Um Sistema de Mensagens Parametrizável e Seguro para SmartPhones e Tablets iOS

Carlos Guilherme Bezerra Aguiar Bechtinger[1], Marco Antonio D'Alessandro Costa[2], Alexandre Sztajnberg[1,2,3]

[1]Bacharelado em Ciência da Computação, IME/UERJ
[2]Mestrado em Engenharia Eletrônica, PEL/FEN
[3]Mestrado em Ciências Computacionais, CComp/IME
Universidade do Estado do Rio de Janeiro (UERJ) - Rua São Francisco Xavier, 524 – Maracanã - Rio de Janeiro – CEP 20550-013 – Rio de Janeiro – RJ - Brasil

gbechtinger@gmail.com, marco.dalessandro@gmail.com, alexszt@uerj.br

Abstract. *The use of messaging systems in professional applications has non-functional requirements such as scalability and security, which are not offered in most common software infrastructure currently available. This paper presents a messaging system supported by mobile devices, which meets specific requirements for use in enterprise environments, including: scalability, high availability, security and flexibility. The presented approach allows the use private servers, third-party, or even an hybrid solution, making the service independent of large providers and is also customizable flexibility by allowing definition of permission settings to groups of users and changing the look of the client software. Besides the architecture, a prototype server and clients for Apple iOS are presented.*

Resumo. *O uso de sistema de trocas de mensagens em aplicações profissionais possui requisitos não-funcionais, como escalabilidade e segurança, que não são oferecidos nas infraestruturas de software mais comuns atualmente disponíveis. Este artigo apresenta um sistema de troca de mensagens com suporte à dispositivos móveis, que atende a requisitos específicos de uso em ambientes corporativos, entre eles: escalabilidade, alta disponibilidade, segurança e flexibilidade. Na abordagem apresentada, é oferecida a possibilidade do uso de servidores privados, de terceiros ou, ainda, uma solução híbrida, tornando o serviço independente de grandes provedores e permitindo, também, a customização do mesmo, definindo permissão de envio de mensagens por grupos de usuários e alterando o aspecto do software cliente. Além da arquitetura, um protótipo do servidor e clientes para o iOS da Apple são apresentados.*

1. Introdução

Aplicações para troca de mensagens textuais, áudio e vídeo são de uso rotineiro por usuários de dispositivos ligados à Internet. A utilidade destas aplicações é indiscutível, substituindo muitas vezes a telefonia tradicional. Seu uso vai desde o lazer até o uso estratégico profissional. Os sistemas disponíveis para o suporte a estas aplicações atualmente oferecem infraestrutura robusta, permeiam as plataformas de software mais

usadas e oferecem a configuração e personalização, por exemplo, a visibilidade de um usuário para os demais usuários, alarmes de som, uso de *emoticons*, etc.

A infraestrutura dos sistemas de troca de mensagens, entretanto, pode não atender a todos os requisitos para o uso destas aplicações em ambientes onde a tolerância a falhas e a segurança sejam características importantes. Por exemplo, os sistemas podem ter arquitetura centralizada, contendo pontos de falha, o que pode tornar o serviço indisponível em vários momentos e por muito tempo. Da mesma forma, a ausência de autenticação forte e de criptografia coloca em risco a privacidade e o sigilo das informações comunicadas durante o trânsito das mensagens na Internet, o que pode ser impeditivo para uso profissional.

Para algumas aplicações comerciais destes sistemas, seria importante oferecer itens de segurança como a autenticação forte dos usuários bem como a criptografia das mensagens transportadas, resolvendo os problemas apontados. Também seria desejável que as infraestruturas disponíveis permitissem a segmentação do acesso de usuários em grupos, baseados em seu papel ou credenciais. Também seria interessante oferecer alguma independência de administração e configuração do comportamento da aplicação, ainda que a infraestrutura fosse provida por um terceiro, por exemplo, manter o banco de usuários e mensagens em um repositório próprio.

Este artigo apresenta um sistema de troca de mensagens com suporte à dispositivos móveis, que atende a requisitos específicos de uso em ambientes corporativos, entre eles: escalabilidade, alta disponibilidade, segurança e flexibilidade para o provimento do serviço. Na abordagem apresentada, é oferecida a possibilidade do uso de servidores privados, de terceiros ou, ainda, uma solução híbrida, tornando o serviço independente de grandes provedores e permitindo, também, a customização do mesmo, definindo permissão de envio de mensagens por grupos de usuários e alterando o aspecto do software cliente.

Foi desenvolvido um protótipo do sistema, com a infraestrutura implementada em Java e implantada sobre serviços gratuitos de nuvem da Amazon, e os módulos de cliente desenvolvidos como aplicações iOS [Apple, 2014] da Apple. Com base neste protótipo foram realizados testes de operação para verificar seu correto funcionamento e a avaliação dos requisitos não funcionais, principalmente os relacionados a escalabilidade, tolerância a falhas e segurança.

O restante do texto está estruturado da seguinte forma. Na Seção 2, são apresentados trabalhos relacionados. A arquitetura do sistema é apresentada na Seção 3. A Seção 4 apresenta detalhes sobre o cliente e a interface gráfica disponível. A interface Administrativa e a integração do banco de dados de usuário do cliente coorporativo são apresentadas na Seção 5. A Seção 6 apresenta as soluções de integração com provedores de nuvem adotados no projeto. A Seção 7 conclui o artigo.

2. Trabalhos relacionados

Atualmente, existem vários sistemas que suportam aplicações de comunicação e troca de mensagens entre usuários. Tais aplicações permitem o envio de arquivos e comunicação via texto ou som. Dentre os aplicativos mais utilizados temos, por exemplo, o WhatsApp [WhatsApp 2013], Skype [Skype 2013] e o Hangouts [Hangouts 2013].

Há também propostas que objetivam oferecer infraestruturas para troca de mensagens a aplicações profissionais ou coorporativas. Por exemplo, o Talkbox Teamwork [Talkbox Teamwork 2013] oferece características como a administração e a customização de grupos fechados, criptografia de dados transmitidos e o escalamento horizontal do sistema acrescentando servidores *back-end* sob demanda. A empresa oferece atualmente um *kit* de desenvolvimento para a instalação de serviços de mensagens dentro de outras aplicações.

Alguns sistemas de troca de mensagens utilizam o protocolo aberto XMPP (*Extensible Messaging and Presence Protoco*l – Protocolo Extensível de Mensagens e Presença), que foi desenvolvido como um conjunto de tecnologias abertas para troca de mensagens e informação de presença, chamadas de voz e vídeo, colaboração e roteamento generalizado de dados XML. É formalizado pelo IETF (*Internet Engineering Task Force*) e disponível através de RFCs [XMPP 2014]. Este protocolo oferece algumas funcionalidades de segurança e proteção contra SPAM. Inicialmente, utilizado pelo serviço aberto Jabber.org, o XMPP já é utilizado por vários clientes de mensagens instantâneas como *middleware*. Outro aspecto de destaque no protocolo é a possibilidade de suportar outros tipos de serviço, como transferência de arquivos, através da utilização de extensões.

A maioria dos sistemas disponíveis, incluindo os mencionados, apresentam alguma forma de autenticação e segurança, através de identificação com senha, que não são robustas o suficiente para sua aplicação em sistemas profissionais. Por exemplo, ao contrário do sistema proposto neste projeto, que utiliza criptografia nos dados transferidos e certificados para a identificação dos servidores, foi descoberto no ano de 2011 que o WhatsApp transfere suas mensagens em texto simples [IDG Now! 2011], o que põe em risco a segurança dos usuários que o utilizam.

Além disso, mesmo nos sistemas que utilizam criptografia como forma de segurança, há evidências de que parcerias com a NSA (*National Security Agency* – Agência de Segurança Nacional dos Estados Unidos da América) estão colocando em risco a privacidade das mensagens [Greenwald 2013] [Naughton 2013], pois, além de a NSA ter acesso aos principais nós de Internet dos EUA, ela está se utilizando de *backdoors* presentes nos aplicativos para ter acesso aos dados, mesmo os criptografados. Isso reforça a segurança como uma característica importante a ser observada em sistemas de troca de mensagens.

Entre os sistemas apresentados, poucos oferecem a flexibilidade de parametrização pelo usuário, utilizada pelo sistema apresentado neste texto. No presente sistema, a empresa ou pessoa interessada em criar um serviço de comunicação pode configurar os usuários a partir de seu próprio banco de dados, restringir as permissões dos usuários, configurar os serviços e os dados que são exibidos e transmitidos. Por outro lado, vários dos sistemas disponíveis oferecem alta disponibilidade, através da replicação de servidores, oferecendo aos usuários uma boa experiência de uso. Por exemplo, estas características permitiram ao WhatsApp a conquista de milhões de clientes [Winkler 2013].

Um último aspecto relacionado a segurança, pesquisado em outros sistemas para troca de mensagem, trata da infraestrutura de comunicação utilizada, que geralmente é implantada sobre a Internet pública. Na abordagem da presente proposta é possível

restringir ou configurar a infraestrutura utilizada, inclusive confinando-a em uma intranet ou isolando servidores *front-end* e *back-end* por VPNs (*Virtual Private Networks* – Redes Virtuais Privadas), além da transmissão das mensagens trocadas entre os servidores de forma criptografada.

3. Arquitetura do sistema

A arquitetura do sistema é composta por uma combinação de quatro elementos: um servidor *front-end*, servidores *back-end*, um banco de dados e aplicativos clientes nos dispositivos móveis (celulares e *tablets*).

O elemento base do sistema é chamado de *front-end.* Ele tem basicamente duas responsabilidades: controlar a lista de usuários ativos com sua localização atual na rede e manter um registro de provedores de serviço disponíveis. Quando um usuário do sistema executa o software em seu dispositivo móvel o primeiro passo do aplicativo consiste em contatar o *front-end*, que deve possuir uma localização pré-definida (por URL ou endereço IP), e requisitar a localização de um servidor que esteja provendo o serviço desejado, de um conjunto de serviços providos (ver Seção 3.1).

