

# Trabalho Prático 1 - Comunicação Distribuída

Filipe Nunes Soares,<sup>1</sup>, Ismael Prado da Cruz Costa<sup>2</sup>

<sup>1</sup>Instituto de Ciências Exatas e Aplicadas(ICEA) - Universidade Federal de Ouro Preto(UFOP)  
R. 36, 115 - Bairro Loanda - CEP 35931-008 – João Monlevade – MG – Brasil

<sup>2</sup>Departamento de Computação e Sistemas(DECSI)

{Filipe.soares.fn@gmail.com, ismael.cruz@aluno.ufop.edu.br}

**Abstract.** *The functioning of this work is made the development of a distributed system for transactions applied to the concepts of objective in the customer service room, and processing threads in the customer room.*

**Resumo.** *O objetivo deste trabalho é fazer o desenvolvimento de um sistema distribuído para transações monetárias aplicando os conceitos de cliente, servidores e threads apresentadas em sala.*

## 1. Introdução

O sistema deve ser capaz de enviar valores de uma máquina para outra utilizando o servidor para controle. Cada um dos pares tem que possuir uma carteira, permitindo o envio de valores para outros pares por meio de uma chave Hash. O servidor é o responsável por fazer o controle e validações.

## 2. Funcionalidades

Cada máquina deve ser inicializada, e instanciada com um valor inicial de 100 reais. O envio do valor é feito por meio de uma chave Hash que é passada manualmente pelo cliente, esta Hash é atualizada de tempos em tempos de modo que mantenha um pouco de segurança. Essa Hash se assimila um pouco a chave PIX.

O servidor é responsável por pegar todas as requisições e trata-las de acordo, ele mantém o controle de todas as máquinas disponíveis, seus IDs e o valor em carteira de cada uma.

## 3. Implementação

Implementação do projeto foi feito em Python e foi dividido em dois arquivos, cliente e servidor. Todas as requisições são feitas por mensagem como (Metodo—Ipmáquina—IdHash), deste modo conseguimos diferenciar qual o método que está sendo chamado.

### 3.1. Cliente

O cliente é uma aplicação que rodará em várias máquinas distintas, e todas elas se conectarão ao servidor que é o responsável por fazer o controle.

```

106 def main(argv):
107     global idHash
108     try:
109         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
110
111             #Conecta o cliente ao servidor
112             s.connect((HOST, PORT))
113             print("\nServidor Conectado!")
114
115             #Configura a maquina
116             configurarMaquina(s)
117
118             #cria a nova Thread pra ficar atualizando a Hash de tempos em tempos
119             newHash = Thread(target=gerarNovaHash, args=(s,))
120             newHash.start()
121
122             #Loop pra mandar o valor sempre que desejado
123             while(True):
124                 #Metodo pra realizar o envio do valor
125                 mandarDinheiro(s)
126                 desconnect = input("Deseja desconectar ?\nS ou N -> ")
127                 if(desconnect == 's'):
128                     s.close()
129                     break
130
131     except Exception as error:
132         print("Exceção - Programa será encerrado!")
133         print(error)
134         return

```

Figure 1. Método Main

### 3.1.1. Main

Inicialmente temos a Main, que realiza a conexão com o servidor e chama os métodos para configurar a maquina e inicia a Thread para a atualização da Hash

### 3.1.2. Configurar Maquina

O método de configuração é responsável por verificar inicialmente se já possui alguma maquina com o mesmo IP cadastrado no servidor, caso já possua, ele pega os dados dessa maquina e as atualiza no cliente. Caso não possua nenhuma maquina com esse IP, ele Gera uma nova Hash, inicia a carteira com o valor de 100 reais e enviar ao servidor para fazer o cadastro.

```

53 def configurarMaquina(s):
54     global idHash, carteira
55
56     #Envia uma mensagem para verificar se alguma maquina com este ip ja esta cadastrada (MetodoIpMaquina)
57     s.send(("getMaquina|" + socket.gethostname(socket.gethostname())).encode())
58     time.sleep(1)
59     data = s.recv(BUFFER_SIZE)
60     resp = repr(data).rstrip("\n")
61     maq = resp.split('|')
62
63     #De acordo com a resposta, ou ele so atualiza os valores na maquina, ou cria uma nova instancia no servidor
64     if(maq[0] == "b'True'"):
65         idHash = maq[1]
66         carteira = int(maq[2])
67     else:
68         #Cria uma hash referente ao ID | You, há 24 minutos • Comentários ...
69         idHash = hash(str(random.randint(10,1000)))
70         #Inicia a carteira com o valor 100
71         carteira = 100
72
73     #Envia o metodo para configurar a maquina (MetodoIpMaquina|idHash)
74     s.send(("configMaquina|" + socket.gethostname(socket.gethostname()) + "|" + str(idHash)).encode())
75     time.sleep(1)
76     data = s.recv(BUFFER_SIZE)
77     resp = repr(data)
78
79     print("\n----- # ----- \nChave Hash: " + str(idHash) + "\nCarteira: " + str(carteira) + ",00\n----- # -----")

