DOCUMENTAÇÃO DA APLICAÇÃO: “SISTEMA DE BANCOS”

Projeto de Implementação: Sistema Distribuído Simples para Manutenção de Contas Bancárias

Trabalho feito pelo aluno Iago Pinto, matrícula 201523035 do curso Ciências da Computação para a disciplina MATA88 ministrado pelo professor Raimundo Macedo.

Foi solicitada uma aplicação com arquitetura cliente-servidor para um problema de um sistema de bancos com o uso de servidor e clientes.

O código foi implementado utilizando o framework 'QtDesigner' para desenhar e definir o layout, a biblioteca 'PyQt5' para converter os elementos do layout em código para importar para o código no Python e as bibliotecas 'Socket' e 'Thread' para definir e estabelecer a comunicação entre o servidor e aplicação para cada cliente.

Instalação

A versão do Python que foi utilizado na implementação do trabalho é 3.9.9 (https://www.python.org/dev/peps/pep-0596/).

Para instalar os pacotes abaixo, é necessário utilizar o pip (https://pip.pypa.io/en/stable/).

Como citado acima, foi utilizado três bibliotecas. Contudo, o socket e o threading são módulos nativos do Python.

A biblioteca PyQt5 não é nativo do Python 3.9.9 e é necessário a instalação.
Para isso, foi utilizado o comando para obter a biblioteca.

```
python3 -m pip install PyQt5
```

Código

O código dos trabalhos estão comentados e disponíveis nos links abaixo.

Arquivo do Cliente - /cliente.py

Arquivo do Servidor - /servidor.py

Demonstração de Uso - Vídeo

[Vídeo](https://youtu.be/4fHmGWs34fA)

Exemplo de Uso - Imagens

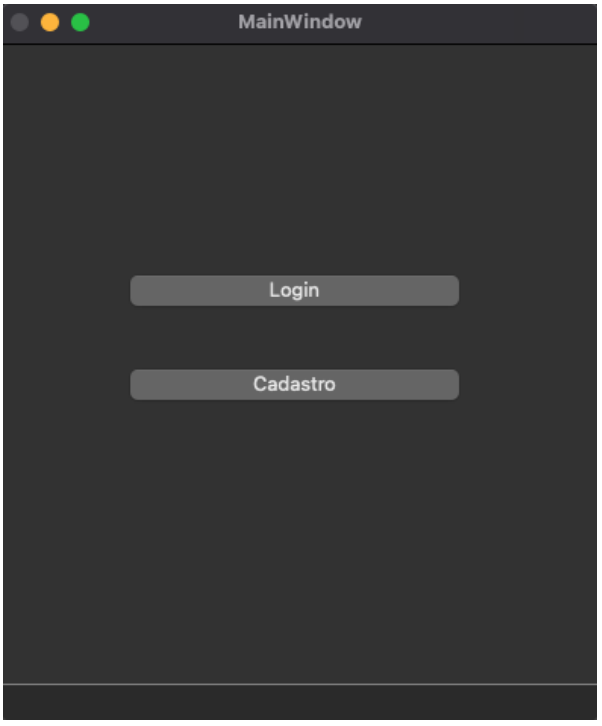
1. Inicialmente, roda o arquivo do servidor:
- Executável/servidor.exe

  - Caso não esteja funcionando o executável acima, pode rodar o arquivo em python servidor.py

2. Logo após, roda o arquivo do cliente:
- Executável/servidor.exe

  - Caso não esteja funcionando o executável acima, pode rodar o arquivo em python cliente.py

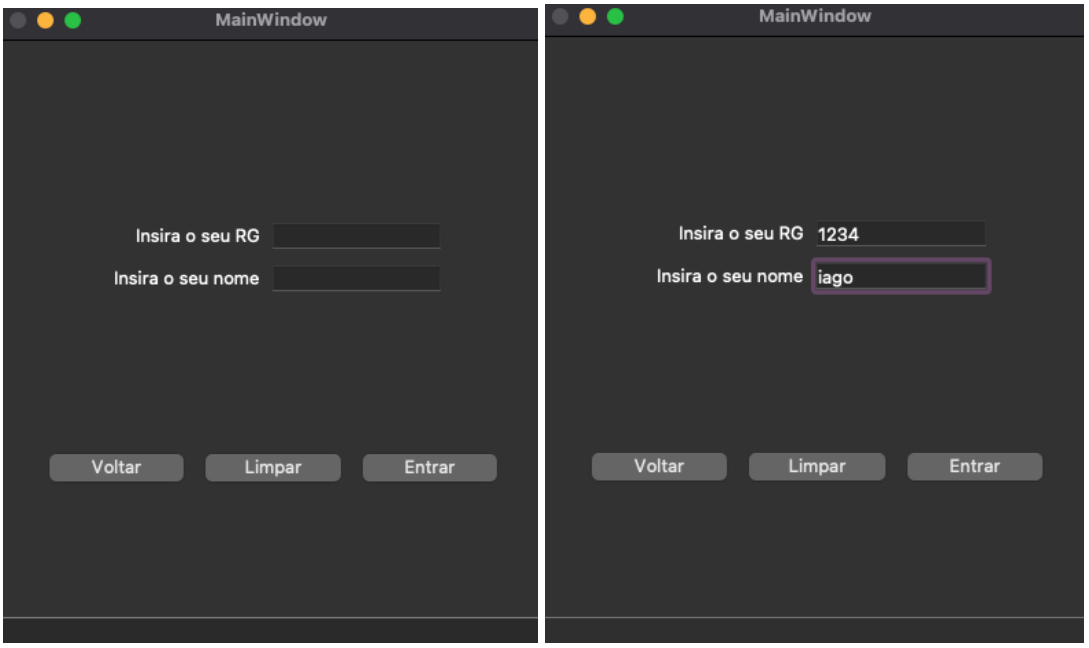
3. Aparece a tela inicial. Há dois botões: um de Login e o outro de Cadastro.



