UNIVERSIDADE FEDERAL DA BAHIA
INSTITUTO DE COMPUTAÇÃO

FELIPE RODRIGUES NASCIMENTO
ICARO BRITO

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

SALVADOR - BA
2022

## 1. Descrição do sistema desenvolvido

O sistema foi criado com sockets como foi requerido para a atividade.

Utilizamos uma classe conta em que foram criados dois objetos com os nomes dos respectivos integrantes da dupla e foram atribuídos valores para o saldo e informações para a conta.

O cliente_banco.py cria uma sessão que troca mensagens com o servidor_banco.py. Todos os processos e informações estão inseridos no servidor.

Utilizamos uma biblioteca nativa do python chamada “socket” para criar a comunicação entre o servidor e o cliente.

Assim como num sistema bancário real, criamos menus onde o usuário consegue escolher o tipo de transação (saldo, saque e trasferência).

O sistema só possui 2 clientes: Icaro e Felipe.

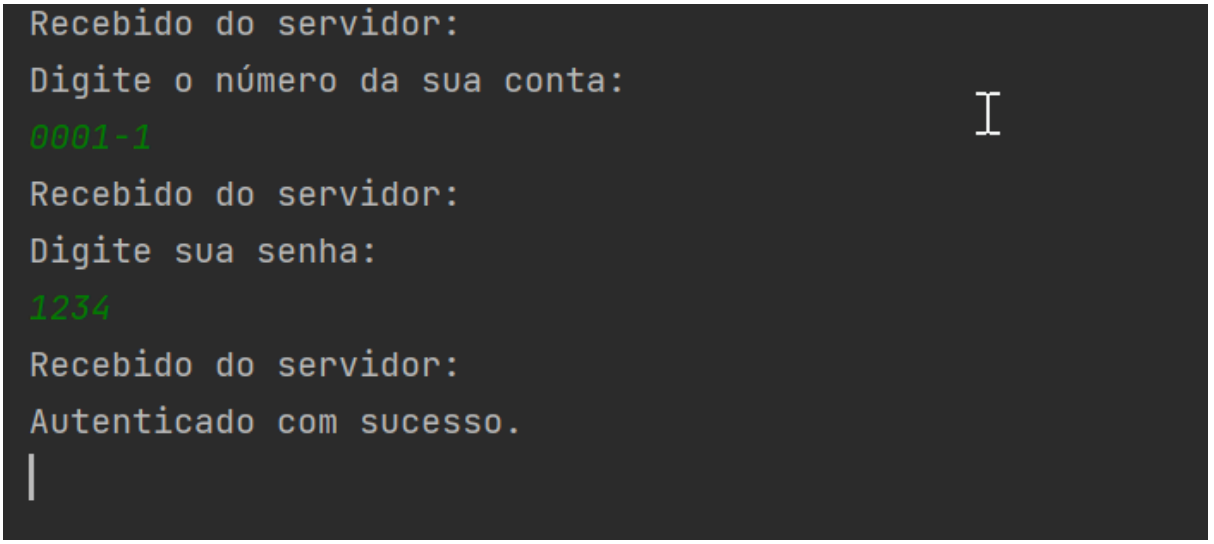
Todas as transações serão executadas por eles

## 2. Exemplo de utilização

Para ter acesso ao sistema, deve-se primeiro rodar o código servidor_banco.py e após rodar o código cliente_banco.py.

A primeira informação será a solicitação do número da conta:


Após digitar o número da conta e senha, o cliente recebe a informação de autenticação:




Deve-se pressionar a tecla <enter> para ter acesso ao menu:

```
Recebido do servidor:

MENU DE OPERAÇÕES
Opções:
1 - Saldo
2 - Depósito
3 - Saque
4 - Transferência
5 - Sair
6 - Novo login
Digite a opção desejada:
```

Digite o número da opção desejada e pressione <enter>. Verificarei o saldo ( “1” e após <enter>)

```
Recebido do servidor:
CLIENTE: Felipe Nascimento RG: 12345678-90
```

Aparecerá o nome do correntista e o seu RG, ao pressionar <enter> novamente: a informação do saldo:

```
Recebido do servidor:
AGÊNCIA: 0001 CONTA: 0001-1
SALDO: R$ 500.00
----------------------------------------
Aperte M para retornar ao MENU
ou qualquer outra tecla para sair:
```

