INSTITUTO POLITÉCNICO DE VISEU

Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Mestrado em Sistemas e Tecnologias de Informação para as Organizações

Agregação de Redes Sociais

Vítor Manuel Seixas da Fonseca

2011

INSTITUTO POLITÉCNICO DE VISEU

Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Mestrado em Sistemas e Tecnologias de Informação para as Organizações

Agregação de Redes Sociais

Vítor Manuel Seixas da Fonseca

Orientador:

Eng.º Paulo da Silva Tomé

2011

Agradecimentos

Vejo na conclusão deste trabalho o culminar de muito esforço e de alguns sacrifícios levados a cabo nos últimos dois anos, período de frequência do Mestrado em Sistemas e Tecnologias da Informação para as Organizações (MSTIO), a par com afazeres profissionais exigentes e sempre tentando não descurar nas relações familiares e pessoais. A todos os meus familiares e amigos, obrigado!

Aos meus colegas e amigos na WIT Software S.A., com quem aprendi imenso nos últimos anos e que sempre me ajudaram a ultrapassar dificuldades e me orientaram em demais projectos profissionais e académicos, obrigado!

A todos os colegas e professores do curso de MSTIO, com quem tive o privilégio de trabalhar, partilhar tantos bons momentos e com os quais aprendi tanto, deixo os mais sinceros agradecimentos.

Em todo este percurso, foram imprescindíveis várias pessoas, que não quero deixar sem uma referência especial neste agradecimento, que apoiaram o projecto durante o seu desenvolvimento e me apoiaram, pessoalmente, em todas as suas etapas.

Os meus agradecimentos ao Eng.º Paulo Tomé, professor em várias cadeiras deste curso e orientador deste projecto, pelos seus ensinamentos, compreensão e apoio.

Agradeço ao Eng.º Rui Oliveira da WIT Software S.A., meu orientador neste projecto e responsável pela equipa que integro na empresa, por colocar à minha disposição todos os meios necessários para o desenvolvimento deste trabalho, por ter apoiado a realização deste projecto académico em paralelo com uma solução comercial, pelas suas orientações e sábios conselhos, em todos os momentos, e amizade pessoal.

Agradeço ao Eng.º Tiago Leitão e ao Eng.º Pedro Gonçalves, ambos colegas de trabalho na WIT Software S.A., pelo apoio técnico no desenvolvimento do servidor e ajuda preciosa na superação de vários obstáculos que foram surgindo.

Por fim, mas sem dúvida a pessoa mais importante, agradeço à pessoa responsável por todo o equilíbrio emocional necessário para a realização deste e outros trabalhos em que me empenho diariamente. À minha amiga, colega, confidente, companheira, a minha mulher, Daniela, pela tua paciência, ajuda, compreensão e amor.

Resumo

Os últimos anos, em especial os anos de 2009 e 2010, foram claramente marcados pela exponencial divulgação e utilização das redes sociais na Internet como forma primária de comunicação. O *Twitter*, uma rede de *micro blogging*, foi dos que mais notoriedade ganhou desde a sua criação, em 2006, devido à simplicidade e rapidez com que se seguem e publicam as actualizações dos utilizadores, quase em tempo real. O *Facebook*, é o fenómeno do momento e teve desde a sua criação, em 2004, o crescimento mais abrupto da história de qualquer ferramenta de comunicação utilizada até à data, conta com cerca de 500 milhões de utilizadores activos de todo o mundo e é um caso de referência da sociedade moderna, independentemente da geração.

Várias outras redes sociais, vocacionadas para vertentes específicas de utilizadores, continuam a nascer ou a sobreviver, contando com comunidades de utilizadores enormes que, com a massificação da Internet, vêem nestas comunidades, a forma mais eficaz e preferencial de obter informações sobre os mais variados assuntos, como exemplos, o *MySpace*, uma rede social que permite a criação de páginas pessoais, vulgarizada por músicos e bandas em ascensão para divulgar o seu trabalho, O *LinkedIn*, uma rede vocacionada para relações profissionais e de apresentação no mercado laboral, o *Foursquare*, que surgiu recentemente, e permite a divulgação da localização geográfica, pontos de interesse e comentários entre utilizadores.

Cada uma no seu ramo, ou tentando competir entre elas quando têm conceitos semelhantes, o importante é que há dezenas de redes sociais, e que continuam a surgir cada vez mais, cada uma com o seu vasto leque de utilizadores e cada utilizador, com presença em várias delas.

A única forma de não se promover apenas algumas das redes sociais em detrimento das demais, é a agregação! Tornar fácil a presença de um utilizador em várias redes, de uma só vez, comunicando com todos os seus contactos, independentemente da sua rede.

Todas as redes sociais actuais disponibilizam acesso às suas plataformas de variadas formas, sendo a maioritária a utilização de *browsers* em computadores. Por vezes fornecem as suas próprias aplicações, mas o mais importante é a disponibilização de APIs que permitam a programadores desenvolver as suas próprias aplicações de interacção com essa rede social abrindo portas a que criativos, externos à organização, criem aplicações que popularizem a sua rede social.

Este trabalho visa à concepção e desenvolvimento de uma plataforma de agregação de várias redes sociais, disponibilizando uma API própria, que permite a implementação de aplicações

cliente unificadas, levando o utilizador a abstrair-se da proveniência das actualizações dos seus contactos e publicando as suas próprias para todos eles.

Neste trabalho, para completar a solução informática, foram ainda implementadas algumas aplicações cliente, perfeitamente escaláveis e que tentam, utilizando a plataforma, colmatar os problemas das aplicações agregadoras que têm vindo, recentemente, a ser criadas, mas que acedem directamente às APIs das redes sociais, e são, assim, constantemente penalizadas pelas alterações, por vezes sem aviso prévio, dos métodos e interfaces de acesso às suas funcionalidades, provocando que estas deixem de funcionar e necessitem de constantes revisões e desenvolvimentos.

Trata-se de um sistema em modelo cliente-servidor, que implementa o conceito de agregação feito no servidor, como que criando uma rede social paralela que agrega as demais, criando uma credencial de acesso única que permite ao utilizador aceder à plataforma em várias aplicações, sem que tenha que configurar consecutivamente as suas contas. No caso de estudo actual, a chave de agregação foi o MSISDN (número de telemóvel) de um operador de comunicações móveis e aplicações vocacionadas para dispositivos móveis[1].

Palavras-chave

Rede social, Internet, agregação, programação, desenvolvimento de software, arquitectura cliente-servidor, comunicação online.

[1] Designados em inglês por *handheld devices*, são computadores de bolso, dotados de um pequeno ecrã e teclado ou ecrã táctil. Os mais relevantes no contexto das redes móveis são os telemóveis e *smartphones*, cuja utilização está cada vez mais vulgarizada.

Abstract

The last few years, especially the years 2009 and 2010 were clearly marked by the exponential use of social networks as primary form of communication. Twitter, a micro blogging network was, probably, the one that had gained the most notoriety since its establishment in 2006, due to the simplicity and speed in which users can follow and publish updates, almost in real time. Facebook is the current Internet phenomenon and, since its creation in 2004, had the most abrupt growth in the history of communication tools, with about 500 million active users worldwide and is, definitely a reference tool to the modern society, regardless of its generation.

Several other social networks directed to specific groups of users, continue to appear or just keep on being used, with large communities, triggered by the popularization of the Internet, looking at online communities as the most effective, and also preferred way, of sharing information on specific topics. Some examples are MySpace, a social network that allows the creation of personal pages, which was popularized by professional musicians and starting bands where they could publish their work, LinkedIn, a network oriented to professionals and personal presentation for the labour market, the Foursquare, which appeared recently, and allows the dissemination of geographic location, points of interest and feedback and ratings from users. Each one focus on each theme, or tries to compete between them when they have similar concepts, but the important thing is that there are dozens of social networks, and growing, each one with its own range of users and each user, states its own presence in several of them.

The only way to avoid promoting and using only a few social networks, forgetting the rest of them, is the aggregation! To make easier the presence of a user across multiple networks, at the same time, communicating with all its contacts, regardless of their network.

All social networks usually provide access to their platforms in several ways, the primary is the use of web browsers on desktop or laptop computers. Sometimes they also provide their own applications, but most important interface is to supply the system with APIs that allow programmers to develop their own applications in order to interact with the social network, opening doors to software designers, from outside of their organization, to create applications that will make their social network popular.

This project aims to design and develop a platform for aggregation of several social networks, providing an API itself, which enables the implementation of unified client applications which will lead the user to abstract himself from the source of the updates and publishing their own for it's contacts.

In this project, in order to fulfil the IT solution, it was also implemented some few client applications, fully scalable, using the platform, bypassing the problems of standard aggregation applications that have been recently created, but which rely on direct API access to each social networks and get affected by the constant changes, sometimes without notice, on their methods and interfaces, causing their applications to cease operating properly and to require constant revisions and developments.

The developed IT system was created over a client-server model, which implements the concept of aggregation on the serve side, as if creating a parallel social network that aggregates the others, creating a single access credential that allows a user to access the platform across multiple applications without having to configure their accounts over and over again. In the current environment, the key of aggregation was the MSISDN (mobile phone number), from a specific mobile operator and applications aimed at handheld devices.

Keywords

Social network, Internet, aggregation, programming, client-server architecture, software development, online communication.

Definições

Para uma correcta compreensão dos textos neste documento, existem termos e/ou expressões que importa clarificar cujo significado se encontra descrito de seguida.

Micro blogging: É uma forma de publicação de conteúdos na Internet, que permite aos utilizadores divulgarem breves actualizações, geralmente com menos de 200 caracteres a um grupo de pessoas ou de domínio público.

Web: É uma forma abreviada de referencia à World Wide Web, que refere o sistema de partilha de informação, em documentos relacionados, na Internet.

Internet: É um aglomerado de redes, à escala mundial que permite o acesso a informações e suporte à transferência de dados, permitindo a existência de serviços de partilha de ficheiros, correio electrónico ou comunicação em tempo real.

Browser: É um programa de computador que permite aos seus utilizadores acederem a páginas com conteúdos produzidos para a Internet.

Online: É um anglicismo que pressupõe a utilização da Internet, "ex. estar online".

Google OpenSocial: É uma plataforma criada pela Google (empresa multinacional de serviços online) que pretende criar um *standard* no acesso a redes sociais ao fornecer uma API aberta que as redes poderão integrar nos seus serviços.

iOS: É um sistema operativo móvel, desenvolvido pela Apple (empresa multinacional do ramo da electrónica e informática), destinado a equipamentos multimédia e aos seus telemóveis: o iPhone.

Android: É um sistema operativo móvel, baseado em Linux, desenvolvido pela Google e pela Open Handset Alliance, que tem vindo a ser adoptado por vários fabricantes de *smartphones* e *tabletPCs*

Security token: É um segmento de texto ou símbolos que tem um significado específico para que possa ser validado entre dois agentes de um sistema, funciona como uma chave electrónica, utilizada para substituir a palavra passe

Back-end: É a camada de software responsável por disponibilizar recursos e serviços à camada de Front-end (o software que utiliza os recursos do back-end, normalmente representa a camada de apresentação, mais visível num sistema informático)

Proxy: É um servidor que filtra e encaminha pedidos numa rede de computadores, tendo, entre outras, capacidades de armazenamento, para caching, ou até de serem usados para ocultação da identidade na Internet

Feeds: São listas de actualização de conteúdos alterados frequentemente, escritos com especificações baseadas em XML (Extented Markup Language)

Siglas

Neste documento são utilizadas siglas para referenciar alguns termos cujo significado se encontra descrito de seguida.

API:	**A**pplication **P**rogramming **I**nterface
MSISDN:	**M**obile **S**tation **I**nternational **S**ubscriber **D**irectory **N**umber
IT:	**I**nformation **T**echnologies
TI:	**T**ecnologias de **I**nformação
SMS:	**S**hort **M**essage **S**ervice
GPS:	**G**lobal **P**ositioning **S**ystem
HTTP:	**H**ypertext **T**ransfer **P**rotocol
CPU:	**C**entral **P**rocessing **U**nit
RAM:	**R**andom **A**ccess **M**emory

Índice

Índice de Figuras

Índice de tabelas

Capítulo 1

1. Introdução

Com o crescente sucesso e aumento do número de redes sociais disponíveis na Internet, sua disseminação e utilização exponencial junto dos utilizadores em todo o Mundo, é bastante comum que cada um de nós tenha várias contas criadas conta criada em várias dessas redes.

Vivemos numa época em que a dependência do acesso constante à informação (Psicologia Digital), aliada à evolução das comunicações em mobilidade e às potencialidades das novas gamas de *smartphones* e outros dispositivos móveis com acesso à Internet, se torna cada vez mais visível, importante ou, até mesmo, imprescindível.

1.1. Objectivos da dissertação

O objecto que origina a presente dissertação consiste no processo de desenvolvimento de uma solução informática, desde a sua conceptualização, viabilização, arquitectura e implementação, a fim de dar suporte à tese de que a agregação de redes sociais, num servidor (plataforma de agregação), é uma solução tecnicamente mais estável e robusta.

Na componente de investigação, foi efectuado um breve estudo às diversas redes sociais, suas APIs, funcionalidades e limitações, a fim de implementar um servidor e clientes que demonstrem a exequibilidade do conceito, em larga escala, adaptado às necessidades e imposições, quanto a disponibilidade, de um operador móvel.

O objectivo desta dissertação é apresentar a solução implementada, sua arquitectura e alguns dos clientes que utilizam a plataforma, descrevendo os processos, funcionalidades, vantagens e desvantagens de uma solução desta natureza.

1.1. Problema

As redes sociais estão notoriamente a ganhar terreno sobre os conceitos que, há alguns anos, definiam a Internet e têm mesmo sido objectos de vários estudos sobre as forma de relacionamento dos seres humanos, com as novas tecnologias (Psicologia Digital). A informação produzida pelos utilizadores dessas redes é de tal maneira avultada, e o valor dessa

informação cada vez maior para quem a recebe, que obriga a que estes consultem os *feeds* com bastante regularidade, onde quer que estejam.

Sendo que cada rede social tem características de utilização orientadas a um grupo de pessoas com um interesse em particular, ou em alguns casos mais generalistas, acaba-se por se fazer parte de várias em simultâneo e a consulta de informação de todas elas, ou a publicação de uma mensagem para os contactos dispersos por cada uma, é um processo moroso e ingrato, mesmo que algumas redes disponham de aplicações ou páginas Web (móveis ou não) simples de usar.

O projecto desenvolvido visa à criação uma plataforma que funcionará como uma camada de abstracção e agregação, permitindo o acesso rápido às redes sociais de que um dado utilizador é membro, podendo aceder e publicar mensagens de uma só aplicação, para todas as suas redes sociais.

1.2. Interesses e motivação

Este projecto visa a simplificar o acesso dos utilizadores das várias redes sociais da Internet, em particular aos que utilizam várias em simultâneo, centralizando e garantindo simplicidade e mobilidade, às actualizações e publicações dos seus contactos e amigos.

Com o crescimento exponencial do acesso à Internet na nossa sociedade, e com a massificação da utilização das redes sociais, sendo estas até consideradas por muitos, como a segunda Internet (Marketingtecnologico.com), é preponderante que estejamos *online*, mesmo em mobilidade, e possamos a qualquer momento publicar conteúdos e comunicar recorrendo à Internet. Podemos abrir algumas dezenas de páginas *web* ou de aplicações para consultarmos as várias redes sociais de que somos membros, ou, por outro lado, podemos fazê-lo de uma só vez, com recurso a uma plataforma que permita a agregação destas.

Com a plataforma de agregação, o desenvolvimento de novas aplicações é mais simples, uma vez que é normalizado no protocolo de comunicação, evitando que se tenham de compreender e analisar todas as APIs proprietárias de cada uma delas, e a partilha de credenciais, quando guardadas num servidor, ao invés de numa aplicação cliente, permite que troquemos de dispositivo (canal de acesso à plataforma), sem qualquer problema, mantendo todas as contas configuradas.

1.3. Abordagens possíveis

Em termos de aplicações que executam funções de agregação de redes sociais, ou acesso individualizado, a abordagem mais comum é estas serem desenvolvidas directamente na plataforma de destino, com toda a lógica implementada no cliente.

Tratando-se de uma aplicação para uma só plataforma, para iPhone por exemplo, não há grandes problemas de escalabilidade, uma vez que a manutenção necessária se irá centrar num só programa. No entanto, se se tratar de um "pacote" de programas para várias plataformas, o esforço de manutenção em caso de necessidade de actualização é multiplicado pelo número de aplicações existentes. A agregação em servidor será bastante mais interessante neste cenário.