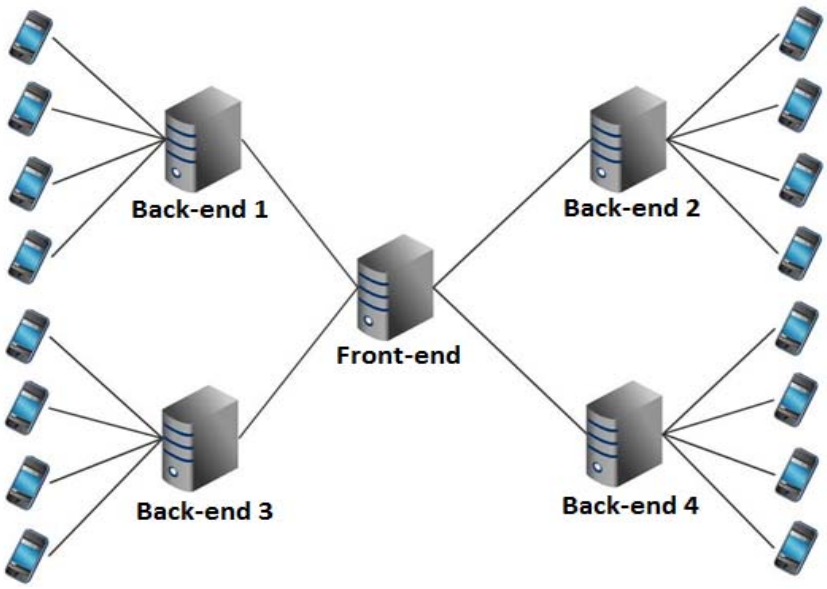

Figura 1. Conexões entre Clientes e *back ends* e entre *back ends* e *front ends*

Os servidores fornecidos aos clientes pelo *front end* são os chamados *back ends*. Eles são os responsáveis por transmitir, gerenciar e armazenar as mensagens transferidas pelos clientes. Estes servidores são capazes de buscar no *front end* a localização de outros usuários do sistema, se conectar a outros servidores e transferir mensagens entre eles caso seja necessário. Estes servidores têm a responsabilidade de manter e gerenciar as conexões de usuários ativos em seu sistema e removê-los quando algum problema for encontrado. As informações de configurações de serviços são enviadas pelo *front end* para os *back ends*. Na Figura 1 podemos ver as conexões entre clientes e *back ends*, e entre *back ends* e *front ends*.

É importante observar que, em paralelo ao *front end*, pode haver um ou mais *back ends*. Tais servidores, ao serem iniciados, também se conectam ao *front end* e se registram como um novo provedor de serviços. Ao estabelecerem a conexão com o

front end o *back end* recebe todas as configurações necessárias para prover um ou mais serviços e passa a estar disponível para qualquer cliente que deseja acessar um dos serviços que ele possua.

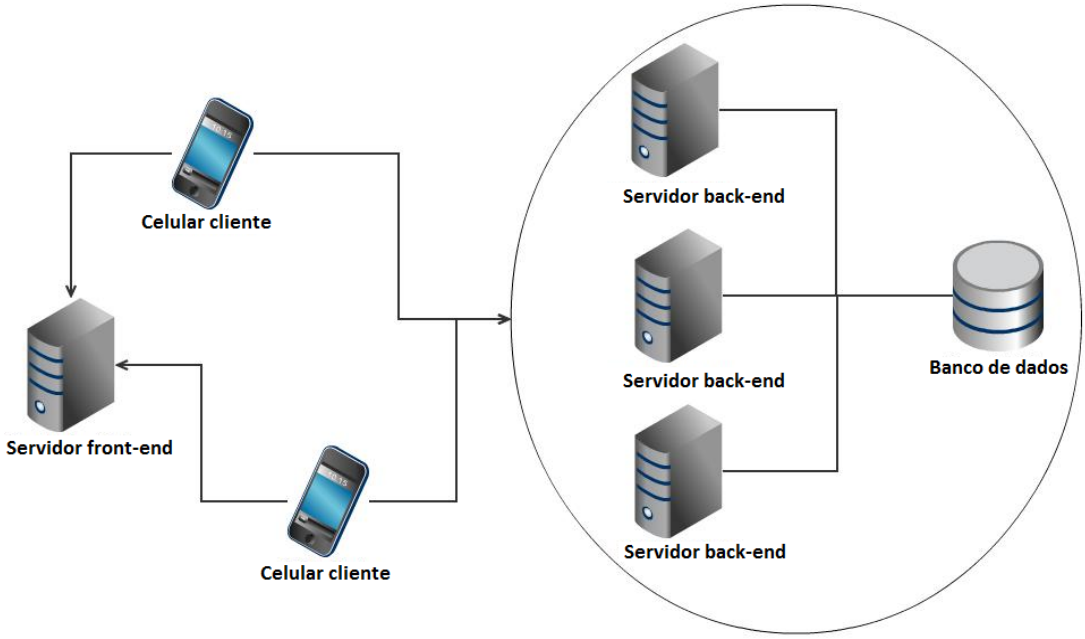

Figura 2. Arquitetura geral do sistema

Na Figura 2 é possível identificar que somente os servidores *back end* possuem acesso ao banco de dados de mensagens e que os clientes se comunicam tanto com o *front end*, durante o primeiro acesso, quanto como o *back end* durante todo o restante da comunicação.

Quando o sistema necessita de novos provedores de serviço, para atender a demanda dos usuários, novos *back ends* podem ser ativados em qualquer lugar da Internet ou em uma rede local. Para isso basta apenas que eles consigam se conectar ao *front end*. A partir do momento em que temos múltiplos *back ends* executando e provendo o mesmo serviço, é necessário que exista uma ligação entre eles, para que usuários do serviço disponibilizado por um *back end* possam se comunicar com outros usuários do mesmo serviço em outro *back end*. Para atingir este objetivo foi utilizada uma técnica de conexão sob demanda que efetua conexões entre os *back ends* somente quando necessário.

Tabela 1. Exemplo de conexões entre servidores *Back end*

Remetente	Destinatário	Conexão criada
Cliente A	Cliente B	Não necessário.
Cliente D	Cliente C	Conexão “a” criada entre *Back end* 3 e 2.
Cliente C	Cliente E	Conexão “b“ criada entre *Back end* 2 e 4.

Na Figura 3, podemos ver as conexões estabelecidas, sob demanda pelo sistema, entre *back ends* criada considerando a necessidade de comunicação entre pares (ou grupos) de usuários, como, por exemplo, na Tabela 1.

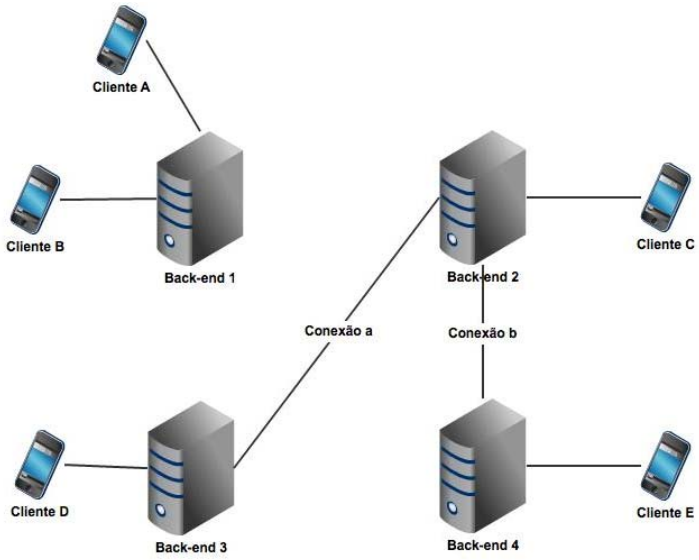

Figura 3. Conexões sob demanda dos *back end*s

Algumas soluções introduzidas na arquitetura estão ligadas à implementação do sistema. O *front-end* e o *back end* foram desenvolvido em Java 1.6 [Oracle 2014], com o apoio dos pacotes JavaMail API [Oracle 2013], o conector PostgresJDBC [PostgreSQL 2014] e o pacote *ConfigurationFiles* que facilita a logística da leitura e interpretação dos arquivos de configuração. A comunicação entre o *front end* e *clientes* ou *back ends* é toda efetuada através da troca de mensagens em formato XML, com o objetivo de compatibilizar o sistema com futuros clientes de acesso para outros dispositivos móveis diferentes dos implementados neste projeto, como os dispositivos com sistema operacional Android. Nas próximas seções discutimos detalhes da arquitetura e da implementação.

3.1 Serviço

Neste sistema, um serviço é um conjunto de configurações necessárias para definir como os servidores devem interagir com os usuários. Tais configurações possuem todas as informações necessárias para que o sistema consiga gerenciar os usuários, efetue controle de acesso e de requisições e também consiga informar a aplicação cliente como exibir e controlar as informações recebidas e enviadas. Um arquivo de configuração contém informações estruturadas de acordo com as classes definidas na Figura 4.