```

Figure 2. Método Configurar Maquina

### 3.1.3. Gerar nova Hash

Este método, roda em paralelo com o código principal, ele é o responsável por atualizar a Hash constantemente de acordo com o tempo determinado. Ele atualiza o valor do ID, e manda ao servidor para atualizar, e caso não o consiga, ele volta a Hash para a anterior.

```
84 def gerarNovaHash(s):
85     global idhash
86     #i=0
87     while(True):
88         #tempo para que a hash seja atualizada
89         time.sleep(180)
90         idhashAntiga = idhash
91         id = str(random.randint(10,100))
92         idhash = hash(id)
93
94         #Manda a atualização da nova hash ao servidor (Metodo[hashAntiga][hashNova])
95         s.send(('updateidhash|'+str(idhashAntiga)+'|'+str(idhash)).encode())
96         data = s.recv(BUFFER_SIZE)
97         respostaServidor = repr(data)
98
99         #De acordo com a resposta do servidor, ou vai mandar a mensagem ao cliente de que a hash foi atualizada, ou vai vol
100         if(respostaServidor == "b'hashAtualizado'"):
101             print('\n\n----- # ----- \nHash atualizada com sucesso\nNova Hash: '+ str(idhash) +' \n----- # ----- \n\n')
102         else:
103             print('Problemas ao atualizar Hash!!')
104             idhash = idhashAntiga
```

Figure 3. Método Gerar Nova Hash

### 3.1.4. Mandar dinheiro

Este método, é o responsável por fazer o envio do dinheiro para outra maquina. Primeiro ele verifica se a Hash passada pelo cliente é válida, e caso seja ele já pergunta qual o valor que deseja enviar. Caso seja um valor inválido, ele reinicia o método para que seja escolhida outra Hash de destino.

Com a Hash já validada, é feito uma consulta ao servidor para ver quanto de dinheiro tem em sua carteira, e de acordo com esse valor, é disponibilizado para fazer o envio, caso o valor seja inválido, como por exemplo maior que o total, ou valores negativos, é perguntado novamente o valor para que seja inserido corretamente.

Assim que finalizado, o valor é enviado ao servidor.

```
12 def mandarDinheiro(s):
13     global idhash, carteira
14     valid = False
15     while(True):
16         #Coleta a hash do destinatario
17         idDestino = int(input("\nChave para envio:"))
18
19         #Verifica se a hash é valida (Metodo[hashDestino])
20         s.send(('checkHash|'+ str(idDestino)).encode())
21         data = s.recv(BUFFER_SIZE)
22         resp = repr(data)
23         if(resp == "b'Check'"):
24             valid = True
25             break
26         else:
27             print("Chave Invalida!\n----- # -----")
28
29         #Caso a hash seja válida, faz a consulta do valor da carteira
30         if(valid == True):
31             #(!Metodo[ipMaquina])
32             s.send(('getMaquina|'+ socket.gethostbyname(socket.gethostname()).encode())
33             time.sleep(1)
34             data = s.recv(BUFFER_SIZE)
35             resp = repr(data).rstrip("\n")
36             maq = resp.split('|')
37             carteira = int(maq[2])
38             while(True):
39                 #Mostra o valor em carteira, e caso o valor desejado pra mandar seja invalido,
40                 print("Carteira R$ "+ str(carteira) +",00")
41                 valor = int(input("----- # ----- \nValor que deseja enviar:"))
42                 if(valor > 0 and valor <= carteira):
43                     print("Valor Enviado!\n----- # -----")
44                     break
45                 else:
46                     print("Valor não aceito!\n----- # -----")
47
48             #Envia de fato o valor (Metodo[idHash][hashDestino][Valor])
49             s.send(('sendMoney|'+ str(idhash) +'|'+ str(idDestino) +'|'+ str(valor)).encode())
```

Figure 4. Método Mandar Dinheiro ao Servidor

## 3.2. Servidor

O Servidor é a aplicação que controla todas as transações e todas as maquinas. Todas as requisições são feitas a ele e por ele são repassadas.

