilla/5.0 (X11; U; Linux i686; pt-PT; rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3*.

Desta forma, o único pré-requisito é que o utilizador tenha dado autorização prévia à aplicação para que possa aceder aos seus dados. Embora a necessidade da integração de um *browser* seja eliminada, a abordagem seguida está muito dependente da interacção HTTP se manter inalterada. Este deve portanto ser um dos pontos a rever numa futura versão do projecto.

Interrogar a API sobre os dados de um determinado objecto é a função do método *get_object_data/1*. Este método recebe num único parâmetro três

dados: o tipo do objecto, o seu nome ou identificador e o *token*, previamente devolvido pelo método *get_token/3* ou *get_token_application/2*. Como resposta será retornado o objecto JSON recebido da plataforma do Facebook, transformado numa estrutura Erlang. Tomando como exemplo prático o utilizador *ejbraga*, ao interrogar a API obteríamos a seguinte estrutura:

```
(erlFBGraph@node)3>
  {ok, Resultado} =
    erlFBGraph:get_object_data({user, "ejbraga", Id_Sessao}).
{ok, {struct,[{<<"id">>,<<"100000963928649">>},
        {<<"name">>,<<"Emanuel Braga">>},
        {<<"first_name">>,<<"Emanuel">>},
        {<<"last_name">>,<<"Braga">>},
        {<<"link">>,
           <<"http://www.facebook.com:443/pg13362">>},
        {<<"birthday">>,<<"04/28/1987">>},
        {<<"hometown">>,
         {struct,[{<<"id">>,<<"109887585700524">>},
                  {<<"name">>,<<"Braga, Portugal">>}]}},
        {<<"quotes">>,
           <<"\"I hope it works\"-Emanuel Braga">>},
        {<<"gender">>,<<"masculino">>},
        {<<"timezone">>,1},
        {<<"locale">>,<<"en_US">>},
        {<<"verified">>,true},
        {<<"updated_time">>,<<"2010-09-13T21:32:57+0000">>}]}}
```

O método *get_connection_data/1* é análogo ao anterior, mas aplicado às relações existentes no grafo social. No entanto a sua aplicação é um pouco mais limitada, visto que apenas é possível aceder a dados das relações estabelecidas pelo utilizador que estiver autenticado. Seguindo o esquema do parâmetro recebido pelo *get_object_data*, este método recebe o tipo de relação a que se pretende aceder e o *token*, previamente obtido pelo método de autenticação apropriado. Os dados são também devolvidos numa estrutura Erlang.

```
(erlFBGraph@node)4>
  {ok, Resultado} =
    erlFBGraph:get_connection_data(
       {friends, "ejbraga", Id_Sessao}
       ).
```

```
{ok, {struct,[{<<"data">>,
       [{struct,[{<<"name">>,<<"Mary Watson">>},
                 {<<"id">>,<<"100001625494904">>}]}]}]}}
```

Por fim, o erlFBGraph disponibiliza um método que tem como única função devolver a localização da imagem do perfil de um dado objecto, *get_picture/1*. Como parâmetro recebe apenas o identificador, ou o nome, do objecto e é o único pode ser invocado sem qualquer autenticação prévia.

```
(erlFBGraph@node)5>
   {ok, Picture} = erlFBGraph:get_picture({page,"coca-cola"}).
{ok,"http://profile.ak.fbcdn.net/profile-ak-snc1/object3/1853/
   100/q40796308305_2334.jpg"}
```

O processo para obter esta localização é à partida bastante simples, bastando interrogar a API no endereço `http://graph.facebook.com/ID/picture`, sendo ID o identificador do objecto. No entanto a resposta que é obtida não é totalmente correcta, visto que apenas funciona como um apontador, através do *header* HTTP *location* para a verdadeira localização. O método implementado no erlFBGraph tem já em conta este pormenor e processa a resposta da plataforma Facebook a fim de devolver a verdadeira localização da imagem.

4.2 Android

Nos últimos anos, a Google tem canalizado parte das suas energias para mundo das tecnologias móveis. Estes esforços traduziram-se, até ao momento, no lançamento de um sistema operativo – Android[2] - e de um dispositivo móvel – Nexus One[3].

Ao longo deste subcapítulo será feita uma abordagem ao Android, descrevendo a sua origem e os seus objectivos, dando particular relevância à plataforma disponibilizada para desenvolvimento de aplicações.

4.2.1 Origem do Android

O sistema operativo Android foi desenvolvido inicialmente por uma empresa chamada Android Inc., empresa que foi posteriormente adquirida pela Google. Esta foi a primeira aposta da empresa no mundo dos *smartphones*,

[2] `http://www.android.com`
[3] `https://www.google.com/phone`

que surgiu como uma resposta face ao crescente sucesso da empresa Apple, através do iPhone e do seu sistema operativo. Actualmente, este sistema operativo está presente em diversos dispositivos móveis, perfazendo cerca de 200 mil dispositivos com Android vendidos por dia[4].

A arquitectura deste sistema operativo baseia-se no *kernel* Linux e em *software* de código aberto. Estas aplicações são desenvolvidas em Java por uma grande comunidade de utilizadores activos, com recurso a um SDK lançado em conjunto com o sistema operativo. A Google disponibiliza também uma loja de aplicações[5] onde os programadores podem publicar as suas aplicações, ficando assim acessíveis a todos os utilizadores que as pretendam instalar.

4.2.2 Plataforma de desenvolvimento

O SDK fornecido pela Google para facilitar o processo de desenvolvimento de aplicações para Android inclui uma basta lista de ferramentas. Entre elas é possível encontrar um *debugger*, um emulador, documentação, várias bibliotecas (por exemplo para produzir soluções que recorrem a mapas), diversos tutoriais e exemplos de código [Murphy, 2010].

O esforço necessário no desenvolvimento de uma aplicação Android é ainda minimizado com a integração do SDK em IDEs como Eclipse[6] ou Netbeans[7], através de *plugins*. O primeiro é aconselhado pela própria documentação oficial da plataforma Android[8]. O segundo projecto encontra-se também bastante funcional, com excepção feita aos casos em que é necessário incluir bibliotecas de terceiros no projecto. Neste caso, a geração do ficheiro de compilação *build.xml* é feita de forma deficiente, sendo necessário adicionar algum texto de forma manual.

No que diz respeito à plataforma, o SDK do Android é disponibilizado para vários sistemas, incluindo Windows, MacOS e ainda todos os sistemas Linux que corram sobre a arquitectura x-86. Os requisitos passam pela instalação prévia do JDK[9], Apache Ant[10] e Python.

[4] `http://techcrunch.com/2010/08/04/android-activations/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed\%3A+Techcrunch+\%28TechCrunch\%29`

[5] `http://www.android.com/market/`

[6] `http://www.eclipse.org/`

[7] `http://netbeans.org/`

[8] `http://developer.android.com/sdk/index.html`

[9] Java Development Kit

[10] `http://ant.apache.org/`

4.2.3 O que compõe uma aplicação Android

As aplicações Android são semelhante às aplicações *desktop*, isto é, têm a capacidade de gerir totalmente as janelas que são mostradas, podem aceder aos serviços fornecidos pelo sistema operativo e ignoram que outros serviços estão em execução no mesmo momento. Contudo, a forma como as aplicações são tratadas é feita de forma diferente com o intuito de que os *smartphones* sejam mais tolerantes a falhas.

Os componentes que os programadores têm à disposição no desenvolvimento de aplicações Android são [Murphy, 2010]:

- **Actividades:** este componente é o componente utilizado na construção das interfaces. Analogamente, a actividade pode ser vista como uma janela numa aplicação *desktop*, embora não tenham necessariamente de estar ligadas a uma interface gráfica. Neste caso, as actividades passarão a ser tratadas como fornecedores de conteúdo ou serviços, dois outros tipos de componentes.

- **Fornecedores de conteúdo:** os fornecedores de conteúdos funcionam como uma camada de abstracção na obtenção de dados de uma ou mais aplicações. O modelo defendido para o desenvolvimento de aplicações Android encoraja a que uma aplicação partilhe os dados obtidos com outras. Desta forma, com fornecedores de conteúdo é possível fazer essa partilha e controlar completamente quais desses dados são disponibilizados.

- **Serviços:** as actividades e os fornecedores de conteúdo não têm, normalmente, tempos de vida muito longos, podendo ser terminados a qualquer momento. Contrariamente, os serviços foram projectados para executarem durante longos períodos e, caso seja necessário, independentes de qualquer actividade. Este componente pode ser bastante útil em aplicações que verificam que dependam de alterações de dados, por exemplo de um serviço RSS[11].

- **Intents:** por último, os intents são mensagens do sistema. A sua utilidade passa pela notificação das aplicações sobre vários eventos, sobre alterações ocorridas ao nível do *hardware*, sobre a chegada de mensagens escritas (SMS), etc.

[11] Really Simple Syndication

4.2.4 Funcionalidades disponibilizadas pelo Android

O sistema operativo Android permite o acesso a vários tipos de funcionalidades do dispositivo móvel, tais como [Murphy, 2010]:

- **Armazenamento:** as aplicações Android permitem o anexo de ficheiros de dados para os casos em que estes não serão alterados, por exemplo ícones ou ficheiros de ajuda. É-lhes inclusive facultado acesso de leitura e escrita a dispositivos de armazenamento externos, como sendo os discos SD[12].

- **Conectividade:** à partida, todos os dispositivos Android estarão ligados à Internet, seja através de *Wi-Fi* ou outra tecnologia qualquer. Visto isto, os programadores podem tirar partido de qualquer tecnologia, desde *sockets* Java até à integração na aplicação de um *browser* baseado em WebKit. Neste ponto, apenas de salientar o facto de o emulador disponibilizado no SDK apenas permitir a conectividade por NAT [Srisuresh and Egevang, 2001] o que pode levantar alguns obstáculos no acesso a máquinas com IPs reservados.

- **Multimédia:** geralmente, a captura e reprodução de áudio e vídeo são funcionalidades asseguradas por todos os dispositivos Android. Estas funcionalidades são também disponibilizadas às aplicações.

- **Global Position System (GPS):** os *smartphones* lançados actualmente têm, na sua maioria, a capacidade de saber a sua posição geográfica, seja através de sensores GPS ou outra tecnologia. Obtendo estes dados, as aplicações podem mostra-los ao utilizador, ou então mostrar a sua posição num mapa.

- **Serviços telefónicos:** não fosse o Android um sistema operativo para *smartphones*, é permitido às aplicações utilizarem o serviço de chamadas, assim como todos os serviços que são normais encontrar num telefone.

4.2.5 Estrutura de um projecto

A estrutura de um projecto Android segue os princípios básicos de um projecto Java [Murphy, 2010]. Esta estrutura pode ser criada utilizando as ferramentas disponibilizadas pelo próprio SDK, utilizando o comando:

```
android create Project
```

[12]Secure Digital

O desta invocação é uma pasta contendo as seguintes pastas e ficheiros:

- **AndroidManifest.xml:** ficheiro XML que descreve a aplicação e os seus componentes. Uma descrição mais detalhada sobre este ficheiro, bem como um exemplo, podem ser encontrados no ponto seguinte desta dissertação.
- **build.xml:** *script* ANT cuja função é compilar e instalar a aplicação desenvolvida num dispositivo.
- **default.properties** e **local.properties:** ficheiros com as propriedades utilizadas no ficheiro build.xml.
- **assets/:** pasta que contem todos os conteúdos estáticos que deverão ser anexados à aplicação no momento de instalação no dispositivo.
- **bin/:** pasta onde a aplicação compilada será guardada.
- **gen/:** pasta onde as ferramentas de geração automática do SDK Android colocarão todo o código gerado.
- **libs/:** pasta onde deverão ser colocados todas as bibliotecas externas necessárias à aplicação.
- **src/:** pasta que contém o código da aplicação.
- **res/:** pasta que contem todos os recursos que deverão ser compilados juntamente com a aplicação, por exemplo ícones, ficheiros XML com as vistas, etc.
- **tests/:** pasta onde devem ser colocados os projectos usados nos testes.

Importa ainda salientar a existência do ficheiro *R.java*, gerado sempre em todas as compilações pelo *script* build.xml. Este ficheiro é o responsável pelo acesso, por parte das actividades, aos recursos contidos na pasta *res/*.

4.2.6 AndroidManifest.xml

A base de qualquer aplicação Android é definida no ficheiro *AndroidManifest.xml*. Neste ficheiro é declarada a composição da aplicação, enunciando todas as suas actividades, serviços, etc. É também neste ficheiro que é especificada qual a actividade com que a aplicação deve iniciar e os serviços do sistema operativos a que esta pretende ter acesso [Murphy, 2010].

```
<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
  package="org.me.localchat_client">
  <application>
   <uses-library android:name="com.google.android.maps" />
    <activity android:name=".Main" android:label="Main">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category
           android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
    <activity
       android:name=".Home" android:label="Home"></activity>
    <activity
       android:name=".Map" android:label="Map"></activity>
  </application>
  <uses-permission
       android:name="android.permission.INTERNET" />
  <uses-permission
       android:name="android.permission.ACCESS_GPS" />
</manifest>
```

Neste caso concreto, o ficheiro AndroidManifest.xml indica descreve uma aplicação que utiliza a biblioteca de mapas e que é composta por três actividades, sendo a actividade *Main* aquela que é iniciada junto com a aplicação. Além das actividades, é também descrito que a aplicação deve ter acesso à internet e á informação vinda do sensor GPS.

4.2.7 Interfaces gráficas - Vistas

As interfaces gráficas Android são normalmente denominadas de *vistas*[13].

Estas vistas estão normalmente associadas a actividades e podem ser definidas utilizando unicamente código Java. Porém, esta abordagem apenas é aconselhada nos casos em que estas vistas são dinâmicas, isto é, não conhecem em tempo de compilação os conteúdos que terão para apresentar. Para os restantes casos, as vistas devem ser definidas em ficheiros XML e integrados nas actividades.

Os ficheiros XML são compostos pela árvore de elementos que constituem a vista. Por sua vez, estes elementos possuem atributos cuja função é des-

[13] *Views* na terminologia inglesa

crever as propriedades visuais e comportamentais de cada elemento, ou seja, a forma como ele deve ser mostrado e qual o comportamento que deve ter. Na Figura 4.1 pode ser observado aspecto correspondente à seguinte vista definida em XML.