Quando um serviço é inicializado, um objeto da classe *ServiceInstance* é instanciada. Esta classe controla diversas classes auxiliares, cada uma com a responsabilidade de gerenciar um aspecto do serviço:

- Classe *Directory* - É responsável por armazenar a lista de usuários ativos em determinado serviço. Ela também gerencia a inserção e remoção de usuários nesta lista pelas outras classes do sistema;
- Classe *Database* - Cabe a ela fornecer acesso ao banco de dados de maneira fácil e prática para as outras classes do sistema. Para tal, ela mantém uma conexão ativa com o banco de dados fornecido no arquivo de configuração, para melhorar o tempo de resposta à uma requisição de outra classe;

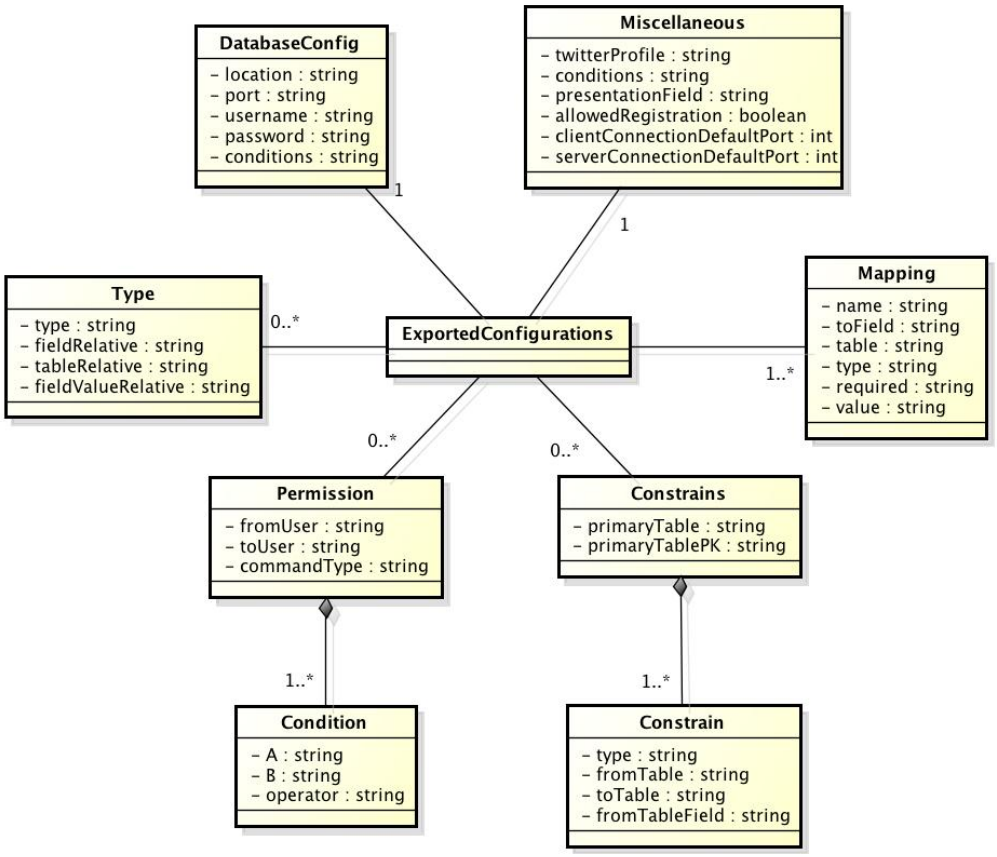

Figura 4. Diagrama de classes do arquivo de configurações

- Classe *Logger* - É quem controla a inserção de informações de *log* em um arquivo específico para cada serviço disponibilizado por um *back end*. Ela utiliza o nome do serviço para criar um arquivo de *log* onde informações de falha, controle e *debug* são armazenadas;
- Classe *Executor* - É responsável por executar os comandos recebidos pelo serviço. Ela possui um objeto do tipo *LinkedBlockedQueue*, que é uma lista encadeada com bloqueio, onde os comandos recebidos são armazenados e posteriormente executados. Ela fornece também um método *Execute* que executa os comandos imediatamente sem a utilização da fila de espera;
- Classe *Authenticator* – É a responsável pela autenticação dos usuários do serviço e por outros métodos necessários para a criação de *token* de acesso dos usuários. Toda vez que um usuário tenta efetuar *login* em um serviço, esta classe é ativada para validar a requisição e retornar um *token* de autenticação, necessário para a troca de mensagens autenticadas, para o usuário;
- Classe *ExportedConfigurations* - É utilizada para armazenar as configurações geradas pela interface de gerenciamento. Esta classe é armazenada pelo serviço e consultada sempre que for necessário o acesso a alguma informação relacionada ao funcionamento do serviço. Esta classe é interface entre a infraestrutura provida e o banco de dados que pode ser exportado pelo administrador do sistema (Seção 4).

3.2 *Front end*

O cliente precisa interagir com o *front end* apenas durante o processo de inicialização, quando ele o consulta em busca de um *back end*. Quando um cliente se conecta ao *front-end*, ele automaticamente recebe uma mensagem XML com a localização de um *front-end*.

A atividade principal do *front end* é receber requisições de clientes, procurando um *back end*, e para isso tem o suporte adequado de *sockets* de rede e mecanismos para selecionar o *back end* adequado para o cliente. Na Figura 5 é apresentado o diagrama de classes da aplicação *front end*. A interação com os clientes é realizada pela classe *UserController.*

A classe *Chooser* que é responsável por escolher um servidor *back end* dentre os disponíveis, utilizando para isso um *ChooserMethod* escolhido pelo cliente. Foi desenvolvido junto com o sistema duas formas de seleção: *RoundRobinChooser* que escolhe um *back end* baseado em uma fila circular e um *ResponseTimeChooser* que escolhe sempre o *back end* com o menor tempo de resposta.

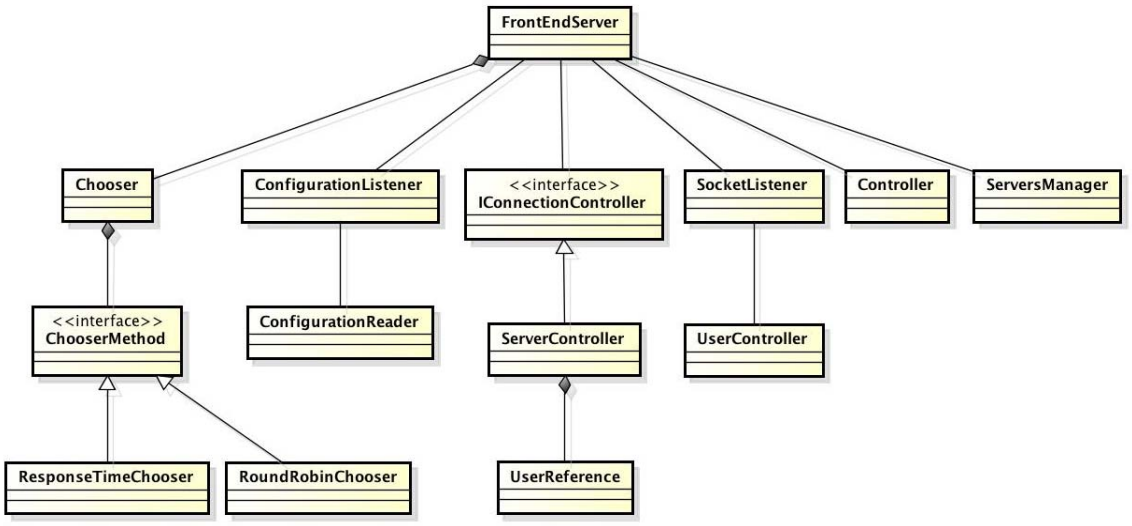

Figura 5. Diagrama de classes do *front-end*

A interface de gerenciamento é tratada pela classe *ConfigurationReader*, que recebe um arquivo de configuração como indicado na Seção 3.1.

O *front end* precisa se comunicar com os *back ends* em algumas situações para realizar a integração entre os elementos. Esta comunicação é gerenciada pela classe *ServerController*

O diagrama da Figura 6 apresenta a sequência de interações para a criação dos tratadores dos 3 tipos de conexões possíveis.

A troca de mensagens entre o *front end* e os *back ends* pode ocorrer para que *back ends* sejam registrados e interligados logicamente, para realizar a manutenção quando existe a movimentação de usuários e para monitoramento dos *back ends*:

- Localização de usuário - Quando um servidor *back end* recebe uma mensagem para ser entregue ele deve primeiro verificar se o destinatário está ativo em seu sistema, e

caso contrário deve buscar a localização atual no *front end*. Com a localização em mãos o *back end* é capaz de se conectar com o *back end* que possui o usuário ativo em seu sistema e transferir a mensagem.