1.4. Estrutura da dissertação

Como referido anteriormente, esta dissertação descreve o processo de desenvolvimento da solução informática da plataforma de agregação e algumas das suas potenciais aplicações cliente.

O primeiro, e actual capítulo, pretende contextualizar e introduzir o âmbito do trabalho desenvolvido, sua natureza, interesses e motivações associadas à sua realização.

No segundo capítulo um breve desenvolvimento às motivações da ideia de criação da plataforma apoiado no estado actual de evolução constante das redes sociais e das tecnologias que lhe dão suporte, obrigando a agilidade e reacção rápida às flutuações e mudanças radicais que estas impõem.

O terceiro capítulo aborda a concepção da solução, análise das redes sociais de uma forma geral e suas funcionalidades e interfaces de acesso (APIs), bem como os requisitos e conceitos técnicos e operacionais para a implementação do sistema. Neste capítulo são delineados os pontos de contacto da plataforma com um operador móvel e as vantagens que a utilização de um servidor centralizado lhe conferem.

O quarto capítulo refere-se ao processo de desenvolvimento do servidor, sua arquitectura e interfaces que expõe para a Internet, permitindo a integração com aplicações cliente.

No quinto capítulo é abordado o processo de desenvolvimento de uma aplicação cliente, de forma genérica, descrevendo e ilustrando as funcionalidades disponibilizadas e a interacção com o utilizador na utilização da plataforma de agregação. São ainda ilustrados alguns exemplos de aplicações cliente desenvolvidas no âmbito do projecto, que ilustram um pequeno lote das potencialidades de clientes baseadas no servidor desenvolvido.

Por fim, o sexto capítulo apresenta uma reflexão sobre o trabalho desenvolvido, enaltecendo os prós e os contras deste tipo de solução.

Capítulo 2

2. Enquadramento

Com a evolução natural de cada rede social, aliada às manifestações e requisitos reclamados pelos utilizadores, todos os dias, e também aos requisitos impostos por entidades reguladoras, regra geral, as plataformas de acesso a cada uma das redes, tende a estar em constante mudança, adaptando-se.

2.1. Dinamismo das redes sociais

Nos últimos tempos, um dos temas mais abordados quando se fala em redes sociais é a privacidade dos seus utilizadores e os riscos de utilização ilícita dos dados publicados (PT.WEBAPPSEC). Face a isto, há um crescente corrupio na adopção de métodos que permitam um maior controlo por parte dos utilizadores, acerca de a quem divulgam quais informações. Estas alterações, obrigam à reformulação constante da arquitectura das redes sociais e consequentemente, afectam todas as suas camadas de *software*, até às API.

2.2. Métodos de autenticação

Um dos dados mais sensíveis quando se fala em qualquer serviço *online*, que pressuponha autenticação dos utilizadores, é a palavra passe, que por motivos óbvios de segurança deve sempre ser salvaguardada e protegida durante os acessos, principalmente os externos, efectuados através de APIs, em ligações inseguras e em que utilizador e palavra passe circulam juntos na rede, sem qualquer encriptação.

As APIs mais antigas de acesso às redes sociais, para aplicações de terceiros, recebiam em todos os seus pedidos o utilizador e palavra passe do utilizador, tendo esta que ficar alojada algures no dispositivo, localmente em memória, permitindo várias formas de violação dos dados pessoais.

Um exemplo de uma rede social que utilizava uma autenticação simples (utilizador + palavra passe), no ínicio deste projecto, era o *Twitter*, que abandonou este método entretanto (Twitter, 2010), mostrando a importância de centralizar os processos de autenticação numa camada que não a aplicação cliente final.

Por forma a proteger os dados pessoais dos utilizadores, evitando que estes tenham que entregar o seu nome de utilizador e palavra passe a aplicações cliente que desconhecem, e podem até ser mal intencionadas, a grande maioria das redes sociais tem vindo a migrar a sua autenticação para o *OAuth* (Open Authorization).

2.2.1.OAuth

O *OAuth* é um protocolo de autenticação que visa a evitar que o utilizador tenha de fornecer as suas credenciais à aplicação cliente (Wikipedia) que vai utilizar para aceder a recursos, fotos, vídeos, contactos, etc., mantendo assim a confidencialidade das suas credenciais, uma vez que estas não ficaram guardadas em lado nenhum, sendo mesmo enviadas já encriptadas em *md5*[2].

O funcionamento deste método de autenticação é bastante simples e baseia-se num fluxo de procedimentos (Imagem 1 – Diagrama de fluxo da autenticação OAuth) em que o utilizador, escreve o seu utilizador e palavra passe numa página *web* proprietária do fornecedor do serviço (rede social para este caso) e após autorizar o acesso às diversas funcionalidades recebe um *security token* que é devolvido à aplicação, e é essa chave que é usada desse momento em diante, sem que tenha necessidade de, sequer, ler a palavra passe ou nome do utilizador.

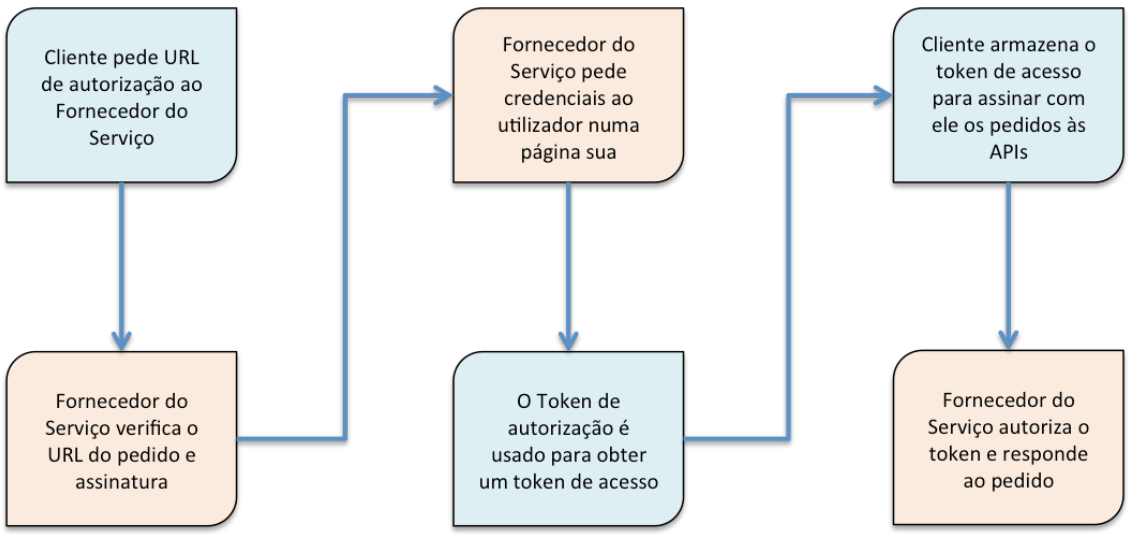

Imagem 1 – Diagrama de fluxo da autenticação OAuth

Como se verá ver mais à frente neste documento, em termos práticos, na arquitectura da solução desenvolvida, o cliente e o local onde são armazenados os *tokens* de acessos, é a própria base de

[2] É um método criptográfico de 128 bits, que gera chaves num só sentido e não tem forma conhecida de se desencriptar. Utilizado por exemplo em palavras passe, comparando-se ambos os valores em *md5*.

dados da plataforma de agregação, para que estes possam estar disponíveis para todos os clientes, mas sempre sem expor a palavra passe em qualquer circunstância.

2.3. Adopção da Google OpenSocial

O *OpenSocial* (Google) é uma plataforma que fornece serviços para acesso a redes sociais, permitindo a criação de simples *widgets*, baseadas em HTML e JavaScript ou através das suas APIs. O *OpenSocial*, encara as redes sociais de uma forma abstracta, fornecendo acesso a informações de forma genérica a perfis de utilizadores, amigos e actividades, objectos esses que correspondem a informações ou conceitos distintos no seio de cada rede social.

As vantagens da adopção do *OpenSocial* pelas redes sociais é o facto de não terem que suportar as APIs directamente, delegando esforço de desenvolvimento para entidades externas, no entanto traz desvantagens, na medida em que não podem contar com funcionalidades demasiado específicas da sua rede (como por exemplo, o Facebook, e as funcionalidades próprias como o Like e outras).

Para quem desenvolve aplicações que acedam a várias redes sociais, há vantagens, na medida em que o acesso às redes sociais que tenham aderido à plataforma *OpenSocial* é feito de maneira igual para todas elas.

Entre outras funcionalidades, a plataforma *OpenSocial* oferece suporte para autenticação *OAuth*, descrita na ponto anterior, e é portanto uma mais valia para redes sociais que não tenham necessidades demasiado especificas.

Actualmente contam com integração *OpenSocial* as redes sociais: *Orkut*, *MySpace*, *Friendster*, *LinkedIn*, *Hi5*, *XING*, *Plaxo*, *Ning*, *Oracle*, *Viadeo* e *SalesForce*. (Google)

Facebook, *Twitter* e muitas outras, continuam a preferir desenvolver as suas próprias APIs, mais robustas e completas ou porque simplesmente, não querem perder o controlo do desenvolvimento de todas as suas camadas de integração e peças de software.

2.4. Volatilidade de aplicações cliente

Naturalmente, o canal primário de acesso à rede social, o *web browser*, não sente os efeitos das constantes alterações, em termos técnicos, uma vez que a aplicação *web* que disponibilizam é

proprietária e têm sobre ela total controlo, os únicos resistentes e aversos à mudança, poderão ser os próprios utilizadores.

Por outro lado, todos os dias, existem programadores que com base nas API de uma rede social, iniciam o desenvolvimento de novas aplicações, estando sujeitos a alterações da API, e consequentemente verem as suas aplicações deixar de funcionar de um momento para o outro. Se estivermos a falar de uma aplicação que se ligue a várias redes em simultâneo, o risco de uma delas modificar a sua API e a aplicação deixar de funcionar, multiplica-se.

Em casos extremos, as redes sociais estão mesmo a descontinuar APIs antigas, remetendo o esforço de desenvolvimentos em novas APIs para terceiros, adoptando plataformas de acesso normalizadas, como o já referido OpenSocial, obrigando novamente os programadores a refazer a camada de comunicação das suas aplicações. Havendo várias versões para diferentes plataformas e sistemas operativos (*iOS* e *Android* como exemplos em plataformas para dispositivos móveis), e dentro de cada plataforma várias versões em simultâneo que podem trazer incompatibilidade (Android.com), o esforço de reconstrução multiplica-se por quantas aplicações com as mesmas funcionalidades tivessem à data.

2.5. Necessidades de um operador móvel

Quando se fala em serviços que possam ser disponibilizados por um operador móvel, há uma série de condicionantes que os tornam viáveis ou não. Principalmente, se houver tráfego de dados proveniente da Internet na equação.

Um operador móvel tem que conseguir mapear os custos de utilização da Internet geral, para descontar de pacotes de dados que o utilizador possa ter, ou facturar por unidade de medida de informação, separando-os de serviços que queira facturar com características particulares ou mesmo permitir o acesso livre, sem contabilizar o tráfego.

Ao acedermos a várias redes sociais em simultâneo, são feitos acessos às APIs destas, sendo os seus endereços de domínio público e variável, tornando impossível a criação de regras de facturação para esse tráfego. A existência de uma plataforma, instalada no operador, centralizada que funcione como *proxy* na comunicação com essas redes, vai colmatar esse problema, na medida em que o acesso é feito exclusivamente via esse servidor e se torna simples perceber quais os dados de um utilizador que dizem respeito a acessos à plataforma de agregação de redes sociais.

Com esta centralização, não só as necessidades de facturação e/ou subscrição do serviço ficam passíveis de ser implementadas, como permitem uma integração com o *back-end* existente e a associação das redes sociais de um utilizador a um cliente seu. O que é um cenário bastante interessante, pois permitirá a configuração e partilha de contas das redes sociais, entre todos os seus canais de comunicação, desde telemóveis, passando por páginas *web* do operador, ou até mesmo, no caso de o operador fornecer outro tipo de serviços, como televisão por cabo, permitindo integrar todas as redes sociais no leque de aplicações disponíveis.

Capítulo 3

3. Estudo preliminar

Nos capítulos anteriores, foi identificada a questão motivadora deste projecto. Para delinear a estratégia de desenvolvimento, houve um processo inicial de análise às soluções semelhantes existentes no mercado, questionando os seus principais pontos de falha e procurando colmatá-los aquando da concepção desta diferente abordagem ao problema.

3.1. Soluções semelhantes

Como referido anteriormente, há várias aplicações no mercado como, por exemplo, a *Nomee*[3] que para além de agregar algumas redes, permite ainda adicionar *feeds* de actualização. A criação de agregadores de redes sociais ganhou força durante o ano de 2010, tendo inclusive a *Microsoft* criado um *website* que agrega algumas delas, o *Spindex*[4], ainda numa fase inicial, mas que denota que a agregação é uma abordagem que tem vindo a ser explorada, principalmente na segunda metade do ano (Wikipedia).

Ainda não surgiu uma plataforma que seja realmente revolucionária e seja, por isso, adoptada em larga escala. Em parte porque as existentes optam por incluir demasiados e divergentes conteúdos nas suas plataformas, descurando na especificidade das redes sociais e nos meios de acesso .

A lacuna deixada nas plataformas existentes é o facto de não fornecerem APIs para programadores externos, não sendo mais do que uma aplicação *web*, ao invés de uma plataforma agregadora.

A possibilidade de integração com um operador móvel foi o elemento que tornou o projecto mais interessante, abrindo portas à interligação de mecanismos que utilizamos diariamente desde há vários anos, como, SMS, voz e e-mail, fazendo uma ponte para os meios de comunicação mais recentes, as redes sociais.

Qualquer pessoa, que viva em sociedade, tem sobre si uma série de dados pessoais que o identificam na sociedade e "mundo real" e outras que o identificam nas diversas comunidades

[3] Aplicação para *iPhone*, *Windows* e *MacOS* que agrega *Facebook*, *Twitter* e *LinkedIn*, bem como *feeds* noticiosos de vários jornais online disponível em: http://www.nomee.com/;
[4] Website agregador de algumas redes sociais, desenvolvido pela *Microsoft*, disponível em: http://fuse.microsoft.com/project/spindex.aspx;

na Internet. Tem uma série de relacionamentos e ligações provenientes de contactos pessoais, profissionais ou familiares.

Que melhor sítio, nos dias que correm, para ter centralizadas todas as informações e contactos, que o nosso telemóvel?

3.2. Análise às redes sociais

Em vez de criar uma plataforma que se especialize em cada uma das redes sociais, aprendendo as suas regras, estrutura de informação e conceitos, optou-se por seguir uma abordagem que encontrasse os pontos comuns entre todas elas, independentemente do seu formato.

Uma publicação não deixa de o ser, estejamos nós a publicar um comentário a uma notícia que esteja a passar na televisão, ou por outro lado estejamos a divulgar a pastelaria em que estamos a lanchar. FourSquare, Twitter, Facebook, ..., todas estas redes são bastante diferentes em conceito e funcionalidades, mas são passíveis de ser abstraídas em pontos comuns: Posts, Amigos/Contactos e informação.

No capítulo seguinte analisam-se as redes sociais, no contexto deste trabalho, em mais pormenor.

3.2.1. Twitter

O Twitter é um serviço de micro *blogging* (Wikipedia), em que os utilizadores subscrevem as publicações de outros obtendo-as, quase em tempo-real, na sua página de perfil. A sua principal característica é o tamanho máximo das mensagens que se podem publicar, apenas 140 caracteres (também conhecido como o SMS da Internet).

3.2.1.1. Funcionalidades e Conceitos

Como referido anteriormente, o conceito essencial da rede social é o "*tweet*" (ou actualização). Ss contactos de cada utilizador são os "friends" (ou seguidores) e existe a possibilidade de republicar uma actualização de outra pessoa, reencaminhando-a para os seus amigos, a esta funcionalidade, chama-se, "*retweet*". Permite ainda mensagens directas públicas e privadas, referindo-se aos nomes de utilizador com o prefixo: caracter @ (arroba). Por fim, existem as tendências do momento, ou tópicos mais discutidos, os "*trending topics*", que são precedidos no texto pelo caracter #, incluídos no texto pelos utilizadores, com o intuito de relacionar as suas actualizações com um tópico em concreto.

3.2.1.2. API

A rede social *Twitter* deve grande parte do seu sucesso à sua API, que fornece acesso a todas as funcionalidades descritas, e à excelente documentação que proporciona a quem queira desenvolver aplicações que interajam com esta. E mesmo empresas que utilizam a massificação desta forma de publicação de opinião, para estarem "à escuta" de referências às suas marcas, para tal compram acessos privilegiados à API, com funcionalidades extra ligadas ao *data mining* (processo de exploração de grandes quantidades de dados, obtendo padrões e dados estatísticos de elevada importância).

