

FUNDAÇÃO EDSON QUEIROZ

UNIVERSIDADE DE FORTALEZA

UNIFOR

Relatório da versão 1.0 do RTPhone

Aluno: Judah Holanda Correia Lima

Sumário

1. Introdução
2. Arquitetura do Sistema
   1. Atual
      1. Geral
      2. Banco de Dados
      3. Protocolos
   2. Propostas de Evolução de Arquitetura
      1. Geral
      2. Banco de Dados
      3. Protocolos
3. Implementação
   1. IDE
   2. Linguagem
   3. Plataformas
   4. Bibliotecas e Recursos Utilizados
   5. Padrão de Projeto
   6. Uso
4. Instalação e Uso
5. Limitações e Correções
   1. Limitações
   2. Propostas para manutenção
6. Bibliografia

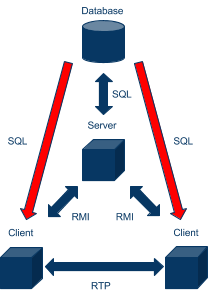
1. Introdução

Este Documento tem como objetivo explicar o funcionamento do RTPhone em sua versão 1.0, mostrar propostas de melhoria e introduzir o padrão de projeto que deve ser utilizado. Além de compor a nota.

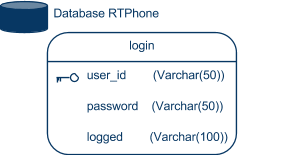
2. Arquitetura do Sistema

# 2.1. Atual

## 2.1.1. Geral



## 2.1.2. Banco de Dados

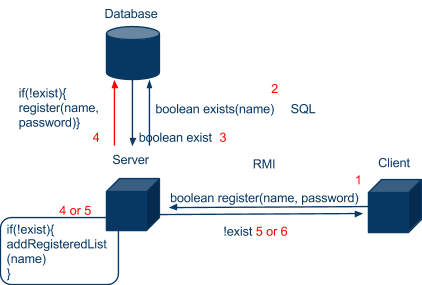


O Banco de Dados atual só possui uma tabela. Cuja tabela possui o username que funciona como chave primária de tabela, uma senha e o IP que se diferente de nulo significa que o usuário está logado.

## 

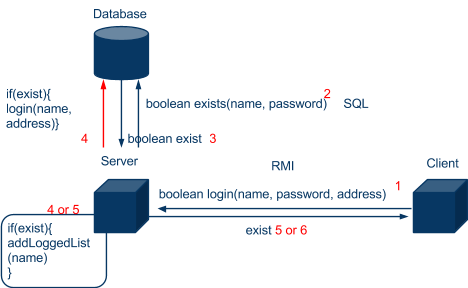
## 2.1.3. Protocolos

### Registro



Para se registrar o Cliente pede para se registrar via RMI para o servidor o retorno da função retornará true pra um registro com sucesso e false caso ocorra erro. Nesta função o Servidor verifica se o usuário já foi registrado, caso tenha, o servidor retorna false, caso não ele o registra e adiciona a lista de registrados e retorna true.

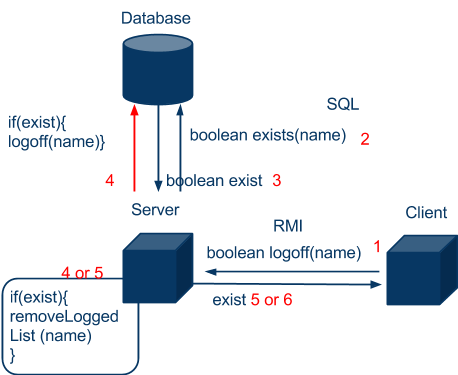
### Login



Para se logar o Cliente pede para se logar via RMI para o servidor o retorno da função retornará true pra um login com sucesso e false caso ocorra erro. Nesta função o Servidor verifica se o usuário não está logado e existe e se a senha está correta, caso esteja, o servidor retorna false, caso não ele o loga e adiciona a lista de logados e retorna true.

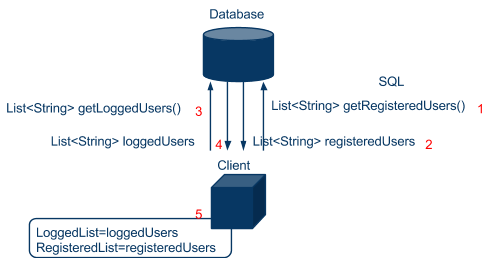
### 

### Logoff



Para se deslogar o Cliente pede para se deslogar via RMI para o servidor o retorno da função retornará true pra um logoff com sucesso e false caso ocorra erro. Nesta função o Servidor verifica se o usuário está logado e existe, caso não esteja, o servidor retorna false, caso esteja ele o desloga e remove a lista de logados e retorna true. Ele é acionado ao fechar a aplicação e isto é seguido em todas as arquiteturas vistas aqui.