Para demonstrar o saque irei pressionar “M” para retornar ao MENU principal:

```
MENU DE OPERAÇÕES
Opções:
1 - Saldo
2 - Depósito
3 - Saque
4 - Transferência
5 - Sair
6 - Novo login
Digite a opção desejada:
3
```

Ao entrar na opção do saque, digite o valor desejado, nesse caso R$499,00:

```
Recebido do servidor: Digite o valor a ser sacado:
499
Recebido do servidor: Saque de R$499.00 realizado com sucesso.
```

Ao pressionar a tecla enter já é retornado o saldo restante da conta, após o saque:

```
Recebido do servidor: Tecle ENTER para continuar
CLIENTE: Felipe Nascimento RG: 12345678-90
AGÊNCIA: 0001 CONTA: 0001-1
SALDO: R$ 1.00
----------------------------------------
```

## 3. Códigos

### 3.1. Código cliente_banco.py

```
# Universidade Federal da Bahia
# Disciplina: Fundamentos de Sistemas Distribuídos
# Professor: Raimundo Macêdo
```

```
# Projeto: Sistema Distribuído Simples para Manutenção de
Contas Bancárias
# Desenvolvedores: Felipe Nascimento (felipern@ufba.br), e
Icaro Brito (icarobss@ufba.br)
# 2022

# Importa biblioteca necessária para o funcionamento da
comunicação
import socket

# Função que administra a conexão no lado do cliente,
# possibilita a troca de mensagens com o seridor
def cliente_banco():
    host = socket.gethostname()  # Recebe o nome do host, que
também é usado pelo servidor por estarem ambos na mesma máquina
    port = 5001  # Número da porta do servidor socket

    client_socket = socket.socket()  # Cria a instância do
socket
    client_socket.connect((host, port))  # Conecta ao servidor

    mensagem = ''  # Define a mensagem

    # Se digitar "finalizar", termina o programa para o cliente
e o servidor
    while mensagem.lower().strip() != 'finalizar':
        data = client_socket.recv(1024).decode()  # Recebe a
resposta do servidor

        # Mostra os dados recebidos do servidor no terminal
        print('Recebido do servidor: ' + data)

        # Se receber do servidor a resposta de encerramento da
operação, finaliza o programa
        if data == 'Encerrando...':
            client_socket.close()
            exit()

        mensagem = input("")  # Aguarda uma nova entrada de
dados do usuario
        client_socket.send(mensagem.encode())  # Envia a
mensagem para o servidor

    client_socket.close()  # Finaliza a conexão


if __name__ == '__main__':
    # Chama a função principal para iniciar
    # a conexão do cliente com o servidor,
    # e solicitar as operações bancárias
    cliente_banco()
```