3.2.2. Facebook

O Facebook é actualmente a rede social mais utilizada em todo mundo, contanto actualmente com cerca de 500 milhões de utilizadores, e baseando-se na criação de um perfil biográfico e no relacionamento, por convite, entre utilizadores, que partilham conteúdos multimédia ou mensagens (Facebook; Facebook).

3.2.2.1. Funcionalidades e Conceitos

O *Facebook* apresenta um leque vasto de funcionalidades, sempre à volta do conceito de perfil, classificando interesses, religião, trabalho, localização geográfica, entre outros, permitindo a adesão a grupos de variadas categorias e, o mais importante, permitindo a publicação de imagens, vídeos, hiperligações[5], e texto, permitindo efectuar comentários a todos esses objectos, ou simplesmente, marcar posição dizendo "gosto" desse conteúdo (Wikipedia). Recentemente a rede social conta também com informações de estado de presença e comunicações instantâneas.

3.2.2.2. API

A API da rede social *Facebook* a que mais evoluiu nos anos, com alterações constantes, seguindo as tendências tecnológicas, cedendo a pressões relacionadas com falhas reportadas nas suas políticas de privacidade. Em 2007, foi lançada a sua primeira plataforma para desenvolvimento de aplicações externas, contando com sete mil aplicações desenvolvidas, apenas um ano depois. Com isto houveram várias aplicações, que em jeito de publicidade invasiva (*spam*) inundaram as páginas de apresentação dos utilizadores, poluindo a informação relevante.

Seguiu-se a criação de várias componentes da plataforma de desenvolvimento para a rede social, com uma série de formas de integração, a alto nível, para *blogs* e páginas *web* por exemplo, e APIs de mais baixo nível para que pudessem ser desenvolvidas aplicações "de raíz".

A base de objectos da rede social é o "*Social Graph*", um grafo[6], que abstrai todos os objectos presentes na rede social e os trata como um todo. Dando ao programador total flexibilidade de acesso e alteração aos elementos neles contidos.

Uma imagem, um texto, uma hiperligação, qualquer objecto é tratado de igual maneira na API do *Facebook*, o que torna bastante interessante o desenvolvimento de aplicações. (Facebook)

[5] É uma referência dentro de um local num documento, ou para outro documento (normalmente utilizados para navegação em páginas *web*);
[6] Utilizado em matemática e ciências da computação, basicamente são pontos (vértices) ligados por arestas e que permitem ser relacionados com problemas da vida real, ajudando à sua resolução;

3.2.3. Hi5

A rede social *Hi5* foi a mais utilizada até ao ano de 2008 (Wikipedia), altura em que o *Facebook* ganhou maior força, foi inclusivamente e o *website* mais utilizado em Portugal[7] em 2007.

3.2.3.1. Funcionalidades e Conceitos

Os utilizadores criam uma página pessoal, personalizada, com fotografias e um leitor de música que toca as suas músicas de eleição. Os utilizadores trocam pedidos de amizade e podem depois comentar as fotografias e aderir a grupos.

3.2.3.2. API

Como programador, terei que tecer a minha opinião acerca do que levou ao declínio daquela que era maior rede, mas que tem caído gradualmente de posição: a API da rede social *Hi5* é extremamente fraca, expõe pouquíssimas das suas funcionalidades e tem uma documentação bastante incompleta, não dispõe de envio de mensagens directas, entre outras limitações.

[7] Fonte: Alexa.com, empresa especializada em estudos analíticos na Internet;

3.2.4. LinkedIn

É uma rede vocacionada para os negócios, que pode facilmente ser comparável a uma rede social das demais referidas neste trabalho. Tem um mercado muito próprio, e conta com cerca de 16 milhões de utilizadores de 400 indústrias diferentes (Wikipedia), que promove as relações profissionais e a divulgação e procura de ofertas de trabalho, entre outras actividades.

3.2.4.1. Funcionalidades e Conceitos

Oferece funcionalidades como consulta de perfis, em contactos com relacionamento directo, em segundo ou terceiro grau, permitindo escalar na hierarquia de conhecimentos, envio de mensagens e recomendações e notificações de alterações de parâmetros do perfil, como empresa ou cargo ocupado.

3.2.4.2. API

A API é bastante boa, no entanto o conceito da rede, desvia-se um pouco do das demais redes sociais, mas ainda assim, é uma rede importante para uma plataforma de agregação de contactos, dado o seu volume de utilização.

Fornece autenticação segura e acesso às suas funcionalidades e encontra-se bem documentada. Faz sentido integrá-la dentro das suas limitações (que não o são realmente, são apenas divergências conceptuais).

3.2.5. MySpace

O *MySpace* é actualmente a segunda mais utilizada em todo mundo, contando com cerca de 110 milhões de utilizadores (Wikipedia) e com características que a tornaram bastante atraente, nomeadamente no universo musical, desde bandas de garagem a músicos profissionais, todos publicaram a sua apresentação e algum do seu trabalho nesta rede social.

3.2.5.1. Funcionalidades e Conceitos

A rede social oferece aos seus utilizadores uma forma simples de criar a sua página pessoal, permitindo a este pertencer a grupos e publicar informação na sua página. O carregamento de ficheiros de música é outra das funcionalidades que fez vingar esta plataforma.

3.2.5.2. API

A API do *MySpace* foi criada em 2008, precisamente quando houve o "boom" das redes sociais, para acompanhar o crescimento das demais redes, sem perder terreno.

A API é relativamente limitada e limitante, mas permite fazer o essencial para a integração neste projecto. A API não oferece suporte ao carregamento de multimédia, por exemplo, mas permite actualizar os dados textuais do perfil dos utilizadores, e obter os seus amigos. Para o caso, é suficiente.

3.2.6. Orkut

O Orkut é uma rede social que foi criada pela *Google* em 2004 (Wikipedia), muito popularizada no Brasil e Estados Unidos, com que foi sempre alvo de muitos ataques e esteve sempre ligada a fenómenos de insegurança da Internet, originou as primeiras discussões sobre privacidade nas redes sociais.

3.2.6.1. Funcionalidades e Conceitos

Oferece aos seus utilizadores funcionalidades de conversação em tempo real, e permite que o seu perfil funcione como um livro de visitas "scrapbook", permitindo que os amigos deixem recados. A organização de eventos é uma das funcionalidades mais utilizadas na rede social.

Actualmente aproxima-se, na rectaguarda, das funcionalidades do *Facebook*, com algum atraso e a perder força.

3.2.6.2. API

A API do *Orkut* apenas fornece o mínimo, amigos e actualizações de estado, mas é o suficiente para a integração na plataforma.

3.2.7. Foursquare

O *Foursquare* (Wikipedia) é baseado na partilha de informação da localização dos utilizadores, por sua vontade, aos seus amigos, ou publicando-a em redes sociais como o *Twitter* e/ou *Facebook*. Em paralelo, funciona como um jogo, em que o número de vezes que um utilizador visita um local, lhe dá pontos e *rankings*, é vocacionada para dispositivos móveis e apoia-se na utilização das suas funcionalidades GPS.

3.2.7.1. Funcionalidades e Conceitos

Permite publicação da localização, com uma mensagem associada, e publicação automática de informações relacionadas com o local em que se faz o "check-in". Conta ainda com acesso aos contactos e uma *timeline* com as suas actualizações.

3.2.7.2. API

A API é completa e oferece todas as funcionalidades de que a rede social dispõe, permitindo o envio de coordenadas para geolocalização das mensagens publicadas.

A plataforma de agregação não irá dar uso às coordenadas (pelo menos não ao nível académico).

Conclui-se que haverá pontos de contacto suficientes para uma integração com esta rede, na API fornecida e que se encontra devidamente comentada.

3.3. Abstracção de objectos e conceitos

Após analisar com cuidado as características e capacidades das diversas redes a integrar, torna-se preponderante para se poderem tratar como iguais, sendo diferentes, identificando nelas conceitos comuns, ou que se enquadrem nos objectos pretendidos.

Repare-se, para construir uma "***timeline***" (informação oriunda da rede social, organizada de forma temporal) basta que haja uma fonte de informação relevante para o utilizador em questão, isto é, que tenha origem nos seus contactos. Contactos esses que podem ser entidades diferentes nas diversas redes, mas que podem facilmente ser reconhecidos da mesma forma, como "**amigos**". Por fim, identificar o propósito essencial da existência da rede, a partilha, o envio de informação para quem a quiser receber. Não há comunicação sem um emissor e sem um receptor. Uma divulgação para a rede social, é um "***Post***", ou publicação.

A tabela (Tabela 1 - Correspondência de conceitos entre as redes e a plataforma de agregação) seguinte ilustra, a alto nível, a correspondência entre a funcionalidade de cada rede social e os conceitos agregadores, assumidos no prisma da plataforma de agregação.

Tabela 1 - Correspondência de conceitos entre as redes e a plataforma de agregação

Plataforma Agregação	Twitter	Facebook	Hi5	LinkedIn	MySpace	Orkut	FourSquare
Timeline	Tweets vindos de todos os que se seguem;	Actualizações de fotos, publicação de hiperligações, mudança de estado, etc.;	Novas publicações oriundas dos amigos;	Novas ligações, alteração de mensagem de estado;	Publicações na página de perfil feitas pelos utilizadores que seguem;	Publicações de perfil e mensagens difundidas por amigos	"check-in" e mensagens publicadas por contactos/amigos ou vindas da plataforma;
Amigos	Contas seguidas pelo utilizador	Amigos	Amigos	Contacto profissional	Amigos ou fã de alguém	Amigos	Amigos
Publicação	"Tweet", publicação de mensagem	Alteração do estado	Publicação na página de perfil	Estado actual / memorando	Publicações de conteudos na página de perfil.	Publicações de alteração ao perfil;	Publicação de mensagem de estado;
Mensagem directa	Mensagem directa, pública ou privada	Mensagem directa, pública ou privada	Não disponível na API;	Mensagem directa, privada;	Não disponível na API;	Mensagem directa, privada;	Não disponível / Não faz sentido;

As funcionalidades específicas de cada rede, que sejam relevantes e imprescindíveis para não se perder a essência da rede social, terão que ser implementadas especificamente.

3.4. Necessidades do operador móvel

Para o cenário concreto de uma integração com um operador móvel, para que a plataforma possa ser um serviço disponibilizado aos seus assinantes, há algumas necessidades que têm que ser preenchidas. Neste ponto faz-se uma breve descrição de algumas das mais importantes que tornam a solução viável e passível de utilização real.

3.4.1. Pluralidade de aplicações cliente

Uma característica importantíssima para um operador móvel é que os seus serviços possam suportar vários terminais, centenas de modelos, novos ou mais antigos, independentemente das suas características. Para suportar vários telemóveis, os operadores dispõem de motores de desenho de páginas com base na identificação do dispositivo e torna-se simples a criação de aplicações WAP (*rendering engines*), minimalistas, mas que por serem suportadas em todo o tipo de dispositivos, mesmo os de baixa gama, são preponderantes para o fornecimento de serviços. Para os modelos mais recentes, com tecnologia mais actual, interessa criar clientes mais específicos e mais trabalhados, como são os casos de aplicações para iPhone, dispositivos com sistema operativo *Android*, ou *Windows*.

Não menos importante que suportar centenas de dispositivos é permitir a centralização do serviço no próprio cliente e permitir que este possa aceder aos seus serviços, em vários suportes. Os operadores móveis evoluíram para empresas com vastíssimo leque de serviços e é a oferta e variedade de serviços que os distingue.

3.4.2. Partilha de credenciais

Para um operador móvel, poder oferecer aos seus clientes serviços na Internet, na Televisão ou no seu dispositivo móvel, é muito importante. A plataforma agregadora, permite que o assinante do operador possa aceder ao serviço de várias aplicações, sem ter que as configurar, sendo reconhecido de imediato, pelo seu número de cliente.

3.4.3. Controlo de tráfego e modelo de negócio

Quando do lado do operador, tecnicamente, é necessário que o tráfego proveniente da Internet seja canalizado pelos seus servidores, para que possam ser criadas regras específicas de nos

endereços utilizados para oferecer a sua utilização, ou, se assim for expectável, associar o serviço ao sistema de facturação e poder cobrar taxas pela sua utilização.

A plataforma, tendo apenas um interface exposto para o exterior, e funcionando de *Proxy* para o carregamento de informações e imagens, permite separar o tráfego gerado pela plataforma do restante que será descontado ao utilizador.

Outro modelo de negócio suportado é a possibilidade de subscrição de utilização, ao invés de contabilizar tráfego, pode fornecer-se acesso livre, mediante subscrição, como descrito mais detalhadamente mais à frente (4.1.1.4.1.2 - Subscrição e facturação).

3.5. Tecnologias de servidor

Num cenário de milhares ou mesmo milhões de utilizadores, um dos factores preponderantes é a disponibilidade dos serviços expostos e a sua escalabilidade. Permitindo o balanceamento de carga nos servidores que alojam os serviços desenvolvidos.

3.5.1. Servidor aplicacional

O servidor foi desenvolvido tendo por base um servidor *web*, o *Apache Tomcat 6*, um servidor livre, baseado em Java (linguagem de programação orientada a objectos), que é muito robusto, e facilmente colocado em várias máquinas em paralelo para assegurar a disponibilidade.

O servidor Tomcat, funciona através de servlets (uma classe Java que permite receber e responder a pedidos HTTP em redes de computadores).

A base de dados de apoio utilizada foi o *Oracle 11G*, a fim de dar suporte ao armazenamento dos *tokens* de autenticação e recolha de registos de utilização para fins estatísticos.

O servidor desenvolvido é a base de todo o funcionamento da solução, sendo que as aplicações cliente representam apenas uma forma de materializar o seu funcionamento interno.

3.5.2. Linguagens de programação

Como foi referido anteriormente, o servidor foi exclusivamente desenvolvido utilizando a linguagem Java, recorrendo a algumas bibliotecas fornecidas pelas redes sociais para a integração com as suas APIs, ou em alguns casos, em que estas não eram disponibilizadas, o processo de acesso foi manual, efectuando pedidos HTTP à rede social e fazendo *parsing* [8]às suas respostas.

3.5.3. Protocolo de comunicação

A comunicação entre servidor e clientes é feita utilizando uma técnica de engenharia de software, denominada de REST (*Representational State Transfer*), que se baseia em padrões de texto para a troca de mensagens.

Aos sistemas que que implementam os princípios do REST chamam-se, vulgarmente, de sistemas RESTful.

[8] É o nome dado ao processo de análise sintática de texto, extraindo informação com base num modelo pré-estabelecido. Em informática é um termo muito associado à comunicação usando protocolos XML;

No caso da plataforma agregação de redes sociais, o interface REST funciona com base em dois protocolos, o XML e o JSON[9], sendo o primeiro muito mais extenso na sua escrita e o segundo bastante mais leve. No entanto, consoante as limitações da aplicação cliente, o programador pode optar por aceder à plataforma em qualquer dos formatos.

Como veremos mais à frente, na arquitectura, a plataforma expõe a sua API com base num interface REST que é posteriormente utilizada por todos os clientes da solução.

[9] Um acrónimo pra *Javascript Object Notation*, é um formato leve para comunicação entre serviços, segue a notação do *Javascript* (linguagem de programação, interpretada, que executa em *browsers web*), mas não tem obrigatoriamente que ser utilizada em aplicações desenvolvida nessa mesma linguagem. É perfeitamente transversal a qualquer plataforma;

3.6. Tecnologias de cliente

Independentemente do cliente que se pretenda desenvolver, desde que o dispositivo em questão e o seu SDK[10] se encontre dotado de capacidade de efectuar pedidos HTTP, este é passível de ter desenvolvida uma aplicação que utilize a plataforma de agregação. Mesmo que não tenha suporte para aplicações "nativas", este poderá sempre utilizar uma abordagem *web* (acedendo a uma aplicação *online*), para aceder à plataforma.

3.6.1. Protocolo de comunicação

A plataforma de agregação fornece um interface REST para acesso à sua API, podendo a comunicação ser feita com recurso a XML ou a JSON, consoante o programador preferir, ou conforme existam ou não bibliotecas pré-fabricadas para a linguagem de programação a utilizar.

3.6.2. Linguagens de programação

Neste projecto, e no contexto académico, foram desenvolvidos dois, de vários, clientes possíveis para integração com a plataforma, como prova de conceito e ferramenta de exploração das funcionalidades disponibilizadas.