### 3.2.1. Main

O Main do servidor juntamente com o `on_new_cliente` são responsáveis por receber a mensagem enviada dos clientes e redireciona-las para o verificador de métodos.

```
94 def on_new_client(clientsocket,addr):
95     while True:
96         try:
97             data = clientsocket.recv(BUFFER_SIZE)
98             if not data:
99                 break
100             mensagemCliente = data.decode('utf-8')
101             print("----- # ----- \nCliente -> {} \nPorta -> {} \nMensagem -> {}".format(addr[0], addr[1], mensagemCliente))
102             #Método para verificar qual serviço será realizado
103             validaMetodos(clientsocket,mensagemCliente)
104         except Exception as error:
105             print("Erro na conexão com o cliente!!")
106             return
107
108 def main(argv):
109     try:
110         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
111             server_socket.bind((HOST, PORT))
112             server_socket.listen(10)
113             while True:
114                 clientsocket, addr = server_socket.accept()
115                 #Mostra qual o cliente que está fazendo a requisição
116                 print("----- # ----- \nConectado ao cliente -> {}, addr, '\n'")
117                 #Inicia a thread daquela requisição
118                 t = Thread(target=on_new_client, args=(clientsocket,addr))
119                 t.start()
120             except Exception as error:
121                 print("Erro na execução do servidor!!")
122                 print(error)
123                 return
```

Figure 5. Método Main

### 3.2.2. Valida Métodos

Este método é responsável por coletar a requisição do cliente e redirecionar ao método correto, passando os parâmetros necessários para tal. Os métodos disponíveis são para adicionar uma nova maquina, atualiza-la, mandar dinheiro, verificar se a Hash é valida e coletar as informações de uma maquina.

```
72 def validaMetodos(clientsocket, mensagemCliente):
73     requisicao = mensagemCliente.split('|')
74     resposta = ""
75
76     #Todas as requisições no começa tem um metodo, de acordo com ele, é redire
77     if(requisicao[0] == 'updateIdHash'):
78         resposta = updateMaquina(requisicao[1],requisicao[2])
79     elif(requisicao[0] == 'configMaquina'):
80         resposta = addMaquina(requisicao[1],requisicao[2])
81     elif(requisicao[0] == 'sendMoney'):
82         resposta = mandarDinheiro(requisicao[1],requisicao[2],requisicao[3])
83     elif(requisicao[0] == 'checkHash'):
84         resposta = checkHash(requisicao[1])
85     elif(requisicao[0] == 'getMaquina'):
86         resposta = getMaquina(requisicao[1])
87
88     #Envia a resposta ao Cliente
89     clientsocket.send(resposta.encode())
```

Figure 6. Método Valida Métodos

### 3.2.3. Add Maquina

De acordo com o IP passado por parâmetro, ele verifica se já possui uma maquina com tal, caso não possua, ele vai adicionar uma nova maquina com uma carteira inicial de 100 reais e a Hash passada pelo cliente.

```
61 def addMaquina(ipMaquina, idHash):
62     try:
63         #Faz o teste se já tem um cadastro para aquele IP
64         maquinas.index(ipMaquina)
65     except:
66         #Caso não tenha, ele adiciona uma nova maquina
67         maquinas.append([ipMaquina, idHash, 100])
68         print(maquinas)
69         return "Maquina adicionada com sucesso!"
```

Figure 7. Método Add Maquina

### 3.2.4. Update Maquina

De acordo com a Hash passada, ele verifica novamente se possui uma maquina com esse ID, e caso possua ele a atualiza e retorna uma mensagem validando ao cliente.

```
51 def updateMaquina(idHashAntiga, idHash):
52     global maquinas
53     for maq in maquinas:
54         #Ele procura dentre as maquinas cadastradas, e atua
55         if(maq[1] == idHashAntiga):
56             maq[1] = idHash
57             print(maquinas)
58             return "hashAtualizado"
```

Figure 8. Método Update Maquina

### 3.2.5. Mandar Dinheiro

O método de mandar dinheiro verifica se a Hash destino realmente existe, e caso a tenha, ele adiciona o valor na maquina referente, e retira o mesmo valor da maquina remetente.

```
32 def mandarDinheiro(idHash, idDestino, valor):
33     global maquinas
34     valid = False
35     for maq in maquinas:
36         #Verifica qual a maquina destino e adciona o v
37         if(maq[1] == idDestino):
38             valid = True
39             maq[2] = int(maq[2]) + int(valor)
40             break
41     if(valid == True):
42         for maq in maquinas:
43             #Se de fato ficer conseguido adcionar o va
44             if(maq[1] == idHash):
45                 maq[2] = int(maq[2]) - int(valor)
46                 print(maquinas)
47                 break
48     else:
49         return "False"
```

