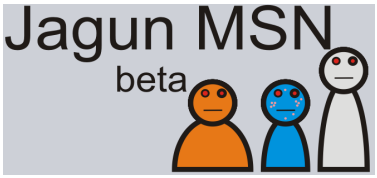
Relatório de Implementação JagunMsn



Índice:

Introdução

JagunMsn é um cliente de mensagens instantâneas desenvolvido como trabalho para a cadeira de Redes de Computadores (CK061) ministrada pelo prof º Miguel

Jagun Msn usa o paradigma Cliente/Servidor, ele pode ser divido em 3 partes:

- JagunMSNClient
- JagunMSNServer
- JagunKet

O Cliente e o Servidor se comunicam atraves do JagunKet, que é baseado em Sockets UDP, só que ao contraio do UDP o JagunKet garante a confiablidade.

Ferramentas usadas

- Desenvolvido em Qt4.5 + C++
- IDE: Qt Creator
- Banco de Dados: Sqlite
- SVN: http://subversion.assembla.com/svn/qcapim/

JagunKet

Socket cliente
O que o socket cliente deve fornecer para a camada acima:

1. Conectar a um servidor
2. Enviar um sinal de conectado
3. Avisar que não pode conecta-se
4. Enviar uma dado ao servidor
5. Avisar que não conseguiu enviar o dado ao servidor
6. Avisar que recebeu um dado do servidor
7. Desconectar
8. Avisar que foi desconectado pelo servidor

Como o socket criará a conexão ao servidor:

1. Cria um pacote do tipo Estabelecer Conexão ao servidor
2. Utiliza-se o sistema de timeout para esperar a resposta do servidor
3. Se não conseguiu estabelecer a conexão envia um sinal a camada acima, que não conseguiu conectar com o servidor
4. Caso receba a resposta, muda o seu estado para conectado
5. Cria os controladores de envio e recebimento
6. Inicializa o buffer de envio

Como o socket enviará um dado ao servidor:

1. Verifica se já está conectado ao servidor. Caso contrario aborta o envio e avisa a camada superior
2. Verifica se o buffer de envio está cheio. Se estiver, aborta a transmissão e avisa a camada superior
3. Caso contrário, coloca o dado na fila (buffer) e chama o método Enviar()
4. O método Enviar() verifica com o controle de envio se já está enviando um dado. Se sim, não faz nada, caso contrário, diz ao controle de envio para enviar o dado
5. Quando o controle de envio terminar de envia um dado, ele avisa a camada socket que a mensagem foi enviada com sucesso.
6. Então o método Enviar() pega o primeiro dado da fila, se existir, e diz ao controle de envio para enviar este dado
7. Caso não foi possível enviar o dado, a controle de envio avisa ao socket
8. O socket, chama o método Enviar(), para tentar enviar a próxima mensagem e avisa a camada superior que não conseguiu enviar o determinado dado

Como o socket receberá um dado do servidor:

1. O servidor enviará um pacote do tipo Inicio de um Novo Dado
2. Verifica-se se o controle de recebimento já está recebendo um dado. Se sim, não faz nada.
3. Caso contrário avisa para o controle de recebimento que vai receber um novo dado.
4. Quando o dado chegar por completo, o controle de recebimento avisa para o socket que um dado chegou do servidor
5. O socket avisa a camada superior que um dado foi enviado pelo servidor

Como o socket fecha a conexão:

1. Verifica se está recebendo ou enviando um dado.
2. Se sim, espera até ser concluído os envios e recebimentos, e fecha a conexão
3. Muda o estado para Sem Conexão
4. A conexão é fechada, enviando ao servidor um pacote do tipo Conexão Fechada e destruindo os controladores de envio e recebimento

Servidor avisa que desconectou:

1. Muda o estado para Sem Conexão
2. Avisa a camada superior que a conexão foi fechada pelo servidor
3. Limpa o buffer e destrói os controladores (?)

