CLIENTE WHATSPROG – INTERFACE

REQUISITOS OBRIGATÓRIOS [COMBATE AO PLÁGIO]:

As regras a seguir devem ser OBRIGATORIAMENTE respeitadas, pois representam modificações na especificação atual em relação à dos anos anteriores. O seu cumprimento garante que o aluno implementou realmente a interface e não apenas buscou e apresentou como seu um material antigo ou de outra pessoa. **O descumprimento dessa regra levará à ATRIBUIÇÃO DE NOTA 0,0 (ZERO) ao trabalho**.

- As tabelas de exibição de Conversas e de Mensagens devem ser baseadas em objetos Qt do tipo `QTableWidget`. **Não podem ser empregados objetos do tipo `QTableView` ou modelos do tipo `QAbstractTableModel`**.
- Na tabela de exibição de Conversas, a conversa que teve adicionada a mensagem mais recente (enviada ou recebida) **deve aparecer como a primeira** da lista de Conversas, sem alteração na cor de fundo.
- A conversa que está sendo visualizada deve aparecer na tabela de Conversas com texto (nº de mensagens e nome do usuário) **em negrito** e não necessariamente na primeira posição da lista de conversas, caso não seja a conversa com mensagem mais recente.
- Na tabela de exibição de Conversas, a primeira coluna contém apenas o número de mensagens da conversa, sem qualquer referência ao número de mensagens não lidas.
- A tabela de exibição de Mensagens deve ter **3 colunas**, sendo exibidos na primeira delas os identificadores (id) das mensagens, e na 2ª e 3ª colunas o texto e status.
- Na 3ª coluna da tabela de Mensagens, não deve ser previsto status que represente mensagem visualizada pelo destinatário (2 tiques azuis).

AS VARIÁVEIS GLOBAIS

Como o programa cliente requer threads funcionando em paralelo, algumas variáveis precisam ser declaradas como globais para que possam ser acessadas em todas as threads:

- `tcp_mysocket sock`: o socket de comunicação com o servidor.
- `WhatsProgDadosCliente DC`: a variável que contém todas as mensagens de todas as conversas. A classe `WhatsProgDadosCliente` já está implementada e pode ser reutilizada do programa cliente em modo console.

Essas variáveis podem ser criadas no programa principal (main). Para que possam ser utilizadas em outros arquivos, devem sempre ser declaradas como variáveis externas:

```
#include "mysocket.h"
#include "dados_cliente.h"
extern WhatsProgDadosCliente DC;
extern tcp_mysocket sock;
```

AS CLASSES Qt

Para representação das 3 janelas que compõem a interface do programa WhatsProg (a principal e as 2 janelas que aparecem quando necessário), devem ser criadas 3 classes Qt:

- `WhatsProgMain`: janela principal do programa, sempre em exibição
- `WhatsProgLogin`: janela exibida quando for se conectar ao servidor, informando os dados (IP, login e senha) do usuário.
- `WhatsProgNovaConversa`: janela exibida quando for criar uma nova conversa.

Além essas, será necessário definir uma classe Qt não gráfica (herda apenas de `QObject`, não de `QWidget`) para gerenciar a thread de leitura do socket:

- `WhatsProgThread`: recebe as informações do socket e armazena na variável `DC` que guarda os dados.

OS DADOS DA CLASSE `WhatsProgMain`

Os objetos da classe `WhatsProgMain` têm os seguintes dados membro, todos privados:

- Um ponteiro `thread` do tipo `WhatsProgThread*`.
- Um ponteiro `loginDialog` do tipo `WhatsProgLogin*`.
- Um ponteiro `novaConversa` do tipo `WhatsProgNovaConversa*`.
- Um ponteiro `msgStatus`, do tipo `QLabel*`, para exibir na barra de status.
- Quatro `QPixmap` para armazenar os ícones com os 4 possíveis status de uma mensagem (enviada, recebida, entregue e outro).

COMUNICAÇÃO ENTRE AS CLASSES:

Para cada sinal da classe remetente, deve haver um slot correspondente na classe destinatária, sendo feita a conexão (`connect`) entre essas duas entidades.