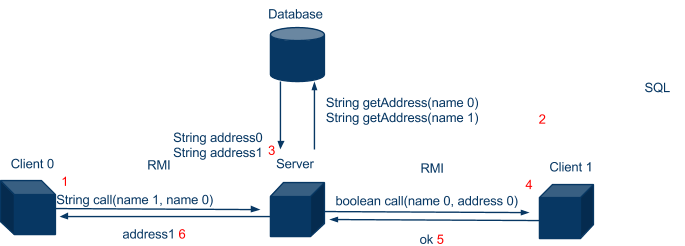
### Refresh



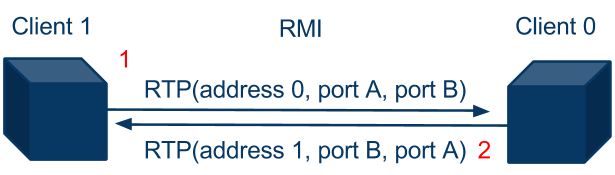
A cada 10 segundo o cliente faz uma requisição pedindo todos os usuários logados e registrados e atualiza a lista, o mesmo acontece com o servidor, mas apenas quando inicia.

### 

### Chamada



Para fazer uma chamada o Cliente que a faz, faz uma requisição ao servidor via RMI, enviando o seu username e o username o qual ele deseja fazer a ligação, então o servidor retorna o endereço IP do usuário o qual o cliente quer se conectar caso o outro cliente aceite a conexão e null caso não. Nesta função o Servidor faz requisições ao Banco de Dados requisitando os endereços IP dos dois clientes, então o servidor se conecta ao outro cliente e faz uma requisição de chamada que retorna se o cliente deseja fazer a conexão ou não (true ou false), enviando o usuário que deseja se conectar e seu endereço IP. Caso seja verdadeiro o Cliente 1 conecta via RTP com o Cliente 0, e após o Cliente 0 receber o endereço IP ele se conecta via RTP ao Cliente 1 usando as portas de forma inversa (A porta que transmitia áudio no Cliente 0, neste agora recebe e vice-versa).

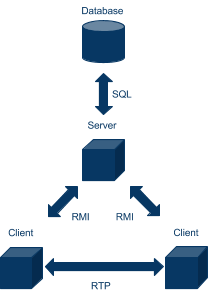


# 2.1. Propostas de Evolução de Arquitetura

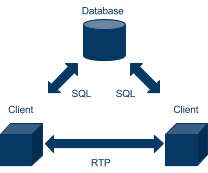
Existem duas propostas óbvias, uma é apenas uma evolução da atual removendo o refresh. A outra é removendo a aplicação Server, ou Utilizando-a apenas como tool, para remover usuários logados e registrado. Deixando a aplicação mais distribuída e retirando a necessidade de uma máquina ligada com o servidor, necessitando apenas conexão com o Banco de Dados que no caso é em nuvem.

## 2.1.1. Geral

### Com Servidor



### Sem Servidor

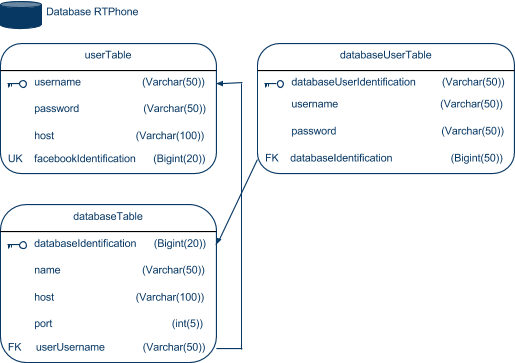


## 2.1.2. Banco de Dados

**Com Servidor**

O Banco continuaria igual.