Foi criada uma aplicação para iPhone e uma Opera *widgets*, mini aplicações com interface gráfico, para telemóveis compatíveis.

A aplicação para iOS foi desenvolvida utilizando a linguagem Objective-C[11], utilizando a camada "nativa" de acesso à rede e para comunicação, o protocolo XML, com recurso à biblioteca *TouchXML*[12].

A *widget*, foi desenvolvida usando JavaScript, HTML e CSS (Cascade Styling Sheet) e conectou-se à API da plataforma utilizando a plataforma REST, utilizando a sintaxe JSON, tornando imediato o acesso aos objectos da resposta do servidor, na aplicação.

A plataforma de desenvolvimento de clientes é claramente irrelevante, na medida em que qualquer linguagem poderá ser usada para a criação de um cliente desta plataforma, bastaria aos programadores seguirem o protocolo descrito no documento em anexo (A.1 Protocolo REST – Especificação da API).

[10] Acrónimo para *Software Development Kit*, corresponde ao conjunto de bibliotecas para o desenvolvimento de *software* em determinado ambiente;
[11] Trata-se de uma linguagem derivada do C, com suporte a objetos e paradigmas de programação mais avançados;
[12] É uma biblioteca desenvolvida para desenvolvimento em Objective-C para facilitar operações com XML, disponível em: http://foobarpig.com/iphone/touchxml-installation-guide.html;

Capítulo 4

4. Processo de desenvolvimento

4.1. Servidor

Desde a concepção, à implementação, do servidor, executaram-se vários processos de análise e especificação de requisitos do sistema e necessidades de utilizadores e operador móvel.

De forma adaptativa refinou-se a arquitectura do sistema, as tecnologias a utilizar e deu-se inicio à implementação.

Este capítulo refere-se ao processo de análise, concepção e desenvolvimento da solução informática, objecto desta dissertação, sem entrar em detalhe quanto à sua codificação.

4.1.1. Arquitectura

A arquitectura do sistema implementado segue um modelo MVC[13] (*Model View Controller*), estruturado em camadas de abstracção que permitem isolar os módulos lógicos, por tarefas e contextos, para que o código e a plataforma seja escalável e seja possível de resolver de forma ágil os problemas que forem surgindo.

4.1.1.1. Arquitectura de alto nível

A maior preocupação ao desenhar a plataforma foi que a camada de dados, ver Imagem 2 - Arquitectura de alto nível do servidor, pudesse ser perfeitamente normalizada desde o primeiro momento, a fim de redigir o documento com o protocolo de comunicação REST (ver anexo: Protocolo REST – Especificação da API), e daí poder partir-se imediatamente para a construção de aplicações cliente. O Objectivo foi que a camada de comunicação ficasse "congelada" e não fosse necessário mexer-lhe quer para a resolução de problemas, quer em caso de evoluções de pequena escala, permitindo assim, que os clientes que fossem entretanto desenvolvidos se mantivessem compatíveis com o servidor, mesmo que nas camadas de acesso às redes sociais, algo mudasse. Em jeito de conclusão antecipada, veio a revelar-se uma boa solução.

[13] É uma arquitectura de *software*, que procura separar o modelo lógico da aplicação das suas operações e do seu interface gráfico, estratificando o sistema desde o modelo de negócio até à interacção com o utilizador.

Como referido anteriormente, e devido às necessidades de controlo de tráfego do operador móvel, foi necessário garantir que todo o tráfego passava pelo servidor da plataforma, assim, para que pudessem ser carregadas, por exemplo, as fotos de perfil dos utilizadores, foi necessário criar uma *servlet* que encaminha os pedidos, redimensiona e converte as imagens para formatos compatíveis com terminais "*low end*" (fim de linha ou antigos). Por uma questão de segurança, todos os acessos são assinados digitalmente com uma chave encriptada, para que não se utilize a *servlet* de imagens para fins que não os propostos pela plataforma.

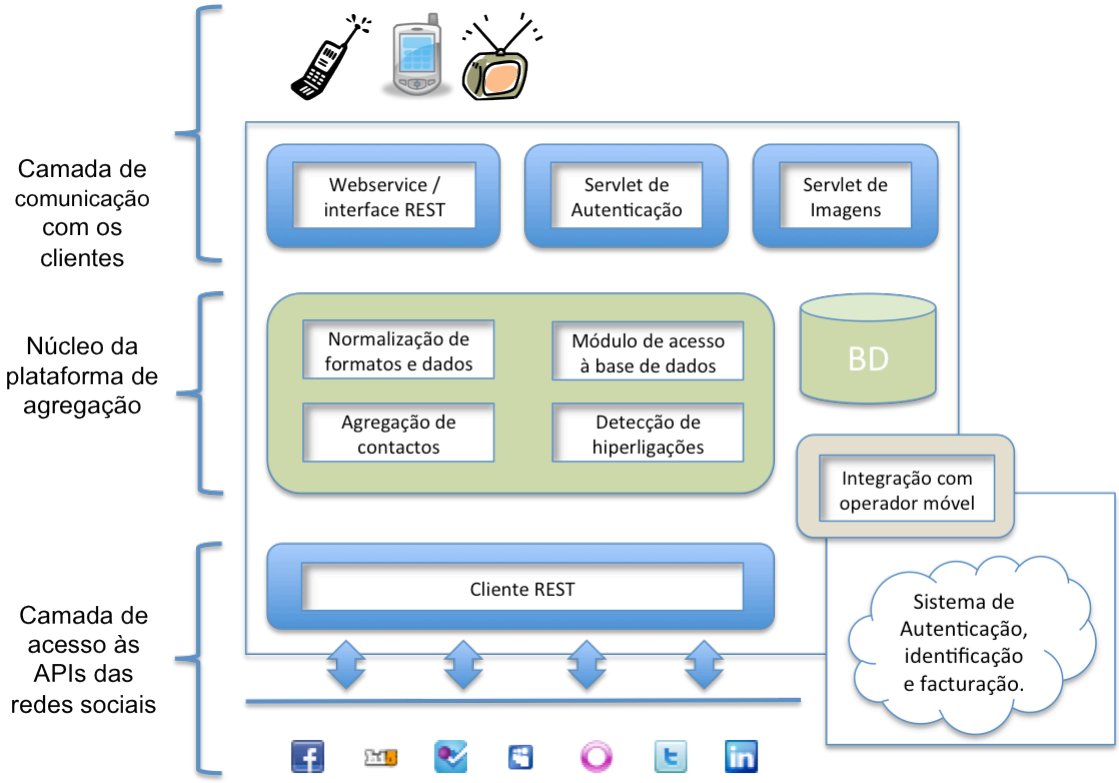

Imagem 2 - Arquitectura de alto nível do servidor

A imagem acima ilustra os módulos de relevo existentes no servidor e algumas funcionalidades implementadas. Nos pontos seguintes é feita uma abordagem mais detalhada a cada um dos componentes de servidor mais importantes.

4.1.1.2. Camada de acesso às redes sociais

A camada de acesso às redes sociais contem classes para os objectos relativos aos pedidos e resposta de cada API e é responsável por faze-la chegar à camada de cima, para processar o seu conteúdo.

Num cenário ideal, esta seria a camada que iria garantir a escalabilidade total, permitindo acrescentar novas redes ao lote de disponíveis. Tal só não foi completamente atingido por haverem funcionalidades específicas, 4.1.1.2.5 , que são preponderantes para justificar a inclusão de uma rede social na plataforma. Não havendo essas "*nuances*" o sistema seria perfeitamente escalável.

4.1.1.2.1. Agregação de amigos

A agregação de contactos, infelizmente, foi a única funcionalidade que não foi implementada como tinha sido previamente idealizada, devido a dificuldades no relacionamento ente contactos de diferentes redes que na prática fossem a mesma pessoa.

A ideia era que pudesse ser estabelecido um campo de comparação que levasse a concluir que um utilizador de uma rede era a mesma pessoa que o utilizador de outra rede, podendo mostrar uma só entrada, com todos os seus contactos. No entanto, os únicos campo que poderiam fazer sentido seriam ou o e-mail (uma vez que não são fornecidos dados pessoais que não os endereços electrónicos), no entanto, várias redes não fornecem sequer o contacto de e-mail ou este não é obrigatório, ou o nome da pessoa, mas nesse cenário era impossível de tratar as repetições, por haverem várias pessoas com mome e sobrenome iguais.

Optou-se por apenas intercalar os contactos, sem os agregar.

4.1.1.2.2. Tratamento de updates

Todo o esforço de tratamento do conteúdo dos *updates* (publicações) provenientes das redes sociais, é feito no servidor, tendo sido criados métodos de normalização de elementos multimédia, convertendo em hiperligações HTML vídeos, fotos, etc.

Em alguns casos, um *update* é bastante complexo e traz consigo uma série objectos relacionados, que são mapeados e interpretados, construindo um *output* exclusivamente textual.

Os *updates* são pedidos às redes sociais subscritas pelo utilizador, em simultâneo, e depois de processadas individualmente, são ordenadas ficando intercaladas na *timeline* do utilizador. Foi implementado um sistema de *cache*[14] de curta duração, para permitir a paginação de resultados nos clientes *web* em terminais com menos capacidades, para evitar ou resolver problemas de falta de memória.

[14] Mecanismo que se situa entre dois sistemas, com capacidades de armazenamento, para agilizar o acesso à informação, evitando acessos desnecessários;

4.1.1.2.3. Publicação de Posts

Entre cliente e servidor um *post* (publicação) é sempre um objecto com texto apenas, sendo limitado em caracteres quanto menor for o limite mínimo de todas as redes sociais enviadas no pedido (redução ao mais fraco).

Todos os métodos do interface REST (o envio de uma publicação (ou alteração de estado) é um com exemplo) com suporte "multi-conta", recebem uma lista de contas, juntamente com o pedido e depois, na prática, o que o servidor faz é executar o pedido correspondente em todas as redes sociais pedidas pelo cliente. A figura seguinte (Imagem 3 - Visão de alto nível do "split" de um pedido) ilustra o processo genérico de estruturação de pedidos "multi-conta".

Imagem 3 - Visão de alto nível do "split" de um pedido

Sem entrar em detalhes sobre o código fonte da solução, este polimorfismo é conseguido com uma hierarquia de classes que tem como base uma rede social, virtual, que representa a agregação de conceito e a normalização para a solução apresentada.

4.1.1.2.4. Respostas e mensagens directas

Por questões técnicas (dificuldades que impossibilitaram a agregação de contactos) não foi possível permitir a resposta para outra rede, por não se poder saber se estaríamos a responder para a mesma pessoa. Assim, para responder a *posts* vindos de determinada rede, ter-se-á obrigatoriamente de ser pela mesma. O mesmo se passa para as mensagens directas, disponíveis nos clientes nos ecrãs de "amigos", em que o envio é directo para um utilizador de uma rede social, apenas.

4.1.1.2.5. Especificidades das redes

A implementação de especificidades vai contra o conceito da agregação e normalização das redes sociais. No entanto, há redes que perdem a identidade se não tiverem as suas funcionalidades exclusivas como o "like" ou os comentários do *Facebook*, ou o "*retweet*" do *Twitter*, entre outros.

Para colmatar esta lacuna, implementaram-se serviços específicos, paralelos ao conceito da normalização de todas as redes sociais, que permitam efectuar essas operações. Apesar de ter introduzido um ponto de falha na solução que choca com os valores iniciais, esta inclusão de funcionalidades torna a solução mais "comercial" e mais funcional.

4.1.1.3. Base de dados de apoio

Como já foi referido, o sistema dispõe de uma base de dados que é utilizada para várias operações: para o processo de autenticação e armazenamento de tokens de contas, para o mapeamento de MSISDN com token de acesso na plataforma (necessário para acessos usando redes sem fios, em que o operador móvel não tem como saber o MSISDN), para armazenamento de dados estatísticos como tempos de resposta das API, etc.

Anteriormente foi descrito o processo *OAuth* para obtenção de um *token* para acesso às APIs de uma rede social. O mecanismo *OAuth*, implicitamente, pressupõe acessos directos do cliente aos seus interfaces, no entanto, no caso da plataforma, após o utilizador introduzir os seus dados, este não recebe o *token* para o guardar localmente (como é hábito), este é guardado na base de dados, associando-se ao seu MSISDN. A variante do mecanismo de autenticação, modificada para a plataforma de agregação, está esquematizada na imagem seguinte (Imagem 4 - Fluxo de autenticação OAuth na plataforma).

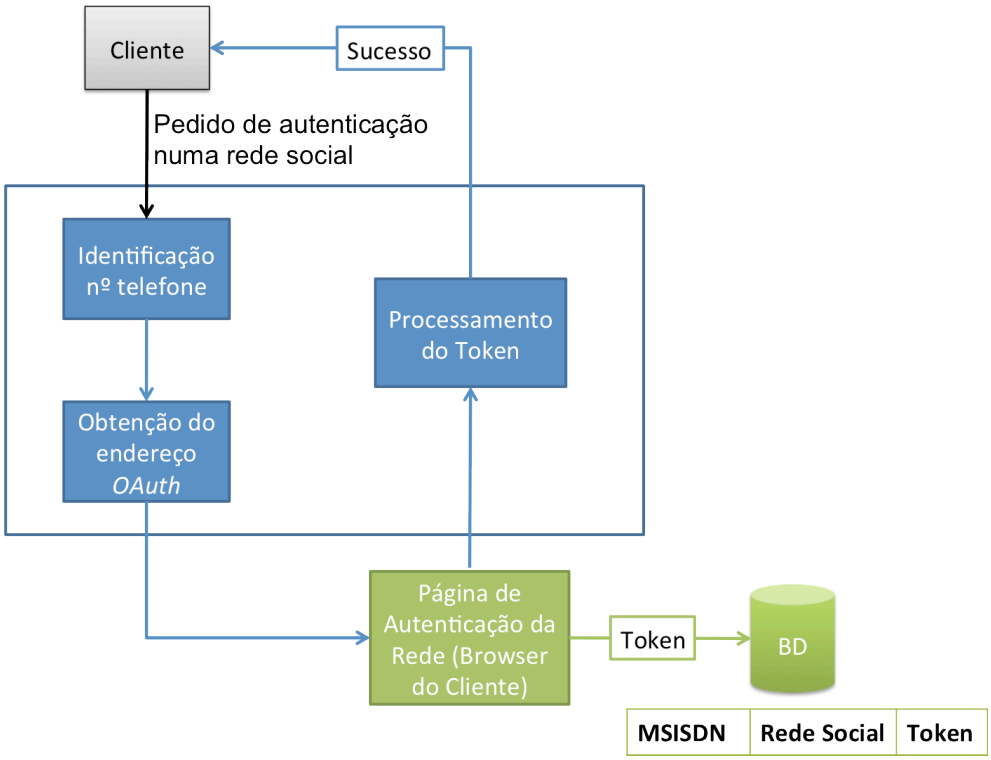

Imagem 4 - Fluxo de autenticação OAuth na plataforma

A base de dados é ainda utilizada para guardar dados temporários entre operações, uma vez que numa solução a larga escala, os servidores poderão perder a sessão devido ao balanceamento de carga entre máquinas.

4.1.1.4. Interfaces de comunicação

Como ilustrado na imagem Imagem 2 - Arquitectura de alto nível do servidor - existem vários sistemas externos a interagir com a plataforma. A comunicação com os serviços necessários para realização de operações de validação do lado do operador móvel, o acesso às APIs das diversas redes sociais e a camada responsável pela comunicação com as aplicações cliente, parte da solução.

4.1.1.4.1. Camada de integração com o operador móvel

A integração com o operador móvel e o acesso aos seus serviços, é possível através de *webservices*[15] de acesso restrito e interno, por motivos de segurança não há serviços (pelo menos não com dados considerados críticos) expostos para a Internet, assim, para integração os

[15] São componentes que permitem às aplicações enviar e receber dados em formato XML;

servidores necessitam de comunicar por um interface que os ligue em rede local, ou seja, encontrarem-se na mesma rede de servidores. Por fim, existem as APIs desenvolvidas para a plataforma e as *servlets* de autenticação e de imagens.

4.1.1.4.1.1. Autenticação

A autenticação junto do operador móvel funciona de forma semelhante ao *OAuth* (ponto 2.2.1), sendo que o nome de utilizador (MSISDN no caso do operador móvel) e palavra passe apenas seguem juntos num único pedido, em ligação segura, para serem encaminhados ao *middleware*[16] do operador móvel para validação. A resposta do serviço de autenticação é um *token* que será usado para validar a identidade do utilizador, junto da plataforma, daí em diante (este *token* fica guardado na aplicação cliente mas não tem relação com o texto da password, é perfeitamente aleatório, nem poderá ser usado noutro terminal.