Sinais enviados pela classe `WhatsProgMain`:

- Após ação “Novo usuário” -> classe `WhatsProgLogin` (exibe janela, novo usuário)
- Após ação “Usuário existente” -> classe `WhatsProgLogin` (exibe janela, usuário existente)
- Após ação “Nova conversa” -> classe `WhatsProgNovaConversa` (exibe janela)
- Ao aceitar um novo usuário, em resposta a um sinal da classe `WhatsProgLogin` -> classe `WhatsProgThread` (iniciar thread)
- Após ação “Desconectar” -> classe `WhatsProgThread` (encerrar thread)

Sinais enviados pela classe `WhatsProgLogin`:

- Após aceitar (pressionar OK) parâmetros do novo usuário -> classe `WhatsProgMain` (tenta conexão, reexibe interface e status)

Sinais enviados pela classe `WhatsProgNovaConversa`:

- Após aceitar criação (pressionar OK) de uma nova conversa -> classe `WhatsProgMain` (reexibe interface)

Sinais enviados pela classe `WhatsProgThread`:

- Após criar nova conversa -> classe `WhatsProgMain` (reexibe interface)
- Após receber nova mensagem -> classe `WhatsProgMain` (reexibe interface)
- Após receber confirmação de mensagem recebida ou entregue da conversa que está aberta e sendo visualizada -> classe `WhatsProgMain` (reexibe)
- Após receber aviso de erro em mensagem -> classe `WhatsProgMain` (reexibe interface e exibe mensagem de erro)
- Após erro no salvamento periódico dos dados -> classe `WhatsProgMain` (exibe mensagem de erro)
- Quando a thread for encerrada -> classe `WhatsProgMain` (mudar interface para modo cliente desconectado)

EXIBIÇÃO DOS DADOS

A origem da informação a ser exibida nas tabelas de conversas e mensagens é a variável `DC`, do tipo `WhatsProgDadosCliente`. Sempre que um novo comando é recebido ou enviado, a informação correspondente deve ser modificada em `DC` e, em seguida, a interface deve utilizar os métodos de consulta da classe `WhatsProgDadosCliente` para recuperar a informação e exibi-la no widget (`QLabel`, etc.) correspondente.