Controle de recebimento de pacotes:

1. Verifica qual o tipo do pacote
    - Conexão estabelecida: o estado já é de Conectado, não faz nada. Caso contrário, muda o estado para Conectado, como informado anteriormente
    - Confirmação de recebimento de pacote: envia o pacote ao controlador de envio
    - Inicio de um Novo Dado: faz o que foi informado anteriormente
    - Pacote de dados: já recebeu um pacote de Inicio de um Novo Dado. Se sim, envia para o controlador de recebimento. Se não, descarta
    - Fim de um Novo Dado: envia o pacote para o controlador de recebimento
    - Conexão fechada: faz o que foi informado anteriormente

Os controladores de envio e recebimento podem ser implementados segundo as formas ditas no livro.

O que o controlador de envio deve fornecer ao socket:

1. Responde se já está enviando um dado
2. Enviar um dado
3. Avisa que o dado foi enviado
4. Avisa que não conseguiu enviar o dado
5. Recebe um pacote de confirmação do socket (?)

O que o controlador de recebimento deve fornecer ao socket:

1. Responder se já está recebendo um dado
2. Avisar que o dado chegou por completo
3. Avisar que não pode completar o recebimento de um dado
4. Cancelar recebimento de dados
5. Receber um pacote de dado do socket(?)

Socket Servidor

O que o socket servidor deve oferecer:

1. Ficar escutando uma porta
2. Avisar que existe alguem querendo estabelecer uma conexão
3. Criar um id de conexão para a nova conexão
4. Retornar o gerenciador de conexão, dado um id de Conexão

Como o soket servidor funciona:

1. Fica escutando um determinada porta especificada

2. Caso receba um pacote do tipo Estabelecer Conexão.
3. Verifica se já existe uma conexão para aquele endereço e porta. Caso sim, apenas envia o pacote de Conexão Estabelecida com o id da Conexão já existente. Caso contrário, cria um id da Conexão, e chama o método novaConexão() com este id
4. Caso o pacote seja de outro tipo, pega-se o id de Conexão, e redireciona para o gerenciador de conexão

Como o gerenciador de conexão deve funcionar:

1. Seus mecanismos são idênticos ao socket cliente, com alguns diferencias
2. Não precisa estabelecer conexão, ela já está estabelecida desde o momento da sua criação
3. Não fica escutando uma porta
4. É uma classe filha do socket servidor

A camada superior deve :

1. Estender o socket servidor
2. Implementar o método novaConexão()
3. Criar uma nova conexão, com o id de Conexão fornecido, instanciando um gerenciador de conexão
4. Capturar os sinais vindos do gerenciador de conexão

Tipos de pacotes:

quint8	código
quint32	Id Confirmação

Tabela 1: pacote do tipo Estabelecer Conexão

quint8		código
quint32		Id Confirmação
quint32		Id Conexão

Tabela 2: pacote do tipo Conexão Estabelecida

quint8		código
quint32		Id Conexão

Tabela 3: pacote do tipo Conexão Fechada

quint8		código
quint32		Id Conexão
quint32		Quantidade de pacotes a serem enviados

Tabela 4: pacote do tipo Novo Dado

quint8		código
quint32		Id Conexão

Tabela 5: pacote do tipo Fim da Transmissão do Dado

quint8		código
quint32		Id Conexão
quint32		Sequencial do pacote
quint16		Tamanho do dado
Variável		Dado

Tabela 6: pacote do tipo Dado

quint8		código
quint32		Id Conexão
quint32		Sequencial do pacote

Tabela 7: pacote do tipo Confirmação de Chegada do Dado
Obs.: O tamanho do pacote não pode ultrapassar o tamanho de 512 bytes

Protocolo do JagunMsn

Descrição do funcionamento do protocolo do JagunMsn:

Cliente :