4.1.1.4.1.2. Subscrição e facturação

Como referido anteriormente, as necessidades do operador móvel quando detém um produto é rentabilizá-lo. Pode também não o fazer, ou simplesmente oferecê-lo a clientes que se insiram num grupo específico.

A questão do tráfego gerado pela plataforma é alheio ao projecto, uma vez que a função do servidor é centralizá-lo num só endereço. Feito isto, é da responsabilidade do operador configurar regras na sua facturação para aquele endereço específico e controlado.

A subscrição é um caso diferente, uma vez que podem haver canais de acessos subscritos temporariamente ou mesmo limitados por funcionalidade. Assim, é necessário que a cada pedido efectuado por uma aplicação cliente, seja validado o estado de subscrição do serviço devolvendo a resposta pretendida em caso afirmativo ou devolvendo na resposta um erro e um URL[17] com uma página para o processo de compra (alheio à plataforma).

4.1.1.4.2. Camada de acesso às redes sociais

Esta camada em termos práticos tem a complexidade (de desenvolvimento) de criar um cliente, sem interface gráfico, para cada uma das redes sociais. Terá sido a camada mais extensa em termos de programação, na medida em que foi necessário implementar todas as funcionalidades da rede, o processo de autenticação (em que alguns não possuem mecanismo *OAuth*), e com o entrave da constante mudança de contexto entre redes.

[16] É um conjunto de componentes de um sistema abrangente, composto por servidores *web*, servidores aplicacionais, entre outros serviços informáticos.
[17] Acrónimo para *Uniform Resource Locator*, identifica a localização de determinado recurso e a forma de obtê-lo (ex. http://www.example.com - protocolo HTTP, endereço www.example.com);

Esta camada é a mais volátil da solução, mas é suposto ser assim. Esta camada representa o ponto de intervenção, como que a caixa de fusíveis onde se resolvem os problemas com acesso às redes ou se introduzem alterações subsequentes a reformulações das APIs das redes sociais.

4.1.1.4.3. Camada de fornecimento de serviços

Após criados os métodos em JAVA, parecendo o ponto mais sensível da aplicação, o interface REST é a componente mais automática em termos de desenvolvimento. Com bibliotecas recentes e ferramentas de desenvolvimento adequadas, todo o protocolo XML associado ao interface é gerado, de forma automática, aquando da compilação do código fonte do servidor.

Apenas quando, e se, forem necessárias alterações nesta camada, poderá haver a necessidade de reajustar e efectuar novos desenvolvimentos nas aplicações clientes.

Se no ponto anterior tínhamos o ponto mais volátil mas controlado, aqui temos o ponto mais crítico de todo o conceito da solução. É claro que os clientes também podem e devem evoluir, mas em termos conceptuais, da plataforma, o que se pretende é que sejam minimizadas essas necessidades de desenvolvimento adicional e se faça toda a manutenção, no servidor (isto obviamente num cenário ideal).

4.1.2. Tecnologias utilizadas

Como referido anteriormente, o servidor é um *Apache Tomcat 6*, com um contexto *web*, expondo um interface REST, implementando com recurso à biblioteca RESTeasy[18], todas as operações e excepções (vulgarmente chamadas de erros), são registadas em ficheiros com recurso à biblioteca log4java, que simplifica a escrita de erros para depuração e controlo. Para acesso à base de dados utilizou-se a última versão do driver ODBC[19] da oracle e a mais recente biblioteca fornecida para Java.

4.1.3. Problemas e Soluções

Durante o período de desenvolvimento do servidor e clientes (Sensivelmente entre Maio e Novembro de 2010), aconteceram várias "revoluções" em termos de mecanismos usados pelas redes sociais. Avanços e recuos na adopção o *OpenSocial*, abandono de mecanismos de autenticação simples (com envio da palavra passe em todos os pedidos) e adopção de mecanismos baseados em *token,* como o *OAuth*, levaram a restruturação de APIs e à reconstrução da camada de comunicação com as redes sociais.

Algum tempo mais tarde, houve a criação a *Graph API* do *Facebook*, impulsionada pela nova arquitectura do sistema, que reformulou praticamente 100% as suas APIs. As APIs antigas forma descontinuadas entretanto.

A solução para essas questões, passou por implementar novamente alguns módulos da camada de acesso, foi penoso, mas confirmou a tese de que a agregação de servidor tem vantagens claras quanto à manutenção do código.

Outra questão que surgiu aquando da implementação dos clientes móveis foi o tamanho e formato das imagens e fotos enviadas pelas redes sociais, que foi solucionado com a inclusão, na *servlet* de imagens já existente (para centralização de tráfego), de mecanismos de redimensionamento e conversão de imagens.

Houve várias dificuldades técnicas que foram sendo ultrapassadas, mas que vieram a reiterar que as redes sociais se baseiam em plataformas muito voláteis e que a agregação em servidor é uma boa solução.

[18] É um projecto que engloba várias ferramenta para acelerar o desenvolvimento de aplicações RESTful em linguagem Java, segue todas as especificações actualmente utilizadas e está disponível em: http://www.jboss.org/resteasy;
[19] Acrónimo para *Open Database Connectivity*, trata-se de um interface normalizado de acesso a sistemas de gestão de bases de dados;

4.2. Clientes

Como exemplo de aplicações baseadas na plataforma, foram desenvolvidas duas versões que demonstram as suas capacidades. Desenvolveu-se uma aplicação para iPhone e uma versão para Opera *widgets*, que pode ser executada no computador pessoal ou em dispositivos móveis que tenham um gestor de *widgets*.

No desenvolvimento da aplicação iPhone, usou-se e testou-se o acesso usando *tags* de XML, à camada REST, ao passo que, na widget, se usou a sintaxe jSON.

Os clientes foram, também, implementados seguindo arquitectura em camadas (MVC). Uma de comunicação com o interface REST da plataforma de agregação, outra com a lógica e funções operacionais e uma que se materializa na interface gráfica das aplicações.

De seguida apresentam-se sinteticamente e de forma ilustrada as funcionalidades já descritas ao longo do trabalho, recorrendo a *screenshots* [20]das versões 100% funcionais das aplicações desenvolvidas, assentes na plataforma.

Os ecrãs apresentados servem apenas para demonstrar as capacidades da plataforma e as possibilidades de implementação de clientes.

[20] Captura de uma região ou de toda a área visível num ecrã de um dispositivo;

4.2.1. Funcionamento / Pontos comuns

Ambas as aplicações se tocam em funcionamento, aproximando-se da mesma experiência de utilização em ambos os casos, apenas variando a forma de interacção e o aspecto gráfico.

Imagem 5 – Cliente iPhone – Splash screen

4.2.1.1. Autorização junto da plataforma

Para concretizar a agregação de contas e sua associação a um assinante da rede móvel, é necessário saber o seu número de telefone, essa é a chave primária [21]do acesso aos *tokens* (na plataforma). Existem dois métodos possíveis de autenticação junto da plataforma de agregação, dependendo da forma de acesso à rede que o terminal está a utilizar.

4.2.1.1.1. Por MSISDN

No caso em que as aplicações acedem à Internet usando as ligações de dados móveis fornecidas pelo operador, a plataforma, por intermédio dos serviços deste, consegue determinar o número de telemóvel de quem está a aceder e daí garantir o funcionamento da aplicação.

[21] É um valor que pode ser usado como índice por nunca se repetir e existir sempre;

4.2.1.1.2. Com credenciais do operador (Login e Password)

No caso de acesso à Internet por redes externas ao operador móvel, uma qualquer rede sem fios por exemplo, esse cenário pressupõe uma forma de autenticação baseada em nome de utilizador e palavra passe, do registo *web* do cliente junto do operador.

As credenciais são encriptadas e enviadas o *token* gerado e o numero de telefone aceite e registado na plataforma de agregação. A partir desse momento, a plataforma, recebe o *token* guardado no cliente e, com ele, permite obter o MSISDN e funcionar de igual forma à autenticação através de "reconhecimento automático". As imagens seguintes ilustram o processo no cliente para iPhone, no cenário em que este acede usando uma rede *Wi-Fi*[22].

Imagem 6 – Cliente iPhone – Autenticação via Wi-Fi

Imagem 7 – Cliente iPhone – Sucesso após autenticação via Wi-Fi

O ecrã de configuração de conta no operador, apenas surge caso não seja possível a identificação automática do MSISDN.

[22] é uma marca registrada da Wi-Fi Alliance, que é utilizada por produtos certificados que pertencem à classe de dispositivos de rede local sem fios;

4.2.1.2. Contas de redes sociais associadas e activas

Assumindo que o cliente com o MSISDN já tem contas configuradas na plataforma, como nas imagens seguintes. Estas podem ser activadas ou desactivadas, sem as remover da plataforma, apenas para que não surjam aquando da obtenção de informação vinda do servidor.

Imagem 8 – Cliente iPhone – Configuração de contas

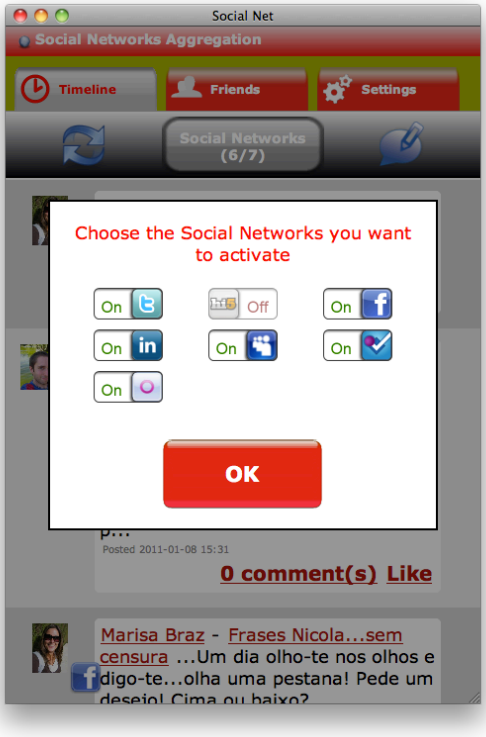

Imagem 9 – Cliente Opera widget – Filtragem de contas apresentadas na *timeline*

4.2.1.3. Configuração de uma conta de uma rede social

No caso de o utilizador ainda não ter associado nenhuma conta, ou se tiver removido e quiser adicionar outra, pode fazê-lo acedendo nas definições da aplicação a um ecrã que se faz valer da *servlet* de autenticação do servidor (ver Imagem 2 - Arquitectura de alto nível do servidor), para todo o processo. O fluxo normal da adição de uma conta é ilustrado nas três imagens seguintes.

Imagem 10 – Cliente iPhone – Acrescentar rede social OAuth – obter URL da página de autenticação

Imagem 11 – Cliente iPhone – Acrescentar rede social OAuth – introduzir credenciais

Ao carregar no botão adicionar conta, é solicitado à *servlet* que redireccione o pedido para o endereço de autenticação *OAuth* da rede social em questão, abrindo-o num browser embutido no ecrã da aplicação. A página exibida é da exclusiva responsabilidade do fornecedor da rede social.

Assim, após introduzir as credenciais, numa página segura, propriedade da rede social, há um redireccionamento um "*callback*", que responde de volta para a plataforma, que "sabe" quem iniciou a autenticação e armazena o *token* que vem na resposta na sua base de dados, para a rede escolhida, para o MSISDN em questão.

A conta em questão está pronta a usar em qualquer dos clientes, uma vez que fica configurada no servidor.

Imagem 12 – Cliente iPhone – Configurações de conta de rede social (activar/desactivar)

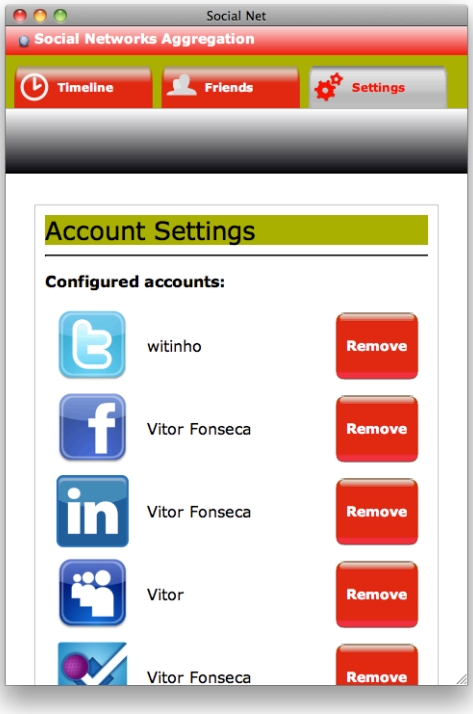

Imagem 13 – Cliente Opera widget – Lista de contas configuradas

No caso da aplicação para iPhone, os botões de ligar ou desligar a rede social (localmente na aplicação) estão acessíveis dentro da configuração de cada uma delas. No caso da *widget*, é possível escolher durante a actualização dos dados da *timeline*[23], da lista de amigos ou da publicação de mensagens.

Outra possibilidade de configuração é a remoção da rede social, que pressupõe desassociar a conta da plataforma definitivamente. Naturalmente, remover uma conta, afecta todos os clientes, pois servidor.

[23] A *timeline* é a lista ordenada cronologicamente, de *posts* recebidos das redes sociais configuradas.

4.2.1.4. Timeline única

A plataforma, depois de obter junto de todas as APIs das redes sociais configuradas, devolve uma lista ordenada cronologicamente que intercala numa só vista todas as mensagens oriundas das diversas redes. Há indicação em cada *post* da rede a que se refere e são apresentadas as opções de interacção comuns e/ou especificas dessa rede.

As imagens seguintes ilustram o aspecto do ecrã em ambas as aplicações.

Imagem 14 – Cliente iPhone – Timeline

Imagem 15 – Cliente Opera widget – Timeline (a actualizar)

4.2.1.5. Lista e pesquisa de amigos

A lista de amigos segue a mesma linha de funcionamento que a *timeline*, com a particularidade de permitir fazer pesquisas (Introduzindo parte do nome de utilizador e obtendo de resposta uma lista com as ocorrências).

Da mesma forma que na *timeline*, é identificada a proveniência do contacto e as possibilidades de interacção que a plataforma permite efectuar.

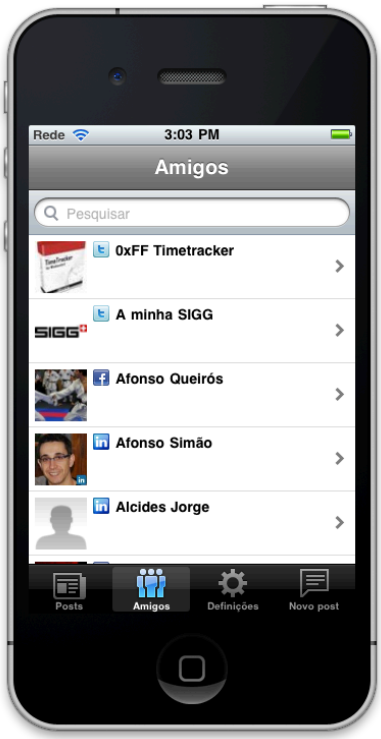

Imagem 16 – Cliente iPhone – Lista de amigos (com pesquisa)

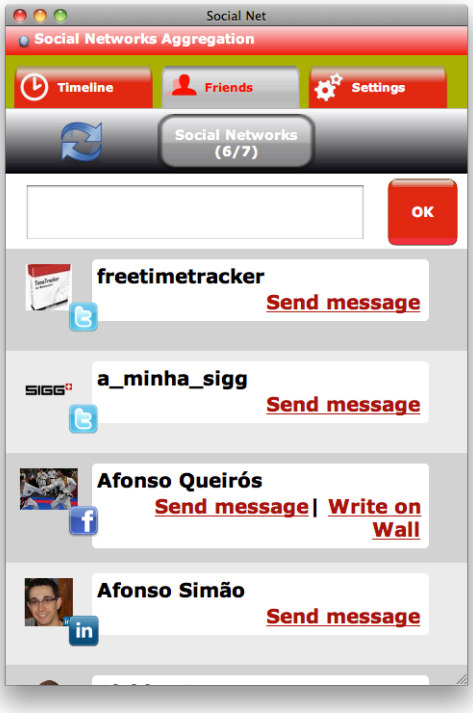

Imagem 17 – Cliente Opera widget – Lista de amigos (com pesquisa)

No caso concreto do Facebook, é dada a possibilidade de escrever no "mural" ou de enviar uma mensagem directa ao amigo. É um exemplo de uma funcionalidade específica de uma rede social, salvaguardada nas funcionalidades da aplicação.