Figure 9. Método Mandar Dinheiro

### 3.2.6. Check Maquina e Get Maquina

O Check Maquina é responsável apenas para verificar se o Hash passado existe ou não, e o Get Maquina é responsável por coletar a maquina de acordo com o IP passado.

```

14 def getMaquina(ipMaquina):
15     global maquinas
16     for maq in maquinas:
17         #Verifica se a maquina é existente e retorna tal
18         if(maq[0] == ipMaquina):
19             print(maq)
20             return "True|" + str(maq[1]) + "|" + str(maq[2]))
21     return "False"
22
23 def checkHash(idDestino):
24     global maquinas
25     for maq in maquinas:
26         #Verifica se o Hash de destino é valido
27         if(maq[1] == idDestino):
28             return "Check"
29     return "False"

```

### Figure 10. Método Check e Get Maquina

## 4. Listagem de Testes

### 4.1. Execução Correta

Execução Correta do sistema.

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda Servidor.py - Sistemas Distribuidos - Visual Studio Code
```

```
◆ Cliente.py x   ...
    ◆ log_cliente.py
    ◆ log_server.py
◆ Cliente.py x   ...
    ◆ mandandoDados
    You hit from 1 other device
```

```
PROBLEMAS SAÍDA CONSOLE DE DEPLAÇÃO TERMINAL GITLINS JUPYTER
```

```
Conectado ao cliente -> ("192.168.2.164", 55092)
----- # -----
Cliente -> 192.168.2.164
Porta -> 55092
Mensagem -> getMapina|192.168.1.132
----- # -----
Cliente -> 192.168.2.164
Porta -> 55092
Mensagem -> configMapina|192.168.1.132|-2498408735903589370
[["192.168.1.132", "-2498408735903589370", 100]]
----- # -----
Conectado ao cliente -> ("192.168.2.126", 64037)
----- # -----
Cliente -> 192.168.2.126
Porta -> 64037
Mensagem -> getMapina|192.168.2.126
----- # -----
Cliente -> 192.168.2.126
Porta -> 64037
Mensagem -> configMapina|192.168.2.126|661426411334488557
[["192.168.1.132", "-2498408735903589370", 100], ["192.168.2.126", "661426411334488557", 100]]
----- # -----
Cliente -> 192.168.2.164
Porta -> 55092
Mensagem -> checktash|661426411334488557
----- # -----
Cliente -> 192.168.2.164
Porta -> 55092
Mensagem -> getMapina|192.168.1.132
["192.168.1.132", "-2498408735903589370", 100]
----- # -----
Cliente -> 192.168.2.164
Porta -> 55092
Mensagem -> sendUniquely|-2498408735903589370|661426411334488557|[50
[["192.168.1.132", "-2498408735903589370", 50], ["192.168.2.126", "661426411334488557", 1
50]]
Irro na conexão com o cliente!
```

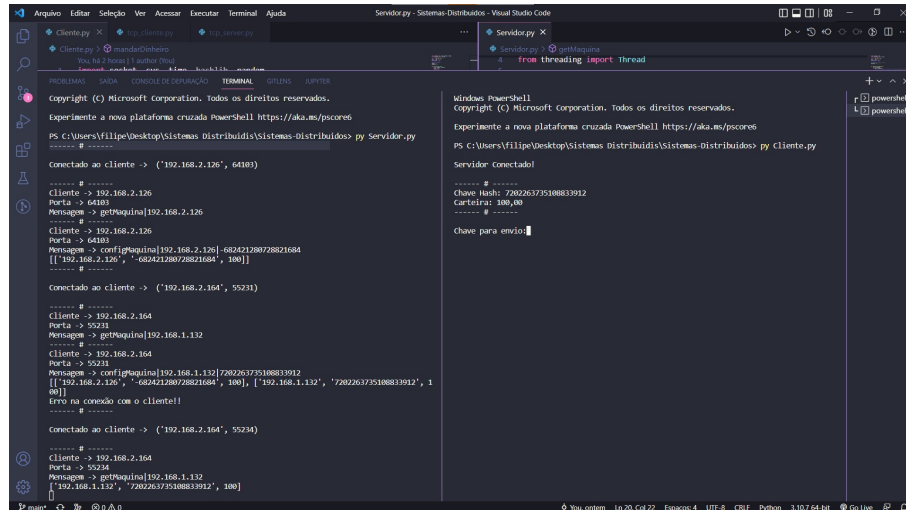
```
Servidor.py x
◆ Servidor.py x
    ◆ getMapina
    A from threading import Thread
```