## 3.2. Código do Servidor_banco.py

```
# Universidade Federal da Bahia
# Disciplina: Fundamentos de Sistemas Distribuídos
# Professor: Raimundo Macêdo
# Projeto: Sistema Distribuído Simples para Manutenção de
Contas Bancárias
# Desenvolvedores: Felipe Nascimento (felipern@ufba.br), e
Icaro Brito (icarobss@ufba.br)
# 2022

# Importa biblioteca necessária para o funcionamento da
comunicação
import socket

# Classe que define uma conta bancária de um cliente
class Conta:
   cliente = 'Nome'
   rg = '12345678-90'
   senha = '123456'
   agencia = '0001'
   numero_conta = '1010-2'
   saldo = 0.00

# Função para realizar um depósito na conta
def depositar(conta, valor, conn):
   conta.saldo += valor # Acrescenta o valor na conta
   mensagem = f'Depósito de R${valor:.2F} realizado com
sucesso.'
   conn.send(mensagem.encode())
   mostrar_saldo(conta, conn) # Chama a função para mostrar o
novo saldo

# Função para realizar um saque na conta
def sacar(conta, valor, conn):
   # Confirma se o usuário possui saldo suficiente para sacar
   if valor <= conta.saldo:
       conta.saldo -= valor # Diminui o valor do saldo da conta
       mensagem = f'Saque de R${valor:.2F} realizado com
sucesso.'
       conn.send(mensagem.encode())
       mensagem = 'Tecle ENTER para continuar'
       conn.send(mensagem.encode())
       mostrar_saldo(conta, conn) # Chama a função para mostrar
o novo saldo
   else:
       mensagem = 'Saldo insuficiente.'
       conn.send(mensagem.encode())
       menu(conta, conn) # Chama a função para exibir o menu
```

```
# Função que realiza a transferência de valores entre contas
def transferir(conta, conta_destino, valor, conn):
   # Confirma se o usuário possui saldo suficiente para
transferir
   if valor <= conta.saldo:
       conta.saldo -= valor # Diminui o valor do saldo da conta
origem
       conta_destino.saldo += valor # Acrescenta o valor na
conta destino
       mensagem = f'Transferência de R${valor:.2F} realizada
com sucesso.'
       conn.send(mensagem.encode())
       mostrar_saldo(conta, conn) # Chama a função para mostrar
o novo saldo
   else:
       mensagem = 'Saldo insuficiente.'
       conn.send(mensagem.encode())
       menu(conta, conn)

# Função contend as operações disponíveis para o usuário, e
chama as funções de acordo com a operação solicitada
def menu(conta, conn):
   mensagem = ('\n\nMENU DE OPERAÇÕES\nOpções:\n1 - Saldo\n2 -
Depósito\n3 - Saque\n4 - Transferência\n5 - Sair\n6 - Novo
login\nDigite a opção desejada: ')
   conn.send(mensagem.encode())
   opcao = str(conn.recv(1024).decode())

   # Chama a opção para ver o saldo da conta
   if opcao == '1':
       mostrar_saldo(conta, conn)

   # Chama a opção para realizar um depósito
   elif opcao == '2':
       mensagem = ('Digite o valor a ser depositado: ')
       conn.send(mensagem.encode())
       valor = float(conn.recv(1024).decode())
       depositar(conta, valor, conn)

   # Chama a opção para realizar um saque
   elif opcao == '3':
       mensagem = ('Digite o valor a ser sacado: ')
       conn.send(mensagem.encode())
       valor = float(conn.recv(1024).decode())
       sacar(conta, valor, conn)

   # Chama a opção para realizar uma transferência
   elif opcao == '4':
       # Solicita o valor de transferência desejado e a conta
de destino
       mensagem = ('Digite o valor que deseja transferir: ')
       conn.send(mensagem.encode())
```

```
        valor = float(conn.recv(1024).decode())
        mensagem = ('Digite a conta para qual deseja transferir:
')
        conn.send(mensagem.encode())
        conta_destino = str(conn.recv(1024).decode())
        beneficiario = ''

        # Verifica se o usuário está tentando transferir para
sua própria conta
        if conta.numero_conta == conta_destino:
            mensagem = ('Não é possível transferir para sua
própria conta.')
            conn.send(mensagem.encode())
            menu(conta, conn)
        else: # Sendo uma conta diferente da sua, identifica que
é o beneficiário
            if conta_destino == pessoa1.numero_conta:
                beneficiario = pessoa1.cliente
                cliente_destino = pessoa1
            else:
                beneficiario = pessoa2.cliente
                cliente_destino = pessoa2

            # Pede confirmação se o beneficiário está correto
            mensagem = (f'Nome do beneficiário: {beneficiario}.
Tecle S para confirmar a operação, ou N para cancelar')
            conn.send(mensagem.encode())
            confirmacao = str(conn.recv(1024).decode())
            if confirmacao == 'S' or confirmacao == 's':
                transferir(conta, cliente_destino, valor, conn)
            else:
                mensagem = ('\nOperação cancelada.\n')
                conn.send(mensagem.encode())
                menu(conta, conn)

    # Chama a opção de sair do sistema