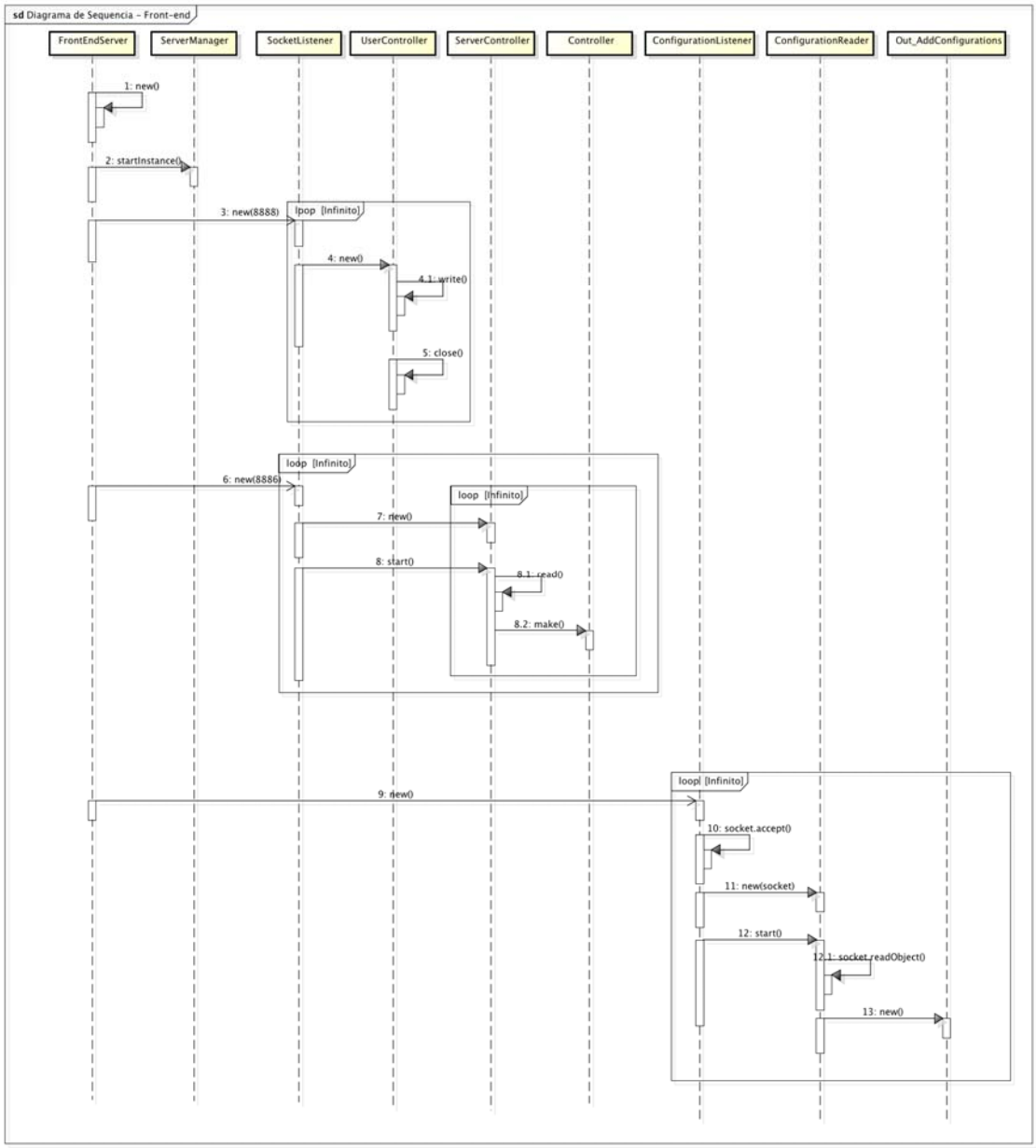

Figura 6. Diagrama de sequência da criação de tratadores de conexão

- Adição e remoção de usuário - Quando um *back end* autentica um usuário com sucesso, ele dever adicionar esse usuário à lista de usuários ativos no *front end* através de uma requisição de adição de usuário. Com o usuário registrado no *front end* qualquer outro *back end* que procure por este usuário irá receber a localização do *back end* que está gerenciando a conexão do mesmo. De forma similar, quando um *back end* detecta que um usuário foi desconectado do sistema, ele deve remover este usuário do *front end* através de uma requisição de remoção de usuário.

- Ping - A requisição de `Ping é utilizada pelo *front-end* para medir o tempo de resposta de um servidor *back-end*. Quando o *front-end* envia uma requisição de Ping ele armazena o *timestamp* do momento em que a mensagem foi enviada e ao receber a resposta calcula o *round-trip* (tempo total de ida e volta da mensagem). Esta informação é utilizada pelo *front-end* para escolher, durante a conexão inicial de um cliente, um servidor que esteja em melhores condições de atender aos pedidos do cliente.
- Registro de *back end* - Quando um servidor *back end* é ativado, ele deve se registrar no *front end*, enviando o seu endereço IP público e a porta que está sendo utilizada para conexão de clientes.
- Envio de configuração para um *back end* - Quando um *front end* recebe uma configuração de serviço enviada pela interface de gerenciamento, ele deve retransmitir esta informação a todos os *back ends* que estão ativos no momento. Este envio é feito através de uma requisição de envio de configurações.

3.3. *Back end*

Como visto até agora, o servidor *back end* é o responsável por receber as mensagens dos usuários e transmitir para o destinatário. Para isto, ele deve ter uma conexão ininterrupta com o *front end* para consultar as localizações dos outros usuários, sempre que necessário, e uma conexão com os bancos de dados utilizados pelos serviços que ele disponibiliza. Durante seu tempo de vida, o *back end* fica a espera de novas conexões de clientes na porta TCP 8890 como visto na Figura 7.

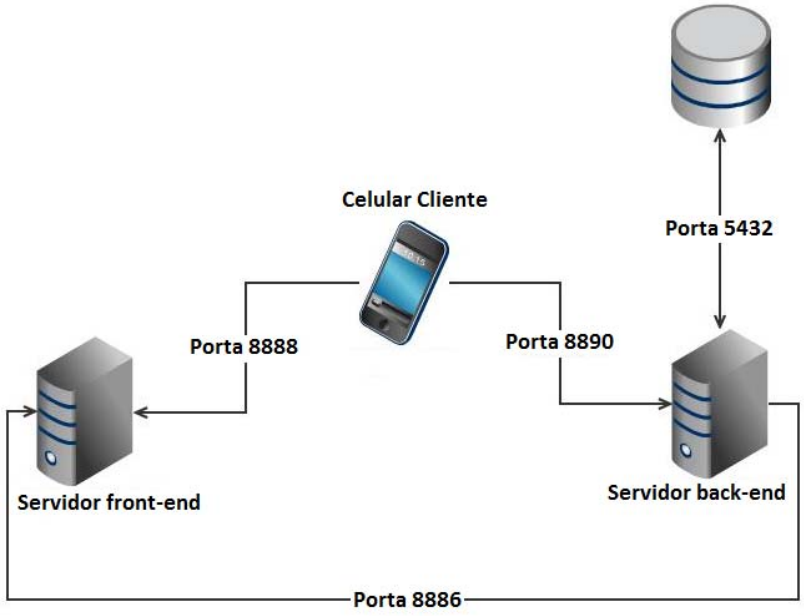

Figura 7. Visão geral do sistema e das portas utilizadas para comunicação

Quando um *back end* recebe uma mensagem de um usuário, ele deve cuidar para que a mensagem seja entregue com sucesso ao destinatário, ou caso não seja possível, deve armazenar a mensagem no banco de dados para que ela seja recuperada e entregue assim que o destinatário se conectar ao serviço novamente.

No diagrama de classes da Figura 8 podemos ver as classes mais importantes do *back end* e suas relações. Dentre elas temos o *FrontEndLink* que é responsável pela conexão do *back end* com o *front end*, a *BackEndLinkers* que é responsável por gerenciar e estabelecer conexões entre *back ends* e a classe *UserLink* que gerencia a conexão com um cliente. A classe *Services* gerencia e armazena as configurações de serviços ativos no *back end* e a classe *Controller* efetua o mapeamento das requisições do usuário em formato XML para classes Java.

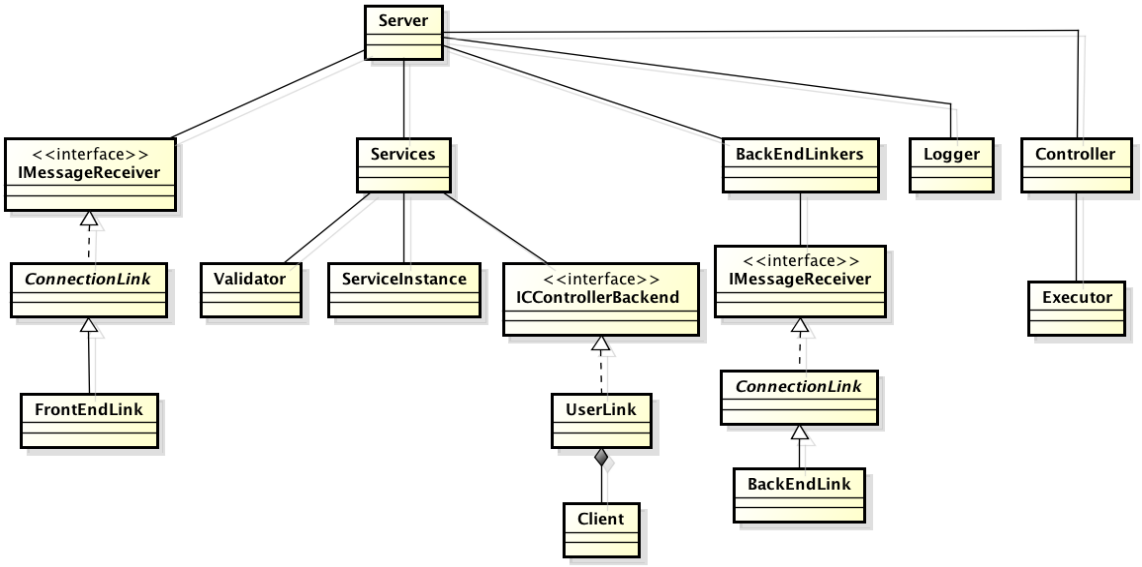

Figura 8. Diagrama de classes do *Back end*

Toda a comunicação entre *back ends* e clientes se dá através de requisições XML. Estas requisições possuem um campo chamado *opcode* que define o tipo da requisição e é utilizado pela classe *Controller*, para mapear a requisição para uma classe Java que possa ser tratada diretamente pelo *back end*. Um *back end* também pode se comunicar através de requisições XML com o *front end*, com outros *back ends* e com os usuários do sistema. As requisições possíveis são:

- Transferência de mensagem entre *back ends* - Quando um *back end* precisa entregar uma mensagem a um usuário que não está ativo em outro *back end*, ele deve transferir a mensagem para o *back end* que possui este usuário ativo através de uma requisição de transferência de mensagem.
- Atualização de foto do usuário - Sempre que um usuário desejar trocar a sua foto de perfil, ele deve ir até a tela de ajustes e selecionar a opção "enviar foto". Desta forma o sistema irá requisitar uma foto ao usuário e irá enviá-la ao servidor *back end* que controla a conexão através de uma requisição de atualização de foto.
- Busca de mensagens armazenadas - Sempre que o sistema cliente efetua *login* em um *back end* ele automaticamente busca mensagens armazenadas nos servidores através de uma requisição de busca de mensagens armazenadas. A resposta a esta requisição são as mensagens armazenadas, caso existam. Caso contrário, nada será enviado ao cliente.

- Busca de foto de um cliente. - Após a busca da lista de usuários disponíveis para comunicação pela aplicação cliente, ela automaticamente busca a foto de cada usuário que é exibido na tela através de uma requisição de busca de foto.

3.4. Cliente

Os módulos cliente para acesso dos usuários foram desenvolvidos como aplicativos para o sistema iOS da Apple usando a linguagem Objective C e bibliotecas fornecidas pela Apple. Além da interface gráfica para o usuário, destacada adiante, o cliente inclui classes para realizar a troca de mensagens usando *sockets*, a conversão e *parsing* de mensagens XML, a persistência e leitura do arquivo de configuração pessoal e os aspectos de segurança (Figura 9).