4. Como não há nenhum cliente ativo, é necessário clicar no botão "Cadastro".

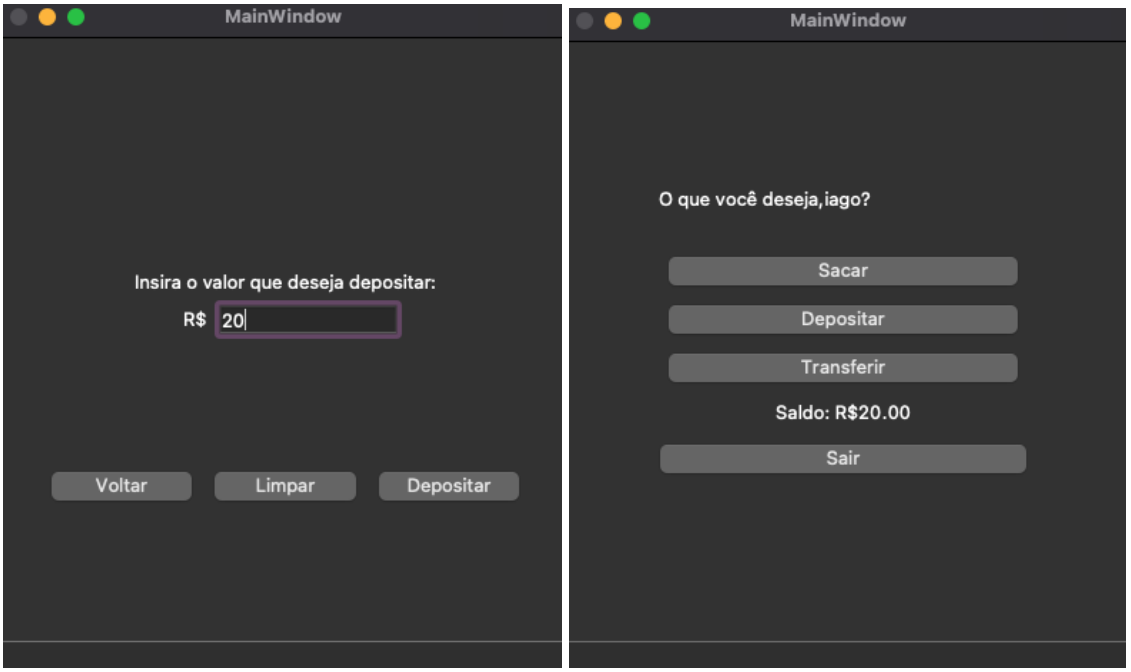
5. Na tela de Cadastro, há dois campos: nome e RG. Há também três botões, um de "Voltar", um de "Limpar" e outro de "Entrar"

- no botão de "Voltar", volta para a tela inicial
- no botão de "Limpar", limpa todos os campos
- no botão de "Voltar", verifica se há o mesmo ID cadastrado. Se não houver, salva e cadastra as instâncias do cliente e direciona para a tela de Operações.

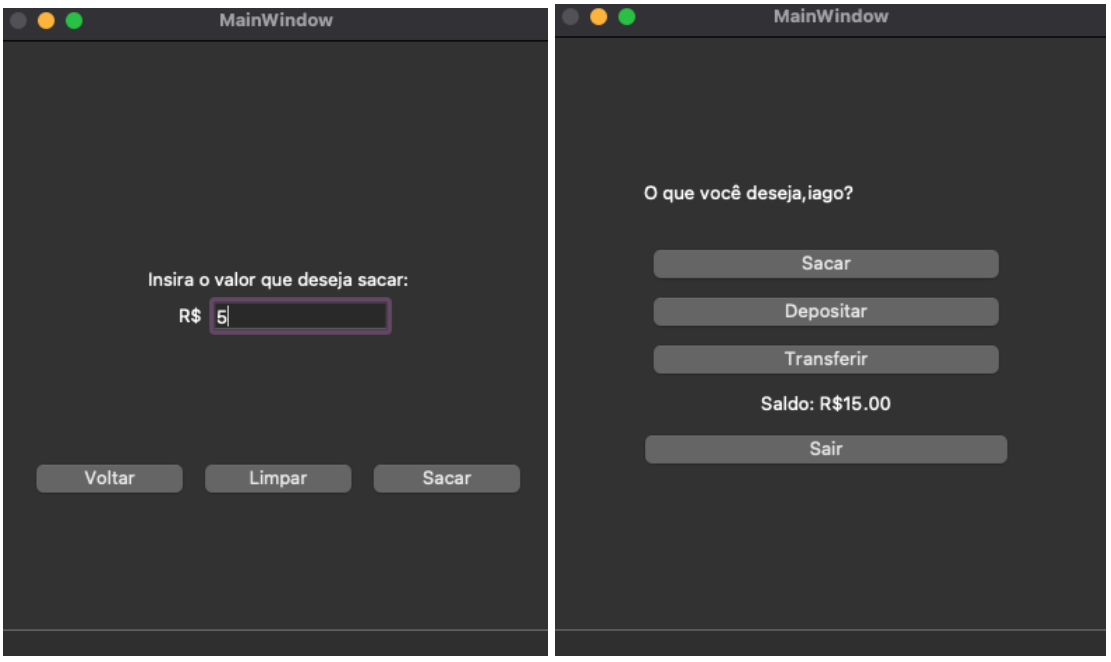


6. Na tela de Operações, há uma mensagem com o nome do cliente, recebido diretamente do servidor. Há também três opções: Sacar, Depositar e Transferir. Sacar subtrai do saldo atual do cliente; Depositar adiciona o valor para o saldo atual e Transferir deposita o valor na conta de outro cliente. Todos as as operações são realizadas diretamente no servidor.

7. Para depositar, há um campo de dinheiro, em reais, para adicionar ao valor atual. Há três botões, um de "Voltar", um de "Limpar" e outro de "Depositar"
- no botão de "Voltar", volta para a tela de Operações
- no botão de "Limpar", limpa todos os campos
- no botão de "Depositar", efetua a operação de depósito. Essa operação é realizada no servidor. Logo após, direciona para a tela de Operações.



8. Para sacar, há um campo de dinheiro, em reais, para substrair o valor atual. Há três botões, um de "Voltar", um de "Limpar" e outro de "Sacar"
- no botão de "Voltar", volta para a tela de Operações
- no botão de "Limpar", limpa todos os campos
- no botão de "Voltar", efetua a operação de saque. Essa operação é realizada no servidor. Logo após, direciona para a tela de Operações.



9. Para transferir, há um campo de dinheiro, em reais, para depositar o valor na conta do destinatário. Há outro campo contendo o RG do destinatário. Também há três botões, um de "Voltar", um de "Limpar" e outro de "Transferir"
   - no botão de "Voltar", volta para a tela de Operações
   - no botão de "Limpar", limpa todos os campos
   - no botão de "Transferir", efetua a operação de transferência. Essa operação é realizada se houver um RG válido no servidor e também é efetuada lá. Logo após, direciona para a tela de Operações.