1. Logar
    1. Envia o login, a senha.
    2. Servidor responde sucesso : retorna o último status do usuário no servidor
    3. Servidor reponde erro : código do erro. Ex.: login não existe, senha inválida
    4. Pede a lista de seus contatos.
    5. Servidor responde com a lista de contato e os status de cada contato (login, apelido, status, imagem pessoal, mensagem pessoal),
    6. Servidor envia pedidos de adicionar contato.
2. Criar nova conta
    1. Envia Email,login e senha.
    2. Servidor responde sucesso.
    3. Servidor responde erro : código do erro.
3. Adicionar contato
    1. Envia o login do contato a ser adicionado e uma mensagem de convite
    2. Servidor responde sucesso.
    3. Servidor responde erro : código de erro. Ex.: login não existe
4. Pedido de adicionamento de contato.
    1. Mostra a mensagem recebida e pergunta para o usuário e envia a reposta para o servidor.
5. Mudança de status
    1. Envia novo status para o servidor, pode-se adicionar pessoas específicas que não estão nos seus contatos para receber esse novo status.
6. Mudança de status de um contato
    1. Servidor envia para o cliente o novo status deste contato.
7. Envio de mensagem
    1. Envia a lista de contatos para onde a mensagem vai, data de envio, quem e quando iniciou o bate-papo e a mensagem.
    2. Servidor pode responder com erro : lista as pessoas para onde as mensagens não foram enviadas.
8. Recebimento de mensagem
    1. Servidor envia quem enviou, para quem além de você ele enviou, a data de envio, quem e quando iniciou o bate papo e a mensagem.
9. KeepAlive
    1. A cada x tempo envia a mensagem de keepalive para o servidor.

Servidor:

1. Pedido de Login
    1. Pega o login e a senha.
    2. Verifica se login e senha estão corretos.
    3. Caso correto : pega o último status, imagem,mensagem de exibição, nick e envia para o usuário e também envia os pedidos de adicionamento de contato.
    4. Caso errado : retorna a mensagem de erro para o usuário
2. Nova Conta
    1. Pega o login e a senha e email
    2. Verifica se o login existe, e se login e senhas são válidos
    3. Caso correto: envia o login e a data criada
    4. Caso errado: retorna a mensagem de erro para o usuário
3. Adicionar contato
    1. Pega o contato a ser adicionado, armazena este pedido
    2. Envia este pedido ao destinatário se estiver online
4. Confirmação de adicionamento de contato
    1. Caso positivo : confirma no banco a aceitação, envia o status do contato a quem pediu o adicionamento
    2. Caso negativo: retira o pedido
5. Remover usuário
    1. Remove o usuário da lista de contato
6. Bloqueio de usuário
    1. Atualiza o contato que ele está bloqueado
    2. Envia ao usuário bloqueado o status de offline do usuário que pediu.
7. Desbloqueio de usuário
    1. Atualizar o contato que foi liberado
    2. Envia o status de quem pediu para o contato desbloqueado
8. Mudança de status
    1. Atualiza status
    2. Envia o novo status a sua lista de contatos online
    3. Envia também os status para os contatos listado na mensagem
    4. Se a mensagem for de logout, fecha a conexão entre este usuário
9. Envia de mensagem
    1. Envia a mensagem para os contatos listados na mensagem e que estejam online
    2. Envia mensagem de erro ao remetente dos contatos que ele não conseguiu enviar a mensagem
10. KeepAlive
    1. Não faz nada, apenas mantém a conexão.

Funcionamento Cliente

O cliente manterá dua listas de contatos : uma dos contatos que tem adicionado e outra de convidados dos contatos que o usuário não tem adicionado. Está segunda lista é importante, para armazenar o status de pessoas que não fazem parte da sua lista de contato, mas o usuário está ou estava conversando, no caso de uma conversa em grupo.

1. Ao entrar no programa, este carrega as informações da última vez que entrou.

2. Logar:
    1. Pede o login , senha
    2. O cliente deve criar uma conexão com o servidor.
    3. Caso não consiga se conectar:
        1. Mostra ao usuário que não conseguiu conecta-se.
    4. Quando conectado
        1. Enviar a mensagem de login.
    5. Esperar confirmação
    6. Caso sucesso
        1. Pega as informações que o servidor passou.
        2. Pede a lista de contatos para o servidor.
        3. Recebe a lista e mostra-a.
    7. Caso erro
        1. A conexão será fechada pelo servidor
        2. Verifica qual o erro e mostra mensagem específica