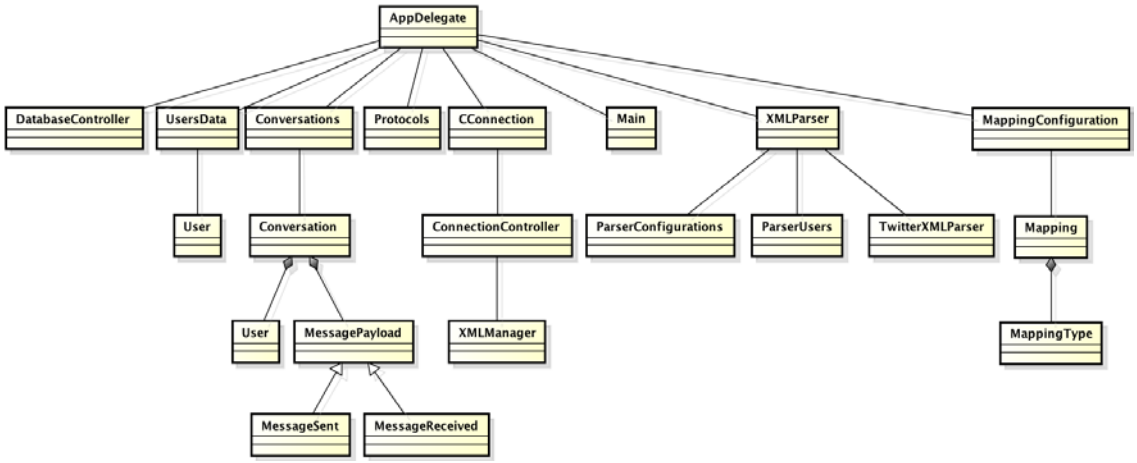

Figura 9. Diagrama de classes da aplicação cliente

Conexão e busca de servidores. O usuário deve instalar a aplicação em seu dispositivo através do serviço da AppStore, disponibilizado em iPads e iPhones e que fornece uma forma fácil de acesso e pagamento às aplicações. Com o aplicativo instalado o usuário ao abri-lo irá se deparar com as telas iniciais de conexão aos servidores e *login*, como visto na Figura 10.

Figura 10. Fluxo de telas inicial da aplicação cliente

Na Figura 10 podemos ver as transições entre as telas inicias do aplicativo. Na transição 1 ao clicar no ícone do software é aberta a tela de inserção do servidor, passo que pode ser omitido caso o aplicativo seja preparado para ser utilizado apenas em um único serviço. Este endereço fornecido deve ser o endereço do *front end*. Para o usuário que utiliza o serviço em seu dispositivo móvel, fica invisível a existência de uma arquitetura de servidores distribuídos, com *front end* e diversos *back ends*, pois todas as requisições ao *front end* para a busca de um *back end* disponível é feita automaticamente e sem a interação do usuário.

Caso o servidor *front end* inserido pelo usuário seja válido e o sistema consiga se conectar com sucesso, a tela de *login* será apresentada e sobre ela uma janela que pergunta se o usuário deseja salvar o servidor inserido como servidor padrão. Caso o usuário responda sim, o servidor *front end* inserido será utilizado automaticamente durante um novo processo de inicialização do aplicativo. Com a localização do servidor *back end* o cliente pode se conectar a ele e buscar as configurações do serviço desejado. As informações de configurações são necessárias para a criação de um novo usuário através do sistema cliente e para posterior exibição e controle dos dados durante o uso.

Principal. Após receber a lista de usuários o cliente será apresentado à tela principal do sistema, visto na transição 5 da Figura 10 e com mais detalhes na Figura 11. Ela possui 2 áreas distintas, uma é a área onde as notícias relacionadas ao serviço são exibidas com suas respectivas datas de postagens. Cada noticia pode conter *links* para *sites* externos que podem ser acessados com apenas um clique do usuário sobre a notícia desejada. Já na parte inferior da tela temos o menu principal que dá acesso a todas as áreas do aplicativo como: *home*, lista de usuários, lista de conversas e configurações gerais. A seguir, seguem mais detalhes de cada uma das áreas.

Figura 11. Tela principal e visualização de notícia

Conversas. A tela de conversas é responsável por exibir todas as conversas ativas com usuários, bem como outras informações básicas sobre a conversa como número de mensagens não lidas. Nesta tela o usuário tem a possibilidade de excluir uma conversa clicando no ícone no canto superior direito "X". Este ícone ativa o modo de edição da tabela que exibe um botão vermelho ao lado de cada conversa, que ao ser clicado remove a conversa da lista e a apaga do banco de dados local do dispositivo móvel.

A informação do número de mensagens não lidas é exibida em duas áreas distintas. No menu inferior, sob o ícone de acesso a janela de conversas, é exibido o número total de mensagens não lidas em todas as conversas, este local é importante pois mesmo que o usuário esteja em outra tela do aplicativo, ele poderá perceber que uma nova mensagem chegou. Já ao lado do nome do usuário com quem esta conversa foi criada é exibido o número de mensagens não lidas da conversa.

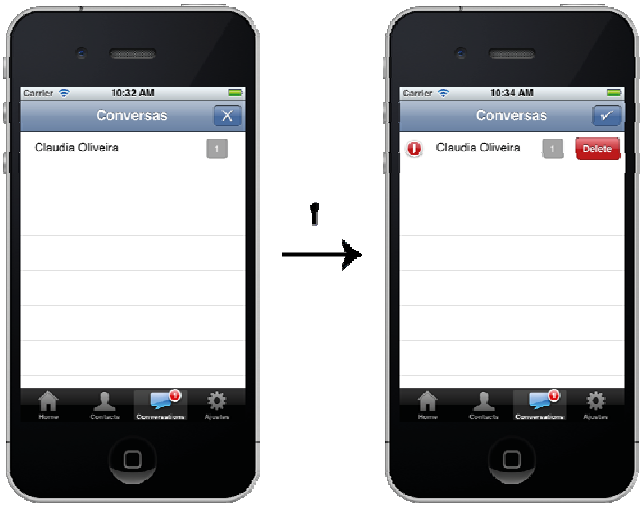

Figura 12. Tela de conversa e exclusão de uma conversa existente

Todos os locais que exibem o número de mensagens não lidas são atualizados automaticamente quando uma conversação é aberta ou quando uma nova mensagem é recebida pelo cliente através de um sistema que utiliza o padrão de projetos *observer* [Gamma 1994]. A célula que representa uma conversação se registra no objeto que representa uma conversa e aguarda por notificações. Quando um objeto conversa recebe uma atualização ele automaticamente avisa todas as áreas do programa que se registraram para receber notificações relacionadas a aquela conversa.

Figura 13. Telas de uma conversa.

Quando uma conversa é aberta pelo usuário, ele é apresentado a uma tela de detalhes onde as mensagens trocadas com o usuário especifico são exibidas. As mensagens de texto enviadas pelo o usuário são alinhadas do lado esquerdo da tela e as mensagens recebidas são alinhadas do lado direito. Mensagens no formato foto são exibidas como uma miniatura da foto transferida, mensagens de localização são exibidas como uma tachinha e mensagens de som são exibidas como um símbolo de caixa de som. A foto pode salva no álbum de fotos do aparelho, encaminhada para outro usuário ou enviada por email.

Configurações pessoais. A tela de ajustes (Figura 14) permite aos usuários a configuração de preferências sobre o funcionamento do programa e o acesso à opções como trocar a foto do perfil e limpar o histórico de mensagens.

Figura 14. Tela de ajustes do cliente de acesso

4. Banco de Dados e Interface Administrativa

O sistema utiliza dois bancos de dados: o banco de dados do cliente que é armazenado no dispositivo do cliente e o banco de dados administrativo utilizado pelos serviços que estão em execução nos servidores. Cada banco de dados possui uma função específica.

4.1 Banco de dados no cliente

O banco de dados do cliente é responsável por manter armazenadas todas as mensagens trocadas pelos usuários a partir daquele dispositivo e informações de preferência de configuração. A implementação utiliza o SGBD SQLite, que tem suporte nativo do iOS, e é utilizado devido a sua simplicidade. Todo o acesso ao banco é feito através de bibliotecas disponíveis no iOS. A tabela de dados é criada durante a primeira inicialização do sistema do cliente em seu dispositivo móvel. Se o cliente mudar de equipamento, a configuração deve ser transferida manualmente ou através do serviço em nuvem iCloud.

Observa-se que as preferências do cliente não são conflitantes com as políticas estabelecidas pelo administrador para cada um dos serviços.

4.2 Banco de dados administrativo

Um diferencial do projeto é a possibilidade do administrador fornecer como entrada de configuração um banco de dados já populado com dados. Por exemplo, é possível indicar um banco contendo a lista de usuários e papéis associados, que serão exportados, em bloco, para o sistema, que passará a usar os dados recebidos na configuração dos serviços e grupos de usuários. O banco de dados fornecido pelo administrador deve ter algumas tabelas auxiliares de controle criadas para permitir que o sistema consiga interagir com qualquer conjunto de dados.

Cada serviço disponibilizado pelos servidores *back end* possui uma tabela no banco de dados administrativo, que é configurada durante a criação da configuração do serviço na interface de gerenciamento administrativo do sistema.

As adaptações comumente necessárias em um banco a ser exportado são a criação de 3 tabelas auxiliares nomeadas de _auth, _messages e _resources. Na Figura 15 é apresentado o diagrama entidade-relacionamento parcial de um banco de dados fornecido pelo cliente e adaptado para ser acessado pelo sistema.

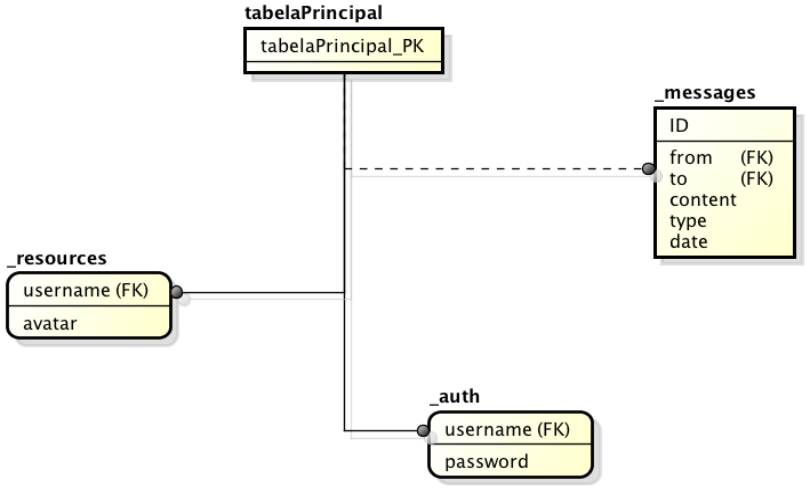

Figura 15. Diagrama entidade-relacionamento do banco de dados de serviços

O sistema cria tabelas adicionais no banco de dados administrativo, complementando os dados fornecidos pelo administrador. Estas tabelas utilizam informações fornecidas pelo administrador, durante a criação da configuração do serviço na interface de gerenciamento, como o nome da tabela, seus campos e o nome do usuário do banco, que identificam unicamente um domínio (se a infraestrutura do sistema estiver sendo usada de forma compartilhada por vários domínios administrativos). Estas informações são necessárias para que as novas tabelas se relacionem corretamente com os dados pré-existentes no banco de dados do sistema.