10. Por último, há o botão de "Sair" que encerra a conexão do cliente atual com o servidor.

servidor.py

```
import socket
from threading import Thread

class Server:
   def __init__(self):
       ### Instancia um soquete para se comunicar com os clientes ###
       server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

       ### Esse método  conecta  o  servidor  para que receba ###
       ### requisições em um endereço e uma porta específicos ###
       server.bind(('127.0.0.1', 9999))

       ### Configura o soquete para ficar no modo de escuta, aguardando conexões
###
       server.listen(5)

       ### Salva a conexão do servidor ###
       self.server = server

       ### Cria uma lista de objetos de clientes ###
       self.li = []
       self.clients = {}

       ### Método para obter todas as conexões requisitidas ###
       self.get_con()

   ### Aguarda todas as conexões requisitadas por clientes, para estabelecer uma
conexão ###
   def get_con(self):
       print("get connection")

       while 1:
           ### Estabele a conexão com o cliente ###
           con, addr = self.server.accept()
           ### Responde ao método begin_thread() do cliente ###
           ### É  somente  para  dar um ping / uma resposta ###
           data = ""
           con.send(data.encode())
           ### Inicia uma thread para habilitar a troca ###
           ### de mensagens com  cada cliente conectado ###
           Thread(target=self.get_msg, args=(con, self.li, addr)).start()

           ### Adiciona a conexão em uma lista de conexões ###
           self.li.append(con)
```

```
   ### Método para realizar a troca de mensagens ###
   ### com o cliente/conexão respectiva ###
   def get_msg(self, con, li, addr):
       print("get msg")

       ### Laço para receber as credenciais do cliente e checar se é cadastro ou
login ###
       while 1:
           ### Recebe a mensagem com as credenciais do cliente ###
           name = con.recv(1024).decode()
           ### Se há um caractere '#',  é porque a operação é  'Cadastro' ###
           ### E foi passado o id e o nome do atual cliente como mensagem ###
           if '#' in name:

               ### Contudo,  se o  id do  cliente já  estiver  na  lista de
clientes, ###
               ### o cadastro é negado, pois já possui um outro usuário com o mesmo
id ###
               ### e retornará  uma mensagem de erro com status  '400'  para o
cliente ###
               ### Caso contrário,  o usuário terá suas credenciais salvas no
servidor ###
               ### e será enviado uma resposta com status '200' com os ids
respectivos ###
               id = name.split("#")

               if id[0] in self.clients:
                   msg = "400"
                   ##print(msg)
                   con.send(msg.encode())
               else:
                   self.clients[id[0]] = {}
                   self.clients[id[0]]["rg"] = id[0]
                   self.clients[id[0]]["nome"] = id[1]
                   self.clients[id[0]]["saldo"] = 0
                   self.clients[id[0]]["conn"] = con
                   msg = "200#" + id[0] + "/" + id[1]
                   con.send(msg.encode())
                   break

           ### Se não houver o caractere '#', então a operação a ser realizada é o
login ###
           ### Se o  cliente já esteja  previamente cadastrado,  ele loga no
sistema e é ###
```

```
            ### enviado  uma mensagem com  o status  '200'  e as credenciais
respectivas ###
            ### Se não, somente é enviado  uma mensagem com o status '401" para o
cliente ###

            else:
                if name in self.clients:
                    msg = "200#" + name + "/" + self.clients[name]["nome"]
                    con.send(msg.encode())
                    break
                else:
                    msg = "401"
                    con.send(msg.encode())

        ### Realiza um  recebimento  e envio de  mensagens cíclicas, indefinidamente
até que ###
        ### a conexão seja  encerrada.  Então, a cada mensagem  enviada pelo
cliente,  haverá ###
        ### uma resposta enviada do servidor dependendo de cada mensagem enviada
pelo cliente ###
        while 1:
            redata = ""
            ##print("ok")
            try:
                ### Recebe  as requisições  enviadas  pelo cliente  através ###
                ### de uma  mensagem.  Com um parser  aplicado,  é obtido a ###
                ### operação, o id do cliente e um valor numeral, se houver ###
                print("waiting answer")
                redata = con.recv(1024).decode()
                ### id[0] é a operação
                ### id[1] é o id do cliente
                ### id[2] é o valor que foi pasasdo em R$, se houver

                ### Há três operações  a serem realizadas:  a de checar o saldo. ###
                ### de realizar  um depósito  ou uma  transferência.  A operação ###
                ### de depósito é feita para a conta do próprio cliente, com uma ###
                ### operação  de somar o  valor recebido  e obter um novo saldo. ###
                ### A operação  de sacar é uma operação  de subtração  do  valor ###
                ### do cliente com o valor que foi inserido,  retornando um novo ###
                ### valor.  A operação de saldo  envia o saldo atual  ao cliente ###

                ### A operação  de  transferência  envia para  o  servidor uma
string ###
                ### contendo duas operações a serem realizadas:  de  depósito  para
o ###
```

```
                #### destinatário somando o valor que  foi inserido com o saldo
atual ###
                ### com o valor inserido,  e de saque,  que subtrai o  saldo atual
do ###
                ### que enviou o valor para o destinatário ###
                ##print(redata)
                if "#" in redata:
                    msg = redata.split("#")
                    if msg[0] == "saldo":
                        ##print(msg[0], msg[1])
                        msg = str(self.clients[msg[1]]["saldo"])
                        con.send(msg.encode())
                    if msg[0] == "deposito":
                        ##print(msg[0], msg[1], msg[2])
                        self.clients[msg[1]]["saldo"] += float(msg[2])
                        if len(msg) > 3:
                            msg = msg[3:6]
                    if msg[0] == "sacar":
                        ##print(msg[0], msg[1], msg[2])
                        self.clients[msg[1]]["saldo"] -= float(msg[2])

            ### Se ocorrer algum erro durante as operações ###
            ### o servidor encerra a conexão com o cliente ###
            except Exception as e:
                print(e)
                for c in self.clients:
                    if con == self.clients[c]["conn"]:
                        self.clients[c]["conn"] = False
                        break
                self.close_client(con, addr)
                break

            ### Se o cliente encerrar  a  aplicação,  é enviado um comando ###
            ### para que o servidor possa encerrar a conexão com o cliente ###
            if (redata.upper() == "QUIT"):
                for c in self.clients:
                    if con == self.clients[c]["conn"]:
                        self.clients[c]["conn"] = False
                        break
                self.close_client(con, addr)
                break

    ### Encerra a conexão com o cliente ###
    def close_client(self, con, addr):
        print("close_client")
```

```
        self.li.remove(con)
        con.close()

if __name__ == "__main__":
    Server()
```