3. Criar nova conta:
    1. Pede o login, senha, email.
    2. Cria uma conexão com o servidor.
    3. Caso não consiga se conectar.
        1. Avisa ao usuário que não conseguiu se conectar.
    4. Envia a mensagem de novo usuário.
    5. Espera confirmação
        1. Caso sucesso:
            1. Mostra aviso de conta criada com sucesso
            2. Entra na parte citada de logar.
        2. Caso erro:
            1. O Servidor fecha a conexão
            2. Verifica qual o erro e mostra a mensagem específica

4. Logoff
    1. Salva as informações do usuário
    2. Envia mensagem de logoff para o servidor
    3. Fecha a conexão
    4. Vai para a janela de logar-se

5. Receber pedido de adicionar contato
    1. Mostra a janela de confirmação, mostrando a mensagem que o possível contato

escreveu e seu login
    2. Caso confirmação do usuário
        1. Envia mensagem de confirmação
        2. O servidor retorna o status do novo contato
        3. Verifica se este contato está na lista de convidados. Se sim, remove-o.
        4. Insere o contato na lista de contatos

6. Enviar pedido de adicionar contato
    1. Pede o login e uma mensagem (não obrigatório)
    2. Envia o pedido para o servidor
    3. Servidor pode retornar erro
    4. Verifica qual foi o erro, e mostra aviso de erro

7. Bloquear contato
    1. Escolhe o contato
    2. Pede confirmação
    3. Caso sim : envia requisição para o servidor
    4. Modifica o status do contato para bloqueado
    5. Procura se existe alguma janela de bate-papo com o contato e altera o status do contato para offline nela

8. Desbloqueio de contato
    1. Escolhe o contato
    2. Envia requisição para o servidor
    3. Servidor retorna sucesso :
        1. Procura se existe alguma janela de bate-papo com o contato e altera o status do contato nela
        2. atualiza o status do contato
    4. Servidor retorna erro : mostra aviso de erro

9. Remover contato
    1. Escolhe o contato
    2. Pede confirmação
    3. Caso sim : envia requisição para o servidor
    4. O Servidor retorna sucesso :
        1. Procura se existe alguma janela de bate-papo com o contato e o retira
        2. remove o contato da lista de contatos
    5. O Servidor retorna erro : mostra aviso de erro

10. Iniciar bate-papo
    1. Escolhe um contato que esteja online
    2. Verifica se já exite uma janela somente com aquele contato
    3. Se sim, só abre a janela
    4. Caso contrário, cria a janela de bate-papo. Armazena a data que criou a janela

11. Enviar mensagem
    1. Envia a mensagem para o servidor para as pessoas que estão no bate-papo
    2. Trata a mensagem
    3. Insere a mensagem tratada no bate-papo

12. Recebimento de uma mensagem

1. Verifica para quem mais a mensagem foi enviada, quem e quando começou o bate-papo
2. Se a conversa for de 1x1 e a contato que enviou a mensagem está bloqueado, descarta a mensagem
3. Verifica se já exite uma janela aberta com essas especificações
4. Caso contrário, cria a janela
5. Verifica se o destinatários estão na sua lista de contatos ou de convidados
6. Para os que não estão, requisita do servidor os status deles e armazena-os na lista de convidados e na janela
7. Trata a mensagem e coloca-a no bate-papo

13. Adicionar contato no bate-papo
    1. Escolhe o contato a ser adicionado
    2. Coloca-o na janela
    3. Envia requisição para o servidor

14. Adicionamento de contato por outro contato no bate-paop
    1. Recebe do servidor o contato a ser adicionado e quem iniciou o bate-papo e quando
    2. Verifica se o contato está na sua lista ou na lista de contato. Caso contrário, pede o status desse contato, o adiciona na lista de convidados
    3. Se existi uma janela que possui os requisitos de quem iniciou o bate-papo e quando iguais insere o contato na janela

15. KeepAlive
    1. Acada x tempo, envia mensagem de keepalive para o servidor

Mensagens do protocolo

Nova Conta

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<NewUser>
<login></login>
<password></password>
</NewUser>
</JaguMSN>
```

Recebido Conta criada

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<UserCreated>
<login></login>
<date></date>
```

```
</UserCreated>
</JagunMSN>
```