Novos usuários e novos serviços são inseridos neste banco de dados com o auxílio da interface administrativa. Além disso, o banco de dados administrativo deve estar acessível pela infraestrutura de redes usada, sendo a mesma uma infraestrutura própria ou a Internet, e as permissões de acesso nos *firewall* devem ser liberadas para o *front-end* e os *back-ends*.

4.3 Notificações do banco de dados

Devido a característica distribuída do sistema, diversos servidores *back-end* podem acessar o banco de dados de um serviço simultaneamente, inserindo, removendo e atualizando usuários que estão conectados. Portanto foi necessário encontrar uma maneira de manter todos os servidores *back-ends* atualizados automaticamente quando um deles efetuar uma modificação na lista de usuários. Uma limitação para resolução do problema é que nem todos os servidores *back-end* possuem conexão a todos os outros, portanto um *back-end* que modifica uma informação não seria capaz de se conectar a todos os outros e alertá-los.

O banco de dados PostgreSQL, escolhido para a implementação do sistema, possui um mecanismo de notificações assíncronas pouco conhecido que foi utilizado neste sistema para ajudar a controlar estas modificações. Este mecanismo pode ser usado para que uma conexão com o banco de dados consiga alertar todas as outras conexões ativas ao mesmo banco de forma fácil e com pouco custo computacional. Por outro lado, o *driver* JDBC (Java Database Connectivity – Conectividade do Banco de dados Java) para Java que efetua a conexão dos servidores *back-end* com o banco de dados PostgreSQL não suporta estas notificações assíncronas, portanto foi necessário utilizar uma técnica sobre o *driver* JDBC que permite o uso de uma forma síncrona de notificações. Esta técnica é detalhada a seguir.

Em cada *back-end*, uma *thread*, mantida pela classe *Database*, é inicializada e mantida em execução contínua. Periodicamente envia em um intervalo uma requisição SQL (*Structured Query Language* - Linguagem de Consulta Estruturada) simples ao banco de dados. Esta requisição não possui utilidade ao programa, mas quando o sistema do banco de dados retorna o resultado da consulta, junto com ele são recebidas as notificações pendentes. Portanto qualquer alerta ativado por um *back-end* no banco de dados será recebido pelas outras conexões dos outros servidores *back-end*. Com o aviso de que algo foi alterado, o serviço correspondente executando no *back-end* pode acionar as devidas ações para atualizar sua lista de usuários e avisar os clientes caso seja necessário.

Para ativar uma notificação, basta que um servidor *back-end* execute uma instrução SQL com conteúdo “notify USERS“, desta maneira todas as *threads* de recebimento são acionadas e seus respectivos *back-ends* são atualizados.

4.4 Interface gráfica do administrador

A interface gráfica do administrador foi dividida em duas partes: criação da configuração e envio. Devido a natureza deste aplicativo de gerência que demanda muitas entradas do usuário, foi necessário um cuidado maior no preparo e na organização das informações na tela, com o intuito de facilitar a interação do administrador com o sistema.

A criação de uma configuração de serviço é realizada em alguns passos que recebem e validam informações fornecidas pelo administrador. Este processo começa na tela de configurações de banco de dados visto na Figura 16, que deve ser preenchida com todos os mapeamentos que serão utilizados pelo sistema. Estes mapeamentos são informações sobre quais campos e tabelas do banco de dados são utilizados pelo serviço, os valores permitidos para cada campo e como eles são utilizados e exibidos pelo aplicativo cliente que é executado nos celulares e *tablets* dos usuários cliente.

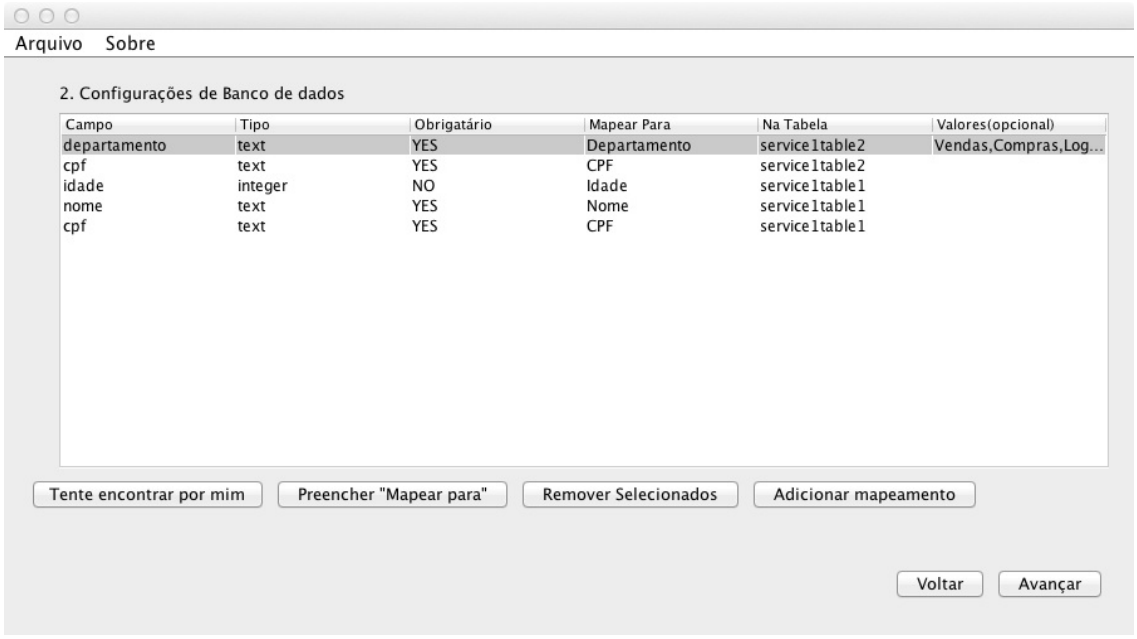

Figura 16. Tela de configuração de mapeamentos de banco de dados

Caso seja o desejo do administrador, é possível utilizar o botão "Tente encontrar por mim", para que o sistema se conecte ao banco e preencha todos os campos automaticamente. Para isso o usuário deve apenas informar os dados de conexão com o banco de dados e após isso quais tabelas são utilizadas pelo serviço (Figura 17).

Figura 17. Tela de busca de mapeamentos automática

Com estas informações o sistema pode passar para as próximas telas onde devem ser escolhidas a tabela principal do sistema com sua respectiva chave primária, a qual todas as outras se interligam através de suas chaves estrangeiras e também deve ser explicitado qual é a chave estrangeira de cada uma das tabelas secundárias.

O próximo passo é a configuração das permissões dos usuários cliente. Tais permissões são importantes para definir o escopo em que cada usuário consegue se comunicar. Na Figura 18, vemos a configuração de permissão onde os usuários do serviço só podem se comunicar com outros usuários do mesmo departamento e uma restrição que define que um usuário só pode se comunicar com usuários com CPF diferente, para evitar que ele se comunique com ele mesmo.

Figura 18. Tela de configurações de permissões

Após todos os passos efetuados com sucesso, a interface de gerenciamento está apta a gerar o arquivo de configuração e enviá-lo para o *front end*. Para isso o usuário deve utilizar a última tela para gravar o arquivo criado em disco, com extensão .cfg, e posteriormente enviá-la através da opção de envio da interface. O tratamento de envio de configurações tem o trabalho de receber um arquivo de configuração gerado previamente pelo sistema, no formato *.cfg*, e enviá-lo para o servidor *front end* especificado pelo usuário.

5. Aspectos de segurança

Para prover segurança ao sistema, os nós hospedeiros têm *firewalls* adequadamente configurados. Além disso, várias técnicas de segurança foram empregadas nas diversas trocas de mensagens do sistema para dificultar ou impedir que usuários mal intencionados tenham acesso a informações indevidas.

5.1 Segurança no meio físico através de *socket* SSL com certificado

Para a conexão entre cliente e servidor *back end* foi utilizado *socket* sobre SSL (Secure Sockets Layer - Camada de Sockets Segura) através do *Java Secure Socket Extension* [Oracle 2011], que oferece o *socket* padrão sobre uma camada de proteção criptografada e certificados para validação dos servidores que garantem que (i) o servidor com o qual o cliente está se conectando é realmente um servidor válido e (ii) os dados transferidos, caso sejam interceptados, são ilegíveis.

Desta forma, qualquer tentativa de simular um servidor válido para receber as requisições dos usuários será invalidada, visto que o servidor falso não possui os certificados necessários para sua identificação positiva. Assim a única maneira de capturar informações seria a escuta da rede e interceptação de dados no meio físico de transporte, porém com a utilização do SSL, toda a informação transmitida é criptografada e não pode ser lida por outra entidade além do servidor real. Com isto, a transferência de dados entre cliente e servidor e entre servidores será segura. O trecho de código da Figura 19 apresenta um exemplo deste tipo de mensagem. Observa-se na linha 6 o conteúdo criptografado.

```
 1. <?xml version="1.0" encoding="UTF-8"?>
 2. <body>
 3.      <opcode>27</opcode>
 4.      <message>
 5.           <type>0</type>
 6.           <content>343dguhf43ifn34fi4bv3i4hv4</content>
 7.           <fromUser>usuário_1</fromUser>
 8.           <toUser>usuário_2</toUser>
 9.           <date>01/01/2012</date>
10.           <uniqueID>id_único_da_mensagem</uniqueID>
11.           <serviceName>nome_serviço</serviceName>
12.      </message>
13. </body>
```

Figura 19. Requisição de transferência de mensagem de um *back end* para outro

Existe outro ponto onde um ataque pode ser efetuado na tentativa de obter dados dos usuários ou criação de mensagens não permitidas, que é o banco de dados. Tendo acesso a informações como os nomes de usuários e senhas dos indivíduos, um indivíduo pode utilizá-las para conectar-se ao sistema e se passar por um outro usuário ou até mesmo acessar outras informações do mesmo, visto que normalmente o usuário utiliza a mesma senha para vários *sites* ou *softwares* diferentes.