```
Experimente a nova plataforma cruzada PowerShell https://aka.ms/powershell
PS C:\Users\filipe\Desktop\Sistemas Distribuidos> python Clientes.py
Servidor Conectado!
----- # -----
Chave Hash: -2498408735903589370
Carteira: 100,00
----- # -----
Chave para envio:661426411334488557
Carteira R$ 100,00
----- # -----
Valor que deseja enviar:50
Valor enviado
----- # -----
Deseja desconectar ?
s ou n > s
```

**Figure 11. Teste da execução correta do sistema**

## 4.2. Maquina já cadastrada

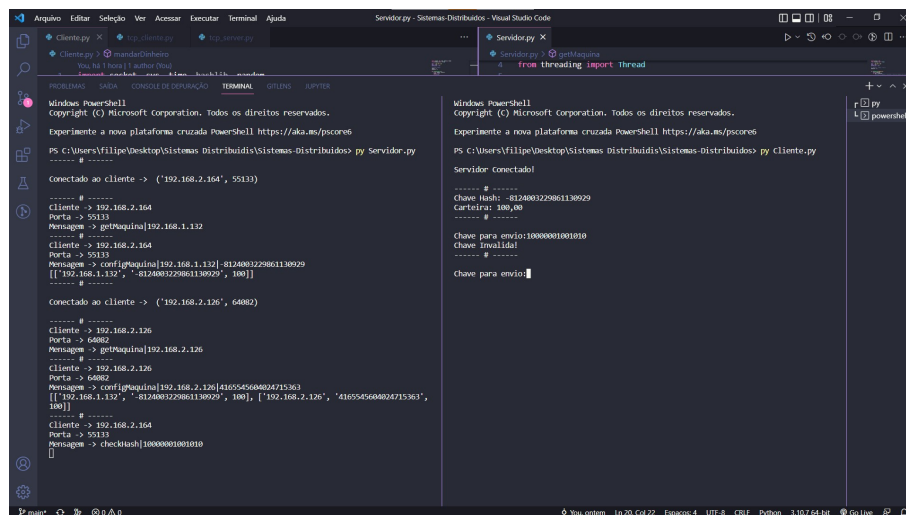
Teste em que uma maquina estava cadastrada, se desconectou e tentou se conectar novamente.



**Figure 12. Maquina já cadastrada**

### 4.3. Hash Inválida

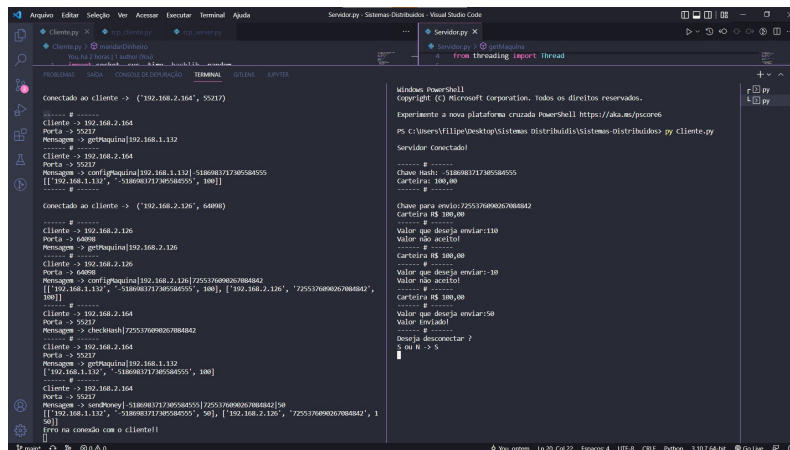
Teste em que a hash está errada.



### Figure 13. Hash Inválida

#### 4.4. Valor inconsistente

Teste em que o valor digitado é inconsistente.



### Figure 14. Valor Inconsistente

## 5. Conclusão

O desenvolvimento do trabalho gerou algumas dificuldades no envio das requisições para o servidor, e como elas seriam tratadas, mas apos o funcionamento do primeiro método o restante funcionou de maneira eficaz.

## 6. Referência

Documentação da linguagem python. Disponível em: <https://docs.python.org/pt-br/3/> .  
Acessado em: 12/09.

## References