cliente.py

```
### Trabalho feito pelo aluno Iago Pinto, matrícula 201523035 ###
### do curso Ciências da Computação para a  disciplina MATA88 ###
### ministrado pelo professor Raimundo Macedo. Foi solicitada ###
### uma aplicação com arquitetura cliente-servidor.  O codigo ###
### foi implementado utilizando o framework 'QtDesigner' para ###
### definir  o layout,  a biblioteca  'PyQt5'  para converter ###
### os elementos do layout em codigo para executar no Python  ###
### e as bibliotecas 'Socket' e 'Thread' para definir e esta- ###
### belecer a comunicação  entre  o servidor e aplicação para ###
### cada cliente.                                             ###



import socket
from threading import Thread
from PyQt5 import QtGui, QtWidgets
import re
from objetos_do_layout import *
import sys
from functools import partial

## Define a classe de Exibição Inicial para inicializar a tela
## e algumas interações com botões
class Client(QtWidgets.QMainWindow, Ui_Program):
    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        ### Instancia um soquete para se comunicar com o servidor  ###
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        ### Estabelece  a conexão com um endereço e uma porta específicos  para ###
        ### enviar solicitar requisições ao servidor com os parâmetros definido ###
        client.connect(('127.0.0.1', 9999))

        ### Salva a instância da conexão para o atual cliente ###
        self.client = client
        ### Cria o campo de RG e nome para salvar as credencias do cliente ###
```

```
        self.rg = ""
        self.nome = ""
        # enable custom window hint
        self.setWindowFlags(self.windowFlags() | QtCore.Qt.CustomizeWindowHint)
        # disable (but not hide) close button
        self.setWindowFlags(self.windowFlags() & ~QtCore.Qt.WindowCloseButtonHint)

        ################################################
        ## Botões que acessam opções na tela Inicial ##
        ################################################
        self.b_login.clicked.connect(self.goLoginScreen)
        self.b_cadastro.clicked.connect(self.goCadastroScreen)

        ##############################################
        ## Botões que acessam opções na tela Login ##
        ##############################################
        self.b_voltar_login.clicked.connect(partial(self.goInicio, self.w_login))
        self.b_limpar_login.clicked.connect(partial(self.goLimparLC,
self.t_rg_login))
        self.b_entrar_login.clicked.connect(partial(self.goLogin, self.w_login,
self.t_rg_login))

        ################################################
        ## Botões que acessam opções na tela Cadastro ##
        ################################################
        self.b_voltar_cadastro.clicked.connect(partial(self.goInicio,
self.w_cadastro))
        self.b_limpar_cadastro.clicked.connect(partial(self.goLimparLC,
self.t_rg_cadastro, self.t_nome_cadastro))
        self.b_entrar_cadastro.clicked.connect(partial(self.goCadastro,
self.w_cadastro, self.t_rg_cadastro, self.t_nome_cadastro))

        ################################################
        ## Botões que acessam opções na tela Operacoes ##
        ################################################
        self.b_sacar.clicked.connect(self.goSacarScreen)
        self.b_depositar_2.clicked.connect(self.goDepositarScreen)
        self.b_transferir.clicked.connect(self.goTransferirScreen)
        self.b_sair.clicked.connect(self.goSair)

        ################################################
        ## Botões que acessam opções na tela Depositar ##
        ################################################
        self.b_voltar_2.clicked.connect(partial(self.goOperacoes, self.w_depositar))
        self.b_limpar_2.clicked.connect(partial(self.goLimparLC, self.t_rs))
```

```
        self.b_depositar.clicked.connect(partial(self.goDepositar, self.w_depositar,
self.t_rs))

        ################################################
        ## Botões que acessam opções na tela Transferir ##
        ################################################
        self.b_voltar_4.clicked.connect(partial(self.goOperacoes,
self.w_transferir))
        self.b_limpar_4.clicked.connect(partial(self.goLimparLC, self.t_rs_2,
self.t_rg_2))
        self.b_transferir_2.clicked.connect(partial(self.goTransferir,
self.w_cadastro, self.t_rg_2, self.t_rs_2))

        ############################################
        ## Botões que acessam opções na tela Sacar ##
        ############################################
        self.b_voltar_5.clicked.connect(partial(self.goOperacoes, self.w_sacar))
        self.b_limpar_5.clicked.connect(partial(self.goLimparLC, self.t_rs_3))
        self.b_sacar_2.clicked.connect(partial(self.goSacar, self.w_sacar,
self.t_rs_3))

    ###################################################
    ## Acessando novas telas a partir da tela Inicial ##
    ###################################################
    ### Exibe a tela de Login ###
    def goLoginScreen(self):
        self.w_bemvindo.hide()
        self.w_login.show()

    ### Exibe a tela de Cadastro ###
    def goCadastroScreen(self):
        self.w_bemvindo.hide()
        self.w_cadastro.show()

    #########################################################
    ## Acessando novas telas a partir da tela Operações ##
    #########################################################
    ### Exibe a tela de Sacar ###
    def goSacarScreen(self):
        self.w_sacar.show()
        self.w_operacoes.hide()

    ### Exibe a tela de Deposito ###
    def goDepositarScreen(self):
        self.w_depositar.show()
```

```
        self.w_operacoes.hide()

    ### Exibe a tela de Transferência ###
    def goTransferirScreen(self):
        self.w_transferir.show()
        self.w_operacoes.hide()

    ##################
    ## Tela de Erro ##
    ##################
    ### Exibe a tela de erro, adicionando a mensagem de erro ###
    def goErroScreen(self, msg):
        self.w_erro.show()
        self.l_erro.setText(msg)
        self.b_ok_2.clicked.connect(self.byeErro)

    ### Fecha a tela de erro ###
    def byeErro(self):
        self.w_erro.hide()
        self.l_erro.setText("")


    ############################################
    ## Funções que acessam opções da tela Login ##
    ############################################
    def goInicio(self, tela_atual):
        tela_atual.hide()
        self.w_bemvindo.show()

    ### Limpa o campo de login ao clicar no botão "Limpar" ###
    def goLimparLC(self, rg, nome=None):
        rg.setText("")
        if(nome is not False):
            nome.setText("")

    ########################################
    ## Funções que checam o campo de texto ##
    ########################################
    #### Checa se há caracteres no campo ####
    def has_char(self, inputString):
        return bool(re.search(r'[a-zA-Z]', inputString))

    #### Checa se há números no campo ####
    def has_number(self, inputString):
        return bool(re.search(r'[0-9]', inputString))
```

```
    ################################################
    ###  Acessando   a    tela   de   Operações  ###
    ################################################
    ### Realiza o saque  obtendo o RG do próprio ###
    ### Configura a mensagem  inicial com o nome ###
    ### do cliente  e solicita o valor  do saldo ###
    ### que possui,  atualizando o valor na tela ###
    ################################################
    def goOperacoes(self, tela_atual):
        ##print("@@-- Operações --@@")
        tela_atual.hide()
        self.w_operacoes.show()
        self.l_operacoes.setText("O que você deseja, " + self.nome + "?")