**Sem Servidor**



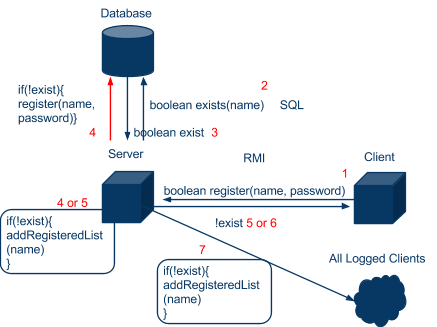
O novo Banco de Dados possuirá três tabelas. Uma tabela possui o username que funciona como chave primária de tabela, uma senha e o IP que se diferente de nulo significa que o usuário está logado e uma unique key com o ID do Facebook do usuário (para integração com o Facebook). Outra terá um conjunto de Banco de Dados que o usuário poderá se logar (a fim de deixar alguns usuários privados de um determinado sistema que tenha esse banco de dados com usuários). E outra que terá um conjunto de usuários e senhas para acessos a estes Bancos de Dados.

## 2.1.3. Protocolos

**Com Servidor**

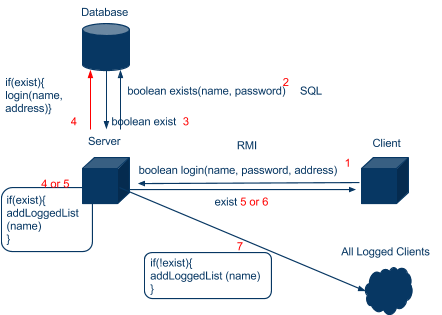
Nesta configuração o servidor faz broadcast aos que estão logados, e caso haja erro de conexão com algum/alguns clientes isto significa que ele/eles não estão mais logados e então o servidor os desloga e envia broadcast a todos os clientes os clientes que se deslogaram.

### Registro



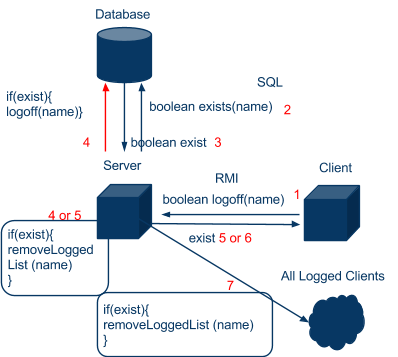
O registro funcionaria da mesma forma, mas após o registro o servidor mandaria um broadcast a todos os clientes os mandando atualizar a tabela de registrados com o novo usuário.

### Login



O login funcionaria da mesma forma, mas após o login o servidor mandaria um broadcast a todos os clientes os mandando atualizarem a tabela de logados com o novo usuário logado.

### Logoff



O logoff funcionaria da mesma forma, mas após o logoff o servidor mandaria um broadcast a todos os clientes os mandando retirarem da tabela de logados o usuário deslogado.

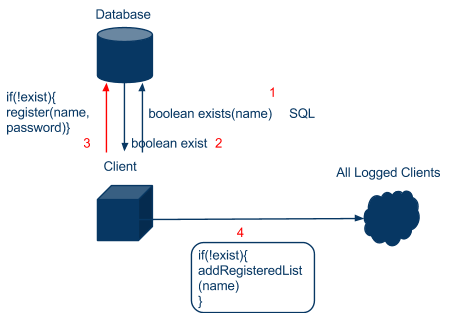
### Chamada

A chamada funcionaria da mesma forma, sem diferenças.

**Sem Servidor**

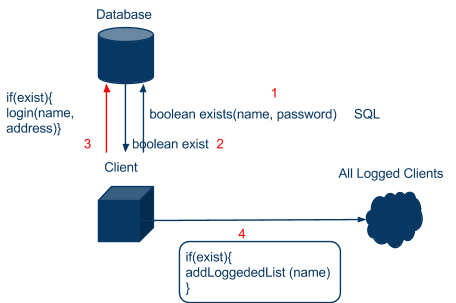
Nesta configuração os próprios clientes fazem broadcast aos que estão logados, e caso haja erro de conexão com algum/alguns clientes isto significa que ele/eles não estão mais logados e este cliente os desloga e envia broadcast a todos os clientes os clientes que se deslogaram.

### Registro



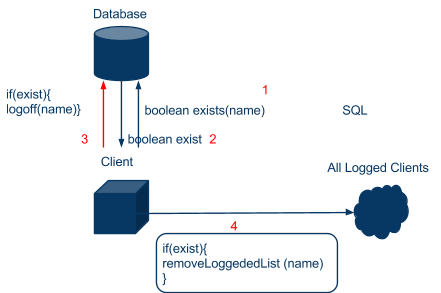
Nesta configuração o próprio cliente vai fazer a consulta no Banco de Dados e verificar se ele existe e caso não exista ele se registra no Banco. Após isto ele faz um broadcast a todos os clientes logados informando que ele acaba de se registrar, para que possam atualizar suas tabelas.