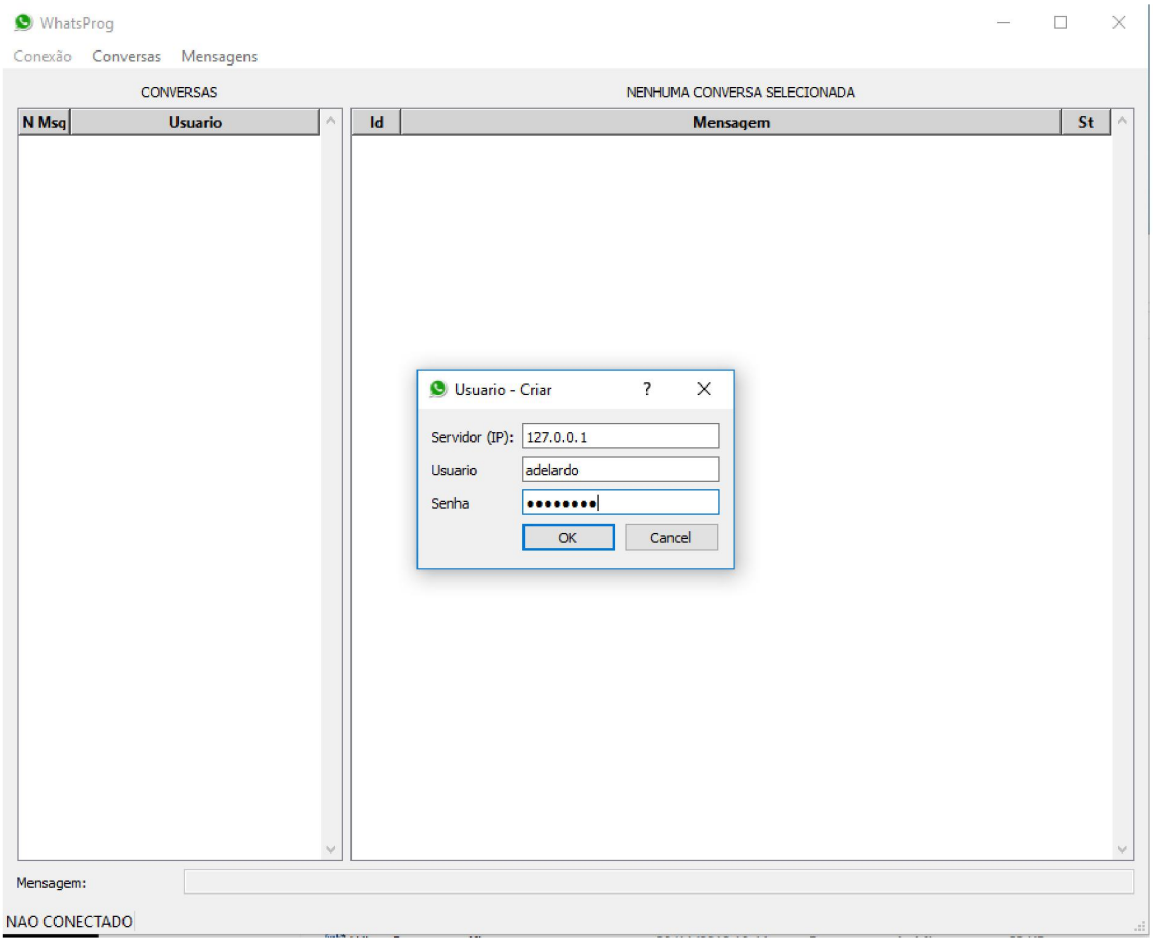
ATENÇÃO

O fato de alterar o conteúdo (`QLabel`, etc.) que está armazenado em uma das células de uma `QTableWidget` **não** faz com que a exibição da tabela seja automaticamente atualizada para refletir o novo conteúdo da célula. Para isso, é necessário chamar:

```
ui->widget_da_tabela->viewport()->update();
```

após ter feito as alterações.

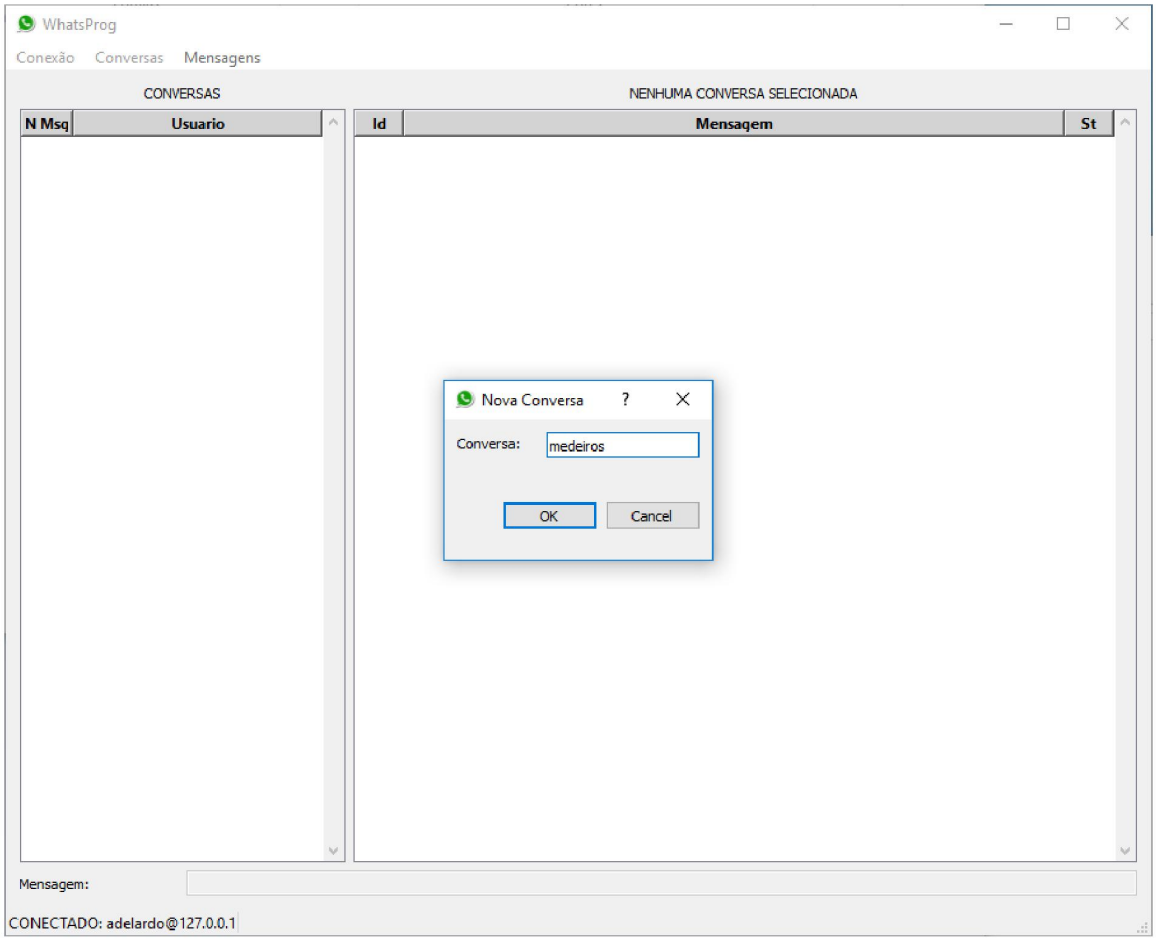
1. Janela de criação de novo usuário



Observações:

- O programa cliente está desconectado. Portanto:
  - A mensagem correspondente aparece na barra de status.
  - O menu Conversas está desabilitado.
  - A lista de Conversas está vazia.
  - Nenhuma conversa é exibida.
  - O menu Mensagens está desabilitado.