        ##print("--id", self.rg)
        msg = "saldo#" + self.rg
        ##print("--msg", msg)
        self.send_msg(msg)
        msg = self.recv_msg()

        ##print("--msg", msg)
        msg = float(msg)
        ##print(msg)
        self.label.setText(self.label.text().split(":")[0] + ": R$" +
str("{:.2f}".format(msg)))

    ################################################
    ### Acessando as telas a partir de Operações ###
    ################################################
    ### Realiza a transferência  obtendo o RG do ###
    ### destinatário e o valor  a ser depositado ###
    ### e envia uma  mensagem para  o  servidor. ###
    ### Logo após, retorna para a tela Operações ###
    ################################################
    def goTransferir(self, tela_atual, rg, valor):
        ##print("@@-- Tranferir --@@")
        msg = "deposito#" + str(rg.text()) + "#" + str(valor.text()) + "#sacar#" +
self.rg + "#" + str(valor.text())
        self.send_msg(msg)
        self.goOperacoes(self.w_transferir)

    ################################################
    ### Realiza o saque  obtendo o RG do próprio ###
    ### cliente e o valor a ser subtraído da sua ###
    ### conta e envia a mensagem para o servidor ###
```

```
    ### Logo após, retorna para a tela Operações ###
    ################################################
    def goSacar(self, tela_atual, valor):
        ##print("@@-- Sacar --@@")
        msg = "sacar#" + self.rg + "#" + str(valor.text())
        self.send_msg(msg)
        self.goOperacoes(self.w_sacar)

    ################################################
    ### Faz o  deposito obtendo o  RG do próprio ###
    ### cliente e o  valor a ser  somado na sua  ###
    ### conta e envia a mensagem para o servidor ###
    ### Logo após, retorna para a tela Operações ###
    ################################################
    def goDepositar(self, tela_atual, rs):
        ##print("@@-- Depositar --@@")
        if self.has_char(rs.text()):
            ##print("Erro deposito char")
            return
        msg = "deposito#" + str(self.rg) + "#" + str(rs.text())
        self.send_msg(msg)
        self.goOperacoes(self.w_depositar)

    ####################################################
    ### Checa se os campos de RG possuem algum erro ###
    ####################################################
    def goErrosRG(self, rg):
        if "#" in rg or "/" in rg:
            ##print("Erro # rg")
            return True

        ### Se o RG possui letras, retorna que é erro ###
        if self.has_char(rg):
            ##print("Erro rg char")
            return True

        ### Se o tamanho do RG é zero, retorna que é erro ###
        if len(rg) == 0:
            ##print("Erro rg len")
            return True

    ######################################################
    ### Checa se os campos de nome possuem algum erro ###
    ######################################################
    def goErrosNome(self, nome):
```

```
        if "#" in nome or "/" in nome:
            ##print("Erro # nome")
            return True

        if self.has_number(nome):
            ##print("Erro nome number")
            return True

        if len(nome) < 4:
            ##print("Erro nome len")
            return True

    ##########################################
    ### Traduz os status de erro em mensagens ###
    ##########################################
    def goErros(self, msg):
        if msg == "401":
            ##print("Erro 401")
            return True
        elif msg == "400":
            ##print("Erro 400")
            return True

    ##########################################
    ### Valida os campos de login, se correto ###
    ##########################################
    def goLogin(self, tela_atual, rg_text):
        ##print("@@-- Entrar --@@")
        rg = rg_text.text()

        ### Checa  se há algum erro na  entrada de RG ###
        ### como se possui letra ou tamanho incorreto ###
        if(self.goErrosRG(rg)):
            return

        ### Inicializa a conexão com o servidor ###
        self.begin_thread()

        ### Envia o RG para o servidor e checar se o id existe ###
        self.send_msg(rg_text.text())

        ### Recebe a resposta do servidor em relação ao RG enviado ###
        msg = self.recv_msg()

        ### Checa se há erros com o login, seja ###
```

```
        ### usuário inválido  ou não cadastrado ###
        if self.goErros(msg):
            return

        ### Salva o RG e o nome do cliente ###
        id = msg.split("#")
        self.rg = id[1].split("/")[0]
        self.nome = id[1].split("/")[1]

        ##print("--id", (self.rg, self.nome))
        ##print("Entrou")

        ### Vai para a tela de operações ###
        self.goOperacoes(tela_atual)

    ##########################################
    ### Realiza o cadastro de novos usuários ###
    ##########################################
    def goCadastro(self, tela_atual, rg_text, nome_text):
        ##print("@@-- Cadastro --@@")
        rg = rg_text.text()
        nome = nome_text.text()

        ### Checa  se há algum erro na  entrada de RG ###
        ### como se possui letra ou tamanho incorreto ###
        if(self.goErrosRG(rg)):
            return

        ### Checa  se há algum erro na  entrada de nome ###
        ### como se possui número  ou tamanho incorreto ###
        if(self.goErrosNome(nome)):
            return

        ### Inicializa a conexão com o servidor ###
        self.begin_thread()

        ### Envia o RG para o servidor e checar se o id existe ###
        self.send_msg(rg + "#" + nome)
         ### Recebe a resposta do servidor em relação ao RG enviado ###
        msg = self.recv_msg()
        ##print(msg)

        ### Checa se há erros com o login, seja ###
        ### usuário inválido  ou não cadastrado ###
        if self.goErros(msg):
```

```
            return

        ### Salva o RG e o nome do cliente ###
        id = msg.split("#")
        self.rg = id[1].split("/")[0]
        self.nome = id[1].split("/")[1]
        ##print("--id", (self.rg, self.nome))
        ##print("Entrou")
        self.goOperacoes(tela_atual)

    ### Encerra a aplicação respectiva do cliente ###
    #### e encerra a conexão com o servidor ###
    def goSair(self):
        self.send_msg("QUIT")
        QtWidgets.QApplication.quit()

    ################################################################
    ### Instancia threads para as funções de receber e enviar ###
    ### mensagem e rodar ambas paralelamente com a aplicação, ###
    ### sem comprometer a aplicação geral de cliente-servidor ###
    ################################################################
    def begin_thread(self):
        Thread(target=self.send_msg).start()
        Thread(target=self.recv_msg).start()

    ### Função responsável por enviar a mensagem para o servidor ###
    def send_msg(self, msg=""):
        ##print(msg)
        self.client.send(msg.encode())
        if (msg.upper() == "QUIT"):
            self.client.close()

    ### Função responsável por receber a mensagem enviada do servidor ###
    def recv_msg(self):
        while 1:
            try:
                data = self.client.recv(1024).decode()
                return data
            except:
                exit()

    ### Encerra a aplicação respectiva do cliente ###
    #### e encerra a conexão com o servidor ###
    def closeEvent(self, QCloseEvent):
        self.send_msg("QUIT")
```

```
        self.client.close()
        QtWidgets.QApplication.quit()

app = QtWidgets.QApplication(sys.argv)
main = Client()
main.show()
app.exec_()
```