### Login



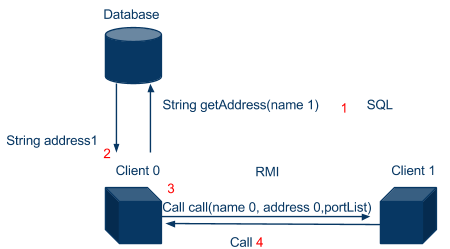
Nesta configuração o próprio cliente vai fazer a consulta no Banco de Dados e verificar se ele existe e se não está logado e se loga no Banco. Após isto ele faz um broadcast a todos os clientes logados informando que ele acaba de se logar, para que possam atualizar suas tabelas.

### Logoff



Nesta configuração o próprio cliente vai fazer a consulta no Banco de Dados e verificar se ele existe e se está logado e se desloga no Banco. Após isto ele faz um broadcast a todos os clientes logados informando que ele acaba de se deslogar, para que possam atualizar suas tabelas.

### Chamada



Nesta configuração o próprio cliente vai fazer a consulta no Banco de Dados e resgatar o endereço do usuário o qual ele deseja se conectar e então chama a função no cliente o qual ele deseja conectar enviando seu username endereço e portas livres para conexão (que nada mais é que uma lista de portas livres para transmissão e recepção), esta função retorna uma classe Call que possui o endereço o qual ele vai se conectar, um áudio de espera (caso haja transferência de chamada), as portas de transmissão e recepção e outros dados de conexão.

3. Implementação

# 3.1. IDE

A IDE utilizada foi o NetBeans. Devido à facilidade de se implementar RMI e devido aos plug-ins que já vem por padrão. Como plug-in para criação e edição de interface gráfica, JDBC e etc.

3.2. Linguagem

A linguagem utilizada foi o Java 7. Devido a sua grande comunidade e documentação além, é claro, das facilidades.

3.3. Plataforma

A plataforma (SO’s e Arquiteturas (x86, x64, alguns Arms com Java convencional)) escolhida foram todas as que o Java dá suporte. Embora se tenha encontrado problemas com o firewall no Windows 8 (Não permite nenhuma conexão nem RMI) e no Linux (Não aceita conexão RTP, mas estabelece quando ele quem a requisita). Nos Sistemas Operacionais Mountain Lion e Windows 7 não houveram problemas. O modelo de interface gráfica escolhido foi a padrão do sistema que utilizará o programa.

3.4. Bibliotecas e Recursos Utilizados

Para comunicação Servidor/Cliente foi utilizado à biblioteca padrão do Java para RMI (Remote Method Invocation). A qual permite uma invocação de métodos remotamente e assincronamente algo que o TCP não permitia facilmente.

Para a conexão RTP foi feita uma pesquisa onde foi encontrado um programa com uma conexão simples de áudio RTP. Então foi extraído o core RTP (que estava fortemente acoplado ao código).

3.5. Padrão de Projeto

Para facilitar a leitura do código algumas regras devem ser seguidas:

1)O código deve ser feito todo em inglês, com exceção dos comentários que podem ser feitos em português por exemplo.

2)Não podem haver siglas em métodos, variáveis e classes, devem ser escritos por extenso.

3)Não pode existir caracteres especiais em métodos, variáveis e classes, como: \_.

4)Deve seguir o notação ”camelo” na declaração e instancia de métodos, variáveis e classes.(Classes devem começar com letra maiúscula e objetos e variáveis com letra minúscula e o resto das letras minúscula, mas para cada nova palavra na variável deve-se começar com maiúscula)

5)Deve-se seguir a identação padrão do NetBeans. (Caso vá utilizar o eclipse no repositório tem o XML para identação).

3.6. Uso

Existem varias maneiras de usar iniciar depois de dado o build no projeto:

-Iniciar:

1)IDE Debug:

Basca clicar no ícone do inseto no NetBeans (ou IDE utilizada)

2)IDE Run:

Basca clicar no ícone do “play” no NetBeans (ou IDE utilizada).

3)Debug:

Entre no terminal:

Digite: cd diretório do projeto/dist

Digite: java -jar projeto.jar

4)Run:

Entre no diretório e basta clicar no jar do projeto

Server:

1-Clique em Start.

2-Caso antes de todos os clientes conectem aparecer algum conectado devido a algum erro na hora de desconectar algum cliente. Clique em Close All Clients.