4.2.1.6. Publicação de estado

A principal utilização para uma aplicação para acesso a redes sociais, quando em mobilidade, é a partilha rápida de pequenos textos ou simples alteração de estado. Como na *timeline* e lista de amigos, ambas as aplicações permitem escolher, para um só envio, quais as redes sociais para as quais a mensagem deve ser difundida.

Imagem 18 – Cliente iPhone – Difusão de uma publicação

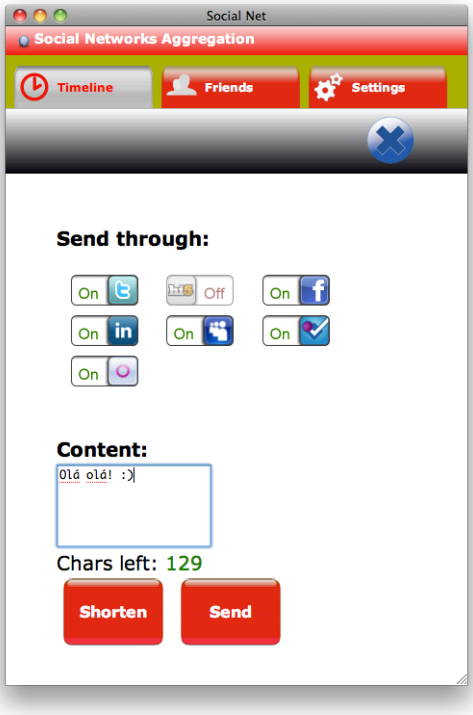

Imagem 19 – Cliente Opera widget – Difusão de uma publicação

Nota: Ao publicar uma mensagem para várias redes, o limite de texto é restringido à "mais fraca" de cada uma delas, de forma a garantir que o conteúdo difundido é publicado de forma igual em todas elas.

4.2.1.7. Mensagens directas

A mensagem directa é suportada em algumas redes e é uma variante dos ecrãs ilustrados nas imagens acima, mas com um destinatário escolhido da lista de amigos. A mensagem enviada pode, ou não, ser privada.

4.2.1.8. Funcionalidades específicas

Foram implementadas algumas funcionalidades específicas, pelos motivos já referidos (o facto de serem importantes para a identidade da rede social), que estão ilustradas e descritas, brevemente, de seguida.

Imagem 20 – Cliente iPhone – Publicação do Facebook com comentários

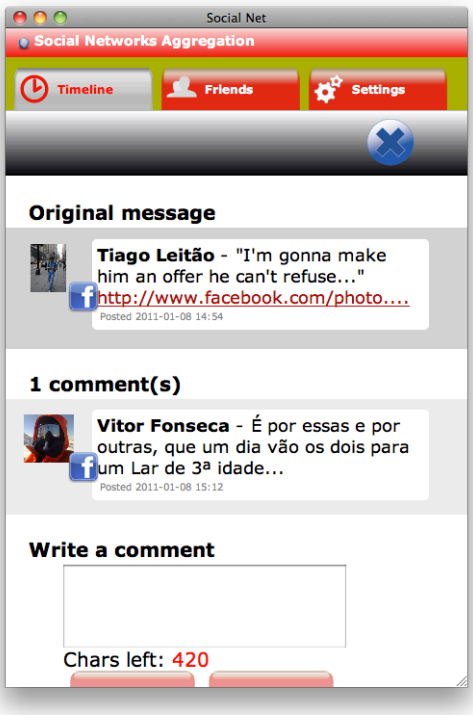

Imagem 21 – Cliente Opera widget – Publicação do Facebook com comentários

No caso da aplicação para iPhone, a página de detalhes de uma publicação permite fazer o "*like*" e ver os comentários já existentes (que são listados imediatamente). O utilizador pode acrescentar o seu próprio comentário.

No caso da *widget*, o "*like*" é suportado directamente no ecrã da *timeline*, como ilustrado na Imagem 15 – Cliente Opera widget – Timeline (a actualizar) – onde também se acede à página de comentários ilustrada acima. A colocação de comentários é feita directamente no mesmo ecrã.

Outra funcionalidade de relevo no caso, por exemplo, do Facebook é a exibição dos dados de perfil de um amigo. Não foi implementado suporte para alteração destes campos na plataforma,

é unicamente um ecrã informativo, apresentando, no entanto, botões com as funcionalidades de mensagem directa (transversal na plataforma) e escrita no "mural" de outro utilizador. A figura seguinte ilustra um perfil do *Facebook* com algumas informações.

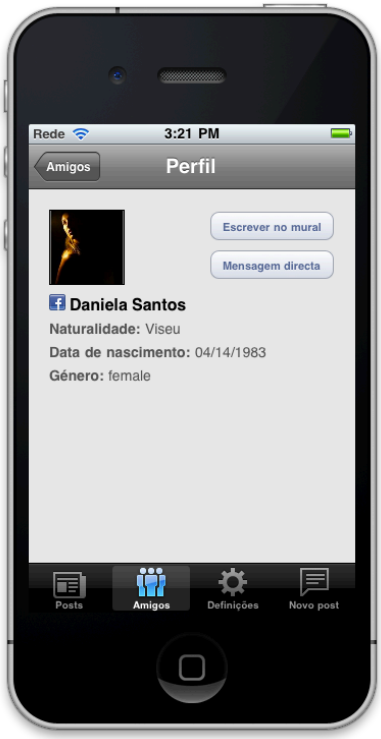

Imagem 22 – Cliente iPhone – Informações de perfil (Facebook)

Imagem 23 – Cliente iPhone – detalhes de um "tweet"

Nas mesmas condições de especificidade, foi implementada a funcionalidade de "retweet" de uma mensagem proveniente do Twitter. É uma funcionalidade exclusiva desta rede social, sendo no entanto preponderante que esta exista.

A introdução destas especificidades reduz a normalização das aplicações clientes, mas torna-a bastante mais robusta e apelativa aos utilizadores das redes sociais, que as conhecem e usam as suas funcionalidades mais conhecidas, com regularidade.

Capítulo 5

5. Análise do sistema

Em termos funcionais, os capítulos anteriores permitem concluir que as vantagens na implementação da plataforma foram bastante positivas e as suas capacidades, apesar de mínimalistas, permitem resolver o problema proposto.

Em termos de disponibilidade do sistema foi necessário desenvolver um banco de testes em que se pudesse analisar os tempos de resposta da API implementada, para compreender se se estaria, ou não, a introduzir um "bottle neck" no acesso às APIs das redes sociais. Foram realizados testes recorrendo ao JMeter[24] , simulando um número crescente de acessos "simultâneos" à API do servidor criado. As imagens seguintes são gráficos gerados pela ferramenta, que demonstram uma eficiência razoável na resposta a centenas de pedidos em simultâneo, validando a solução para um cenário mais realista (apesar de realizados num servidor instalado num computador pessoal de características comuns).

A imagem seguinte representa o número de pedidos a que a API consegue responder, em função de atrasos provocados em cascata pelas APIs das redes sociais. Isto é, por exemplo, quando o tempo de resposta das redes sociais é de aproximadamente 0ms, teríamos capacidade para responder a 350 pedidos, com um atraso de 500ms nas redes, já só seria possível responder a 100, etc., em simultâneo.

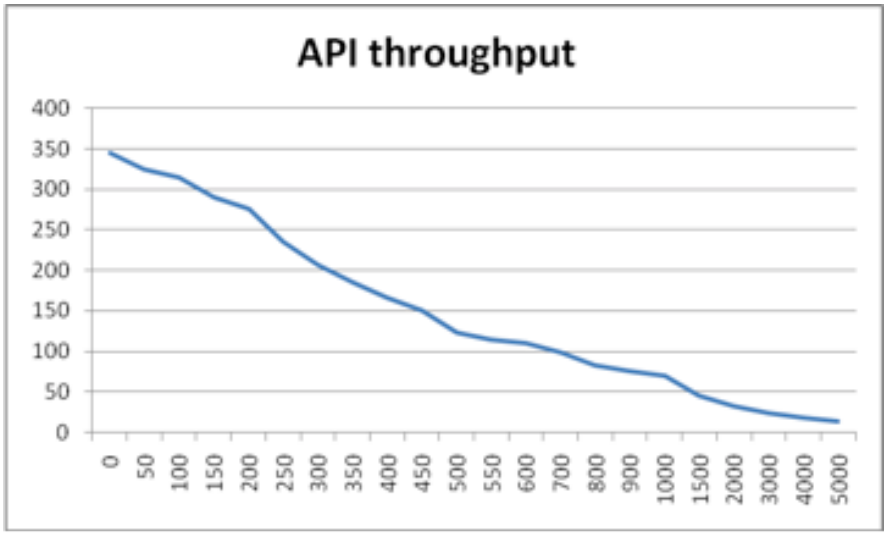

Imagem 24 – Capacidade de resposta da API da plataforma face à resposta das APIs das redes sociais

Durante o processamento dos pedidos, o CPU da máquina oscilou entre 70% e 90%, não restringindo os tempos de resposta. Estes tempos apenas foram afectados por "atrasos" relacionados com tempos de respostas na rede sociais e relacionados com latência na rede.

[24] Ferramenta utilizada para efectuar testes de carga em sistemas informáticos;

O comportamento do servidor foi bastante consistente, num período de testes de cerca de uma hora em que foram efectuados cerca de 80000 pedidos à API.

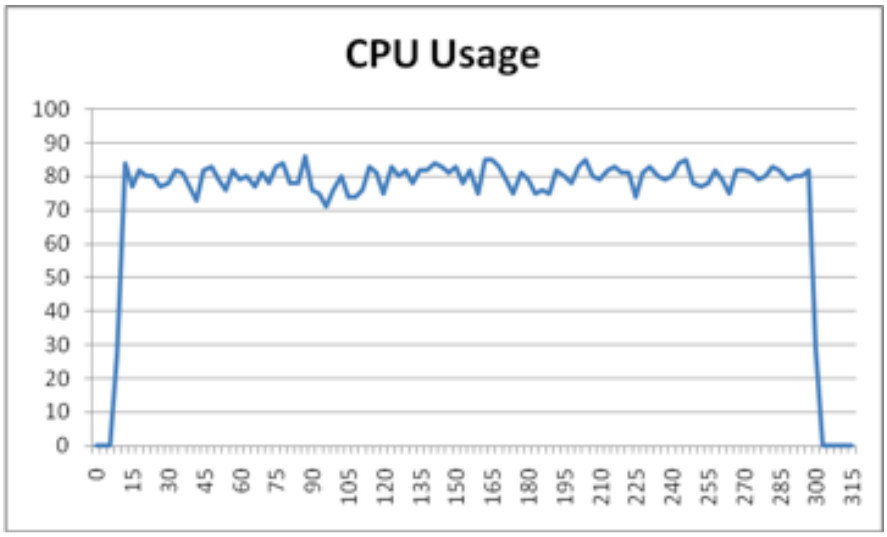

Imagem 25 - Utilização do CPU durante o processamento dos pedidos

O servidor foi configurado para alocar apenas até 256Mb de memória RAM, revelando-se um valor aceitável para o volume gerado, uma vez que usando Java o seu mecanismo de "garbage colection", alocava objectos até atingir cerca de 95% da memória disponível e depois libertava objectos desnecessários, caindo para cerda de 20% de utilização (ver imagem seguinte).

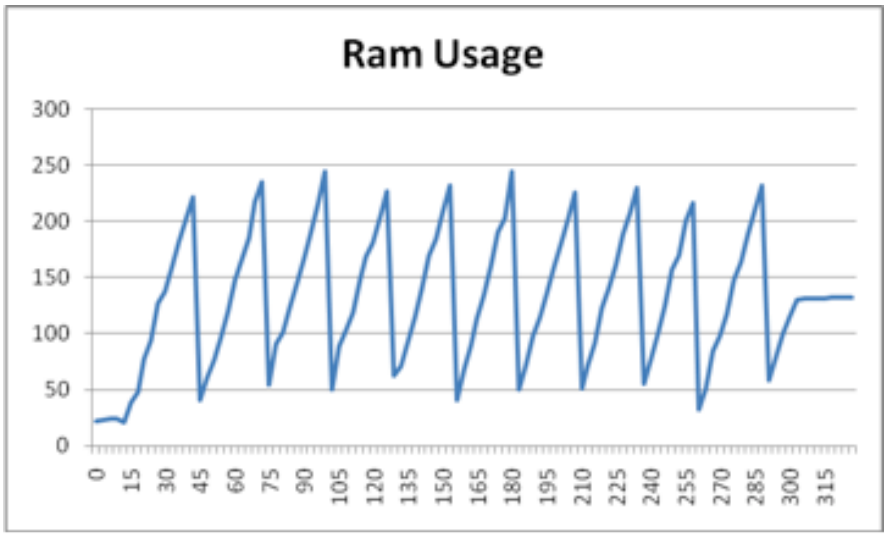

Imagem 26 - Utilização da memória RAM durante o processamento dos pedidos

Pode assim concluir-se que o sistema tem capacidade de resposta mesmo quando submetido a elevada carga. Com melhores máquinas e usadas em simultâneo (load balancing), a plataforma poderia suportar um cenário real, com acesso público aos clientes do operador móvel.

Capítulo 6

6. Conclusões

Conforme referido na introdução, pode voltar a reiterar-se neste capítulo, de conclusão, o quão dinâmicas e voláteis são os sistemas da maioria das redes sociais.

Sendo a maioria das redes de sociais empresas valiosíssimas pela sua colocação no mercado, apresentam um modelo de angariação de utilizadores quase viral e se tiverem funcionalidades apelativas, as pessoas vão aderir. Por outro lado ninguém paga para as utilizar e como tal não há aqui uma legitimidade intrínseca para reclamar com o serviço. Obviamente não falo dos utilizadores comuns da rede social, falo sim dos programadores que para elas desenvolvem, que a maior parte das vezes não são "tidos nem achados" aquando de mudanças radicais, e por vezes sem aviso prévio, de toda a estrutura em que basearam as suas aplicações. Mas, se querem continuar a rentabilizar ou simplesmente preservar o funcionamento destas, têm que ir atrás dessa evolução (ou descontinuá-las), adaptando-se vezes sem conta.

A plataforma de agregação de redes sociais desenvolvida no âmbito deste projecto é essencialmente destinada a minimizar este impacto e reduzir algum deste, ingrato, trabalho (que é refazer a mesma coisa várias vezes), ao sabor das alterações impostas por fornecedores de serviços em redes sociais, com monopólio e com milhares de pessoas nos seus quadros, pagos para terem ideias e para as implementarem todos os dias.

Em termos de serviço, a agregação é bastante prática para o utilizador final, mas também o seria se fossem criados clientes de agregação directamente. Aqui o trunfo é a plataforma centralizar e permitir colmatar alguns dos pontos de falha das aplicações "*stand alone*", bem como permitir que se detenha um lote enorme de clientes (exemplos: box de TV por cabo, SMS, aplicação Java, aplicação *Symbian*[25], etc.) e não seja necessário refazer "N" clientes em caso de mudanças nas APIs das redes sociais suportadas (e em cinco meses, como exemplo, houve três alterações críticas, só no *Facebook*).

Fazer um sistema que dependa de uma organização externa, a que não temos acesso directo, é bastante complicado. Fazer um sistema que dependa de, neste caso, sete organizações externas, em que cada uma conta com largas centenas de funcionários a trabalharem incessantemente é obviamente um processo iterativo e cheio de surpresas (dificuldades) recorrentes. Assim, ter

[25] Linguagem de programação e sistema operativo (Symbian OS) criado pela empresa Nokia para os seus telefones, que tem evoluído constantemente e, ainda, é usado em muitos terminais actualmente;

uma plataforma que, no mínimo, agilize o processo de manutenção da solução é, por si só, motivo suficiente para justificar a sua existência.

O aumento do número de redes suportadas na plataforma não é por si só limitador em termos de *performance* da plataforma, uma vez que o sistema foi tecnologicamente desenvolvido para suportar vários pedidos em paralelo e assíncronos. No entanto, em termos de resposta aos pedidos efectuados, a plataforma apenas pode devolver a resposta, aos clientes, quando a obtém de todas as redes sociais. Ou seja, introduzir uma rede de baixa disponibilidade na camada de integração, irá reduzir a prestação do servidor a valores mínimos impostos pela mais fraca, em disponibilidade e *performance*, rede social.

No prisma do operador móvel, a plataforma torna-se ainda mais interessante porque abre portas à criação de novos serviços que nela se possam basear. Ao usar a plataforma e ao tê-la disponível para todos os seus clientes e segmentos, conseguem idealizar novas funcionalidades de uma forma simplificada, de desenvolvimento rápido e sem expor os seus clientes a riscos desnecessários relacionados com o acesso às redes, podendo ainda aproveitar para reunir novos dados sobre os seus clientes, estudar tendências e formas de comunicação, para com eles, mais adaptadas à sociedade actual.