Para a conexão do *back-end* com o banco de dados, permitindo a consulta ao mesmo durante a operação, o sistema utiliza uma conexão protegida com SSL. Além disso, toda senha armazenada no banco de dados é criptografada com o algoritmo de *hash* SHA-1 [Apple 2007] que gera um *hash* de 160 bits.

5.2 Validação das requisições recebidas

Além da segurança da transferência e do armazenamento de dados, mais algumas precauções são tomadas pelos servidores *back end* para garantir a confiabilidade do sistema. Cada requisição enviada pelo cliente e recebida por um servidor *back end* passa por um sistema de validação para garantir que não possui nenhuma informação inválida ou maliciosa.

Para tal, as requisições recebidas pelo servidor são classificadas em duas categorias: requisição simples e requisição autenticada. Uma requisição simples não requer que o cliente que a envia esteja autenticado, como exemplo o registro de um novo usuário no sistema ou um teste da conexão.Estas requisições não colocam em risco a segurança do sistema, e podem ser criadas e enviadas para o servidor por um cliente não autenticado. Já as requisições que envolvem dados pessoais e mensagens de usuários participantes do grupo de usuários cadastrados devem ser autenticadas e necessitam que o usuário que as envia esteja identificado e autenticado, como exemplo de requisições autenticadas temos: o envio de mensagem, busca de usuários ou a troca da foto do usuário.

As mensagens autenticadas têm também a sua estrutura validadas através de arquivos XSD [Sperberg-McQueen and Thompson 2012] que provê formas de controlar a estrutura, conteúdo e semântica dos dados contidos em um documento XML.

5.3 *Token* de autenticação

Todas as requisições autenticadas incluem um campo chamado *token*. O *token* do usuário serve para evitar a necessidade de envio de *login* e senha em cada requisição autenticada e é gerado pelo servidor após uma requisição de autenticação efetuada com sucesso. Este *token* é enviado para o cliente, e deve ser anexado a todas as requisições autenticadas. É através dele que o sistema verifica se o usuário que envia a requisição está autenticado no sistema e passou por todos os testes de validação.

5.4 Controle de autenticação

Quando um cliente se conecta a um servidor *back end* fornecido pelo servidor *front end*, o primeiro passo é buscar as configurações do serviço desejado, através de uma requisição de configurações. A Figura 20 apresenta um exemplo deste tipo de requisição.

```
1.   <?xml version="1.0" encoding="UTF-8"?>
2.   <body>
3.          <opcode>8</opcode>
4.          <serviceName>nome_serviço_exemplo</serviceName>
5.   </body>
```

Figura 20. Requisição de configuração de um serviço

Ao receber esta requisição o servidor *back end* extrai do XML o campo "*serviceName*" de tipo *string* e consulta os serviços ativos no sistema para buscar as configurações necessárias para que o cliente consiga se conectar e comunicar através do serviço desejado. Caso um serviço com o nome especificado seja encontrado, o servidor retorna ao cliente em um documento XML as seguintes informações: campo de apresentação utilizado para exibir os usuários para o cliente, campo que utilizado para identificar unicamente um usuário, perfil do Twitter atrelado ao serviço para a exibição de notícias e a informação de que o serviço permite ou não registro de usuário remoto. A Figura 21 apresenta a resposta do *back end*.

```
 1.   <?xml version="1.0" encoding="UTF-8"?>
 2.   <body>
 3.       <id>id_unico_da_mensagem</id>
 4.       <opcode>9</opcode>
 5.       <envelope>
 6.           <isValid>true</isValid>
 7.           <configurations>
 8.               <presentationField></presentationField>
 9.               <userIDField></userIDField>
10.               <twitterProfileName></twitterProfileName>
11.               <allowedRegistration></allowedRegistration>
12.           </configurations>
13.           <mappings>
14.               <mapping>
15.                  <realField></realField>
16.                  <friendlyField></friendlyField>
17.                  <values></values>
18.                  <type></type>
19.                  <required></required>
20.               </mapping>
```

```
21.            </mappings>
22.        </envelope>
23.    </body>
```

Figura 21. Resposta a uma requisição de configuração de serviço

Além das informações já descritas, que são padrão para todos os serviços disponibilizados, na mesma resposta ao cliente, o *back-end* também envia uma lista de campos, tipos e possíveis valores utilizados pelo serviço requisitado. Com estas informações em mãos o cliente pode se registrar como um novo usuário do serviço, caso o serviço permita registro remoto ou se autenticar caso ele já possua uma conta no serviço.

Neste ponto o cliente já está apto a enviar uma requisição de autenticação ao *back-end*. Além do campo padrão "*opcode*" e "*id*", a mensagem de autenticação recebe o nome do serviço para o qual o usuário deseja se autenticar, um nome de usuário e uma senha, que é criptografada em SHA1 [Apple 2007] no envio, além de ser armazenada também criptografada no banco de dados administrativo. Isso evita que um invasor em qualquer ponto do sistema, ou ao banco de dados, tenha acesso às senhas dos usuários

5.5 Gerência dos usuários conectados

Quando um usuário se conecta a um serviço disponibilizado por este sistema, ele passa a ter uma conexão persistente ao *back-end* designado a ele e pode tanto receber quanto enviar mensagens para outros usuários do mesmo sistema.

Devido às características do sistema operacional iOS, um aplicativo em *background* pode ter seus *sockets* terminados sem aviso pelo *kernel* do sistema operacional e só pode identificar esta situação ao voltar à execução. Portanto ele deve ser capaz de identificar que suas conexões foram terminadas e restabelecê-las. Isso pode prejudicar a experiência de uso no caso em que o usuário deseja estar conectado ao sistema para receber mensagens a qualquer momento, mas não deseja tomar a iniciativa de enviar uma mensagem.

Por este motivo, o servidor *back end* efetua um controle rigoroso de suas conexões para evitar que recursos do servidor se esgotem e para que usuários inativos sejam removidos do sistema automaticamente, diminuindo assim a possibilidade de ataque ao sistema utilizando o *token* de um outro usuário. Todos os clientes conectados no sistema passam periodicamente por testes que verificam seu estado.

Com uma periodicidade de 10 segundos o sistema envia uma requisição de *keep_alive* aos clientes e aguarda as respostas. Caso um sistema de um usuário não responda estas requisições por mais de 30 segundos, o sistema do *back end* presume que o usuário colocou o aplicativo em *background* e que deve demorar para ativá-lo novamente ou que houve algum erro na conexão. Logo, a conexão do cliente é fechada e o usuário é removido da lista de usuários ativos. Desta forma a capacidade do sistema é mantida em níveis aceitáveis, pois não é necessário manter conexões de clientes inativos e a experiência do usuário não será afetada, visto que quando o aplicativo voltar à execução ele irá identificar que a conexão foi fechada e irá restabelecê-la efetuando todos os procedimentos necessários para encontrar um servidor *back end* disponível, que talvez nem seja o mesmo que o da conexão anterior.

5.6 Permissões de usuários

Neste sistema, qualquer usuário que deseja se comunicar com outro deve ter permissão para isso. Tais permissões são dadas pelo administrador, criador do serviço, na interface administrativa de gerenciamento do sistema. Quando um usuário requisita a lista de usuários disponíveis para conversa, ele só recebe os usuários aos quais ele tem permissão, portanto a tentativa de contato com usuários não permitidos se torna impossível.

O sistema de permissão verifica todas as permissões inseridas na interface de gerenciamento em cada um dos usuários do mesmo serviço. Qualquer usuário que não satisfaça nenhuma das permissões não é inserido na lista que é enviada ao usuário requisitante.

Dentro das classes que formam uma configuração de serviço está a *Permission*, que possui uma lista de Condições "*Conditions*", sendo que cada condição possui um operador e dois operandos, A e B. A verificação de uma condição consiste na comparação do valor do operando A no usuário "*from*" e do valor do operando B no usuário "*to*" através do operador fornecido. Caso o operador B não seja um atributo de banco de dados, mas um número ou texto, a verificação se dá entre o valor do operando A no usuário "*from*" e valor do operando B, não sendo necessária a verificação do usuário "*to*".

6. Integração com o Amazon Elastic Compute Cloud

Devido à característica de escalabilidade do sistema, foi necessário buscar uma ferramenta que atendesse a esta necessidade com confiabilidade e de preferência com um custo-benefício satisfatório. Dentre várias opções disponíveis, a que melhor se adequou ao sistema foi a Amazon Elastic Compute Cloud [Amazon EC2 2013] que além de prover uma interface fácil de gerenciamento baseado na web, possui um sistema que permite aumentar a capacidade do software em minutos, ativando milhares de servidores ao comando do usuário ou automaticamente pelo software através de uma API de gerenciamento [EC2 API 2013] disponibilizada.

A Amazon EC2 possui um sistema de pagamento onde não existe valor mínimo, logo apenas o que for utilizado será cobrado do cliente. Há um nível de serviço gratuito, ideal para sistemas em fase de testes, que fornece os recursos listados na Tabela 3.

Tabela 3. Serviços oferecidos no nível gratuito da Amazon EC2

750 horas de utilização de EC2, instância Linux/Unix Micro
750 horas de utilização de EC2, instância Microsoft Windows Server Micro
750 horas de Elastic Load Balancing além de 15 GB de processamento de dados
30 GB de Amazon Elastic Block Storage (EBS), 2 milhões de E/S e 1 GB de armazenamento de *snapshot*
15 GB de largura de banda para fora agregado em todos os serviços AWS
1 GB de Transferência de dados regional

Foi utilizado neste sistema uma micro instância [Amazon T1 Micro 2013], que está dentro do nível gratuito, possui 600MB de Memória RAM e fornecem uma pequena quantidade de recursos computacionais, mas que podem ter sua capacidade aumentada em pequenas rajadas, quando existirem ciclos disponíveis.