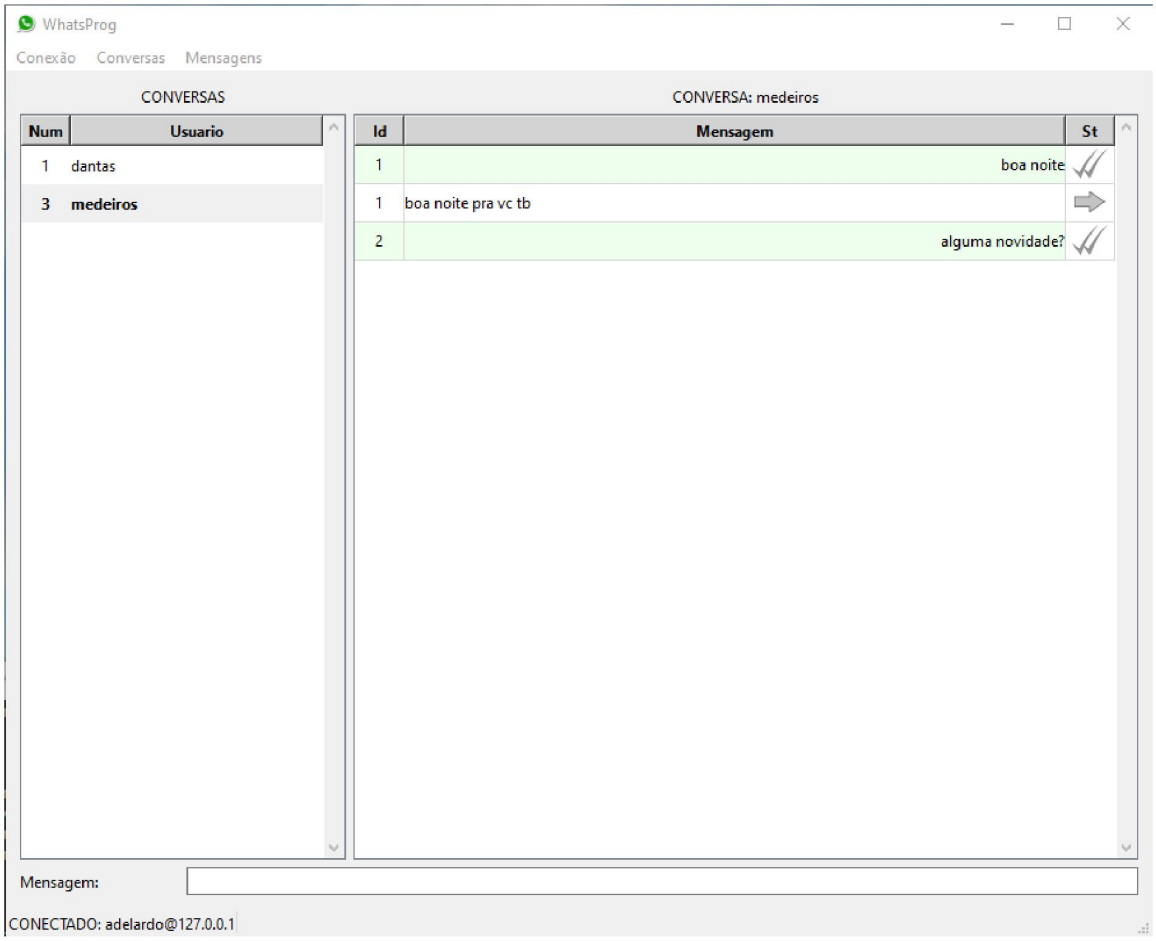
2. Janela de criação de nova conversa



Observações:

- O programa cliente está conectado. Portanto:
  - A mensagem correspondente (usuário@servidor) aparece na barra de status.
  - O menu Conversas está habilitado.
- A lista de Conversas está vazia. Portanto:
  - Nenhuma conversa é exibida.
  - O menu Mensagens está desabilitado.

3. Exibição de conversa



Observações:

- A conversa “medeiros” está selecionada, sendo exibidas suas mensagens.
    - Como ela é a conversa que está sendo exibida na janela de Mensagens, o texto aparece em negrito e seu nome aparece acima da janela de Mensagens.
    - A conversa tem 3 mensagens no total.
    - Das 3 mensagens, 1 é de autoria do correspondente (medeiros) e aparece em branco e alinhada à esquerda. As outras 2 são de minha autoria (Adelardo) e aparecem em fundo verde e alinhadas à direita. Todas as mensagens de minha autoria já foram entregues.
- A conversa “dantas” não está selecionada e, portanto, suas mensagens não são exibidas.
    - A conversa tem 1 mensagem no total.
    - A última mensagem recebida ou enviada foi na conversa “dantas”, o que faz com que ela apareça na primeira posição na lista de conversas.