Todos os dias se assiste à introdução/"intromissão" das redes sociais em serviços e no quotidiano das pessoas, e é certo que o crescimento do fenómeno das redes sociais é exponencial e vai continuar a sê-lo no futuro, com novos conceitos e evoluções na forma de comunicar, é importante que se mantenha a mente aberta e a vontade necessária para não boicotar a evolução, adaptando-se rapidamente.

O mais importante na vida são as pessoas e os relacionamentos que estas desenvolvem entre si. Na sociedade digital em que vivemos, os valores de um relacionamento humano estará, para alguns, desvirtuado com as redes sociais, que os banalizam, sendo que para outros é apenas um complemento e uma forma rápida de estar em contacto. A tecnologia é feita de pessoas, e é para essas pessoas que este género de soluções trará valor acrescentado, às suas vidas.

Bibliografia

Zakas, N. C. (2005). *Professional Javascript for Web dev ISBN: 978-0764579080.* Wrox.

Zdziarski, J. *Iphone – Open Application Development. ISBN: 978-0596-51855-4.* 2008: O'Reilly.

Wikipedia. (s.d.). *Facebook*. Obtido em 20 de Dezembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/Facebook

Wikipedia. (s.d.). *Foursquare*. Obtido em 15 de Novembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/Foursquare

Wikipedia. (s.d.). *Hi5*. Obtido em 21 de Dezembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/Hi5

Wikipedia. (s.d.). *LinkedIn*. Obtido em 21 de Dezembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/LinkedIn

Wikipedia. (s.d.). *MySpace*. Obtido em 20 de Dezembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/MySpace

Wikipedia. (s.d.). *OAuth*. Obtido em 20 de Novembro de 2010, de Wikipédia, a enciclopédia livre: http://en.wikipedia.org/wiki/OAuth

Wikipedia. (s.d.). *Orkut*. Obtido em 12 de Novembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/Orkut

Wikipedia. (s.d.). *Social network aggregation*. Obtido em 19 de Novembro de 2010, de Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Social_network_aggregation

Wikipedia. (s.d.). *REST*. Obtido em 21 de Novembro de 2010, de Wikipédia, a enciclopédia livre: http://pt.wikipedia.org/wiki/REST

Wikipedia. (s.d.). *Twitter*. Obtido em 21 de Dezembro de 2010, de http://pt.wikipedia.org/wiki/Twitter

Alexa. (s.d.). *Alexa*. Obtido em 14 de Novembro de 2010, de The web information company: http://www.alexa.com

Android.com. (s.d.). *Android developers*. Obtido em 28 de Dezembro de 2010, de Platform Versions - Current distribution: http://developer.android.com/resources/dashboard/platform-versions.html

Blueoxen. (s.d.). *Wiki Rest*. Obtido em 2010 de Novembro de 2010, de Rest: http://rest.blueoxen.net/cgi-bin/wiki.pl

Bourke, T. (2001). *Server Load Balancing. ISBN: 978-0596000509*. O'reilly.

Chappell, D. A. (2002). *Java Web Services. ISBN: 0-596-00269-6*. O'Reilly.

Facebook. (s.d.). *Facebook*. Obtido em 17 de Novembro de 2010, de Press room - statistics: http://www.facebook.com/press/info.php?statistics

Facebook. (s.d.). *Facebook Developers*. Obtido em Setembro de 2010, de Graph API: http://developers.facebook.com/docs/api/

Foursquare. (s.d.). *Foursquare Developers*. Obtido em Outubro de 2010, de Foursquare APIv2: http://developer.foursquare.com/docs/

Google. (s.d.). *Google code*. Obtido em Outubro de 2010, de Orkut Developer Home: http://code.google.com/apis/orkut/

Google. (s.d.). *OpenSocial*. Obtido em 20 de Novembro de 2010, de Google Code: http://code.google.com/apis/opensocial/

Hi5. (s.d.). *H5 Developer platform*. Obtido em Outubro de 2010, de Hi5 API: http://api.hi5.com/

LinkedIn. (s.d.). *LinkedIn - Developer Network*. Obtido em Novembro de 2010, de APIs: http://developer.linkedin.com/community/apis

loadbalancing.org. (s.d.). *Load Balancing FAQ*. Obtido em 10 de Novembro de 2010, de Load Balancing: http://www.loadbalancing.org/

Muiomuio. (s.d.). Obtido em Janeiro de 2011, de Estatísticas e factos do Facebook em 2010: http://muiomuio.net/estatisticas-e-factos-do-facebook-em-2010/

MySpace. (s.d.). *MySpace Developer Plataform*. Obtido em Novembro de 2010, de RESTful API Overview: http://wiki.developer.myspace.com/index.php?title=Category:RESTful_API

Macclure, W. b.-0. (2010). *Professional Iphone programming wit. ISBN: 978-0470637821*. Wrox.

Marketingtecnologico.com. (s.d.). *Marketing Tecnológico*. Obtido em Janeiro de 2011, de As surpreendentes estatisticas sobre redes sociais: http://www.marketingtecnologico.com/marketingtecnologico/artigos/default.asp?id=260

Mathai, J. (s.d.). *Home*. Obtido em 15 de Novembro de 2010, de Hacker::getInstance()->Articles();: http://www.jaisenmathai.com/articles/twitter-php-oauth.html

Psicologia Digital. (s.d.). *A Psicologia dos Relacionamentos na Internet*. Obtido em 18 de Novembro de 2010, de Psicologia Digital: http://www.psicologiadigital.com/2009/09/13/a-psicologia-dos-relacionamentos-na-internet/

PT.WEBAPPSEC. (s.d.). *A Privacidade e a Segurança nas Redes Sociais*. Obtido em 10 de Dezembro de 2010, de Segurança de Aplicações Web: http://webappsec.netmust.eu/2010/02/06/a-privacidade-e-a-seguranca-nas-redes-sociais/

Twitter. (s.d.). *Twitter Developers*. Obtido em Setembro de 2010, de API Documentation: http://dev.twitter.com/doc

Twitter. (24 de Novembro de 2010). *Transitioning from Basic Auth to OAuth*. Obtido de Twitter developers: http://dev.twitter.com/pages/basic_to_oauth

Anexos

A.1 Protocolo REST – Especificação da API

Este anexo descreve o protocolo de comunicação entre clientes e servidor, apresentando os métodos disponíveis e a estrutura de pedidos e respostas ao servidor.

Com base na API criada no âmbito do projecto, e com auxilio à documentação presente no documento de especificação, seria possível a terceiros (programadores independentes) desenvolver aplicações cliente utilizando a plataforma de agregação.

Social Networks Aggregation

REST Interface - API Specification

Summary

1. Pre-set parameters

The fields of all the REST Interface calls described below are treated as a java String. However, in some cases, they can only assume pre-determined values to be correctly parsed. For these fields, if an unauthorized value is sent (or the field is missing) the UNKNOWN value is assumed, which can turn the required option impossible to execute (e.g., for the account type).

All values are **case-insensitive**. The fields expecting pre-determined values are:

- **applicationType** (mapped as <type> on POST rquests)
  - WIDGET
  - WAP
  - IPHONE
  - …
  - UNKNOWN
- **accountType** (mapped as <type> inside <account> fields on POST requests)
  - TWITTER
  - FACEBOOK
  - HI5
  - LINKEDIN
  - MYSPACE
  - FOURSQUARE
  - ORKUT
  - UNKNOWN

In the specification for some POST calls to the interface clients have the option to pass multiple accounts to perform the same operation in all of them. In GET calls only one account per request is allowed.

To avoid the repetition of the same description for all the methods, the **account** parameter was included. In fact, the interface is not expecting any parameter with this name. The account information is mapped in the following parameters:

- **name** *(optional) account name*
- **accountId** *(optional) account username or ID*
- **accountType** *(mandatory) as described above*
- **key** *(mandatory) account key*
- **secret** *(mandatory, if applicable) account key*

2. Authentication

To authenticate in the social networks, simply call

http://server/context/Auth?snName=<SocialNetworkName>[&callback=http://desiredCallback.com][&userToken=yourUserToken][&failureURL=http://callbackWhenAuthFails.com]

and follow the on-screen indications.

The parameter **snName** is mandatory and represents the Social Network in which the caller wants to authenticate

The parameters **callback** and **userToken** are optional and their purpose is:

-callback: if the caller wants to be redirected to a specific website when the authentication is sucessful, he should pass it as the value of this parameter; if this parameter is not specified, caller will be redirected to a page in the proxy which states that authentication is complete;

-failureURL: similar to callback, but to the case where the caller wants to be redirected to a specific URL when authentication fails;

-userToken: the caller must either pass the userToken parameter or to be in a network which has a gateway that translates the caller's IP to a MSISDN; userToken or MSISDN are used to authentication purposes;

Example:

http://serverurl/socialnet/Auth?snName=Twitter

http://serverurl/socialnet/Auth?snName=Facebook?callback=http://my.site.com/authOk&failureURL=http://my.site.com/authFail

http://serverurl/socialnet/Auth?snName=LinkedIn&userToken=ABC123

Available Social Networks:

- Twitter
- Facebook
- LinkedIn
- MySpace
- FourSquare

When authentication is complete, call list accounts (refer to 3.1) to get account details.

3. Account Management

3.1. ListAccounts

Call URL

/rest/xml/acc/list

/rest/json/acc/list

@GET Request

- userToken *(optional: if specified, will list accounts by user token; otherwise will list accounts by authenticated user's MSISDN)*
- *applicationType* *(mandatory)*

@POST Request

```
<ListProxyAccountsRequest>
<userToken>user token here</userToken>
<type>WAP</type>
</ListProxyAccountsRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<ListProxyAccountsResponse>
    <message>OK</message>
    <status>0</status>
    <accounts>
        <account>
            <name>My Hi5 Name</name>
            <id>1223456789</id>
            <username>userInHi5</username>
            <type>Hi5</type>
            <key>authKeyHi5</key>
            <secret>authSecretHi5</secret>
           <bthdySmsStatus>1</bthdySmsStatus>
        </account>
        <account>
             <name>My LinkedIn Name</name>
             <id>56565458</id>
             <username>userInLinkedIn</username>
             <type>Linkedin</type>
              <key>authKeyLinkedIn</key>
              <secret>authSecretLinkedIn</secret>
             <bthdySmsStatus>0</bthdySmsStatus>
        </account>
    </accounts>
</ListProxyAccountsResponse>
```

Response (JSON):

```
{"status":"0",
 "message":"OK",
```

```
"accounts":[
   {"name":"My Hi5 Name",
    "id":"1223456789",
    "username":"userInHi5",
    "type":"Hi5",
    "key":"authKeyHi5",
    "secret":"authSecretHi5",
    "bthdySmsStatus":"1"},
   {"name":"My LinkedIn Name",
    "id":"56565458",
    "username":" userInLinkedIn",
    "type":"Linkedin",
    "key":"authKeyLinkedIn",
    "secret":"authSecretHi5",
    "bthdySmsStatus":"0"}]}
```

3.2. AddAccount

Call URL

/rest/xml/acc/add

/rest/json/acc/add

@GET Request

- name *(mandatory)*
- accountId *(mandatory)*
- accountType *(mandatory)*
- key *(optional)*
- secret *(optional)*
- userToken *(optional)*
- applicationType *(mandatory)*

@POST Request

```
<AddAccountRequest>
<account>
        <name>username here</name>
        <accountId>account id here</accountId>
        <type>Linkedin</type>
        <key>user token here</key>
        <secret>user token here</secret>
</account>
<userToken>user token here</userToken>
<applicationType>WIDGET</applicationType>
</AddAccountRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<Response>
    <message>OK</message>
    <status>0</status>
</Response>
```

Response (JSON):

```
{"status":"0",
 "message":"OK"}
```

3.3. RemoveAccount

Call URL

/rest/xml/acc/remove

/rest/json/acc/remove

@GET Request

- name *(mandatory)*
- accountId *(mandatory)*
- accountType *(mandatory)*
- key *(optional)*
- secret *(optional)*
- userToken *(optional)*
- applicationType *(mandatory)*

@POST Request

```
<RemoveProxyAccountRequest>
<accounts>
 <account>
  <name>username here</name>
  <accountId>account id here</accountId>
  <type>Linkedin</type>
  <key>user token here</key>
  <secret>user token here</secret>
 </account>
…
</accounts>
<msisdn>912345678</msisdn>
<userToken>user token here</userToken>
<applicationType>WIDGET</applicationType>
</RemoveProxyAccountRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<Response>
    <message>OK</message>
    <status>0</status>
</Response>
```

Response (JSON):

```
{"status":"0",
 "message":"OK"}
```

3.4. MobileOperatorAuthenticate

Call URL

/rest/xml/acc/authenticate

/rest/json/acc/authenticate

@GET Request

- username *(mandatory: the account username)*
- password *(mandatory: the account password)*
- applicationType *(mandatory)*

@POST Request

```
<MobileOperatorAuthenticateRequest>
<username>xpto</username>
<password>xpto</password>
<type> WIDGET</type>
</VMobileOperatorAuthenticateRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<MobileOperatorAuthenticateResponse>
    <message>OK</message>
    <status>0</status>
    <userToken>userTokenHere</userToken>
</MobileOperatorAuthenticateResponse>
```

Response (JSON):

```
{"status":"0","message":"OK","userToken":"userTokenHere"}
```

4. Shared calls

4.1. GetTimeline

Call URL

/rest/xml/getTimeline

/rest/json/getTimeline

@GET Request

- start *(optional: default value is 0)*
- count *(optional: default value is 10)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<GetTimelineRequest>
<start>0</start>
<count>10</count>
<accounts>
<account>
<name></name>
<id></id>
<type></type>
<key></key>
<secret></secret>
</account>
<account>...</account>
</accounts>
<type>WIDGET</type>
</GetTimelineRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<GetTimelineResponse>
    <message>OK</message>
    <status>0</status>
    <updates>
        <update>
            <text> text is here</text>
            <date>2010-07-08T12:29:14+01:00</date>
           <formattedDate>Postado em Mon, 08 de Jul de 2010</formattedDate>
            <name>Jogn Doe</name>
            <userId>111222333</userId>
            <id>123456789</id>
            <imageURL>http://image.com</imageURL>
            <shortImageURL>http://image.com</shortImageURL>
            <type>FACEBOOK</type>
            <retweeted>true</retweeted>
            <retweetId>0</retweetId>
            <likeCount>0</likeCount>
            <commentCount>0</commentCount>
            <targetUser>Joe</targetUser>
```

```
            <targetUserId>0</targetUserId>
            <liked>false</liked>
        </update>
        <update>(…)</update>
    </updates>
</GetTimelineResponse>
```

Response (JSON):

```
{"status":"0",
 "message":"OK",
 "updates":[
    {"text":"text is here",
     "date":"Thu Jul 08 12:29:14 WEST 2010",
     "formattedDate":"Postado em Mon, 08 de Jul de 2010",
     "name":"John Doe",
     "userId":"111222333",
     "id":"123456789",
     "imageURL":"http://image.com",
     "shortImageURL":"http://image.com",
     "type":"FACEBOOK",
     "likeCount":"0",
     "commentCount":"0",
     "liked":"false",
     "fullUpdate":
        {"to":[],
         "message":"message here",
         "link":"http://www.facebook.com/",
         "icon":"http://icon.com",
         "type":"link"}},
    {"text":"other text",
     "date":"Wed Jul 07 17:09:08 WEST 2010",
     "name":"Mister John",
     "userId":"22233344",
     "id":"987654321",
     "imageURL":"http://otherimage.com",
     "type":"FACEBOOK",
     "likeCount":"0",
     "commentCount":"0",
     "targetUser":"Miss Jane",
     "targetUserId":"444555666",
     "liked":"false",
     "fullUpdate":
        {"to":[
            {"name":"Target Name",
             "id":"666555444"}],
         "message":"Message text",
         "picture":"http://image.com",
         "link":"http://link.com",
         "caption":"www.youtube.com",
         "description":"message desc",
         "source":"http://source.com",
         "icon":"http://icon.com",
         "name":"name here",
         "type":"video"}}]}
```

4.2. GetPost

Call URL

/rest/xml/getPost

/rest/json/getPost

NOTE: this method is only implemented for Facebook and Twitter.

@GET Request

- postId *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<LoadAlbumRequest>
<postId>17657897609</postId>
<account>
<name></name>
<id></id>
<type></type>
<key></key>
<secret></secret>
</account>
<applicationType>WAP</applicationType>
</LoadAlbumRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LoadAlbumResponse>
<message>OK</message>
        <status>0</status>
        <update>
            <text> text is here</text>
            <date>2010-07-08T12:29:14+01:00</date>
           <formattedDate>Postado em Mon, 08 de Jul de 2010</formattedDate>
            <name>Jogn Doe</name>
            <userId>111222333</userId>
            <id>123456789</id>
            <imageURL>http://image.com</imageURL>
            <shortImageURL>http://image.com</shortImageURL>
            <type>FACEBOOK</type>
            <retweeted>true</retweeted>
            <retweetId>0</retweetId>
            <likeCount>0</likeCount>
            <commentCount>0</commentCount>
            <targetUser>Joe</targetUser>
            <targetUserId>0</targetUserId>
            <liked>false</liked>
            <fullUpdate>
                <icon>http://icon.com</icon>
                <link>http://www.facebook.com/</link>
                <message>message here</message>
                <to />
```

```
                <type>link</type>
            </fullUpdate>
        </update>
</LoadAlbumResponse>
```