Cada instância micro possui 8GB de armazenamento para sua partição de inicialização e permite que diversos volumes Elastic Block Store [Amazon EBS 2013] sejam montados em determinado caminho do sistema operacional como um disco rígido, para aumentar a capacidade de armazenamento. O EBS funciona independente da vida da instancia que o utiliza. Mesmo que a instância seja terminada, o armazenando em blocos do EBS mantém sua informação intacta. Todo EBS é seguro, pois é automaticamente duplicado dentro de sua zona de disponibilidade. Existe também a possibilidade de criação de *snapshots* de EBS, que funcionam como um ponto de partida para a criação de novos volumes. Um resumo das configurações utilizadas pela instância que disponibiliza o *front end* e no *back end* está listado na Tabela 4.

Tabela 4. Configurações utilizadas para o *Front end* e nos *Back ends*

Tipo de instância	T1.Micro
Sistema operacional	Amazon Linux
Partição de inicialização	1 GB
Área de disponibilidade	sa-east-1ª
Portas TCP abertas no *firewall*	22: Secure Shell (SSH). 8000: Recebimento de configurações. 8886: Recebimento de conexões de *back end*. 8888: Recebimento de conexões de clientes. 8887: Conexões entre *back ends*. 8889: Recebimento de conexões com os clientes.

Para a execução do *back end*, foi criado uma instância no Amazon EC2 configurada para inicializar o aplicativo automaticamente através do *script* /etc/rc.d/rc.local. Qualquer comando inserido neste arquivo é automaticamente executado depois de todos os outros *scripts* de inicialização do Linux.

Utilizando esta configuração o Auto Scaling [Amazon Auto Scaling 2013] junto com o CloudWatch [Amazon CloudWatch 2013] da Amazon podem ser configurados para automaticamente inicializar uma nova cópia de instância de *back end* quando a utilização da CPU de uma instância ultrapassar um nível pré-definido. Desta maneira, novas instâncias automaticamente irão iniciar o aplicativo *back end* que automaticamente irá se registrar no *front end* e passar a disponibilizar seus serviços para clientes.

7. Avaliação

O protótipo desenvolvido foi avaliado através de testes unitários e testes de funcionamento das várias características do sistema. Testes de escalabilidade foram realizados de forma simulada, através da criação de aplicações cliente que executaram operações de registro, conexão e troca de mensagens no sistema. Foi verificado que as opções de configuração adotadas no uso dos serviços da Amazon foram satisfatórias. É

possível ainda o uso de opções de alocação dos back ends em sites geograficamente distribuídos para melhorar a percepção de desempenho do usuário.

Uma outra avaliação foi realizada com 10 usuário alfa-testadores para se constatar a correta operação do sistema e suas funcionalidades. Esta avaliação, ainda que simples, permitiu verificar que as opções e serviços providos pelo sistema funcionam adequadamente.

8. Conclusão

Neste artigo foi apresentada a arquitetura de um sistema que oferece a infraestrutura necessária para aplicações corporativas de troca de mensagens, com suporte a dispositivos móveis. Este sistema possui características de escalabilidade, tolerância a falhas, segurança e privacidade, necessárias para o uso profissional de tais aplicações. Este conjunto de características não é encontrado com facilidade em outras soluções disponíveis.

Servidores *back end* distribuídos atendem às características de escalabilidade e tolerância a falhas. A implantação destes servidores com uma distribuição geográfica coerente, ainda melhora o tempo de resposta verificado pelos clientes. Estes servidores podem, ainda, estar fisicamente dentro da rede coorporativa, isolando sistema da Internet, ou podem ser hospedados na nuvem, e ainda podem ser usadas soluções híbridas de hospedagem.

A troca de mensagens entre os vários elementos do sistema e clientes é suportada por mecanismos de autenticação e criptografia, atendendo aos aspectos de segurança. Algumas outras características importantes para o uso profissional também foram atendidas como a segmentação do acesso dos usuários em grupos, baseados em seu papel ou credenciais e a independência para configuração do comportamento da aplicação através de parâmetros dos serviços suportados pelo sistema.

O uso de um banco de dados fornecido e gerenciado pelo administrador coorporativo oferece independência e flexibilidade na configuração dos vários serviços e características de segurança da aplicação. Através desta abordagem é possível registrar grupos clientes e suas respectivas credenciais, já existentes no banco, em operações simples.

As características incluídas no sistema proposto não são encontradas com facilidade, em conjunto, em sistemas com objetivos similares.

Um protótipo do sistema foi desenvolvido para o sistema operacional iOS da Apple. Devido a uma característica do iOS, foi criada a solução de reconexão ao *back end* pelo aplicativo cliente, já que o sistema operacional pode encerrar os *sockets* dos aplicativos em *background*. Para criar o servidor *front end* e os *back ends*, foi utilizado o serviço de nuvem da Amazon numa versão gratuita.

Em trabalhos futuros alguns elementos podem ser introduzidos no sistema, como políticas para designação de um *back end* para um usuário baseado na geolocalização do mesmo, melhorando assim o tempo de resposta na comunicação. É possível automatizar o escalonamento dos servidores em nuvem utilizando a API fornecida pela Amazon e a inclusão de valor chamado “sal” aleatório ao valor original de senha, impedindo a utilização de uma *rainbow table* [Wikipedia Rainbow Table 2013] para

descobrir a senha de autenticação. O sistema do *front end* já tem sua arquitetura preparada para aceitar políticas distintas para a escolha dos *back ends*, podendo se incluir a localização nestas métricas.

Bibliografia

Amazon Auto Scaling (2013), "Auto Scaling", Amazon Web Services Inc, 2013, http://aws.amazon.com/pt/autoscaling/ [Acesso: Nov/2013]

Amazon CloudWatch (2013), "Amazon CloudWatch", Amazon Web Services Inc, 2013, http://aws.amazon.com/pt/cloudwatch/ [Acesso: Nov/2013]

Amazon EBS (2013), "Amazon Elastic Block Store (EBS)", Amazon Web Services Inc, 2013, http://aws.amazon.com/pt/ebs/ [Acesso: Nov/2013]

Amazon EC2 (2013), "Amazon Elastic Compute Cloud (Amazon EC2)", Amazon Web Services Inc, 2013, http://aws.amazon.com/pt/ec2/,[Acesso: Nov/2013]

Amazon T1 Micro (2013), "Micro Instances – Amazon Elastic Compute Cloud", Amazon Web Services Inc, Outubro, 2013, http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/concepts_micro_instances.html [Acesso: Nov/2013]

Apple (2007), "Mac OS X Manual Page For CC_SHA1", Apple Inc., 2007, https://developer.apple.com/library/ios/documentation/System/Conceptual/ManPages_iPhoneOS/man3/CC_SHA1.3cc.html [Acesso: Nov/2013]

EC2 API (2013), "Amazon EC2 API Tools", Amazon Web Services Inc, Setembro, 2013, http://aws.amazon.com/developertools/351, [Acesso: Nov/2013]

Gamma, E., at all. (1994), Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley.

Greenwald, G, at all (2013), "Microsoft handed the NSA access to encrypted messages", Julho, 2013, http://www.theguardian.com/world/2013/jul/11/microsoft-nsa-collaboration-user-data [Acesso: Nov/2013]

Hangouts (2013), "Google+ Hangouts", Google Inc, http://www.google.com/intl/us-en/+/learnmore/hangouts/ [Acesso: Nov/2013]

IDG Now! (2011), "WhatsApp envia dados sem criptografia e facilita ação de bisbilhoteiros", IDG Now!, Maio, 2011, http://idgnow.uol.com.br/seguranca/2011/05/19/whatsapp-envia-dados-sem-criptografia-e-facilita-acao-de-bisbilhoteiros/ [Acesso: Nov/2013]

Apple Inc. (2014), "iOS 7", 2014, https://www.apple.com/ios/what-is/ [Acesso: Mar / 2014]

Oracle Inc. (2014), "Java SE 6 Release Notes", Oracle, http://www.oracle.com/technetwork/java/javase/webnotes-136672.html [Acesso: Mar / 2014]

Oracle Inc. (2013) "JavaMail", Oracle, http://www.oracle.com/ technetwork/ java/ javamail/ index.html [Acesso: Mar/2014]

Naughton, J. (2013), “Edward Snowden's not the story. The fate of the internet is”, Julho, 2013, http://www.theguardian.com/technology/2013/jul/28/edward-snowden-death-of-internet [Acesso: Nov/2013]

Oracle (2011), “Java Secure Socket Extension (JSSE) Reference Guide”, 2011, http://docs.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html [Acesso: Nov/2013]

PostgreSQL (2014), “PostgreSQL JDBC Driver”, The PostgreSQL Global Development Group, 2014, http://jdbc.postgresql.org/download.html [Acesso: Mar/2014]

Skype (2013), “Skype”, Skype, http://www.skype.com [Acesso: Nov/2013]

Sperberg-McQueen, C. M. and Thompson, H. (2012), “XML Schema”, W3C, Junho, 2012, http://www.w3.org/XML/Schema [Acesso: Nov/2013]

Talkbox Teamwork (2013), “Talkbox Teamwork”, TalkBox Limited., http://talkboxapp.com/teamwork [Acesso: Nov/2013]

WhatsApp (2013), ”WhatsApp”, WhatsApp Inc, http://www.whatsapp.com [Acesso: Nov/2013]

Wikipedia Rainbow Table (2013), “Rainbow table”, Wikipedia, Novembro, 2013, http://en.wikipedia.org/wiki/Rainbow_table [Acesso: Nov/2013]

Winkler, R. (2013) “WhatsApp Surpasses 250 Million Active Users”, Junho, 2013, http://blogs.wsj.com/digits/2013/06/20/whatsapp-surpasses-250-million-active-users/ [Acesso: Nov/2013]

XMPP Standards Foundation (2014), “Extensible Messaging and Presence Protocol (XMPP): Core”, RFC 6120, Março, 2011, http://xmpp.org/rfcs/rfc6120.html [Acesso: Mar/2014]