objetos_do_layout.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'bemvindo.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.

from PyQt5 import QtCore, QtGui, QtWidgets

### Interface do programa definido nesse arquivo, ###
### com janelas, labels, frames e botões definidos ###

### O layout foi feito com o auxílio do programa QtDesigner, ###
### e, com o código em XML pronto, foi utilizado uma extensão ###
### para converter para elemento da biblioteca PyQt5 ###
class Ui_Program(object):
    ### Essa função instancia todos os elementos gráficos da aplicação ###
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(398, 449)
        MainWindow.setMaximumSize(QtCore.QSize(16777215, 16777215))
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        ## Elementos da interface inicial
        self.w_bemvindo = QtWidgets.QWidget(self.centralwidget)
        self.w_bemvindo.setGeometry(QtCore.QRect(80, 110, 231, 171))
        self.w_bemvindo.setObjectName("w_bemvindo")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.w_bemvindo)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setSpacing(1)
        self.verticalLayout.setObjectName("verticalLayout")
        self.b_login = QtWidgets.QPushButton(self.w_bemvindo)
```

```
        self.b_login.setObjectName("b_login")
        self.verticalLayout.addWidget(self.b_login)
        self.b_cadastro = QtWidgets.QPushButton(self.w_bemvindo)
        self.b_cadastro.setObjectName("b_cadastro")
        self.verticalLayout.addWidget(self.b_cadastro)

        ## Elementos da interface de Login
        self.w_login = QtWidgets.QWidget(self.centralwidget)
        self.w_login.setGeometry(QtCore.QRect(30, 70, 341, 261))
        self.w_login.setObjectName("w_login")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.w_login)
        self.verticalLayout_2.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.formLayout = QtWidgets.QFormLayout()
        self.formLayout.setFormAlignment(QtCore.Qt.AlignCenter)
        self.formLayout.setObjectName("formLayout")
        self.l_rg = QtWidgets.QLabel(self.w_login)
        self.l_rg.setObjectName("l_rg")
        self.formLayout.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.l_rg)
        self.t_rg_login = QtWidgets.QLineEdit(self.w_login)
        self.t_rg_login.setObjectName("t_rg_login")
        self.formLayout.setWidget(0, QtWidgets.QFormLayout.FieldRole,
self.t_rg_login)
        self.verticalLayout_2.addLayout(self.formLayout)
        self.gridLayout = QtWidgets.QGridLayout()
        self.gridLayout.setContentsMargins(-1, 50, -1, -1)
        self.gridLayout.setObjectName("gridLayout")
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        self.gridLayout.addLayout(self.horizontalLayout_3, 0, 2, 1, 1)
        self.b_entrar_login = QtWidgets.QPushButton(self.w_login)
        self.b_entrar_login.setObjectName("b_entrar_login")
        self.gridLayout.addWidget(self.b_entrar_login, 0, 5, 1, 1)
        self.b_limpar_login = QtWidgets.QPushButton(self.w_login)
        self.b_limpar_login.setObjectName("b_limpar_login")
        self.gridLayout.addWidget(self.b_limpar_login, 0, 3, 1, 1)
        self.b_voltar_login = QtWidgets.QPushButton(self.w_login)
        self.b_voltar_login.setObjectName("b_voltar_login")
        self.gridLayout.addWidget(self.b_voltar_login, 0, 0, 1, 1)
        self.verticalLayout_2.addLayout(self.gridLayout)

        ## Elementos da interface de Cadastro
        self.w_cadastro = QtWidgets.QWidget(self.centralwidget)
        self.w_cadastro.setGeometry(QtCore.QRect(30, 80, 341, 250))
        self.w_cadastro.setObjectName("w_cadastro")
```

```
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.w_cadastro)
        self.verticalLayout_3.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.formLayout_3 = QtWidgets.QFormLayout()
        self.formLayout_3.setFormAlignment(QtCore.Qt.AlignCenter)
        self.formLayout_3.setObjectName("formLayout_3")
        self.l_rg_3 = QtWidgets.QLabel(self.w_cadastro)
        self.l_rg_3.setObjectName("l_rg_3")
        self.formLayout_3.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.l_rg_3)
        self.t_rg_cadastro = QtWidgets.QLineEdit(self.w_cadastro)
        self.t_rg_cadastro.setObjectName("t_rg_cadastro")
        self.formLayout_3.setWidget(0, QtWidgets.QFormLayout.FieldRole,
self.t_rg_cadastro)
        self.l_nome_2 = QtWidgets.QLabel(self.w_cadastro)
        self.l_nome_2.setObjectName("l_nome_2")
        self.formLayout_3.setWidget(1, QtWidgets.QFormLayout.LabelRole,
self.l_nome_2)
        self.t_nome_cadastro = QtWidgets.QLineEdit(self.w_cadastro)
        self.t_nome_cadastro.setObjectName("t_nome_cadastro")
        self.formLayout_3.setWidget(1, QtWidgets.QFormLayout.FieldRole,
self.t_nome_cadastro)
        self.verticalLayout_3.addLayout(self.formLayout_3)
        self.gridLayout_3 = QtWidgets.QGridLayout()
        self.gridLayout_3.setContentsMargins(-1, 50, -1, -1)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_5.setObjectName("horizontalLayout_5")
        self.gridLayout_3.addLayout(self.horizontalLayout_5, 0, 2, 1, 1)
        self.b_entrar_cadastro = QtWidgets.QPushButton(self.w_cadastro)
        self.b_entrar_cadastro.setObjectName("b_entrar_cadastro")
        self.gridLayout_3.addWidget(self.b_entrar_cadastro, 0, 5, 1, 1)
        self.b_limpar_cadastro = QtWidgets.QPushButton(self.w_cadastro)
        self.b_limpar_cadastro.setObjectName("b_limpar_cadastro")
        self.gridLayout_3.addWidget(self.b_limpar_cadastro, 0, 3, 1, 1)
        self.b_voltar_cadastro = QtWidgets.QPushButton(self.w_cadastro)
        self.b_voltar_cadastro.setObjectName("b_voltar_cadastro")
        self.gridLayout_3.addWidget(self.b_voltar_cadastro, 0, 0, 1, 1)
        self.verticalLayout_3.addLayout(self.gridLayout_3)

        self.w_depositar = QtWidgets.QWidget(self.centralwidget)
        self.w_depositar.setGeometry(QtCore.QRect(30, 80, 341, 251))
        self.w_depositar.setObjectName("w_depositar")
        self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.w_depositar)
        self.verticalLayout_4.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_4.setObjectName("verticalLayout_4")
```

```
        self.l_deposito = QtWidgets.QLabel(self.w_depositar)
        self.l_deposito.setAlignment(QtCore.Qt.AlignBottom|QtCore.Qt.AlignHCenter)
        self.l_deposito.setObjectName("l_deposito")
        self.verticalLayout_4.addWidget(self.l_deposito)
        self.formLayout_4 = QtWidgets.QFormLayout()