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
   android:id="@+id/widget0"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
>
   <TextView
       android:id="@+id/title"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="MinhaAplicação"
       android:textSize="28sp"
       android:gravity="center"
       android:layout_x="0px"
       android:layout_y="126px"
   >
   </TextView>
   <Button
       android:id="@+id/enter"
       android:layout_width="200px"
       android:layout_height="wrap_content"
       android:text="Entrar"
       android:textStyle="bold"
       android:layout_x="58px"
       android:layout_y="194px"
   >
   </Button>
</AbsoluteLayout>
```

A separação entre o código Java das actividades e os ficheiros XML das vistas permite que os dois componentes sejam desenvolvidos de forma quase independente, apenas é necessário estabelecer parâmetros como seja o nome dos campos, etc. Desta forma, o processo de codificação pode ser atribuído a um programador, sem quaisquer conhecimentos de interfaces gráficas em Android, e o processo criativo das vistas delegado a um especialista em interfaces, que não precisa sequer de ter conhecimentos em Java. Esta separação

Figura 4.1: Aspecto de uma vista Android.

permite ainda a evolução das vistas ou do modelo lógico da aplicação sem que haja necessidade de alterar o outro componente.

Já foi referida a possibilidade de integração do SDK Android nos IDEs Netbeans e Eclipse. Contudo, estes IDEs não facilitam em muito o processo de criação das vistas. O projecto DroidDraw[14], é independente da plataforma e permite a criação de vistas através do arrasto dos elementos da interface. Permite a sua organização por *layouts* (absoluto, relativo, lista, etc.) e a edição de algumas das suas propriedades mais relevantes. Como resultado, este programa gera um ficheiro XML com a interface criada. Estando ainda me fase beta, o DroidDraw possui ainda algumas limitações, visto não possibilitar a edição de todas as propriedades possíveis dos elementos, e também alguns problemas de funcionamento. Por exemplo, ao nível da importação de vistas já geradas.

[14] `DroidDraw`

4.3 Extensible Messaging and Presence Protocol (XMPP)

O protocolo XMPP [Saint-Andre, 2005b], anteriormente denominado Jabber, foi desenvolvido para aplicações com necessidades de troca de mensagens em tempo real e informação acerca da presença dos utilizadores. Foi, portanto, criado como uma alternativa aberta aos protocolos existentes de troca de mensagens instantâneas utilizados nas aplicações de conversação. Tendo sido desenhado de forma a ser extensível, este protocolo tem evoluído e neste sentido têm sido adicionadas novas funcionalidades como salas de conversação multi-utilizador ou a possibilidade de marcação de favoritos.

O XMPP segue a filosofia cliente-servidor em que cada servidor pode formar uma rede privada, como mostra a Figura 4.2 ou então estar integrado na rede XMPP global, que é pública. Todo o processo de comunicação entre os clientes e o servidor é feito utilizando mensagens em XML.

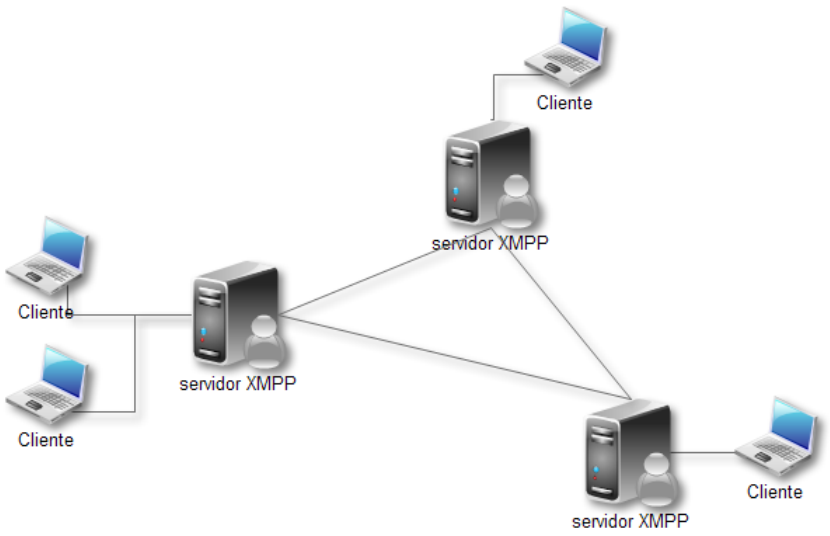

Figura 4.2: Arquitectura de uma rede XMPP

Cada utilizador do serviço XMPP possui uma conta com um identificador único, normalmente denominado de *JabberID* [Saint-Andre et al., 2009]. É composto por dois campos: o nome do utilizador e o domínio do servidor, tal como podemos ver no seguinte *JabberID* de exemplo:

```
utilizador@servidor.com
```

As principais vantagens deste protocolo são:

- Descentralização: a sua arquitectura faz com que qualquer entidade possa ter o seu próprio servidor, não havendo um servidor central. Desta forma não existe a dependência de um servidor único, aumentando a tolerância a falhas.

- Segue especificações abertas: a Internet Engineering Task Force (IETF) aprovou o protocolo como uma tecnologia para troca de mensagens instantâneas e controlo de presença. Sendo um protocolo aberto, não existe também a necessidade de se pagar qualquer valor pela sua utilização.

- Segurança: os servidores XMPP podem, como já foi referido, ser configurados para correr de forma isolada, em redes privadas ou integrados na rede XMPP global. Para qualquer um dos casos, existe a possibilidade de utilizar mecanismos de segurança como Simple Authentication and Security Layer (SASL) [Myers, 1997] e Transport Layer Security (TLS) [Dierks and Rescorla, 2008].

- Flexibilidade: o protocolo XMPP foi desenhado de forma a ser extensível. Desta forma, qualquer desenvolvedor pode criar e integrar novas funcionalidades sob a forma de extensões. Estas novas adições são geridas pela própria XMPP Software Foundation.

- Integração: este protocolo permite a integração com outros protocolos de conversação baseados em mensagens instantâneas. Desta forma é possível a um utilizador do protocolo XMPP comunicar com um outro que utilize um protocolo concorrente, como o por exemplo o Windows Messenger.

- Casos práticos: as tecnologias baseadas neste protocolo têm sido utilizadas desde 1998 e são suportadas por grandes empresas como a Google. Esta empresa utiliza mesmo o protocolo XMPP no serviço de conversação Google Talk[15], que se encontra actualmente integrado no serviço Gmail e na rede social Orkut. Este serviço possui também um cliente *desktop* disponível para a plataforma Microsoft Windows. Além do Google Talk, o protocolo XMPP é também utilizado no serviço de conversação do Facebook, tal como já foi referido neste documento.

Em contrapartida, o XMPP possui alguns pontos contra:

[15] `http://www.google.com/talk/`

- *Overhead* de dados de presença: existe algum *overhead* na entrega dos dados relacionados com a presença dos utilizadores.
- Ineficiente transmissão dos dados binários: sendo as mensagens XMPP formatadas num único e longo documento XML, os dados binários têm de previamente ser codificados em base de 64 bits. Visto isto, este protocolo não é aconselhado para a transferência de ficheiros. No entanto já existe uma extensão capaz de minimizar este problema[16].

4.3.1 Servidores

Existe uma grande lista de servidores XMPP, pelo que vamos apenas referir alguns:

- djabberd[17]
- ejabberd (já abordado no capítulo sobre a linguagem Erlang)
- iChat server[18]
- inetdxtra[19]
- Openfire[20]
- OpenIM[21]
- Sun Java System Instant Messaging[22]
- ...

4.3.2 Clientes

Tal como o caso dos servidores, a lista de clientes é também bastante extensa, existindo aplicações para variadas plataformas:

- **Apple MacOS**: Adium[23], iChat[24];

[16] `http://xmpp.org/extensions/xep-0166.html`
[17] `http://www.danga.com/djabberd/`
[18] `http://www.apple.com/server/macosx/features/ichat-server.html`
[19] `http://inetdxtra.sourceforge.net/#jabberd`
[20] `http://www.igniterealtime.org/projects/openfire/index.jsp`
[21] `http://www.openim.techlab.smk.fr/en/`
[22] `http://www.sun.com/software/products/instant_messaging/`
[23] `http://adium.im/`
[24] `http://www.apple.com/macosx/what-is-macosx/ichat.html`

- **Consola / Modo de texto**: GNU Freetalk[25];
- **Linux**: Kopete[26];
- **Microsoft Windows**: MirandaIM[27], Trilian Pro[28];
- **Plataformas móveis**: Pidgin[29], Psi[30];
- **Multi-plataforma**: IM+[31];
- **Web**: ijab[32], JWChat[33].

4.3.3 Bibliotecas

Além de clientes e servidores XMPP, existem também disponíveis bibliotecas desenvolvidas nas mais variadas linguagens de programação, tais como C, C++, C#, Erlang, Java, Javascript, Perl, PHP, etc. Estas bibliotecas facilitam o desenvolvimento de novas aplicações que necessitem de se integrar com servidores XMPP.

4.4 Resumo

As tecnologias exploradas ao longo deste capítulo terão um papel extremamente relevante no sistema LocalChat. A biblioteca erlFBGraph permite agilizar a interacção com a plataforma do Facebook, implementando métodos capazes de obter quaisquer dados dos seus objectos ou das relações entre eles. Por sua a vez, a tecnologia Android está normalmente associada a dispositivos com boa capacidade de processamento, conectividade e sensores capazes de sentir o contexto do utilizador. Este facto, aliado à sua plataforma de desenvolvimento poderosa e bem documentada justificam a sua escolha. Por último, o XMPP é uma alternativa aberta aos protocolos de troca de mensagens instantâneas em tempo real. Possui uma variada lista de clientes, servidores e bibliotecas para diversas linguagens, incluindo Erlang e Java,

[25] http://www.gnu.org/software/freetalk/
[26] http://kopete.kde.org/
[27] http://www.miranda-im.org/
[28] http://www.trillian.im/
[29] http://pidgin.im/
[30] http://psi-im.org/
[31] http://www.shapeservices.com/en/products/details.php?product=im&platform=none
[32] http://code.google.com/p/ijab/
[33] http://blog.jwchat.org/jwchat/

e é utilizado em vários serviços bem sucedidos. Será, portanto, uma parte fundamental do serviço de conversação do serviço de conversação do LC.

Capítulo 5

Caso de estudo

O trabalho desenvolvido nesta dissertação propõe-se a criar uma arquitectura conceptual para serviços capazes de potenciar a interacção social e sensível ao contexto dos seus utilizadores. Propõe-se também a desenvolver um serviço deste género que evidencie a utilidade e comprove a arquitectura proposta. Este serviço foi denominado de LocalChat e, para uma melhor compreensão sobre qual o seu público-alvo, é apresentado um cenário na secção 5.1. Nas secções seguintes, é explorada e justificada a abordagem arquitectural pensada.

5.1 Pessoas interessantes na minha região

O David, um jovem das novas tecnologias e aficionado por tudo o que é social, possui um *smartphone* Android de última geração, com conectividade à internet e sensor GPS integrado. O seu vício em redes sociais leva-o a actualizar o seu perfil onde quer que esteja e a estar sempre em busca de novas pessoas interessantes.

No Facebook, seguindo o grafo social dos seus contactos, dos contactos dos seus contactos e assim sucessivamente, o David conta já com uma imensa lista de amigos e conhecidos oriundos dos mais variados pontos do país e do mundo. No entanto, chegou à conclusão que aquela infinidade de contactos não corresponde verdadeiramente à realidade, isto é, com tamanha dispersão geográfica, a maioria dos contactos não passam de um mero perfil pessoal. Concluindo isto, o David deseja encontrar e interagir com pessoas da sua região e que estejam integradas no seu meio, aumentando as hipóteses de as vir realmente a conhecer pessoalmente.

Para tal, o David necessita de uma aplicação que lhe permita pesquisar outros utilizadores tendo como base a sua localização geográfica e que permita

a interacção com eles. A visualização de informação pessoal, tal como o filtro por estado são também funcionalidades importantes. Desta forma, o David pode saber mais sobre os outros utilizadores antes de enviar um convite para conversação e, através do filtro, pode receber apenas aqueles resultados que estejam disponíveis a aceitar esse convite.

Com esta aplicação, o David experimenta um novo conceito de interacção social, o qual depende inteiramente da localização onde se encontra e lhe permite encontrar pessoas com as mesmas necessidades que ele.

5.2 Decisões relevantes

Tal como descrito durante a apresentação do problema, a localização geográfica e o estado dos utilizadores são informações de contexto que podem ser interessantes de ser abordados. Todavia, embora estas tenham sido as informações referidas, outras podem ser considerados relevantes.

Para que se possa considerar o contexto do utilizador, é necessária uma forma de o obter. Podem ser utilizados dispositivos próprios para o efeito. Porém, esta abordagem não seria muito prática nem económica. Este caso obrigaria cada utilizador a transportar dispositivos extra e além da usabilidade, esses dispositivos têm também o seu custo. Outra solução passa por utilizar *smartphones* como o do David, personagem do caso apresentado na secção anterior. Estes dispositivos mais actuais contam já com conectividade à internet e vários sensores embebidos, entre os quais sensores GPS. Utilizando esta abordagem, o utilizador não tem a despesa de adquirir novos dispositivos, nem necessita de os transportar consigo apenas para a obtenção do contexto. Em alternativa aos sensores, pode ser utilizada uma obtenção do contexto mais passiva, sendo o próprio utilizador a inserir dados relevantes do seu contexto. No entanto, esta seria a forma menos precisa e mais passível de se utilizarem informações falsas.

Sendo a localização geográfica actual dos utilizadores a base de todo o processo, é imperativo que esta informação seja obtida da forma mais precisa. Não interessa a um utilizador em viagem que, a pesquisa seja efectuada considerando uma localização errada. Neste caso, toda a ideia fundamental de promover a interacção entre utilizadores próximos seria deitada por terra. O processo de procura pode ser efectuado utilizando tecnologias sem fios como o Bluetooth, no entanto o curto alcance implicaria que o resultado das pesquisas fosse vazio na maioria dos casos. Como alternativa mais fiável e abrangente podem ser consideradas as coordenadas GPS, que se podem obter a partir dos mesmos *smartphones* utilizados com sensores de contexto.

Tendo solucionado o problema da obtenção de informações contextuais do

utilizador, existe agora a necessidade de obter informações pessoais acerca dos restantes. A necessidade desta funcionalidade prende-se com o facto de os utilizadores terem acesso a dados mais ricos e actualizados das pessoas com quem pretendem interagir. Uma opção é a auto-inserção destas informações directamente no serviço. No entanto estas informações podem ficar rapidamente obsoletas e, como tal, nada melhor do que utilizar os serviços de redes sociais existentes, onde os seus utilizadores actualizam as suas informações regularmente. O acesso a estes dados é possibilitado pelas APIs remotas e plataformas disponibilizadas por estes serviços. Uma questão importante quando se acede a este tipo de informações é a privacidade dos utilizadores.

Neste caso específico, foi escolhida a rede social Facebook, pelas seguintes razões:

- **Número de utilizadores:** neste momento o Facebook é uma das redes sociais mais utilizadas em todo o mundo, chegando recentemente aos 500 milhões de utilizadores.

- **Tipo de rede social:** tal como já foi referido na secção 2.1.6, esta é uma rede social que funciona como uma extensão social da vida real dos seus utilizadores e em que os seus contactos são normalmente os do dia-a-dia. Visto isto, é natural os utilizadores estarem constantemente a partilhar novas informações pessoais, e é neste tipo de informações que o sistema aqui apresentado está interessado.

- **API:** o Facebook disponibiliza uma plataforma rica para desenvolvimento de aplicações.

Para garantir este direito, é fundamental que estes sejam informados e autorizem previamente os dados que serão tornados públicos. Um sistema que permita a pesquisa de outros utilizadores, geograficamente próximos, e que forneça algumas informações pessoais sobre eles, não soluciona só por si o problema referido na secção anterior. É necessário algo que promova a uma interacção social mais activa entre os utilizadores, como um sistema de conversação.

5.3 Arquitectura conceptual

Tomando em consideração os pontos abordados na secção anterior, foi elaborada uma lista de requisitos e uma arquitectura contendo a melhor solução para cada um deles e que satisfizesse as necessidades dos utilizadores.

5.3.1 Requisitos

Os requisitos funcionais do problema apresentado no início deste capítulo foram compilados no diagrama de casos de uso da Figura 5.1.

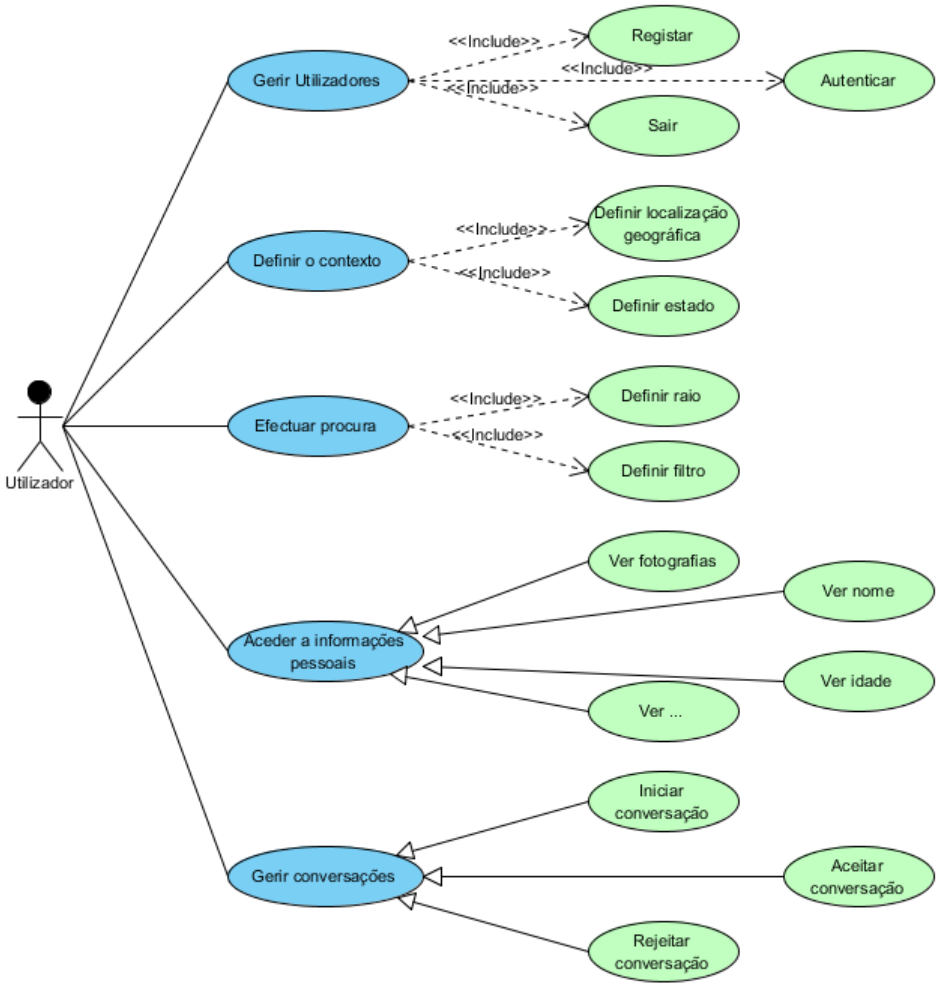

Figura 5.1: Compilação dos requisitos funcionais do caso de estudo.

A identidade do utilizador, bem como os seus dados pessoais e o seu contexto, são de extrema relevância para este serviço. Como tal, é importante garantir a sua segurança e privacidade e, embora não seja este o principal objectivo desta dissertação, foram tidos em consideração alguns requisitos básicos a este nível. Em termos práticos, são requisitos o registo de novos utilizadores e que estes se autentiquem ao entrarem no serviço.

A definição do contexto por parte do utilizador é fundamental e a base

de todo o serviço. Neste âmbito, o utilizador pode e deve definir informações como a sua localização geográfica ou o seu estado.

O processo de procura, além de personalizado com a localização geográfica, tem também em consideração um raio definido pelo utilizador. O utilizador pode ainda definir um filtro relativo ao estado dos utilizadores a incluir no resultado da pesquisa.

Sendo o resultado do processo de procura uma lista de utilizadores processada de acordo com os parâmetros definidos, é dada a possibilidade ao utilizador de aceder a algumas informações pessoais. A título de exemplo podemos destacar o nome, a idade, fotografias, etc.

A interacção entre utilizadores é também um requisito fundamental, sendo que sem ele este sistema perde grande parte da sua utilidade. Esta necessidade é colmatada com um sistema de conversação onde é possível ao utilizador enviar, recusar e aceitar convites para salas de conversação privadas. A possibilidade de enviar convites apenas é permitida caso o destinatário tenha definido o seu estado como disponível, sendo mais uma vez o contexto do utilizador relevante.

5.3.2 Arquitectura

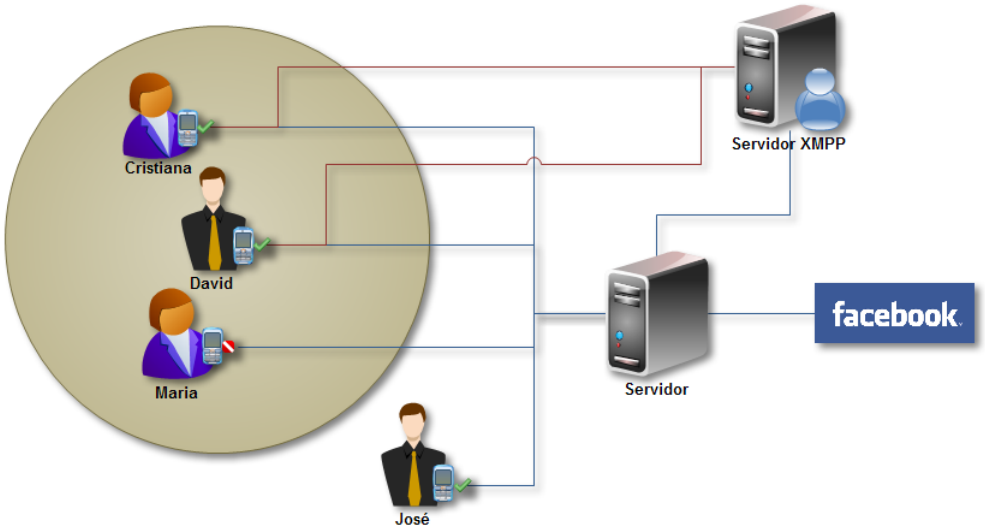

Figura 5.2: Arquictura geral da solução.

A Figura 5.2 apresenta uma visão geral da solução. A arquitectura segue o modelo cliente-servidor, comunicando entre si através do protocolo HTTP.

Neste sistema, os clientes agregam as responsabilidades de interacção com o utilizador e de obtenção do contexto. Para tal, tiram partido das capacidades dos *smartphones* actuais e dos funcionalidades que estes e os seus sistemas operativos disponibilizam. Entre elas encontram-se o acesso a informação contextual, como localização geográfica, e a facilidade de desenvolvimento e integração de aplicações.

Desta forma, e como os dispositivos estão actualmente sempre junto aos seus utilizadores, fazem com que sejam a melhor opção para obtenção de informações e interacção com a aplicação. Para os utilizadores, este é o único ponto de interacção com o sistema. É através das aplicações móveis que acedem às funcionalidades fornecidas e actualizam as informações relacionadas com o seu contexto.

Por sua vez, o servidor é a entidade comum a todos os clientes, tal como mostra a Figura 5.2, e tem como função processar os dados dos utilizadores e as funcionalidades a que eles acedem. Sendo este um serviço onde a identidade dos utilizadores e os seus dados são importantes, o servidor é também o responsável pela sua autenticação.

É possível observar na Figura 5.2 a existência de um outro servidor, o servidor XMPP. Este componente é incluído nesta arquitectura para satisfazer o requisito de ser possível a conversação entre utilizadores. Na mesma figura é também possível observar a ligação entre este componente e o servidor. Esta ligação simboliza, na prática, a gestão das contas dos utilizadores no servidor XMPP, como sendo a criação, eliminação e alteração de dados. Ainda relacionado com este componente e recorrendo à Figura 5.2, pode-se observar a ligação entre dois clientes (David e Cristiana), representando uma conversação. Desta forma, todas as mensagens enviadas pelo David à Cristiana, e vice-versa, passam pelo servidor XMPP que se encarrega de as encaminhar ao seu destino. A rede social Facebook permite a integração de aplicações externas com o seu serviço de conversação, no entanto não permite a adição e remoção dinâmica de contactos. Assim sendo, embora seja uma solução escalável e que simplificaria o processo de desenvolvimento, não satisfaz os requisitos deste sistema, pelo que se optou por um servidor XMPP autónomo.

A ligação do servidor às redes sociais, mais precisamente ao Facebook, é também evidente na Figura 5.2. A integração destes serviços foi a opção escolhida para obter informações pessoais acerca dos utilizadores. Estas informações são posteriormente analisadas e processadas pelo servidor e devolvidas como resposta às solicitações dos clientes.

Por último, a disposição dos utilizadores simboliza uma pesquisa feita por parte do David. Neste caso, o resultado dessa pesquisa foram a Cristiana e a Maria, sendo impossível ao David iniciar uma conversa com esta última visto

que definiu o seu estado como *ocupada*[1]. O José não foi incluído no resultado por se encontrar fora do raio de pesquisa definido pelo David, representado pelo círculo que envolve as duas utilizadoras e o David.

5.4 Resumo

Neste capítulo foi explorado o âmbito do caso de estudo desta dissertação, isto é, foi enquadrada a necessidade de criação de funcionalidades sociais sensíveis ao contexto. Foram discutidas algumas estratégias que possibilitariam a criação de serviços deste tipo e apresentados os requisitos funcionais. No geral podem ser destacados: (1) a utilização de informação contextual, pela utilização da localização geográfica e do estado do utilizador, (2) a interacção social, pela possibilidade de os utilizadores poderem comunicar entre si e (3) a extensão das redes sociais, com a utilização da rede social Facebook como fonte de dados pessoais.

A solução arquitectural apresentada tem em consideração os requisitos mencionados. É baseada no modelo cliente-servidor, sendo os clientes constituídos por aplicações móveis para uma interacção mais rica por parte do utilizador e para uma obtenção do contexto mais precisa. Este modelo possibilita também a extensibilidade do serviço com a criação de clientes distintos. O componente mais crítico do sistema é o servidor, que agrega toda a informação dos clientes e faz a gestão das informações pessoais vindas do Facebook e do serviço de conversação entre utilizadores, através do protocolo XMPP

O sistema LocalChat, que implementa a arquitectura apresentada está descrito nos capítulos 6 e 7.

[1]simbolizado pelo traço ao lado do dispositivo móvel, enquanto o estado *disponível* é simbolizado pelo tic

Capítulo 6

Servidor

Os requisitos e a arquitectura apresentada no capítulo 5.3.2 foram concretizados num sistema de *software*, o LocalChat (LC). Ao longo deste capítulo será abordada a vertente de servidor apresentada na arquitectura, evidenciando as decisões tomadas, as tecnologias e a forma como foram utilizadas.

6.1 Tecnologia

Sendo um componente central do serviço LC, o servidor adiciona aos requisitos funcionais do sistema alguns não funcionais. Entre eles encontram-se os da escalabilidade, performance e disponibilidade.

O estudo realizado sobre a linguagem de programação Erlang, e descrito na secção 3, mostram que esta tecnologia foi pensada de raiz para o desenvolvimento deste tipo de soluções. Foram mostradas algumas funcionalidades extremamente úteis, como a plataforma OTP e o servidor Web YAWS. Além disso, foi também possível observar a sua utilização em vários sistemas reais. Entre eles incluem-se componentes das redes sociais e em serviços críticos como sendo a rede GPRS. Considerando este estudo, a escolha da tecnologia para o desenvolvimento do servidor LC recaiu sobre Erlang, tentando tirar o máximo proveito das suas características.

6.2 Arquitectura

A Figura 6.1 apresenta uma visão geral da arquitectura do servidor LC. Nela podemos observar que segue uma abordagem multi-camada, havendo uma divisão entre os componentes responsáveis pelos dados (dados.app), lógica aplicacional (lógica.app) e interface com o cliente (interface.app). Os detalhes

e as decisões tomadas em cada camada aplicacional, bem como as ligações entre camadas, serão abordados nas respectivas secções ao longo do capítulo.

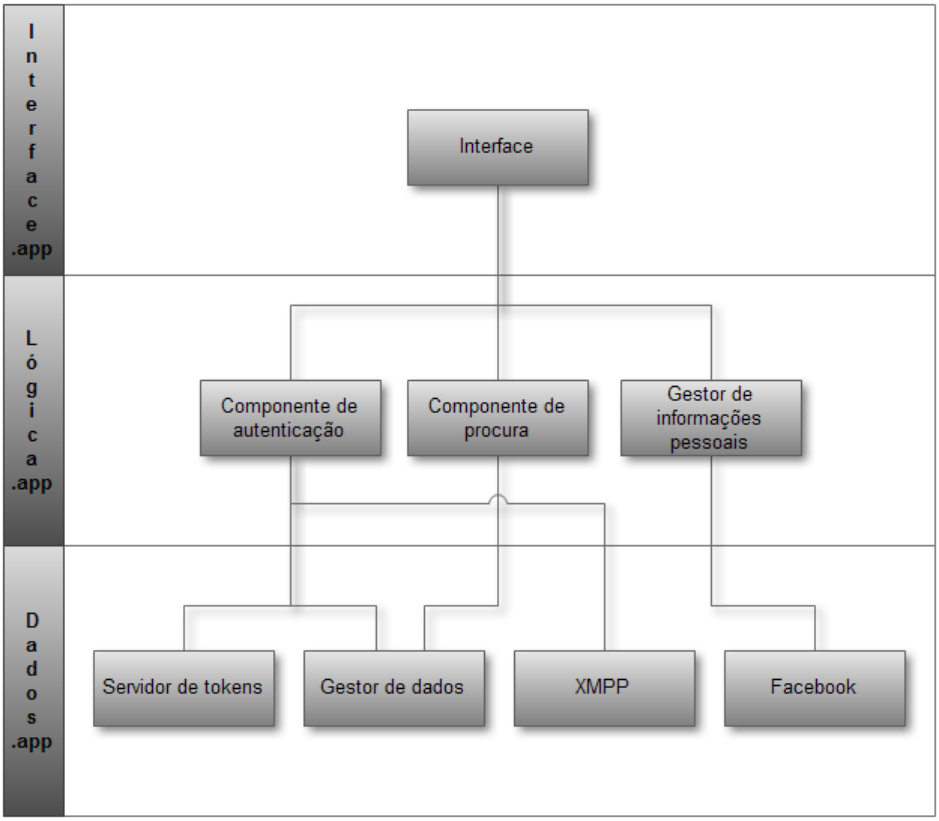

Figura 6.1: Arquitectura do servidor LocalChat.

As extensões *.app* encontradas no nome das camadas representadas na Figura 6.1 simbolizam que cada camada está desenvolvida como uma aplicação Erlang isolada. Todas elas, no entanto, encontram-se incluídas numa aplicação principal(LC.app). Esta inclusão está descrita na Figura 6.2, onde os círculos mais pequenos simbolizam as aplicações correspondentes às camadas, enquanto que o círculo superior, e maior, representa a aplicação principal. Esta opção, além de facilitar o processo de manutenção do código, permite também um desenvolvimento praticamente autónomo das três camadas, em que apenas é necessário ter conhecimento das APIs de cada uma. É inclusive possível a compilação e execução autónoma de cada aplicação Erlang, permitindo uma mais fácil e célere detecção de erros.

A inclusão das aplicações interface.app, logica.app e dados.app pela aplicação

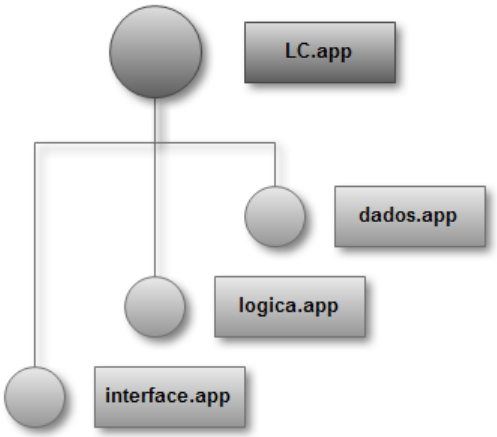

Figura 6.2: Integração das aplicações Erlang que compõem o sistema Local-Chat.

principal (LC.app) é definida seguinte coniguração:

```
{application, LC, [
 {description, "Servidor LocalChat - Aplicação principal."},
 {vsn, "0.1"},
 {modules, [LC, interface, logica, dados]},
 {registered, [LC_app]},
 {included_applications, [interface, logica, dados ]},
 {applications, [kernel, stdlib, sasl]},
 {mod, {app_test, []}},
 {start_phases, []}
]}.
```

A variável responsável por esta inclusão é denominada *included_applications*.

6.3 Árvore de supervisão

A tecnologia Erlang disponibiliza aos programadores a plataforma OTP[1]. Nela podemos encontrar desde várias bibliotecas úteis a um sistema de base de dados distribuído, que facilitam bastante o processo de desenvolvimento de aplicações escaláveis, distribuídas e disponíveis.

[1]plataforma já abordada ao longo da secção 3.4

A plataforma OTP possibilita também que os programadores apliquem alguns comportamentos ao seu código, obtendo como resultado uma maior tolerância a faltas e consequentemente uma maior disponibilidade das aplicações.

No caso do servidor LC foram aplicados dois comportamentos: o *supervisor* e o *server*. Este segundo foi utilizado em todos os componentes presentes na Figura 6.1, visto serem componentes desenvolvidos para realizarem determinadas acções específicas da aplicação. No entanto, adoptando a filosofia de desenvolvimento não defensiva encorajada pelo Erlang, é assumido que qualquer um dos componentes pode falhar. Nesta filosofia, mais importante que a falha é a forma como o sistema consegue recuperar dela, e para tal foi criado um novo processo que implementa o comportamento *supervisor* em cada camada aplicacional.

Este processo supervisor é executado pela aplicação e fica responsável por iniciar, e reiniciar sempre que ocorram falhas, todos os processos com comportamento de servidor, tal como mostra a Figura 6.3 para o caso da aplicação *logica.app*.

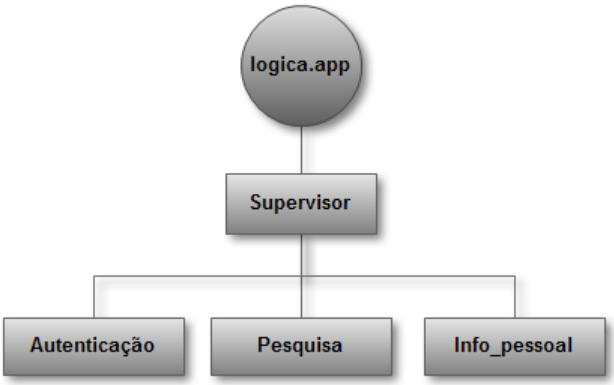

Figura 6.3: Arquitectura de componentes da aplicação *logica.app*.

Visto não haver dependências entre os componentes de uma aplicação, foi adoptada uma política em que apenas é reiniciado o processo que falhou (*one-for-one*). Em contra-partida poderia ter sido utilizada a política de reiniciar todos os processos, mas neste caso isso não traria nenhuma vantagem, podendo mesmo matar processos que estariam a executar correctamente.

6.4 Interface com o cliente

O processo de comunicação entre o cliente e o servidor é assegurado, neste último, através da aplicação Erlang *interface.app*. Esta aplicação corresponde à camada de interface do sistema e, tal como demonstra a Figura 6.1, é composta apenas por um componente, denominado *RESTWebService*. O seu objectivo é receber os pedidos vindos dos clientes, encaminhá-los para a camada de lógica aplicacional, que os executará, e formatar e encaminhar os resultados a esses pedidos de volta para o cliente. Esta aplicação implementa um serviço Web do tipo RESTfull, acessível via HTTP. A escolha por esta tecnologia justifica-se essencialmente pela sua acessibilidade e performance.

A utilização do protocolo HTTP permite que o tráfego gerado na comunicação, entre o cliente e o servidor, não encontre obstáculos na generalidade das *firewalls*. A utilização de um serviço Web, torna também possível o acesso ao servidor a partir de qualquer dispositivo com ligação à internet, seja ele fixo ou móvel. Mais do que a acessibilidade em termos de conectividade, desta forma o servidor permite a comunicação com clientes desenvolvidos nas mais diversas linguagens, sendo apenas necessário respeitar a API disponibilizada.

Para melhorar o desempenho e diminuir o volume de dados a ser transferidos, foi convencionado que o serviço Web seria do tipo RESTfull. Desta forma evita-se a utilização de um protocolo bastante verboso como é o XML, diminuindo o tempo de processamento necessário para a análise dos pedidos vindos do cliente e criação das respostas a esses pedidos. Com um protocolo menos verboso diminui também a carga na rede, visto que os dados gerados são menores.

Continuando com a ânsia de conseguir uma boa performance, foi escolhido o servidor Web YAWS. É um servidor desenvolvido em Erlang e cuja performance foi comparada favoravelmente em relação a outros servidores, tal como foi referido na secção 3.

6.5 API

Foi referida a opção por um serviço Web do tipo RESTfull, em detrimento da tecnologia WSDL[2]. Desta forma evitamos a utilização de um protocolo verboso como XML. Contudo, a utilização de uma formatação standard é tida como uma mais-valia no processo de desenvolvimento, sendo possível recorrer a bibliotecas cliente já existentes.

A sua leveza e a existência de bibliotecas para as linguagens mais populares e Erlang foram as razões que fizeram com que a escolha recaísse sobre a

[2]Web Services Description Language

formatação JSON. Desta forma, o servidor continua a dar liberdade à criação de clientes em diversas linguagens e reduz o tamanho das mensagens trocadas.

A aplicação *interface.app* disponibiliza todos os métodos necessários para dar resposta à globalidade dos requisitos abordados no capítulo 5.3.1. Estes métodos encontram-se divididos em três grupos (Autenticação, Procura e Informações pessoais), tendo em consideração o seu propósito. Além de uma divisão lógica dos métodos, este agrupamento permite desde logo que os pedidos sejam encaminhados para diferentes componentes da camada de lógica aplicacional, que será abordada na secção seguinte.

O grupo de autenticação contém métodos relacionados com a autenticação, criação e eliminação de registos no serviço LC:

- **/auth/login/ -** método responsável pela recepção dos pedidos de autenticação no sistema. Recebe como parâmetros o nome do utilizador e a sua palavra-chave, retornando em caso de sucesso alguns dados relativos ao utilizador e o valor que identifica a sessão a ser utilizado nos métodos em que é necessária autenticação. Em termos práticos, a resposta a um pedido de autenticação bem sucedido no formato JSON será algo como:

```
{
   "info":"Success",
   "token":"vudlaX\\Vn#?T+\"FV",
   "xmpp_server":"xmppserver:5222",
   "xmpp_username":"david",
   "xmpp_password":"david_password",
   "picture":"http://profile.ak.fbcdn.net/
       hprofile-ak-snc4/hs459.snc4/
       48587_100000963928649_4066_q.jpg"
}
```

- **/auth/logout/ -** Este método tem o propósito contrário ao anterior, isto é, permite que um utilizador saia do sistema, apagando os dados relativos à sua sessão. Como parâmetros recebe o identificador de sessão devolvido pelo método de */auth/login/*. Em caso de sucesso retorna uma mensagem informativa.

- **/auth/new_user/ -** permite o registo de um novo utilizador no serviço LocalChat. Os parâmetros a receber são o nome do utilizador, a palavra-chave pretendida, e o identificador da sua conta Facebook para

a aplicação possa posteriormente aceder aos seus dados pessoais. Alternativamente ao identificador pode ser fornecido o *username* do utilizador. Em caso de sucesso retorna uma mensagem informativa.

- **/auth/delete/ -** método oposto ao anterior, ou seja, possibilita que um utilizador autenticado elimine de forma definitiva a sua conta no serviço LocalChat. Para tal, deve receber o identificador de sessão do utilizador. Tal como nos métodos anteriores, em caso de sucesso é retornada uma mensagem informativa.

Por sua vez, o grupo de procura é composto pelos métodos responsáveis pela actualização do contexto do utilizador e pela procura de utilizadores próximos:

- **/search/update_location/ -** método que permite a actualização da posição geográfica do utilizador. Além do identificador de sessão, recebe também como parâmetros a latitude, a longitude, a altitude e pode ainda receber o estado do utilizador. Uma mensagem informativa é retornada em caso de sucesso.

- **/search/update_status/ -** actualiza o estado do utilizador. Este estado pode tomar os valores de "Disponível", "Ausente" e "Ocupado". Além deste valor, deve também receber o identificador de sessão. Caso o método seja processado com êxito, é enviada uma mensagem informativa como resposta.

- **/search/search/ -** recebe os pedidos de pesquisa efectuados pelos utilizadores. Para tal, além do identificador de sessão, recebe também as coordenadas actuais do utilizador, o raio de pesquisa e três valores booleanos, definindo o filtro por estado que deve ser aplicado aos resultados. Ao cliente é devolvida uma lista contendo os nomes dos utilizadores que satisfazem os requisitos definidos na pesquisa.


```
{
   "users":[
       {"username":"David",
       "latitude":"4.157350054999999985545e+01",
       "longitude":"-8.396691000000000057094e+00",
       "altitude":"0.000000000000000000000e+00",
       "status":"Available",
       "picture":"http://profile.ak.fbcdn.net/
           hprofile-ak-snc4/hs459.snc4/
```

```
            48587_100000963928649_4066_q.jpg"},
        {"username":"Maria",
        "latitude":"4.1503500000000025011e+01",
        "longitude":"-8.3066899999999968509e+00",
        "altitude":"0.0000000000000000000e+00",
        "status":"Available",
        "picture":"http://profile.ak.fbcdn.net/
            hprofile-ak-snc4/hs446.snc4/
            49142_100001625494904_2353_q.jpg"}]
}
```


Por último, a constituição do grupo que contem os métodos relacionados com a obtenção de informações pessoais é:

- **/search/update_fb_token/ -** método que recebe o identificador de sessão que será utilizado na interacção com a plataforma do Facebook para obtenção de dados pessoais. Este valor é proveniente do processo de autenticação OAuth ocorrido no cliente. Este método recebe também como parâmetro a chave de sessão do utilizador para que o servidor possa fazer a devida associação.

- **/user_info/get_info/ -** em caso de sucesso retorna uma lista de informações pessoais acerca de um utilizador passado como parâmetro. Entre essas informações encontram-se os dados retirados do Facebook e o endereço XMPP, caso esse utilizador tenha definido o seu estado como "Disponível". Visto tratar informações sensíveis, este é um método em que é necessária autenticação, logo deve também ser passado como parâmetro o identificador de sessão obtido do método */auth/login/*.

- **/user_info/get_feed/ -** através deste método, é possível ter acesso às últimas actualizações do perfil Facebook de um dado utilizador, isto é, aos últimos comentários na sua *wall*. Para este efeito, este método recebe o identificador de sessão e o nome do utilizador alvo.

- **/user_info/get_photos/ -** método que devolve uma lista contendo os endereços das fotografias publicadas no perfil Facebook de um dado utilizador. Tal como os dois outros métodos deste grupo, este recebe como parâmetros o identificador de sessão e o nome do utilizador sobre o qual incide este pedido.

Importa ainda salientar que todos os métodos devolvem uma mensagem de erro em caso de insucesso e estão apenas acessíveis através de pedidos HTTP do tipo POST, visto ser o método HTTP mais indicado para transporte de dados.

6.6 Camada de lógica aplicacional

Após sua recepção e análise, os pedidos oriundos dos clientes são encaminhados para a aplicação *logica.app*. Esta aplicação implementa a lógica de negócio do sistema, isto é, implementa os algoritmos necessários que dão resposta aos requisitos funcionais. Para tal, pode recorrer à camada de dados para obter as informações necessárias, além das provenientes dos clientes através da camada de interface.

Tal como é possível observar na Figura 6.1, a aplicação *logica.app* é composta por três componentes: autenticação, procura e informações pessoais. Estes componentes têm uma correspondência directa com grupos em que foi dividida a API do sistema LC que foi abordada na secção anterior. Os pedidos chegados à aplicação Erlang *interface.app*, após analisados, são encaminhados para os respectivos componentes da camada de lógica de negócio.

6.6.1 Componente de autenticação

O componente de autenticação, tal como o próprio nome indica, é o responsável pelo processamento das funcionalidades relacionadas com a autenticação dos utilizadores no sistema e pelo registo de novos utilizadores.

Embora haja a consciência que a segurança e a privacidade dos utilizadores são factores de extrema relevância quando se trabalha com informações pessoais, estes não eram os principais objectivos desta dissertação. Visto isto, foi implementado um sistema elementar de autenticação baseado na correspondência entre o nome do utilizador e a sua palavra-chave. Em caso de sucesso é criada uma chave de sessão temporária e enviada ao cliente. À excepção da criação de novos utilizadores, que não necessita de qualquer autenticação, a utilização desta chave é obrigatória para aceder a todas as funcionalidades do serviço. É através desta chave que o utilizador é identificado sempre que invoca uma funcionalidade no servidor.

O processo de registo de novos utilizadores no serviço LC é simples, bastando ao utilizador inserir um total de três dados relativos à sua autenticação no sistema (nome e *password*) e no serviço Facebook (identificador do utilizador). Externamente, o utilizador deve também dar autorização na plataforma Facebook a que o sistema LocalChat tenha acesso a ao seu perfil, incluindo às suas fotografias e *feed*.

6.6.2 Componente de procura

A função deste componente consiste em processar todos os pedidos provenientes do grupo de procura da API, isto é, todas as actualizações do contexto

do utilizador e todas as pesquisas efectuadas.

O contexto do utilizador que pode ser actualizado consiste na sua localização geográfica e no seu estado. Visto não passarem de actualizações, a lógica aplicacional envolvida nestes processos é bastante reduzida, sendo apenas necessária a substituição das informações anteriores pelas actuais.

O mesmo não se passa em relação à lógica aplicacional que em que assenta o processo de procura de utilizadores próximos. Neste caso são percorridos todos os utilizadores activos cujo estado corresponde aos estados são aceites pelo utilizador e filtrados aqueles que se encontram dentro do raio definido.

Atendendo a que são usadas coordenadas GPS, foi escolhido a fórmula de Haversine. Esta equação calcula a distância entre dois pontos de uma esfera através da sua longitude e latitude. Considerando todos os ângulos em radianos e as coordenadas dos dois pontos diferenciadas pelo seu índice, matematicamente, a fórmula traduz-se da seguinte forma (6.1):

$$R = raio\,da\,terra(raio\,médio = 6,371km) \tag{6.1a}$$
$$\triangle latitude = latitude_2 - latitude_1 \tag{6.1b}$$
$$\triangle longitude = longitude_2 - longitude_1 \tag{6.1c}$$
$$a = sin^2(\triangle latitude/2) + cos(latitude_1).cos(latitude_2).sin^2(\triangle longitude/2) \tag{6.1d}$$
$$c = 2.atan2(\sqrt{a}, \sqrt{(1-a)}) \tag{6.1e}$$
$$d = R.c \tag{6.1f}$$

A sua implementação em Erlang produz o seguinte código:

```
%%
 % @edoc Calcula a distância entre dois pontos geográficos,
 % utilizando a fórmula de Haversine.
 %

distancia(Latitude1, Longigude1, Latitude2, Longitude2) ->
   R = 6371,      % metros
   % 1 grau = 0.0174532925 radianos
   Delta_latitude = (Latitude2 - Latitude1) * 0.0174532925,
   Delta_longitude = (Longitude2 - Longitude1) * 0.0174532925,
   A = math:pow(math:sin(Delta_latitude/2), 2)
        + math:cos(Latitude1 * 0.0174532925)
        * math:cos(Latitude2 * 0.0174532925)
        * math:pow(math:sin(Delta_longitude / 2), 2),
```

```
C = 2 * math:atan2(math:sqrt(A), math:sqrt(1 - A)),
Distancia = R * C,
Distancia.
```

Sendo a variável *Distancia* o valor da distância entre os dois pontos cujas coordenadas geográficas são representadas pelas variáveis Latitude1 e Longitude1, no caso do ponto 1, e Latidude2 e Longitude2, no caso do ponto 2.

6.6.3 Gestor de informações pessoais

O último componente implementa a lógica funcional dos requisitos que envolvem as informações pessoais dos utilizadores. Como foi descrito na secção 5.3.1, o sistema LC permite que um utilizador tenha acesso a dados pessoais de um outro, tais como nome, idade, sexo, fotografias e as ultimas actualizações da sua *wall* do Facebook.

Em termos práticos, este componente é responsável por devolver o identificador do utilizador em questão na plataforma Facebook, para que a camada de dados possa aceder às suas informações pessoais. No caso em que o cliente deseja aceder às fotografias de um utilizador, este componente opera também uma análise dos dados, para que as fotografias de todos os álbuns sejam agrupadas numa única lista.

Neste caso, a distribuição das fotografias por álbuns não acresce informação útil ao utilizador e simplifica o processamento das fotografias que teria de ser efectuado no cliente.

6.7 Camada de dados

Por último, mas não menos importante, existe a camada de dados. Esta camada é implementada na aplicação Erlang *dados.app* e é encarregue de receber, persistir e fornecer dados à camada de lógica.

Existindo um grande volume de dados envolvidos no sistema LC foi necessário dividir a aplicação em vários componentes, tendo como referência o propósito dos dados e a sua fonte. Visto isto, e tal como pode ser observado na Figura 6.1, a aplicação *dados.app* é composta por quatro componentes: *Servidor de tokens*, *gestor de dados*, *XMPP* e *Facebook*.

6.7.1 Servidor de tokens

O componente *servidor de tokens*, tal como mostra a Figura 6.1, é acedido pelo componente responsável pelos métodos de autenticação da camada de

lógica aplicacional.

A sua função passa por criar e gerir os identificadores de sessão dos utilizadores activos. Além disso, recebendo um identificador válido retorna o nome do utilizador ao qual o identificador pertence. É através desta funcionalidade que o servidor traduz os identificadores de sessão recebidos, como dados da generalidade dos métodos da API, e os traduz nos utilizadores reais que invocaram esses métodos. Esta é, portanto, uma funcionalidade crítica e bastante utilizada.

A correspondência entre estes identificadores e os utilizadores activos, bem como a sua validade são persistidos utilizando o sistema de base de dados Mnésia. Esta base de dados tem a capacidade de persistir qualquer estrutura Erlang e permite que o desenvolvedor defina o suporte onde os dados devem ser persistidos, isto é, RAM ou disco rígido. Estas foram duas funcionalidades, bem como a possibilidade de distribuir o sistema por várias máquinas, foram preponderantes na escolha do sistema Mnésia.

Neste caso em concreto, os dados foram definidos numa estrutura Erlang e apenas persistidos em RAM. A opção por um suporte volátil prende-se pela necessidade de acessos de leitura bastante rápidos, visto que para a generalidade dos pedidos feitos pelos utilizadores, é necessária autenticação.

A estrutura Erlang que agrega os dados necessários para a autenticação dos utilizadores é definida da seguinte forma:

```
-record(access_token, {
  token,
  username,
  expiration_time
  }).
```

Sendo *access_token* o nome da tabela, *token* o identificador de sessão do utilizador e chave da tabela, *username* o nome do utilizador e *expiration_time* a data em que o identificador expira.

6.7.2 Gestor de dados

Tal como o componente anterior, este utiliza como única fonte de dados o sistema de base de dados Mnésia. Tem como tarefa a gestão e persistência dos dados relacionados com os utilizadores. Entre estes dados é possível distinguir aqueles que são respeitantes aos registos dos utilizadores no sistema LC, daqueles que fazem parte da actividade do utilizador. Utilizando o mesmo raciocínio que no componente *servidor de tokens*, cada um dos tipos de dados foi definido numa estrutura Erlang.

A estrutura respeitante aos dados dos registos dos utilizadores no sistema está abaixo definida e alberga o seu nome e a sua palavra-chave. Além disso, é composta também pelo identificador, ou *username*, do utilizador no Facebook e ainda a sua identificação e palavra-chave no servidor XMPP (dados gerados pelo componenten *XMPP*). Na estrutura é traduzida numa tabela denominada *person*, tendo *username* a chave e aparecendo os restantes dados pela mesma ordem como foram supramencionados.

```
-record(person, {
  username,
  password,
  fb_user,
  ejabberd_username,
  ejabberd_password
  }).
```

Sendo estes dados relativos ao registo dos utilizadores no sistema LC, a sua persistência em disco é fundamental para evitar perdas de dados em falhas de hardware[3].

No que diz respeito aos dados relacionados com a actividade actual do utilizador, isto é, os dados da sua sessão, esta questão não se coloca, visto que apenas a replicação ou distribuição do sistema podem solucionar uma falha de hardware. Como tal, deu-se maior relevância à performance no acesso a estes dados e optou-se pela sua persistência em RAM. A estrutura que define estes dados está abaixo descrita e é composta da seguinte forma:

```
-record(active_user, {
  username,
  status,
  latitude,
  longitude,
  altitude,
  fb_token
  }).
```

A tabela correspondente a esta estrutura tem é designada por *active_user*, tem o campo *username*, que corresponde ao nome do utilizador e que é a sua chave. Além do nome do utilizador é também persistido o seu estado (*status*), as suas coordenadas GPS (*latitude*, *longitude* e *altitude*) e ainda a chave de sessão do utilizador no Facebook (*fb_token*), fornecida pelo componente Facebook, que ainda irá ser abordado.

[3]exceptuando claro os casos em que a falha for o próprio disco. Nestes casos apenas a replicação prévia dos discos ou a distribuição da base de dados servem de solução

6.7.3 XMPP

O servidor XMPP escolhido para integrar o sistema LC foi o ejabberd. Como foi já descrito, este servidor está desenvolvido em Erlang e é utilizado em alguns casos de sucesso, tais como o serviço de conversação do Facebook.

A interacção entre o servidor LC e o servidor ejabberd é suportada pelo componente *XMPP*. Este componente permite a criação e eliminação das contas dos utilizadores no serviço de conversação e fornece ainda a localização do servidor, para que os clientes se possam conectar.

As contas dos utilizadores são criadas no acto do seu registo no sistema LC e apagadas quando estes eliminam as suas contas.

6.7.4 Facebook

O último componente tem a seu cargo a obtenção de dados da rede social Facebook. Para tal, possui os dados para ligação à sua plataforma, tais como o identificador da aplicação, a palavra-chave e a chave da plataforma.

A interacção propriamente dita é realizada, com consciência das suas virtudes e limitações, com recurso à biblioteca cliente erlFBGraph, que já foi abordada no capítulo 4.1.

Os métodos disponibilizados por este componente centram-se no acesso a informações pessoais dos utilizadores, tais como os dados do seu perfil, da sua *wall*, as suas fotografias e ainda a sua imagem de apresentação do perfil. Disponibiliza, portanto, quatro métodos, sendo um por cada um dos tipos de dados descritos.

A biblioteca cliente erlFBGraph permite ainda a autenticação da aplicação na plataforma Facebook. No entanto, as permissões dadas às aplicações pelo Facebook não chegam para servir os propósitos do sistema LC, tal como descrito na secção 4.1. Nesta mesma secção é referido que o erlFBGraph permite a autenticação dos utilizadores, simulando a interacção HTTP do protocolo OAuth. No entanto foi também mencionado que é um processo instável. Visto isto, optou-se pela autenticação dos utilizadores, via OAuth, no cliente e passagem da chave de sessão obtida para o servidor.

Sendo que o processo de autenticação da plataforma Facebook é realizado no cliente, levanta-se a questão de o porquê de não ser esta mesma entidade a processar a restante interacção com a rede social. Esta opção prende-se com o agregar de toda a lógica do serviço no próprio servidor, possibilitando assim a criação de diversos clientes e a delegação do desenvolvimento destas aplicações para os próprios utilizadores.

6.8 Resumo

O presente capítulo foi dedicado ao servidor LC. Foram apresentadas as tecnologias utilizadas bem como a sua arquitectura interna. Relativamente ao primeiro ponto, convém destacar que apenas foram utilizadas tecnologias desenvolvidas em Erlang, desde o servidor Web YAWS ao servidor XMPP *ejabberd* e base de dados Mnesia, com o fim de aumentar a robustez do sistema.

A arquitectura do servidor segue uma abordagem multi-camada, onde cada camada está concretizada numa aplicação Erlang distinta, e em que todas estas estão incluídas numa aplicação principal. Esta opção foi justificada pela divisão dos componentes tendo em consideração a sua função e para que existisse um processo de desenvolvimento mais autónomo e célere em cada camada.

Para garantir a disponibilidade do sistema, foi implementada uma árvore de supervisão em cada uma das aplicações, utilizando os padrões OTP servidor e supervisor. Em caso de falha, a política adoptada foi de apenas reiniciar o componente em causa, visto que não existe uma dependência directa entre eles.

O servidor LC disponibiliza uma API para interacção com as aplicações cliente. Esta API está implementada segundo um serviço Web do tipo *REST-Full*. A autenticação dos utilizadores é feita pela correspondência entre o seu nome e a sua palavra-chave, ao qual é devolvida uma chave de sessão temporária e que deve ser utilizada nos restantes métodos da API.

Relativamente à lógica aplicacional, é de destacar a utilização da fórmula de Haversine no cálculo da distância entre dois utilizadores através das suas coordenadas geográficas.

Por último, na camada de dados estão contidos todos os componentes cuja missão é interagir com fontes de dados, como é o caso da base de dados Mnesia e a rede social Facebook, e com componentes externos como é o caso do servidor *ejabberd*.

Capítulo 7

Aplicação Cliente

No capítulo anterior foi apresentado o servidor LC, descrevendo as tecnologias utilizadas, a sua arquitectura e as decisões tomadas mais relevantes. O mesmo processo é aplicado ao cliente LC ao longo deste capítulo.

A aplicação criada pretende atingir dois objectivos. Primeiro fornecer ao utilizador uma forma mais rica para interacção com o sistema. Segundo, actuar como sensor do contexto do utilizador e fornece-lo ao servidor LC, para que este consiga implementar os requisitos descritos na secção 5.3.1.

7.1 Funcionalidades suportadas

O cliente LocalChat foi desenvolvido com o objectivo de implementar todas as necessidades expostas no cenário apresentado em 5.1. Nessa secção, o David, personagem fictícia, sentia a necessidade de um novo tipo de experiencia social mais personalizado e contextualizado. Mais específicamente, pretendia um serviço que lhe permitisse interagir com outras pessoas, mas que tivessem em comum o facto de estarem próximas geograficamente.

O cliente LocalChat, com recurso ao servidor apresentado na secção 6, permite que os seus utilizadores encontrem outros que estejam dentro de um dado raio e cujo estado obedeça a um dado filtro. Por outras palavras, através do cliente LocalChat, um utilizador pode pesquisar por outras pessoas próximas, não necessariamente já conhecidas, saber mais sobre elas e interagir através de um serviço de conversação. Estas são as três grandes vertentes do serviço LocalChat e que são disponibilizadas aos utilizadores através do cliente Android aqui apresentado.

Outros clientes LocalChat podem ser desenvolvidos de forma a implementar e mesmo evoluir estes serviços. A arquitectura conceptual proposta e a implementação do servidor LC assim o permitem.

7.2 Tecnologias

A solução arquitectural proposta em 5.3 permite a criação de diversas aplicações clientes, disponíveis para as mais variadas plataformas. É, no entanto, necessário garantir que essas plataformas possuem os mecanismos necessários para a captura do contexto do utilizador, mais precisamente as suas coordenadas geográficas.

Para este efeito, foi já referido na secção 2.2.1 que a melhor opção é a utilização de dispositivos móveis, mais precisamente telemóveis, dado que estão sempre próximos dos utilizadores. Estes dispositivos têm sofrido uma evolução assinalável e, actualmente, não é difícil encontrar exemplos com sensores GPS integrados e com conectividade à internet. A evolução dos sistemas operativos tem acompanhado esta inovação ao nível do *hardware*, sendo já possível o desenvolvimento de aplicações ricas ao nível da usabilidade e funcionalidades. Um dos sistemas operativos mais populares é o Android, abordado na secção 4.2.

Visto preencher todos os requisitos do sistema LC, optou-se por desenvolver a aplicação cliente na plataforma Android, tirando partido das suas características e dos dispositivos para a qual foi concebida.

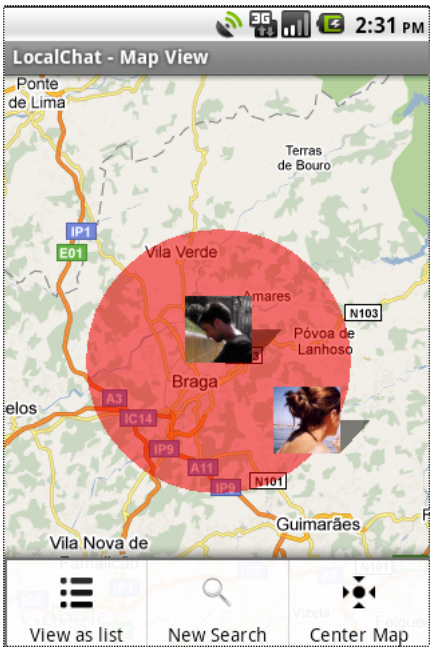

Figura 7.1: Interface que mostra os resultados de uma pesquisa dispostos num mapa.

Adicionalmente, foram incluídas três bibliotecas de terceiros: Google Maps[1], Asmack[2] e Facebook-Android-SDK[3]. A primeira é inserida com o objectivo de enriquecer a interface gráfica da aplicação, permitindo ao utilizador ver os utilizadores próximos representados num mapa, tal como mostra a Figura 7.1. Por sua vez, a segunda biblioteca facilita a interacção com o servidor XMPP, utilizado para conversação entre utilizadores. O aspecto gráfico de uma janela de conversação pode ser observado na Figura 7.2. Por último, a Facebook-Android-SDK é a aplicação cliente oficial do Facebook para a plataforma Android e é utilizada para implementar o sistema de autenticação dos utilizadores na rede social, utilizando o protocolo OAuth. Na Figura 7.3 está representado o aspecto gráfico da interface que mostra a informação pessoal de um utilizador. Além desta vista, é também possível a visualização das fotografias existentes no seu perfil do Facebook e as suas últimas publicações.

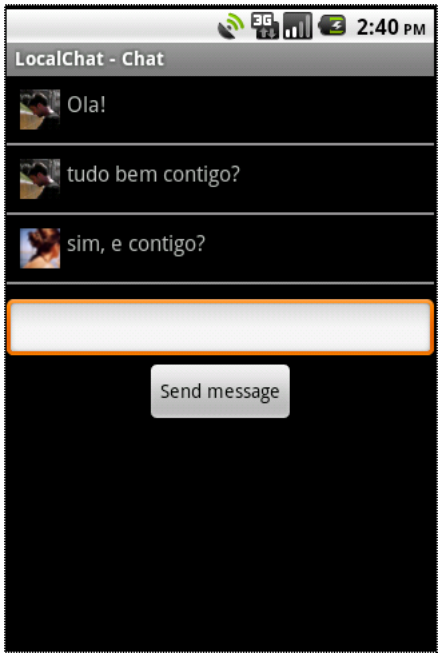

Figura 7.2: Interface que permite a conversação entre dois utilizadores.

[1] http://code.google.com/intl/pt-PT/android/add-ons/google-apis/
[2] http://code.google.com/p/asmack/
[3] http://github.com/facebook/facebook-android-sdk

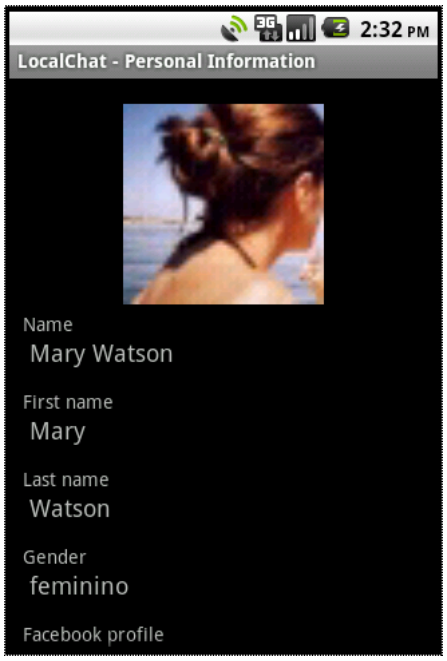

Figura 7.3: Interface que permite a visualização das informações pessoais de um utilizador.

7.3 Arquitectura

A aplicação cliente LC segue uma arquitectura multi-camada, tal como demonstra a Figura 7.4. A interacção entre o utilizador e a aplicação é assegurada por um consideravel número de interfaces gráficas. Visto isto, e para melhor evidenciar a forma como os componentes da aplicação se relacionam, optou-se por apresentar uma versão minimalista da arquitectura, em detrimento de uma versão completa que apenas aumentaria a entropia.

A arquitectura do cliente LC foi dividida em três camadas: interface, negócio e dados. A primeira agrega os componentes responsáveis pela construção das interfaces gráficas. A segunda implementa a lógica inerente à aplicação e por último, a camada de dados processa as rotinas de interacção com o servidor LC, servidor XMPP, autenticação via OAuth e *download* de imagens.

7.3.1 Camada de interface

Na camada superior, existem dois tipos de componentes: as vistas e as actividades, que em conjunto são responsáveis pelo aspecto e interactividade de

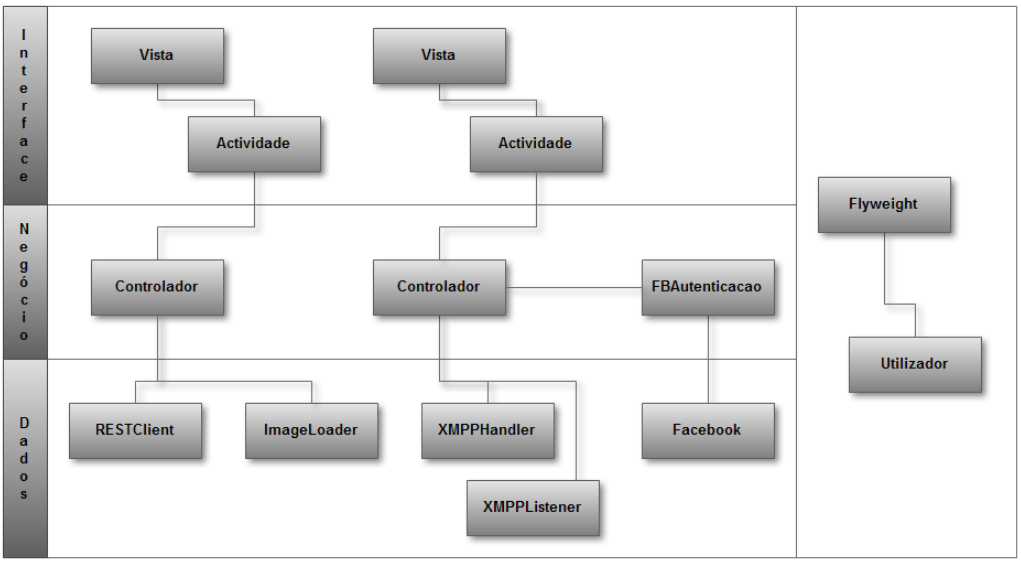

Figura 7.4: Arquictura do cliente LocalChat.

cada interface gráfica. As vistas definem o aspecto e os *widgets* das interfaces gráficas utilizando o formato XML, tal como foi descrito na secção 4.2.7. Por sua vez, o papel das actividades passa por processar as vistas e adicionar-lhes interactividade. Por outras palavras, é nas actividades que o comportamento de cada *widget* é definido, tal como a transição entre interfaces.

Como exemplo de uma acitivade apresentamos o seguinte código, implementa a actividade associada à vista de registo de novos utilizadores.

```
public class New_account extends Activity {

  /** Chamado quando a actividade é criada */
  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    // define a vista a implementar
    setContentView(R.layout.register);

    // associa uma variável Java ao widget
    final Button register =
                    (Button) findViewById(R.id.register);
    // adiciona um listener ao widget
    register.setOnClickListener(new View.OnClickListener() {
```

```
// define o comportamento a executar quando o
//  utilizador premir o botão
public void onClick(View v) {

  // acede aos campos editados pelo utilizador
  EditText username =
      (EditText) findViewById(R.id.username);
  EditText password =
      (EditText) findViewById(R.id.password);
  EditText fb_username =
      (EditText) findViewById(R.id.fb_username);
  try {
    if (!username.getText().toString().equals("")
      && !password.getText().toString().equals("")
      && !fb_username.getText().toString().equals("")
    ) {

      // cria uma instancia do controlador
      // associado a esta interface gráfica
      New_account_controler new_account_controler =
          new New_account_controler();
      // invoca o método que implementa a acção
      // espelotada pelo utilizador
      new_account_controler.register(
          username.getText().toString(),
          password.getText().toString(),
          fb_username.getText().toString());

      // redirecciona o utilizador para a
      // intergace gráfica anterior
      Intent intent = new Intent();
      setResult(RESULT_OK, intent);
      finish();
    } else {
      throw new
          Exception("All fields are required!");
    }
  } catch (Throwable e) {

  // mostra uma mensagem de erro
```

```
        new AlertDialog.Builder(New_account.this).
          setTitle("LocalChat - Warning").
          setMessage(e.getMessage()).
          setNeutralButton("Ok",
                new DialogInterface.OnClickListener() {
            public void onClick(
                            DialogInterface dialog,
                            int which) {
            }
          }).show();
        }
      }
    });
  }
}
```

A actividade começa por definir a vista XML que deve implementar. Isto é feito com recurso à classe *R* e ao código *setContentView(R.layout.register)*. Esta classe está presente em todos os projectos Android e que tem como missão fazer o mapeamento entre os ficheiros estáticos, como as vistas XML, em objectos Java que possam ser utilizados.

É também associada uma variável, *register*, ao *widget* do tipo botão e adicionado um *listener*, que espoletará uma acção quando o utilizador o premir. Esta acção é definida no próprio *listener* e, neste caso, passa por aceder aos campos editados pelo utilizador (*username*, *password* e *fb_username*) e invocar o método no controlador que processe o registo do novo utilizador. Este controlador é representado pelo objecto *new_account_controler*. Por último, o utilizador é redireccionado para a interface gráfica anterior. Caso ocorra uma excepção ao longo do processo, é mostrada uma mensagem de erro através da classe *AlertDialog*.

A totalidade das interfaces gráficas que compõem o sistema, bem como a interacção entre si, está representada através do diagrama de estados da Figura7.5, onde cada estado representa uma interface gráfica. O diagrama certifica ainda que, através desta aplicação é possível solucionar as necessidades descritas no cenário da secção 5.1, fornecendo aos utilizadores um serviço social sensível ao seu contexto.

Quando o utilizador inicia o cliente LocalChat no seu dispositivo Android é-lhe apresentada uma interface gráfica que lhe permite autenticar-se, caso já esteja registado, ou então efectuar o registo no serviço. Neste último caso é reencaminhado para a vista "Novo utilizador", onde deve inserir os dados necessários (nome, palavra-chave e o seu identificador na plataforma

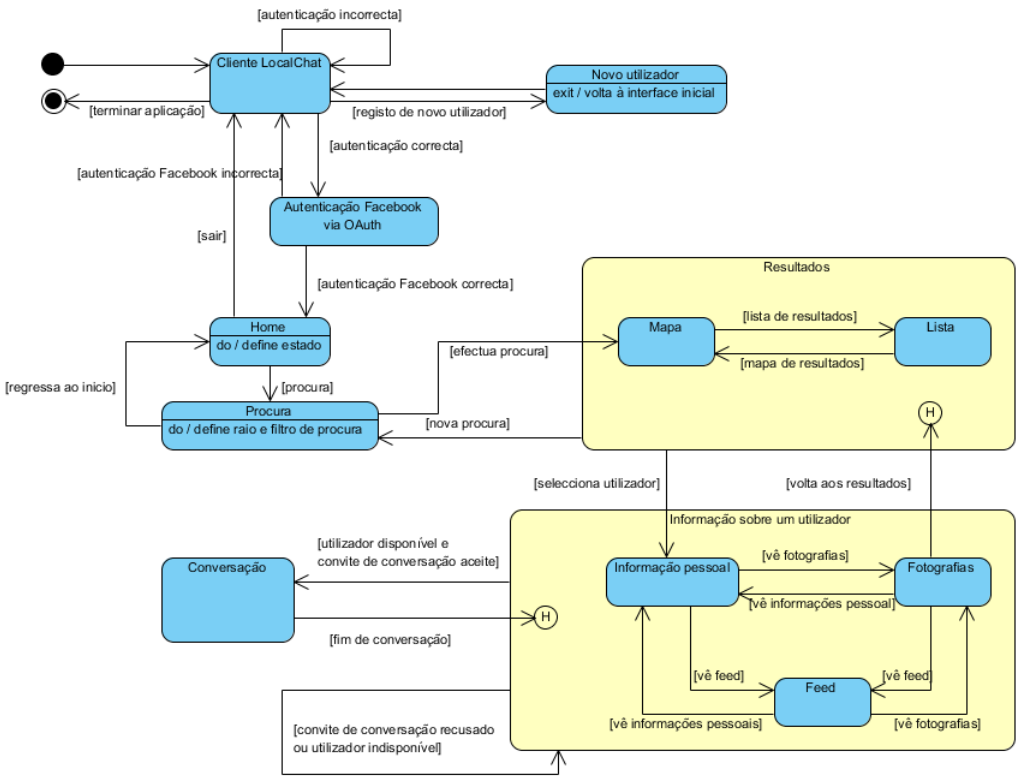

Figura 7.5: Diagrama de estados correspondente à interacção entre as várias vistas da aplicação cliente LocalChat.

Facebook) para que o registo seja concluído com sucesso.

Assumindo que se encontra registado, o utilizador pode efectuar o processo de autenticação. Caso este processo seja bem sucedido, é apresentada ao utilizador a vista "Autenticação Facebook via OAuth". Esta vista recorre a um navegador Web e, tal como o nome indica, permite a autenticação do utilizador na plataforma do Facebook através do protocolo OAuth. Em caso de sucesso é obtida uma chave de sessão válida para a obtenção dos dados pessoais utilizador e este é reencaminhado para a vista "Home". Caso contrário, volta à vista inicial.

Na vista "Home", o utilizador é convidado a definir o seu estado, estando disponível por defeito e iniciar um processo de procura por outros utilizadores que se encontrem na mesma região. Escolhendo este caso, é-lhe apresentada a vista "Procura" onde pode definir o raio, em quilómetros, a ser usado na procura e um filtro, por estado, a ser aplicado ao resultado deste processo. O utilizador pode ainda retroceder para a vista "Home".

A visualização dos resultados é o passo seguinte no processo de interacção entre o utilizador e o cliente LocalChat. Existem duas formas distintas de os ver: ou dispostos num mapa de acordo com a sua localização geográfica, ou através de uma lista. A primeira opção é concretizada pela vista "Mapa" e a segunda pela "Lista". Está contemplada a possibilidade de o utilizador poder alternar entre as duas. A partir deste ponto, o utilizador pode efectuar novas procuras, voltando à vista "Procura", ou escolher um elemento e aceder a mais informações sobre ele.

Após seleccionar um elemento da lista de resultados, o utilizador é direccionado para a vista "Informação Pessoal". A partir desta, ele tem a possibilidade de deambular entre esta vista, e as vistas "Fotografias" e "Feed", sendo todas elas alimentadas por dados obtidos da rede social Facebook. A primeira mostra a informação pessoal sobre o utilizador em questão, a segunda mostra fotografias presentes no seu perfil e por último, a vista "Feed" apresenta as ultimas actualizações que o utilizador inseriu na sua *wall*. Em qualquer uma destas três vistas, o utilizador pode decidir enviar um convite para conversação.

Enviado o convite existem duas possibilidades: ou o convite é aceite e o utilizador é encaminhado para a vista "Conversação", para que possa trocar mensagens, ou o convite é recusado pelo sistema (caso o utilizador esteja já indisponível) ou pelo destinatário do convite. Em ambos os casos onde o convite é recusado, o utilizador mantém-se na vista actual.

Iniciada a conversação, o utilizador pode por fim a esta e é redireccionado para a vista em que estava quando enviou o convite.

No diagrama de estados apresentado não estão contempladas os erros que podem ocorrer, pelas mais diversas razões, em tempo de execução. Nestes casos é apresentada uma janela *pop-up* ao utilizador a informar do sucedido, voltando posteriormente à vista onde se encontrava.

7.3.2 Camada de negócio

A execução das acções espoletadas pelo utilizador é da responsabilidade dos controladores existentes na camada de negócio. Na secção anterior foi exposto código relativo integrante da actividade de registo de novos utilizadores. O seguinte fragmento faz parte do controlador que implementa a lógica aplicacional que permite concluir este processo, e representa o método que é invocado na actividade apresentada.