    elif opcao == '5':
        mensagem = ('\nEncerrando...')
        conn.send(mensagem.encode())
        conn.close()
        exit()

    # Chama a opção para fazer o login em uma nova conta
    elif opcao =='6':
        mensagem = ('Insira seus dados para fazer um novo
login')
        conn.send(mensagem.encode())
        fazer_login(conn, autenticacao=False)

# Função que mostra o saldo atual da conta do usuário
autenticado no sistema
def mostrar_saldo(conta, conn):
    # Exibe os dados do usuário da conta e saldo
```

```
   mensagem = (f'\nCLIENTE: {conta.cliente} RG: {conta.rg}')
   conn.send(mensagem.encode())
   mensagem = (f'\nAGÊNCIA: {conta.agencia} CONTA:
{conta.numero_conta}')
   conn.send(mensagem.encode())
   mensagem = (f'\nSALDO: R$ {conta.saldo:.2F}')
   conn.send(mensagem.encode())
   mensagem = ('\n' + ('-' * 40))
   conn.send(mensagem.encode())

   # Após exibição dos dados, oferece a opção de voltar ao menu
ou de encerrar a operação
   mensagem = '\nAperte M para retornar ao MENU\nou qualquer
outra tecla para sair: '
   conn.send(mensagem.encode())
   aperte = str(conn.recv(1024).decode())
   if aperte == 'M' or aperte == 'm':
       menu(conta, conn)
   else: # Digitando qualquer tecla encerra a conexão
       mensagem = ('\nEncerrando...\n')
       conn.send(mensagem.encode())
       conn.close()
       exit()

# Função que realiza o login do usuário que irá realizar as
operações bancárias
def fazer_login(conn, autenticacao):
   while True:
       # Se ainda não foi feito login, solicita os dados da
conta bancária
       if autenticacao == False:
           mensagem = '\nDigite o número da sua conta: '
           conn.send(mensagem.encode())
           num_conta = str(conn.recv(1024).decode())
           if not num_conta:
               print("\nNenhuma mensagem recebida do
usuario.\n")
               break

           mensagem = '\nDigite sua senha: '
           conn.send(mensagem.encode())
           senha = str(conn.recv(1024).decode())
           if not senha:
               print("\nNenhuma mensagem recebida do
usuario.\n")
               break

           # Confirma se o número da conta bancária e senha
correspondem ao usuário da conta 1
           if num_conta == pessoa1.numero_conta:
               if senha == pessoa1.senha:
                   mensagem = '\nAutenticado com sucesso.'
                   conn.send(mensagem.encode())
```

```
                    autenticacao = True
                    menu(pessoa1, conn)
                else: # Se a senha estiver incorreta, encerra o
login e a conexão em seguida
                    mensagem = '\nSenha incorreta.\n'
                    conn.send(mensagem.encode())
                    break

            # Confirma se o número da conta bancária e senha
correspondem ao usuário da conta 2
            elif num_conta == pessoa2.numero_conta:
                if senha == pessoa2.senha:
                    mensagem = '\nAutenticado com sucesso.'
                    conn.send(mensagem.encode())
                    autenticacao = True
                    menu(pessoa2, conn)
                else: # Se a senha estiver incorreta, encerra o
login e a conexão em seguida
                    mensagem = '\nSenha incorreta.'
                    conn.send(mensagem.encode())
                    break

# Função que administra a conexão no lado do servidor,
# possibilita a troca de mensagens com o cliente, e operações
financeiras
def servidor_banco():
   host = socket.gethostname() # Solicita o nome do host
   porta = 5001  # Inicia a porta
   autenticacao = False # Define se o usuário realizou login

   server_socket = socket.socket() # Cria instância do socket
   server_socket.bind((host, porta)) # A função "bind" hospeda
o endereço do host e da porta juntos
   server_socket.listen(2) # Configura quantos clientes o
servidor pode escutar simultâneamente
   conn, address = server_socket.accept() # Aceita uma nova
conexão
   print("Conectado com: " + str(address)) # Informa o
dispositivo com o qual está conectado

   fazer_login(conn, autenticacao) # Chama a função para fazer
login do usuário

   conn.close() # Finaliza a conexão


if __name__ == '__main__':

   # Cria uma nova conta de cliente no banco,
   # definindo os dados e saldo inicial
   pessoa1 = Conta()
   pessoa1.cliente = 'Felipe Nascimento'
   pessoa1.rg = '12345678-90'
```

```
pessoa1.senha = '1234'
pessoa1.numero_conta = '0001-1'
pessoa1.saldo = 500.00

# Cria uma nova conta de cliente no banco,
# definindo os dados e saldo inicial
pessoa2 = Conta()
pessoa2.cliente = 'Icaro Brito'
pessoa2.rg = '12345678-09'
pessoa2.senha = '4321'
pessoa2.numero_conta = '0002-1'
pessoa2.saldo = 500.00

# Chama a função principal para iniciar
# a conexão do servidor com o cliente,
# e acessar as funções de operações
servidor_banco()
```