self.formLayout_4.setFormAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
        self.formLayout_4.setObjectName("formLayout_4")
        self.l_rs = QtWidgets.QLabel(self.w_depositar)
        self.l_rs.setObjectName("l_rs")
        self.formLayout_4.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.l_rs)
        self.t_rs = QtWidgets.QLineEdit(self.w_depositar)
        self.t_rs.setObjectName("t_rs")
        self.formLayout_4.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.t_rs)
        self.verticalLayout_4.addLayout(self.formLayout_4)
        self.gridLayout_2 = QtWidgets.QGridLayout()
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.b_depositar = QtWidgets.QPushButton(self.w_depositar)
        self.b_depositar.setObjectName("b_depositar")
        self.gridLayout_2.addWidget(self.b_depositar, 0, 3, 1, 1)
        self.b_limpar_2 = QtWidgets.QPushButton(self.w_depositar)
        self.b_limpar_2.setObjectName("b_limpar_2")
        self.gridLayout_2.addWidget(self.b_limpar_2, 0, 2, 1, 1)
        self.gridLayout_4 = QtWidgets.QGridLayout()
        self.gridLayout_4.setObjectName("gridLayout_4")
        self.gridLayout_2.addLayout(self.gridLayout_4, 0, 1, 1, 1)
        self.b_voltar_2 = QtWidgets.QPushButton(self.w_depositar)
        self.b_voltar_2.setObjectName("b_voltar_2")
        self.gridLayout_2.addWidget(self.b_voltar_2, 0, 0, 1, 1)
        self.verticalLayout_4.addLayout(self.gridLayout_2)

        self.w_operacoes = QtWidgets.QWidget(self.centralwidget)
        self.w_operacoes.setGeometry(QtCore.QRect(60, 90, 271, 221))
        self.w_operacoes.setObjectName("w_operacoes")
        self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.w_operacoes)
        self.verticalLayout_5.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_5.setObjectName("verticalLayout_5")
        self.l_operacoes = QtWidgets.QLabel(self.w_operacoes)
        self.l_operacoes.setObjectName("l_operacoes")
        self.verticalLayout_5.addWidget(self.l_operacoes)
        self.verticalLayout_6 = QtWidgets.QVBoxLayout()
        self.verticalLayout_6.setObjectName("verticalLayout_6")
        self.b_sacar = QtWidgets.QPushButton(self.w_operacoes)
        self.b_sacar.setObjectName("b_sacar")
        self.verticalLayout_6.addWidget(self.b_sacar)
```

```
        self.b_depositar_2 = QtWidgets.QPushButton(self.w_operacoes)
        self.b_depositar_2.setObjectName("b_depositar_2")
        self.verticalLayout_6.addWidget(self.b_depositar_2)
        self.b_transferir = QtWidgets.QPushButton(self.w_operacoes)
        self.b_transferir.setObjectName("b_transferir")
        self.verticalLayout_6.addWidget(self.b_transferir)
        self.verticalLayout_5.addLayout(self.verticalLayout_6)
        self.label = QtWidgets.QLabel(self.w_operacoes)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")
        self.verticalLayout_6.addWidget(self.label)
        self.b_sair = QtWidgets.QPushButton(self.w_operacoes)
        self.b_sair.setObjectName("b_sair")
        self.verticalLayout_5.addWidget(self.b_sair)

        self.w_transferir = QtWidgets.QWidget(self.centralwidget)
        self.w_transferir.setGeometry(QtCore.QRect(20, 60, 361, 251))
        self.w_transferir.setObjectName("w_transferir")
        self.verticalLayout_7 = QtWidgets.QVBoxLayout(self.w_transferir)
        self.verticalLayout_7.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_7.setObjectName("verticalLayout_7")
        self.l_rg_dest = QtWidgets.QLabel(self.w_transferir)
        self.l_rg_dest.setAlignment(QtCore.Qt.AlignBottom|QtCore.Qt.AlignHCenter)
        self.l_rg_dest.setObjectName("l_rg_dest")
        self.verticalLayout_7.addWidget(self.l_rg_dest)
        self.formLayout_5 = QtWidgets.QFormLayout()

self.formLayout_5.setFormAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
        self.formLayout_5.setObjectName("formLayout_5")
        self.l_rg_2 = QtWidgets.QLabel(self.w_transferir)
        self.l_rg_2.setObjectName("l_rg_2")
        self.formLayout_5.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.l_rg_2)
        self.t_rg_2 = QtWidgets.QLineEdit(self.w_transferir)
        self.t_rg_2.setObjectName("t_rg_2")
        self.formLayout_5.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.t_rg_2)
        self.verticalLayout_7.addLayout(self.formLayout_5)
        self.l_valor = QtWidgets.QLabel(self.w_transferir)
        self.l_valor.setAlignment(QtCore.Qt.AlignBottom|QtCore.Qt.AlignHCenter)
        self.l_valor.setObjectName("l_valor")
        self.verticalLayout_7.addWidget(self.l_valor)
        self.formLayout_6 = QtWidgets.QFormLayout()

self.formLayout_6.setFormAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
        self.formLayout_6.setObjectName("formLayout_6")
        self.l_rs_2 = QtWidgets.QLabel(self.w_transferir)
```

```
        self.l_rs_2.setObjectName("l_rs_2")
        self.formLayout_6.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.l_rs_2)
        self.t_rs_2 = QtWidgets.QLineEdit(self.w_transferir)
        self.t_rs_2.setObjectName("t_rs_2")
        self.formLayout_6.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.t_rs_2)
        self.verticalLayout_7.addLayout(self.formLayout_6)
        self.gridLayout_5 = QtWidgets.QGridLayout()
        self.gridLayout_5.setObjectName("gridLayout_5")
        self.b_transferir_2 = QtWidgets.QPushButton(self.w_transferir)
        self.b_transferir_2.setObjectName("b_transferir_2")
        self.gridLayout_5.addWidget(self.b_transferir_2, 0, 3, 1, 1)
        self.b_limpar_4 = QtWidgets.QPushButton(self.w_transferir)
        self.b_limpar_4.setObjectName("b_limpar_4")
        self.gridLayout_5.addWidget(self.b_limpar_4, 0, 2, 1, 1)
        self.gridLayout_6 = QtWidgets.QGridLayout()
        self.gridLayout_6.setObjectName("gridLayout_6")
        self.gridLayout_5.addLayout(self.gridLayout_6, 0, 1, 1, 1)
        self.b_voltar_4 = QtWidgets.QPushButton(self.w_transferir)
        self.b_voltar_4.setObjectName("b_voltar_4")
        self.gridLayout_5.addWidget(self.b_voltar_4, 0, 0, 1, 1)
        self.verticalLayout_7.addLayout(self.gridLayout_5)

        self.w_sacar = QtWidgets.QWidget(self.centralwidget)
        self.w_sacar.setGeometry(QtCore.QRect(20, 80, 361, 251))
        self.w_sacar.setObjectName("w_sacar")
        self.verticalLayout_8 = QtWidgets.QVBoxLayout(self.w_sacar)
        self.verticalLayout_8.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_8.setObjectName("verticalLayout_8")
        self.l_sacar = QtWidgets.QLabel(self.w_sacar)
        self.l_sacar.setAlignment(QtCore.Qt.AlignBottom|QtCore.Qt.AlignHCenter)
        self.l_sacar.setObjectName("l_sacar")
        self.verticalLayout_8.addWidget(self.l_sacar)
        self.formLayout_7 = QtWidgets.QFormLayout()

self.formLayout_7.setFormAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
        self.formLayout_7.setObjectName("formLayout_7")
        self.l_rs_3 = QtWidgets.QLabel(self.w_sacar)
        self.l_rs_3.setObjectName("l_rs_3")
        self.formLayout_7.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.l_rs_3)
        self.t_rs_3 = QtWidgets.QLineEdit(self.w_sacar)
        self.t_rs_3.setObjectName("t_rs_3")
        self.formLayout_7.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.t_rs_3)
        self.verticalLayout_8.addLayout(self.formLayout_7)
        self.gridLayout_7 = QtWidgets.QGridLayout()
        self.gridLayout_7.setObjectName("gridLayout_7")
```