```
public void register(String username,
 String password,
 String fb_username)
```

```
  throws Exception {
   try {
     RestClient rest = new RestClient();
     this.rest.new_account(username, password, fb_username);
   } catch (Exception e) {
     throw new Exception(e.getMessage());
   }
 }
```

Neste caso em particular, a lógica de negócio envolvida é bastante simples. Os parâmetros inseridos pelo utilizador não sofrem qualquer tipo de processamento e são de imediato passados enviados para o servidor LC, através do componente da camada de dados *RestClient*. No caso de algum erro ocorrer durante o processo, é lançada uma excepção para a camada superior.

Além dos controladores, esta camada é também composta por uma outra classe denominada *FBAutenticacao*. A função deste componente passa pela integração de um navegador Web na aplicação, viabilizando assim o processo de autenticação por OAuth na plataforma Facebook. Este processo é executado recorrendo à biblioteca cliente Facebook-Android-SDK.

No caso do serviço LC, este processo envolve quatro entidades: o utilizador, a aplicação cliente, o servidor e a plataforma de aplicações do Facebook. O diagrama de sequência apresentado na Figura 7.6 ajuda a perceber a interacção existente até que o servidor LC obtenha a chave necessária para que possa obter informações da rede social. Este diagrama assume que o utilizador está já registado no serviço LC e que o processo de autenticação é bem sucedido.

O processo de autenticação na plataforma Facebook, via OAuth, é espoletado quando o utilizador utiliza a aplicação cliente para fazer uma tentativa de autenticação no sistema LC (passos 1 e 1.1). Caso o processo seja bem sucedido, pela validação ocorrida no passo 2, é retornado pelo servidor um *token*, que o cliente deve usar como autenticação em futuros acções (passo 3).

Neste momento, está concluída a autenticação do utilizador no serviço LC, no entanto não é possível ao servidor obter informações pessoais a seu respeito. Como tal, é iniciado o processo de autenticação na plataforma Facebook.

É apresentada ao utilizador uma nova interface (passo 1.2), implementada com recurso a um navegador Web, que permite ao utilizador autenticar-se pelo protocolo OAuth (passo 4). Após submetidos os dados, a plataforma do Facebook procede à sua validação (passo 4.1) e, dependendo do resultado, ocorre uma de duas coisas.

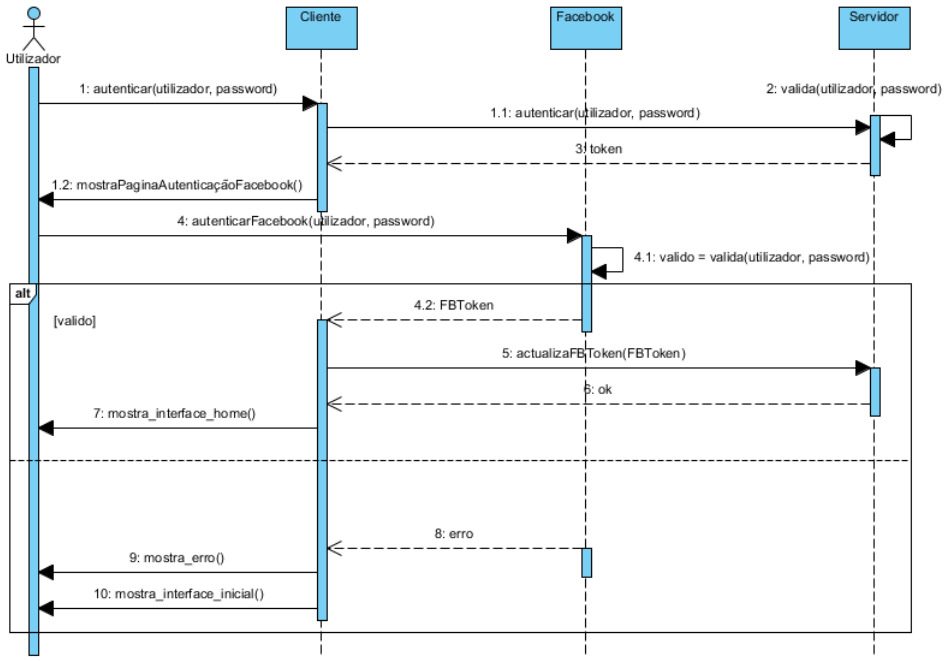

Figura 7.6: Diagrama de sequência que ilustra o processo de autenticação dos utilizadores na plataforma do Facebook, através do protocolo OAuth.

Caso a validação seja bem sucedida, isto é, caso as credenciais do utilizador estejam correctas, a plataforma devolve uma chave de sessão (representado pela mensagem *FBToken*) que é capturada pelo cliente e permite a obtenção de dados pessoais (passo 4.2). A obtenção destes dados é da responsabilidade do servidor LC, pelo que a chave de sessão é enviada no passo 5. Decorrendo este processo sem problemas, a aplicação cliente reencaminha o utilizador para a interface gráfica principal (passo 7).

Por outro lado, se a validação do passo 4.1 não terminar da forma esperada, a plataforma do Facebook devolve uma mensagem de erro (passo 8), que é mostrada ao utilizador (passo 9) pelo cliente. Visto que, desta forma, não é possível a obtenção dos dados pessoais, cruciais para o bom funcionamento do serviço LC, é barrada a entrada do utilizador no serviço.

7.3.3 Camada de acesso a dados

A interacção com serviços externos e obtenção de dados fazem parte das competências da camada de dados[4]. Para isso, a camada de dados é constituída por cinco entidades: *RESTClient*, *ImageLoader*, *Facebook*, e *XMPPHandler* e *XMPPListener*.

Pelo primeiro componente passa toda a comunicação entre o cliente e o servidor LC, possuindo mecanismos capazes de interagir com todos os métodos remotos disponibilizados. Continuando a recorrer ao exemplo do registo de novos utilizadores, tal como nas restantes camadas, é agora possível observar a forma como a interacção entre o cliente e o servidor LC é processada.

```
public void new_account(String username,
   String password,
   String fb_username) throws Exception {
  try {
    // Transformar parâmetros em objectos JSON
    JSONObject obj = new JSONObject();
    obj.accumulate("username", username);
    obj.accumulate("password", password);
    obj.accumulate("fb_username", fb_username);

    // Construir o pedido HTTP
    StringEntity data = new StringEntity(obj.toString());
    data.setContentEncoding(
      new BasicHeader(HTTP.CONTENT_TYPE, "application/json")
    );
    String service =
       this.server + ":" + this.port + "/auth/new_user/";
    HttpPost request = new HttpPost(service);
    request.setEntity(data);

    // Enviar o pedido
    HttpResponse response = client.execute(request);

    if (response != null) {
      // Processamento da resposta ao pedido enviado
    }
```

[4]exceptuando a captura do contexto do utilizador que é efectuada pelos controladores directamente na plataforma Android

```
  } catch (Exception e) {
      throw new Exception(e.getMessage());
  }
}
```

No método apresentado, é possível observar que antes de a interacção ser iniciada, é necessário que os dados a ser enviados sejam transformados em objectos JSON. Estes dados são posteriormente inseridos num pedido HTTP, em conjunto com o endereço do serviço a que se pretende aceder.

Por fim, este pedido é enviado e processada a resposta enviada pelo servidor. Neste processamento existe a necessidade de transformar os dados que vêm como objectos JSON em tipos de dados simples, para serem enviados para as camadas superiores.

Por sua vez, o *ImageLoader* fornece um método genérico para a obtenção do imagens do perfil dos utilizadores. Este processo é implementado pelo seguinte método Java.

```
public Bitmap downloadImage(String fileUrl) throws Exception {
  URL myFileUrl = null;
  try {
    // constroi o pedido HTTP
    myFileUrl = new URL(fileUrl);
    HttpURLConnection conn =
       (HttpURLConnection) myFileUrl.openConnection();
    conn.setDoInput(true);
    conn.setInstanceFollowRedirects(true);
    conn.connect();

    // descarrega a imagem
    InputStream is = conn.getInputStream();

    // retorna convertido para Bitmap
    return BitmapFactory.decodeStream(is);
  } catch (Exception e) {
    throw new Exception(e.getMessage());
  }
}
```

Começa por construir um pedido HTTP tendo como base o endreço da imagem recebido como parâmetro. Construído o pedido, é iniciada uma conexão e descarregada a imagem, que no fim é retornada como um Bitmap[5].

[5]formato de imagem suportado pela plataforma Android.

Convém realçar a definição do parâmetro HTTP *InstanceFollowRedirects*, visto que alguns endereços de imagens devolvidos pelo Facebook são apenas redirecções para os verdadeiros endereços.

O componente *Facebook* representa a biblioteca Facebook-Android-SDK, constituída por diversas classes e que facilita a interacção com a rede social. Embora permita mais do que isso, neste caso a biblioteca foi apenas utilizada no processo de autenticação dos utilizadores, descrito na Figura 7.6.

Ainda na camada de dados, o *XMPPHandler* e o *XMPPListener* têm a seu cargo a interacção com o servidor XMPP que implementa o serviço de conversação. A primeira classe processa o envio das mensagens. Por sua vez, a segunda espera por novas mensagens vindas do servidor, reencaminhando-as para a camada superior.

7.3.4 Clases comuns entre camadas

Por último, há ainda a destacar dois componentes transversais a todas as camadas: *Flyweight* e *Utilizador*. Estes dois componentes estão desenvolvidos recorrendo a classes estáticas, que implementam o padrão de desenvolvimento Flyweight[Gamma et al., 1995]. O seu objectivo é, por um lado, facilitar o processo de comunicação entre camadas e por outro, minimizar o consumo de memória que por vezes é um recurso escasso nestes dispositivos.

O primeiro componente reúne todas as informações necessárias à interacção cliente-servidor, tais como chaves de sessão, credenciais e localização do servidor XMPP, entre outros. Considerando normal que o utilizador pretenda aceder às informações pessoais de um outro várias vezes durante a mesma sessão, o *Flyweight* agrega também estas informações.