Resposta de erro

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<Error code="">
<instructions></instructions>
</Error>
</JagunMSN>
```

Logar

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<LoginUser>
<login></login>
<password></password>
</LoginUser>
</JagunMSN>
```

Resposta de login aceito

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<LogOn>
<login></login>
<nickname></nickname>
<message></message>
<image></image>
<status></status>
</LogOn>
</JagumSN>
```

Adicionar contato

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<AddContact>
```

<contact></contact>
<message></message>
</AddContact>
</JagunMSN>

Recebido pedido de adicionar contato

<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<RequestAddContact>
<from></from>
<message></message>
</RequestAddContact>
</JagunMSN>

Confirmacao de adicionar contato

<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<RequestAddContactConfirmation>
<to></to>
<confirmation></confirmation>
</RequestAddContactConfirmation>
</JagunMSN>

Adicionar contato aceito

<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<ApprovedAddContact>
<login></login>
<nickname></nickname>
<status></status>
<image></image>
<message></message>
</ApprovedAddContact>
</JagunMSN>

Pedir de lista de contatos

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
</JagunMSN>
```

Recebido lista de contatos

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<ContactList>
<Contact>
<user></user>
<nickname></nickname>
<image></image>
<message></message>
<email></email>
<status></status>
<blocked></blocked>
</Contact>
</ContactList>
</JagunMSN>
```

Remover contato

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<Remove>
<to></to>
</Remove>
</JagunMSN>
```

Bloquear ou desbloquear contato

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<Block type="">
<to></to>
</Block>
</JagunMSN>
```

Mudar de status

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<ChangeStatus>
<nickname></nickname>
<status></status>
<image></image>
<message></message>
</ChangeStatus>
<SendToo>
<to></to>
</SendToo>
</JagunMSN>
```

Pedir status de um contato

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<GetStatusContact>
<from></from>
</GetStatusContact>
</JagunMSN>
```

Mudanca de status de um contato

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
```

```
<ChangeStatusContact>
<contact></contact>
<nickname></nickname>
<status></status>
<image></image>
<message></message>
</ChangeStatusContact>
</JagunMSN>
```

KeppAlive

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
</JagunMSN>
```

Enviar mensagem

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<SendMessage>
<SendTo>
<to></to>
</SendTo>
<StartChat>
<who></who>
<date></date>
</StartChat>
<dateSend></dateSend>
<message></message>
</SendMessage>
</JagunMSN>
```

Recebido mensagem

```
<?xml version="1.0" encoding="UTF-8"?>
<JagunMSN>
<Message code="/>
<ReceiveMessage>
<from></from>
<SendToo>
<to></to>
</SendToo>
<StartChat>
```

<who></who>
<date></date>
</StartChat>
<dateSend></dateSend>
<message></message>
</ReceiveMessage>
</JagunMSN>

Modelagem do Banco de Dados

O JagunMSN tem uma classe para tratar todas as consultas feitas no banco, ele usa um SqLite para armazenar as tabelas. Segue uma descrição das tabelas envolvidas.

Tabela usu_usuario:

Tabela que contem as informações do usuário,quando um novo cadastro de usuário é efetuado é atribuído um 'Id' único para esse usuário.

Os campos 'usu_login' e 'usu_pass' são escolhidos pelo usuário no momento do seu cadastro. 'usu_login' é único no sistema, as principais consultas são feitas em cima desse parâmetro.

Os campos da tabela 'usu_nick' 'usu_status' 'usu_mensagem', mudam de acordo com a vontade do usuário. O _nick define o nome de exibição dele para os outros usuarios, _status define o status dele ex: Online, Ocupado e etc. Por ultimo _mensagem define uma mensagem de exibição do usuário ex: “Hoje estou de bom humor”.

O 'usu_laststatus' armazena o ultimo status do usuário antes de ele se deslogar do sistema, esse status sera armazenado para que na próxima vez em que o usuário se logar, automaticamente o sistema mudará _status para _laststatus

Já 'usu_imagem' mantem o nome da imagem de exibição atual do cliente, o cliente tem algumas opções de imagens padrões que ele pode escolher.

O 'usu_idcon' descreve o atual Id de conexão da sessão do usuário, a cada nova conexão com um servidor um novo idcon é atribuído. O valor para quando o usuário estiver offline é -1

Enum de _status e _laststatus:

```
enum ECStatus

{
	ECSOffline = 0, // contato offline
	ECSOnline = 1, // contato online
	ECSBusy = 2, // contato ocupado
	ECSAway = 3, // contato ausente
	ECSInvisible = 4, // contato invisível

}
```

Descrição resumida da tabela

Id	Int(PK)	Campo com o número de identificação único do usuário (PK)
usu_login	Varchar(UNIQUE)	Campo com o identificador usado para login
usu_pass	Varchar	Password do usuário
usu_email	Varchar	Email do usuário
usu_nick	Varchar	Apelido que aparece no programa
usu_status	Int	Status atual do usuário
usu_laststatus	int	Guarda o ultimo Status do usuário
usu_imagem	Int	Imagem de exibição atual do usuário
usu_mensagem	Varchar	Mensagem Pessoal do usuário
usu_idcon	BigInt	Id de conexão atual do usuário

Tabela lsc_listcont:

A função dessa tabela é armazenar as relações de amizades entre os clientes, quando um pedido de adicionar contato é enviado pela aplicação o servidor age da seguinte forma:

- Verificar se o contato que o usuário X quer adicionar realmente existe(Verificar se o usuário Y existe), se não existe retornar uma mensagem de erro para o usuário. O usuário X tem a opção aqui de enviar uma mensagem de convite ao usuário Y ex: “Olá quer ser meu amigo”.

- Se esse contato existir verificar se ambos já não são amigos, se já são amigos retorna uma mensagem de erro para o usuário.

- Se ambos não são amigos é criado 2 relações na Tabela a relação original (X,Y) e a relação inversa (Y,X), ambas tem seu campo de status definido como não confirmado. A mensagem de convite é armazenada somente na relação (X,Y) e assim que o usuário Y se logar no sistema ele receberá um aviso sobre o pedido de adição do usuário X.

- Se Y aceitar o pedido de X, então as 2 relações no banco são definidas como aceitas e agora eles passam a se ver na lista de contato.

- Caso contrario as 2 relações são apagadas do banco.

Essa tabela também é usada para preencher a lista de amigos do usuário, é feita uma consulta ao banco para pegar todos os contatos já aceitos do usuário e após isso são feitas varias consultas para pegar os dados dos contatos.

Os campos 'usu_login1 ' 'usu_login2 ' são os identificadores da relação de amizade, eles indicam que o usuario de _login1 tem ou quer ter o _login2 como amigo.

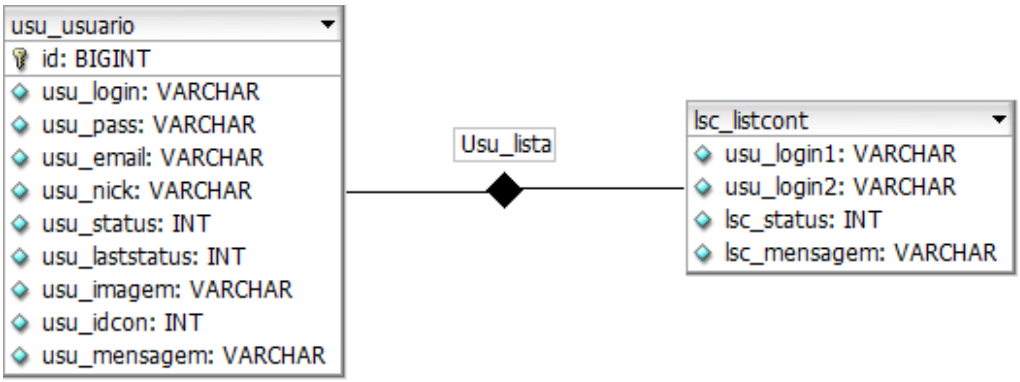
'lsc_status' Descreve o status da relação de amizade entre os contatos.

'lsc_mensagem' Define a mensagem de Convite enviado pelo usuário X ao usuario Y.

Descrição resumida da tabela:

usu_login1	Varchar	Usuario id1 tem como amigo usuario id2
usu_login2	Varchar	*
lsc_status	Int	Status da relação
lsc_mensagem	Varchar	Mensagem de saudação

Modelagem do Banco



Modo de Uso

A seguir a primeira tela que o usuário tem acesso ao iniciar o JagunMsnClient, ela contem os campos necessários para autenticação um botão de login e um botão de novo usuário caso queira se registrar no sistema.