Client:

1-Digite o IP do Server.

2-Digite o Usuário e senha.

3-Caso o usuário ainda não esteja registrado: Clique em Register.

4-Clique em Login.

5-Caso queira ligar para outro usuário selecione a aba Logged Users e selecione o usuário o qual deseja fazer a ligação, depois de selecionado clique em Call.

6-Para desligar clique em Hung Up (Mesmo que o outro usuário já tenha desligado).

Release: O programa jar executável vai estar na pasta dist então passe apendas para o usuário o conteúdo desta pasta.

4. Instalação e Uso

# Requisitos:

-Internet

-Windows 7 ou Mac (Ou Linux (e com IP configurado no arquivo hosts) e Windows 8 com firewall configurados)

# Instalação:

Não tem, basta ter o programa, o executável está na pasta com a extensão .jar .

# Uso:

## Cliente:

1-Caso esteja no sistema Linux:

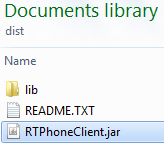
Entre no terminal:

Digite: cd diretório do projeto/dist

Digite: java -jar RTPhoneClient.jar:



Caso não esteja basta clicar no executável:



2-Digite o IP do Server.

3-Digite o Usuário e senha.

4-Caso o usuário ainda não esteja registrado: Clique em Register.

5-Clique em Login.

6-Caso queira ligar para outro usuário selecione a aba Logged Users e selecione o usuário o qual deseja fazer a ligação, depois de selecionado clique em Call.

7-Para desligar clique em Hung Up (Mesmo que o outro usuário já tenha desligado).

## Servidor:

1-Caso esteja no sistema Linux:

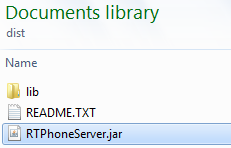
Entre no terminal:

Digite: cd diretório do projeto/dist

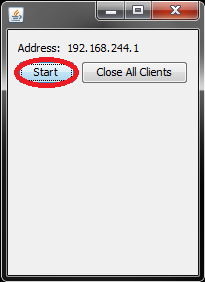
Digite: java -jar RTPhoneServer.jar



Caso não esteja basta clicar no executável:



2-Clique em Start.



3-Caso antes de todos os clientes conectem aparecer algum conectado devido a algum erro na hora de desconectar algum cliente. Clique em Close All Clients.

5. Limitações e Correções

Limitações

1) Devido a baixa capacidade do Banco de Dados em nuvem utilizado existe um número máximo de requisições ao banco e como no refresh dos usuários logados e registrado no banco são feitas requisições ao banco que retornam estes usuários. O cliente fica um tempo sem atualizar.

2)Não existe janela de pergunta para saber se deseja aceitar a ligação, aceitando automaticamente.

3) As janelas de erros não dizem que erro é, nem seu motivo.

Exemplo: Falha no Registro. Usuário já registrado.

4)Não transmite vídeo.

5)Sem conferências.

6)Sem encaminhamento de chamadas.

7)Sem status de usuário.

8)Sem conexão com a linha telefônica.

9)Sem chat de texto nem privado nem salas.

10)Todos os usuários se veem. Não apenas os “amigos”.

11)Sem nome de usuário apenas username.

12)Sem versão mobile.

13)Ausência de semáforos nos métodos do RMI. Que pode causar erros caso exista conexão simultânea.

14) Não se pode deixar uma chamada em espera.

15)Não se pode conectar a mais de um Banco de Dados.

5.2. Propostas de Manutenção

1) Utilizar a nova arquitetura de Protocolo RMI proposta.

6)Utilizar a Classe Call pra encaminha para o IP desejado.

7)Utilizar o Banco de Dados proposto

8)Criar projeto de cliente em firmware com conexão a linha telefônica. Configurável por uma pagina na rede.

9)Criar chat de Texto via RTP. Ou utilizando o Facebook.

10)Utilizar o Facebook.

11)Utilizar o Facebook.

15)Utilizar a arquitetura sem servidor proposta.

+) Separar Conexões ao Banco de Dados em uma nova classe (Database).

+) Renomear Classes que possuem sigla RMI para RemoteMethodInvocation.

6. Bibliografia

* Disponível em <<http://bcook.cs.georgiasouthern.edu/cs523/2taj/index.htm>>

Acesso em 10/2013

* Disponível em <<http://restfb.com/>>

Acesso em 10/2013