```
HashMap<String, Utilizador> utilizadores;
```

Este tipo de cache é implementado utilizando uma estrutura do tipo Map, tendo como chave o identificador de cada utilizador e como valor um objecto do tipo *Utilizador*, que contém as referidas informações pessoais. Estas informações incluem também as suas imagens de perfil, contrariamente às fotografias existentes no seu perfil, que são eliminadas por questões de espaço.

7.4 Resumo

O cliente LocalChat apresentado neste capítulo foi desenvolvido para implementar um serviço social sensível ao contexto do seus utilizadores e que solucionasse todos os requisitos expostos no cenário apresentado em 5.1.

A plataforma escolhida foi a Android, no entanto, uma hipótese igualmente válida seria a opção pela plataforma iPhone, visto que os dispositivos associados a ambas possuem as funcionalidades necessárias para o bom funcionamento do serviço LC.

A arquitectura do cliente segue uma abordagem multi-camada, tal como mostra a Figura 7.4, existindo a distinção entre os componentes responsáveis pela interacção com o cliente (camada de interface), os que têm como função implementar a lógica aplicacional (camada de negócio) e os que fazem a ponte com as fontes de dados (camada de dados). Na camada de interface é feita a distinção entre o aspecto e o comportamento das interfaces gráficas, existindo para tal dois tipos de ficheiros: as vistas, definidas em XML, e as actividades. A camada de negócio é composta pelos controladores, que processam os pedidos efectuados pelos utilizadores. Para tal recorrem aos componentes da camada de dados, onde existem objectos capazes de interagir com os servidores LocalChat e XMPP, de carregar imagens oriundas da Web e implementar o método de autenticação na plataforma do Facebook via OAuth.

Além das três camadas, existem ainda dois componentes (*Flyweight* e *Utilizador*), concretizados por duas classes estáticas, e cuja função passa por facilitar o processo de comunicação entre camadas e minimizar o consumo de memória. Estes dois componentes implementam o padrão de desenvolvimento Flyweight.

Capítulo 8

Conclusões

Actualmente, as redes sociais ocupam um lugar de destaque na vida das pessoas. Estes serviços são alvo de uma evolução muito acentuada, principalmente após o momento em que permitiram a integração de aplicações externas. As APIs remotas disponibilizadas pelos serviços (ou aplicações) de redes sociais, em conjunto com a evolução das tecnologias móveis, permitiram a criação de novas funcionalidades e formas de interacção. No entanto, a inclusão do contexto do utilizador apenas agora começa a ter relevo. Com um papel mais relevante por parte da informação contextual, é possível melhorar os serviços por forma a ficarem mais personalizados e úteis.

Ao longo deste trabalho, foi também possível observar, pela revisão de literatura e projectos existentes, que cada vez mais a criação de funcionalidades baseadas em contexto é uma tendência a seguir pelas redes sociais. Como exemplo mais recente foi apresentado o Facebook Places. Além da sua utilização social, é também evidente a opção por tornar esta funcionalidade rentável através de publicidade baseada em localização. Mais do que isso, antevê-se que num futuro próximo, outros tipos de serviços baseados em contexto tirem proveito das redes sociais. A título de exemplo podem destacar-se os serviços de notícias locais. Neste caso, as notícias podem ser marcadas com temas e informação geográfica e acedidas tendo como base essas marcações.

Tanto no Facebook Places como no serviço de notícias locais, a informação contextual mais utilizada é a localização geográfica. De facto, isto acontece na generalidade das aplicações sensíveis ao contexto. No entanto, com a evolução ao nível do *hardware* dos dispositivos móveis, é expectável que outros tipos de sensores sejam incorporados. A acontecer, esta possibilidade abre portas para que mais informação contextual seja utilizada, permitindo assim o aparecimento de novas funcionalidades e uma evolução das existentes.

8.1 Contributos

Esta dissertação propõe uma arquitectura conceptual que permite a criação de aplicações sociais sensíveis ao contexto dos seus utilizadores e com componentes críticos desenvolvidos em Erlang. Além da arquitectura, e como prova de conceito, apresenta-se o sistema LC. Este sistema é composto por dois componentes: o cliente e o servidor, sendo o primeiro implementado como uma aplicação móvel para a plataforma Android e responsável pela interacção com o utilizador. Este componente possui também mecanismos capazes de obter a localização geográfica do utilizador (principal informação contextual utilizada neste serviço). O segundo componente centraliza toda a informação vinda dos clientes e implementa a lógica de negócio do sistema. Foi desenvolvido utilizando a linguagem de programação Erlang, tirando proveito das suas características que incentivam a criação de aplicações altamente escaláveis e disponíveis. A sua interface consiste num serviço Web do tipo RESTfull e com formatação JSON. Desta forma, embora esta dissertação tenha apenas produzido uma prova de conceito, é possível a criação de outras e para diferentes plataformas.

Para obtenção das informações pessoais do utilizador, úteis para solucionar os requisitos do sistema LC, foi utilizada a rede social Facebook. Tida em conta como o serviço social com mais utilizadores no mundo inteiro, disponibiliza também uma plataforma que se revelou apropriada para o desenvolvimento de aplicações externas.

Existindo a necessidade de o servidor LC comunicar com a plataforma Facebook, esta dissertação produziu também uma biblioteca cliente. Este projecto foi denominado de erlFBGraph e está desenvolvido em Erlang, sendo suficientemente genérico para obter qualquer dado disponibilizado através da GraphAPI. Embora existam outros projectos semelhantes, a opção pelo desenvolvimento de um novo projecto prendeu-se com o facto de, até então, nenhum dos existentes estar actualizado para a nova versão da plataforma Facebook.

Tendo em consideração as tendências seguidas pelos serviços de redes sociais, o caso de estudo apresentado revelou-se extremamente apropriado. Prova disso é o caso do Facebook Places. Os primeiros rumores sobre a integração da localização na rede social sempre estiveram associados ao negócio da publicidade local. No entanto, só muito recentemente, e apenas numa fase avançada da dissertação em que os requisitos tinham já sido definidos, foram conhecidas as verdadeiras funcionalidades deste serviço. Comparando os princípios fundamentais de cada serviço, como a utilização da localização, da rede social e interacção entre utilizadores, pode ser concluído que os dois serviços não são assim tão diferentes. Este facto certifica que a criação de

aplicações sociais com recurso ao contexto do utilizador é alvo de um forte investimento de serviços populares como o Facebook. Mais do que isto, atesta também que o âmbito do caso de estudo escolhido é actual e está em linha com a evolução escolhida pelas redes sociais para as suas funcionalidades.

8.2 Trabalho Futuro

O sistema apresentado não completa todo o trabalho que poderia ainda ser realizado na integração de informação contextual nas redes sociais. O LC poderia ser enriquecido com a utilização de mais e diferente informação contextual, personalizando ainda mais o serviço, ou então explorando ainda mais a localização geográfica, através da criação de listas de utilizadores dinâmicas. Os elementos destas listas, além de terem em comum a sua proximidade geográfica, poderiam também ser filtrados de acordo com as informações pessoais obtidas.

Da mesma forma, poderia ser evoluída a utilização das redes sociais, quer integrando o sistema LC com outros serviços, quer utilizando estes serviços não apenas como fontes de dados. A título de exemplo, poderia ser dada a possibilidade de publicar no perfil dos utilizadores as pessoas com quem interagiram através do sistema LC. Outro exemplo interessante seria a integração de informação geográfica com sistemas de gestão de informações pessoais [Jones, 2007]. Neste caso, poder-se-ia evoluir serviços de agendas pessoais e de tarefas, com integração directa com a agenda social do utilizador no Facebook.

Além da evolução ao nível das funcionalidades, seria também importante prover este sistema de melhores mecanismos de segurança e privacidade dos utilizadores.

Por fim, seria também interessante a publicação do serviço de forma a poder ser testado e utilizado em ambientes reais. Desta forma poder-se-ia testar verdadeiramente a escalabilidade e utilidade do serviço.

Bibliografia

[Cen, 2007] (2007). Cenceme – injecting sensing presence into social networking applications. pages 1–28.

[AB, 2010] AB, E. (2010). Otp design principles user's guide.

[Armstrong, 2003] Armstrong, J. (2003). Concurrency oriented programming in Erlang. *Invited talk, FFG*.

[Armstrong, 2007] Armstrong, J. (2007). A history of erlang. In *HOPL III: Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pages 6–1–6–26, New York, NY, USA. ACM.

[Armstrong et al., 1997] Armstrong, J., Arts, T., and Ab, E. T. (1997). Erlang and its applications.

[Barkhuus and Dey, 2003] Barkhuus, L. and Dey, A. (2003). Is context-aware computing taking control away from the user? three levels of interactivity examined.

[Beach et al., 2008] Beach, A., Gartrell, M., Akkala, S., Elston, J., Kelley, J., Nishimoto, K., Ray, B., Razgulin, S., Sundaresan, K., Surendar, B., Terada, M., and Han, R. (2008). Whozthat? evolving an ecosystem for context-aware mobile social networks. *IEEE Network*, 22(4):50–55.

[Beach et al., 2010] Beach, A., Gartrell, M., Xing, X., Han, R., Lv, Q., Mishra, S., and Seada, K. (2010). Fusing mobile, sensor, and social data to fully enable context-aware computing. In *HotMobile '10: Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pages 60–65, New York, NY, USA. ACM.

[Boyd and Ellison, 2007] Boyd, D. and Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1).

[Brown et al., 2005] Brown, B., Chalmers, M., Bell, M., Hall, M., Maccoll, I., and Rudman, P. (2005). Sharing the square: collaborative leisure in the city streets. In *ECSCW'05: Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*, pages 427–447, New York, NY, USA. Springer-Verlag New York, Inc.

[Brown et al., 2007] Brown, B. A. T., Taylor, A. S., Izadi, S., Sellen, A., Kaye, J., and Eardley, R. (2007). Locating family values: A field trial of the whereabouts clock. In *Ubicomp*, pages 354–371.

[Bruner and Kumar, 2007] Bruner, G. I. and Kumar, A. (2007). Attitude toward location-based advertising. *Journal of Interactive Advertising, Spring*, 7(2).

[Bryson and Vinoski, 2009] Bryson, D. and Vinoski, S. (2009). Build Your Next Web Application with Erlang. *IEEE Internet Computing*, 13(4):93–96.

[Carlsson and Rémond, 2006] Carlsson, R. and Rémond, M. (2006). Eunit: a lightweight unit testing framework for erlang. In *ERLANG '06: Proceedings of the 2006 ACM SIGPLAN workshop on Erlang*, pages 1–1, New York, NY, USA. ACM.

[Cesarini and Thompson, 2009] Cesarini, F. and Thompson, S. (2009). *Erlang programming*. O'Reilly Media.

[Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA.

[Cole et al., 2010] Cole, C., Russell, C., and Whyte, J. (2010). *Building OpenSocial apps : a field guide to working with the MySpace platform*. Addison-Wesley.

[Dey, 2001] Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7.

[Dierks and Rescorla, 2008] Dierks, T. and Rescorla, E. (2008). Rfc 5246 - the transport layer security (tls) protocol version 1.2. Technical report.

[Eagle and Pentland, 2005] Eagle, N. and Pentland, A. (2005). Social serendipity: mobilizing social software. *Pervasive Computing, IEEE*, 4(2):28–34.

[Ellison et al., 2007] Ellison, N. B., Steinfield, C., and Lampe, C. (2007). The benefits of facebook " friends:" social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication*, 12(4):1143–1168.

[Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R. E., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA.

[Gregory D. Abowd, 1999] Gregory D. Abowd, A. K. D. P. J. B. N. D. M. S. P. S. (1999). Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK. Springer-Verlag.

[Hopper et al., 1993] Hopper, A., Harter, A., and Blackie, T. (1993). The active badge system (abstract). In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 533–534, New York, NY, USA. ACM.

[Jang et al.,] Jang, S., Ko, E. J., and Woo, W. Unified user-centric context: Who, where, when, what, how and why. *Personalized Context Modeling and Management for UbiComp Applications*, 149:26–34+.

[Joly et al., 2009] Joly, A., Maret, P., and Daigremont, J. (2009). Context-awareness, the missing block of social networking. *International Journal of Computer Science and Applications*, 4(2).

[Jones, 2007] Jones, W. (2007). Personal information management. *Annual Review of Information Science and Technology*, 41(1):453–504.

[Krumm et al., 2008] Krumm, J., Davies, N., and Narayanaswami, C. (2008). User-generated content. *IEEE Pervasive Computing*, 7:10–11.

[Lamming and Flynn, 1994] Lamming, M. and Flynn, M. (1994). Forget-me-not: intimate computing in support of human memory. In *Proceedings FRIEND21 Symposium on Next Generation Human Interfaces*.

[Lampe et al., 2006] Lampe, C., Ellison, N., and Steinfield, C. (2006). A face(book) in the crowd: social searching vs. social browsing. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 167–170, New York, NY, USA. ACM.

[Letuchy, 2009] Letuchy, E. (2009). Erlang at Facebook.

[LOGAN et al., 2010] LOGAN, M., MERRITT, E., and CARLSSON, R. (2010). Erlang and otp in action.

[Marmasse et al., 2004] Marmasse, N., Schm, C., and Spectre, D. (2004). Watchme: Communication and awareness between members of a closely-knit group. pages 214–231.

[Marmasse and Schmandt, 2000] Marmasse, N. and Schmandt, C. (2000). Location-aware information delivery with commotion. pages 361–370.

[Mattsson et al., 1999] Mattsson, H., Nilsson, H., Wikström, C., and Ab, E. T. (1999). Mnesia - a distributed robust dbms for telecommunications applications. In *In Practical Applications of Declarative Languages: Proceedings of the PADL'1999 Symposium, G. Gupta, Ed. Number 1551 in LNCS*, pages 152–163. Springer.

[Mitchell-Wong et al., 2007] Mitchell-Wong, J., Kowalczyk, R., Roshelova, A., Joy, B., and Tsai, H. (2007). Opensocial: From social networks to social ecosystem. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 361–366.

[Murphy, 2010] Murphy, M. L. (2010). *Beginning Android 2*. Apress.

[Myers, 1997] Myers, J. (1997). Simple authentication and security layer (SASL). RFC 2222, Internet Engineering Task Force.

[Nagy and Nagyné Víg, 2008] Nagy, T. and Nagyné Víg, A. (2008). Erlang testing and tools survey. In *ERLANG '08: Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*, pages 21–28, New York, NY, USA. ACM.

[Nakanishi et al., 2002] Nakanishi, Y., Takahashi, K., Tsuji, T., and Hakozaki, K. (2002). icams: a mobile communication tool using location information and schedule information. In *Information", International Conference on Pervasive Computinge (Pervasive2002*, pages 26–28.

[Nick Gerakines and Goffinet, 2008] Nick Gerakines, M. Z. and Goffinet, C. (2008). Developing Erlang At Yahoo.

[Oliveira, 2009] Oliveira, L. M. V. C. (2009). Desenvolvimento de aplicações móveis sensíveis ao contexto. Master's thesis.

[Pessemier et al., 2009] Pessemier, T. D., Deryckere, T., and Martens, L. (2009). Context aware recommendations for user-generated content on

a social network site. In Geerts, D., Cesar, P., and Grooff, D. D., editors, *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pages 133–136, New York, NY, USA. ACM.

[Phones et al., 2005] Phones, O. M., Sohn, T., Li, K. A., Lee, G., Smith, I., Scott, J., and Griswold, W. G. (2005). Place-its: A study of location-based reminders. In *In Ubicomp*, pages 232–250. Springer.

[Saint-Andre, 2005a] Saint-Andre, P. (2005a). Streaming xml with jabber/xmpp. *IEEE Internet Computing*, 9:82–89.

[Saint-Andre, 2005b] Saint-Andre, P. (2005b). Streaming XML with Jabber/XMPP. *IEEE internet computing*, 9(5):82–89.

[Saint-Andre et al., 2009] Saint-Andre, P., Smith, K., and Tronçon, R. (2009). *XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies*. O'Reilly Media, Inc.

[Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US.

[Schlossnagle, 2006] Schlossnagle, T. (2006). *Scalable internet architectures*. Sams, Indianapolis, IN, USA.

[Smith et al., 2005] Smith, I., Consolvo, S., Lamarca, A., Hightower, J., Scott, J., Sohn, T., Hughes, J., Iachello, G., and Abowd, G. D. (2005). Social disclosure of place: From location technology to communication practices. In Gellersen, H. W., Want, R., and Schmidt, A., editors, *Pervasive Computing*, volume 3468 of *Lecture Notes in Computer Science*, pages 134–151. Springer Berlin / Heidelberg.

[Srisuresh and Egevang, 2001] Srisuresh, P. and Egevang, K. (2001). Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational).

[Sumi et al., 1998] Sumi, Y., Etani, T., Fels, S., Simonet, N., Kobayashi, K., and Mase, K. (1998). C-map: Building a context-aware mobile assistant for exhibition tours. In *Community Computing and Support Systems, Social Interaction in Networked Communities [the book is based on the Kyoto Meeting on Social Interaction and Communityware, held in Kyoto, Japan, in June 1998]*, pages 137–154, London, UK. Springer-Verlag.

[Torstendahl, 1997] Torstendahl, S. (1997). Open telecom platform. *Ericsson Review(English Edition)*, 74(1):14–23.

[Whitehead and Wiggins, 1998] Whitehead, E. J. and Wiggins, M. (1998). Webdav: Ieft standard for collaborative authoring on the web. *Internet Computing, IEEE*, 2(5):34 –40.