- **Se Logando no Sistema:**

Nome de usuário: Aqui o Cliente indica seu login único para autenticação no sistema.

Senha: Digite aqui senha do usuário.

Entrar: Botão que após precionado tenta logar o usuário no sistema, caso não consiga é retornado uma mensagem de erro.

Novo Usuário: Apertando esse botão levará a um novo formulário .

- **Se cadastrando no sistema**

Caso o cliente queira se cadastrar no sistema ele precionará o botão de “Novo usuário?” e a seguinte tela aparecerá:

Email: Aqui o Cliente indica seu email de contato.

Nome de Usuário: Aqui ele indica seu login de autenticação, esse login é único no sistema.

Senha: Senha para autenticação.

Confirmar Senha: Confirmar a senha digitada anteriormente .

Cadastrar: Tenta Cadastrar um Novo usuário no sistema, é indicado um erro caso algum dos campos seja invalido.

Cancela: Retorna para a tela de Login.

- **Logado no sistema**

Após se logar o usuário tem a opção de mudar seu Nick, que é o nome que aparece para os outros usuários do sistema, sua mensagem pessoal, seu status e escolher sua imagem de exibição.

A primeira caixa de texto Indica o nick atual do usuário, por padrão na primeira vez que ele se loga ao sistema o valor do nick é definido como o Login dele. Para mudar esse valor basta digitar um novo e pressionar ENTER.

O desenho ao lado esquerdo do nome indica o atual status do usuário, para mudar o status basta clicar nele. Já no lado direito podemos escolher a imagem de exibição com um clique uma lista com as possíveis imagem aparece.

A segunda caixa de texto indica a mensagem de exibição do usuário, assim como o nick basta digitar e pressionar ENTER para mudar o valor.

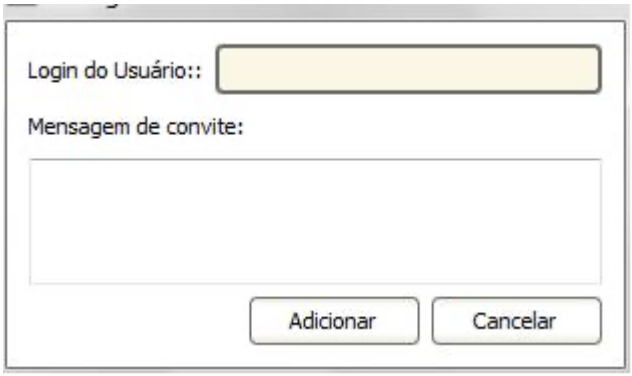
A caixa de Texto com a lupa é a busca por usuários, assim se o mesmo tiver uma grande lista de amigos, basta digitar por exemplo “A” , para serem listados somente os contatos que comecem com a letra “A”

Embaixo da busca de contatos, tem a lista de contatos, aqui é mostrado todos os contatos do usuário junto com as fotos, os nicks,mensagens de exibição desses usuários, com um clique duplo sobre o usuário é possível abrir a janela de conversação com o mesmo.

E por ultimo temos o botão de adicionar usuário.

- **Adicionando um Usuário**

  Após clicar no botão de adicionar surge a seguinte janela:



Aqui o usuário pode indicar o login de identificação do usuário que ele quer adicionar e uma mensagem de convite para esse usuário.

- Conversando com outros usuários

Após escolher qual contato o usuário deseja conversar surge a seguinte janela, nela é possiel enviar e receber mensagens:

Desenvolvedores

- Adyson Magalhães Maia
- Fábio Rodrigo Moura Maia
- Schwarzenegger Alves Lima