```
        self.b_sacar_2 = QtWidgets.QPushButton(self.w_sacar)
        self.b_sacar_2.setObjectName("b_sacar_2")
        self.gridLayout_7.addWidget(self.b_sacar_2, 0, 3, 1, 1)
        self.b_limpar_5 = QtWidgets.QPushButton(self.w_sacar)
        self.b_limpar_5.setObjectName("b_limpar_5")
        self.gridLayout_7.addWidget(self.b_limpar_5, 0, 2, 1, 1)
        self.gridLayout_8 = QtWidgets.QGridLayout()
        self.gridLayout_8.setObjectName("gridLayout_8")
        self.gridLayout_7.addLayout(self.gridLayout_8, 0, 1, 1, 1)
        self.b_voltar_5 = QtWidgets.QPushButton(self.w_sacar)
        self.b_voltar_5.setObjectName("b_voltar_5")
        self.gridLayout_7.addWidget(self.b_voltar_5, 0, 0, 1, 1)
        self.verticalLayout_8.addLayout(self.gridLayout_7)

        self.w_saldo = QtWidgets.QWidget(self.centralwidget)
        self.w_saldo.setGeometry(QtCore.QRect(130, 160, 160, 80))
        self.w_saldo.setObjectName("w_saldo")
        self.verticalLayout_10 = QtWidgets.QVBoxLayout(self.w_saldo)
        self.verticalLayout_10.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_10.setObjectName("verticalLayout_10")
        self.l_saldo = QtWidgets.QLabel(self.w_saldo)
        self.l_saldo.setAlignment(QtCore.Qt.AlignCenter)
        self.l_saldo.setObjectName("l_saldo")
        self.verticalLayout_10.addWidget(self.l_saldo)
        self.b_ok = QtWidgets.QPushButton(self.w_saldo)
        self.b_ok.setObjectName("b_ok")
        self.verticalLayout_10.addWidget(self.b_ok)

        self.w_erro = QtWidgets.QWidget(self.centralwidget)
        self.w_erro.setGeometry(QtCore.QRect(130, 160, 160, 80))
        self.w_erro.setObjectName("w_erro")
        self.verticalLayout_11 = QtWidgets.QVBoxLayout(self.w_erro)
        self.verticalLayout_11.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_11.setObjectName("verticalLayout_11")
        self.l_erro = QtWidgets.QLabel(self.w_erro)
        self.l_erro.setAlignment(QtCore.Qt.AlignCenter)
        self.l_erro.setObjectName("l_erro")
        self.verticalLayout_11.addWidget(self.l_erro)
        self.b_ok_2 = QtWidgets.QPushButton(self.w_erro)
        self.b_ok_2.setObjectName("b_ok_2")
        self.verticalLayout_11.addWidget(self.b_ok_2)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 398, 24))
        self.menubar.setObjectName("menubar")
```

```
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.w_bemvindo.show()
        self.w_login.hide()
        self.w_cadastro.hide()
        self.w_depositar.hide()
        self.w_operacoes.hide()
        self.w_transferir.hide()
        self.w_sacar.hide()
        self.w_saldo.hide()
        self.w_erro.hide()

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    ### Essa função renomeia todas as labels e títulos que a aplicação possui ###
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.b_login.setText(_translate("MainWindow", "Login"))
        self.b_cadastro.setText(_translate("MainWindow", "Cadastro"))
        self.l_rg.setText(_translate("MainWindow", "Insira o seu RG"))
        self.b_entrar_login.setText(_translate("MainWindow", "Entrar"))
        self.b_limpar_login.setText(_translate("MainWindow", "Limpar"))
        self.b_voltar_login.setText(_translate("MainWindow", "Voltar"))
        self.l_rg_3.setText(_translate("MainWindow", "Insira o seu RG"))
        self.l_nome_2.setText(_translate("MainWindow", "Insira o seu nome"))
        self.b_entrar_cadastro.setText(_translate("MainWindow", "Entrar"))
        self.b_limpar_cadastro.setText(_translate("MainWindow", "Limpar"))
        self.b_voltar_cadastro.setText(_translate("MainWindow", "Voltar"))
        self.l_deposito.setText(_translate("MainWindow", "Insira o valor que deseja
depositar:"))
        self.l_rs.setText(_translate("MainWindow", "R$"))
        self.b_depositar.setText(_translate("MainWindow", "Depositar"))
        self.b_limpar_2.setText(_translate("MainWindow", "Limpar"))
        self.b_voltar_2.setText(_translate("MainWindow", "Voltar"))
        self.l_operacoes.setText(_translate("MainWindow", "O que você deseja, "))
        self.b_sacar.setText(_translate("MainWindow", "Sacar"))
        self.b_depositar_2.setText(_translate("MainWindow", "Depositar"))
        self.b_transferir.setText(_translate("MainWindow", "Transferir"))
        self.label.setText(_translate("MainWindow", "Saldo: R$ "))
```

```
        self.b_sair.setText(_translate("MainWindow", "Sair"))
        self.l_rg_dest.setText(_translate("MainWindow", "Insira o RG do
destinatário:"))
        self.l_rg_2.setText(_translate("MainWindow", "RG"))
        self.l_valor.setText(_translate("MainWindow", "Insira o valor que deseja
transferir:"))
        self.l_rs_2.setText(_translate("MainWindow", "R$"))
        self.b_transferir_2.setText(_translate("MainWindow", "Transferir"))
        self.b_limpar_4.setText(_translate("MainWindow", "Limpar"))
        self.b_voltar_4.setText(_translate("MainWindow", "Voltar"))
        self.l_sacar.setText(_translate("MainWindow", "Insira o valor que deseja
sacar:"))
        self.l_rs_3.setText(_translate("MainWindow", "R$"))
        self.b_sacar_2.setText(_translate("MainWindow", "Sacar"))
        self.b_limpar_5.setText(_translate("MainWindow", "Limpar"))
        self.b_voltar_5.setText(_translate("MainWindow", "Voltar"))
        self.l_saldo.setText(_translate("MainWindow", "saldo"))
        self.b_ok.setText(_translate("MainWindow", "OK!"))
        self.l_erro.setText(_translate("MainWindow", "erro"))
        self.b_ok_2.setText(_translate("MainWindow", "OK!"))
```