Response (JSON):

```
{"status":"0",
 "message":"OK",
 "update":
{"text":"this is the post text",
 "name":"John Doe",
 "date":"Wed Oct 13 09:46:57 BST 2010",
 "userId":"12434",
 "id":"12344_3242354355",
 "imageURL":"http://image.com",
 "shortImageURL":"http://shortImage",
 "type":"FACEBOOK",
 "likeCount":"1",
 "commentCount":"0",
 "liked":"false"}]}}
```

4.3. GetFriends

Call URL

/rest/xml/getFriends

/rest/json/getFriends

@GET Request

- start *(optional: default value is 0)*
- count *(optional: the value "all" will retrieve the entire list of friends)*
- search *(optional: if not defined, no filter is applied)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<GetFriendsRequest>
<start>0</start>
<count>10</count>
<search>hello</search>
<accounts>
<account>
<name></name>
<id></id>
<type></type>
<key></key>
<secret></secret>
</account>
<account>...</account>
</accounts>
<type>TV_WIDGET</type>
</GetFriendsRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetFriendsResponse>
<message>OK</message>
<status>0</status>
<friends>
<friend>
<name>Name Lastname</name>
<username>username</username>
<id>1</id>
<imageURL>image1</imageURL>
<shortImageURL>shortImage1</shortImageURL>
<numPhotos>10</numPhotos>
<follower>false</false>
<type>FACEBOOK</type>
</friend>...</friend>
</friends>
<totalCount>100</totalCount>
</GetFriendsResponse>
```

Response (JSON):


```
{"status":"0",
 "message":"OK",
 "totalCount":"2",
 "friends":[
    {"name":"Friend one",
     "username":"",
     "id":"111222333",
     "imageURL":"http://image.com",
     "shortImageURL":"http://shortImage.com",
     "numPhotos":"not supported yet",
     "type":"FACEBOOK"},
    {"name":"Friend two",
     "username":"",
     "id":"22233344",
     "imageURL":"http://image.com",
     "shortImageURL":"http://shortImage.com",
     "numPhotos":"not supported yet",
     "type":"FACEBOOK"}]}
```

4.4. GetProfile

Call URL

/rest/xml/getProfile

/rest/json/getProfile

@GET Request

- id *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<GetProfileRequest>
<id>109</id>
<account>
<name></name>
<id></id>
<type></type>
<key></key>
<secret></secret>
</account>
<type>WAP</type>
</GetProfileRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetProfileResponse>
<message>OK</message>
<status>0</status>
<profile>
<name>John Doe</name>
<hometown>Hometown as String</hometown>
<imageURL>image</imageURL>
<shortImageURL>image</shortImageURL>
<id>111222333</id>
<birthday>MM/DD/YYYY</birthday>
<gender>male</gender>

<political>User political view</political>
<groups>
	<group>
		<name>Group One</name>
		<id>1122</id>
	</group>
	<group>
		<name>Group Two</name>
		<id>2233</id>
	</group>
</groups>
<pages>
	<page>
		<name>Page One</name>
		<category>CAT1</category>
		<id>123</id>
```

```
        </page>
        <page>
                <name>Page Two</name>
                <category>CAT2</category>
                <id>1234</id>
        </page>
</pages>
<works>
        <work>
                <employer>Last Employer</employer>
                <position>Bartender</position>
                <location>Las Vegas, USA</location>
                <startDate>Wed Aug 01 00:00:00 WEST 2007</startDate>
        </work>
        <work>
                <employer>First Employer</employer>
                <position>Footballer</position>
                <location>London, United Kingdom</location>
                <startDate>Thu Jun 01 00:00:00 WEST 2006</startDate>
                <endDate>Sun Jul 01 00:00:00 WEST 2007</endDate>
        </work>
</works>
<educations>
        <education>
                <degree>Master in Psychology</degree>
                <school>University of Texas</school>
                <year>2007</year>
        </education>
        <education>
                <school>Best School in the World</school>
                <year>1234</year>
        </education>
</educations>
<interests>
        <interest>
                <name>Ps3</name>
                <id>12345</id>
                <category>Interest</category>
        </interest>
        <interest>
                <name>Cinema</name>
                <id>54321</id>
                <category>Interest</category>
        </interest>
</interests>
<affiliations>
        <affiliation>
                <nid>12345</nid>
                <name>Ps3</name>
                <year>2004</year>
                <type>work</type>
                <status></status>
        </affiliation>
</affiliations>
<friendsInCommonCount>44</friendsInCommonCount>
<pagesCount>36</pagesCount>
<albumsCount>5</albumsCount>
<postsCount>39</postsCount>
<type>FACEBOOK</type>
<status>This is my status and it rocks!</status>
</profile>
</GetProfileResponse>
```

Response (JSON):

```
{"status":"0",
 "message":"OK",
 "profile":
    {"name":"John Doe",
     "hometown":"Hometown as String",
     "imageURL":"http://image.com",
     "shortImageURL":"http://shortImage.com",
```

```
      "id":"111222333",
      "relationshipStatus":"In a Relationship",
      "birthday":"MM/DD/YYYY",
      "gender":"male",
      "political":"User political view",
      "groups":[
         {"name":"Group One",
          "id":"1122"},
         {"name":"Group Two",
          "id":"2233"}],
      "pages":[
         {"name":"Page One",
          "category":"CAT1",
          "id":"123"},
         {"name":"Page Two",
          "category":"CAT2",
          "id":"1234"}}],
      "works":[
         {"employer":"Last Employer",
          "position":"Bartender",
          "location":"Las Vegas, USA",
          "startDate":"Wed Aug 01 00:00:00 WEST 2007",
          "endDate":"Mon Dec 01 00:00:00 WET 2"},
         {"employer":"First Employer",
          "position":"Footballer",
          "location":"London, United Kingdom",
          "startDate":"Thu Jun 01 00:00:00 WEST 2006",
          "endDate":"Sun Jul 01 00:00:00 WEST 2007"}],
      "educations":[
         {"degree":"Master in Psychology",
          "school":"University of Texas",
          "year":"2007"},
         {"school":"Best School in the World",
          "year":"2002"}],
      "interests":[
         {"id":"12345",
          "name":"Ps3",
          "category":"Interest"},
         {"id":"54321",
          "name":"Cinema",
          "category":"Interest}],
      "affiliations":[
         {"nid":"672891364",
          "name":"BigCorp",
          "year":"0",
          "type":"work",
          "status":""}],
      "friendsInCommonCount":"44",
      "pagesCount":"36",
      "albumsCount":"5",
      "postsCount":"39",
      "type":"FACEBOOK",
      "status":"This is my status and it rocks!"}}
```

4.5. Update

Call URL

/rest/xml/update

/rest/json/update

@GET Request

- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<UpdateRequest>
<text>update text here</text>
<accounts>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<account>...</account>
</accounts>
<type>WAP</type>
</UpdateRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Response>
<message>OK</message>
<status>0</status>
</Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

NOTE: Maximum number of characters for Social Networks:

Table 1 - Social networks maximum post length

Social Network Name	Max chars
Twitter	140
Hi5	200
Facebook	420
LinkedIn	140
MySpace	140
FourSquare	140
Orkut	1000

5. Facebook calls

5.1. GetFeeds

Call URL

/rest/xml/facebook/getFeeds

/rest/json/facebook/getFeeds

@GET Request

- start *(optional: default value is 0)*
- count *(optional: default value is 10)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<FacebookGetFeedsRequest>
<start>0</start>
<count>10</count>
<accounts>
<account>
<name></name>
<id></id>
<type></type>
<key></key>
<secret></secret>
</account>
<account>...</account>
</accounts>
<type>WIDGET</type>
</FacebookGetFeedsRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<GetTimelineResponse>
    <message>OK</message>
    <status>0</status>
    <updates>
        <update>
            <text> text is here</text>
            <date>2010-07-08T12:29:14+01:00</date>
            <formattedDate>112323423545</formattedDate>
            <name>Jogn Doe</name>
            <userId>111222333</userId>
            <id>123456789</id>
            <imageURL>http://image.com</imageURL>
            <shortImageURL>http://image.com</shortImageURL>
            <type>FACEBOOK</type>
            <retweeted>true</retweeted>
            <retweetId>0</retweetId>
            <likeCount>0</likeCount>
            <commentCount>0</commentCount>
            <targetUser>Joe</targetUser>
```

```
            <targetUserId>0</targetUserId>
            <liked>false</liked>
        </update>
        <update>(…)</update>
    </updates>
</GetTimelineResponse>
```

Response (JSON):

```
{"status":"0",
 "message":"OK",
 "updates":[
    {"text":"text is here",
     "date":"Thu Jul 08 12:29:14 WEST 2010",
     "formattedDate":"112323423545",
     "name":"John Doe",
     "userId":"111222333",
     "id":"123456789",
     "imageURL":"http://image.com",
     "shortImageURL":"http://image.com",
     "type":"FACEBOOK",
     "likeCount":"0",
     "commentCount":"0",
     "liked":"false",
},
    {"text":"other text",
     "date":"Wed Jul 07 17:09:08 WEST 2010",
     "name":"Mister John",
     "userId":"22233344",
     "id":"987654321",
     "imageURL":"http://otherimage.com",
     "type":"FACEBOOK",
     "likeCount":"0",
     "commentCount":"0",
     "targetUser":"Miss Jane",
     "targetUserId":"444555666",
     "liked":"false",
     "fullUpdate":
        {"to":[
            {"name":"Target Name",
             "id":"666555444"}],
         "message":"Message text",
         "picture":"http://image.com",
         "link":"http://link.com",
         "caption":"www.youtube.com",
         "description":"message desc",
         "source":"http://source.com",
         "icon":"http://icon.com",
         "name":"name here",
         "type":"video"}}]}
```

5.2. Like

Call URL

/rest/xml/facebook/like

/rest/json/facebook/like

@GET Request

- id *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<FacebookLikeRequest>
<id>123</id>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</FacebookLikeRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Response>
<message>OK</message>
<status>0</status>
</Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

5.3. Unlike

Call URL

/rest/xml/facebook/unlike

/rest/json/facebook/unlike

@GET Request

- id *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<FacebookUnlikeRequest>
<id>123</id>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</FacebookUnlikeRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Response>
<message>OK</message>
<status>0</status>
</Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

5.4. ListComments

Call URL

/rest/xml/facebook/listComments

/rest/json/facebook/listComments

@GET Request

- id *(mandatory)*
- start *(optional)*
- count *(optional)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<FacebookListCommentsRequest>
<id>123</id>
<start>0</start>
<count>10</count>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</FacebookListCommentsRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FacebookListCommentsResponse>
<message>OK</message>
<status>0</status>
<comments>
<comment>
<text>hello world</text>
<date>2010-05-07T16:06:11.423+01:00</date>
<formattedDate>112323423545</formattedDate>
<name>tiago</name>
<userId>4352</userId>
<id>id</id>
<imageURL>image</imageURL>
</comment>
<comment>...</comment>
</comments>
</FacebookListCommentsResponse>
```

Response (JSON):

```
{"status":"0",
```

```
"message":"OK",
"comments":[
   {"text":"hello world",
    "date":"Fri May 07 16:05:45 BST 2010",
    "formattedDate":"112323423545",
    "name":"tiago",
    "userId":"user",
    "id":"id",
    "imageURL":"image"},
   {"text":"hello world",
    "date":"Fri May 07 16:05:45 BST 2010",
    "name":"name",
    "userId":"user",
    "id":"id",
    "imageURL":"image"}]}
```

5.5. Comment

Call URL

/rest/xml/facebook/comment

/rest/json/facebook/comment

@GET Request

- id *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*

@POST Request

```
<FacebookCommentRequest>
<id>7</id>
<text>hello world</text>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</FacebookCommentRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

5.6. DirectMessage

Call URL

/rest/xml/facebook/directMessage

/rest/json/facebook/directMessage

@GET Request

- id *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<FacebookDirectMessageRequest>
<id>7</id>
<text>hello world</text>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</FacebookDirectMessageRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

5.7. WriteOnWall

Call URL

/rest/xml/facebook/writeOnWall

/rest/json/facebook/writeOnWall

@GET Request

- id *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<FacebookWriteOnWallRequest>
<id>7</id>
<text>hello world</text>
<account>
<name></name>
<id></id>
<type>FACEBOOK</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</FacebookWriteOnWallRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

6. Twitter

6.1. Reply

Call URL

/rest/xml/twitter/reply

/rest/json/twitter/reply

@GET Request

- id *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<TwitterReplyRequest>
<id>7</id>
<text>hello world</text>
<account>
<name></name>
<id></id>
<type>TWITTER</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</TwitterReplyRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

6.2. Retweet

Call URL

/rest/xml/twitter/retweet

/rest/json/twitter/retweet

@GET Request

- id *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<TwitterRetweetRequest>
<id>7</id>
<account>
<name></name>
<id></id>
<type>TWITTER</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</TwitterRetweetRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TwitterRetweetResponse>
    <message>OK</message>
    <status>0</status>
    <retweetID>25038357973</retweetID>
</TwitterRetweetResponse>
```

Response (JSON):

```
{"status":"0","message":"OK","retweetID":"12345"}
```

6.3. UndoRetweet

Call URL

/rest/xml/twitter/undoRetweet

/rest/json/twitter/undoRetweet

@GET Request

- id *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<TwitterUndoRetweetRequest>
<id>7</id>
<account>
<name></name>
<id></id>
<type>TWITTER</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</TwitterUndoRetweetRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

6.4. DirectMessage

Call URL

/rest/xml/twitter/directMessage

/rest/json/twitter/directMessage

@GET Request

- screenName *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<TwitterDirectMessageRequest>
<screenName>user</screenName>
<text>hello world!</text>
<account>
  <name></name>
  <id></id>
  <type>TWITTER</type>
  <key>a</key>
  <secret>b</secret>
</account>
<type>WAP</type>
</TwitterDirectMessageRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

7. Hi5

7.1. Authorize

Call URL

/rest/xml/hi5/authorize

/rest/json/hi5/authorize

@GET Request

- username *(mandatory)*
- password *(mandatory)*
- userToken *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<Hi5AuthorizeRequest>
<username>user@xpto.com</username>
<password>hello world!</password>
<userToken>token</userToken>
<type>WAP</type>
</Hi5AuthorizeRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Hi5AuthorizeResponse>
<message>OK</message>
<status>0</status>
<account>
<id>23453</id>
<type>HI5</type>
<key>fmnvc9pdh</key>
<secret></secret>
</account>
</Hi5AuthorizeResponse>
```

Response (JSON):

```
{"status":"0","message":"OK","account":{"id":"23453","key":"fmnvc9pdh","secret":"","type":"HI5
"}}
```

8. LinkedIn

8.1. DirectMessage

Call URL

/rest/xml/linkedin/directMessage

/rest/json/linkedin/directMessage

@GET Request

- id *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<LinkedInDirectMessageRequest>
<id>7</id>
<text>hello world</text>
<account>
<name>User name</name>
<id>User ID</id>
<type>LINKEDIN</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</LinkedInDirectMessageRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```

9. Orkut

9.1. DirectMessage

Call URL

/rest/xml/orkut/directMessage

/rest/json/orkut/directMessage

@GET Request

- id *(mandatory)*
- text *(mandatory)*
- account *(mandatory)*
- applicationType *(mandatory)*

@POST Request

```
<OrkutDirectMessageRequest>
<id>7</id>
<text>hello world</text>
<account>
<name>User name</name>
<id>User ID</id>
<type>ORKUT</type>
<key>a</key>
<secret>b</secret>
</account>
<type>WAP</type>
</OrkutDirectMessageRequest>
```

Response (XML):

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Response><message>OK</message><status>0</status></Response>
```

Response (JSON):

```
{"status":"0","message":"OK"}
```