

**CENTRO UNIVERSITÁRIO MULTIVIX**  
**ENGENHARIA DA COMPUTAÇÃO**

**PROGRAMAÇÃO DISTRIBUÍDA E PARALELA**

**Professor: Breno Krohling**

**Laboratório II – Chamada de Procedimento Remoto (RPC)**

**Maria Eduarda Felipe Fasollo**

**VITÓRIA, ES**

**2025**

## **Introdução**

Este relatório técnico abrange a implementação e a análise comparativa de dois projetos distintos que utilizam a comunicação por Chamada de Procedimento Remoto (RPC): uma Calculadora baseada na arquitetura tradicional ONC RPC em C e um Minerador implementado com a arquitetura moderna gRPC em Python. O objetivo é demonstrar a capacidade de construir sistemas distribuídos robustos, avaliando as diferenças metodológicas entre os ambientes C/rpcgen e Python/gRPC, e validar as lógicas específicas de cada aplicação, como a validação de dados e o gerenciamento de concorrência e estado.

# ATIVIDADE 1 – Calculadora RPC

## 1. Metodologia de Implementação

O projeto utiliza a arquitetura cliente-servidor para executar operações matemáticas remotamente. A implementação foi dividida em três componentes principais:

### 1.1 Definição do Protocolo e Geração de Stubs

A comunicação é regida pelo arquivo de protocolo **rpcCalc.x**. Este arquivo define o contrato da interface de serviço, incluindo:

- Estrutura de Dados: Definição da estrutura **operandos**, que encapsula as duas variáveis de entrada (x e y), ambas do tipo **inteiro (int)**.
- Funções Remotas: Declaração das quatro operações básicas (ADD, SUB, MUL, DIV) que o servidor deve implementar.
- Geração de Código: A ferramenta **rpcgen** processou o protocolo, gerando um código intermediário de rede e interface necessário para converter estruturas de dados (como o int x e int y) para ser enviado pela rede (e vice-versa).

### 1.2 Implementação do Servidor

O servidor (rpcCalc\_server.c) atua como provedor de serviço. A lógica é:

1. Registro: O servidor se registra no serviço rpcbind, permitindo que os clientes localizem o serviço na rede.
2. Execução: Implementação das funções de cálculo definidas no protocolo. Ao receber uma solicitação, o servidor extrai os operandos, realiza a operação e retorna o resultado como um ponteiro para um inteiro (int\*).

### 1.3 Implementação do Cliente

O cliente (rpcCalc\_client.c) fornece a interface de usuário e orquestra as chamadas remotas.

1. Conexão: O cliente se conecta ao servidor, utilizando 'localhost' como endereço.
2. Menu interativo: O cliente apresenta um menu de opções (soma, subtração, multiplicação, divisão, sair).
3. Validação de Entrada: A lógica de entrada foi implementada de forma a garantir que **apenas números inteiros válidos** sejam aceitos para os operandos, protegendo o sistema contra dados inválidos, além de garantir acesso correto no momento de escolha no menu.

## 2. Testes e Considerações

Os testes concentraram-se em duas áreas críticas: Configuração da Comunicação RPC e Integridade da Entrada de Dados.

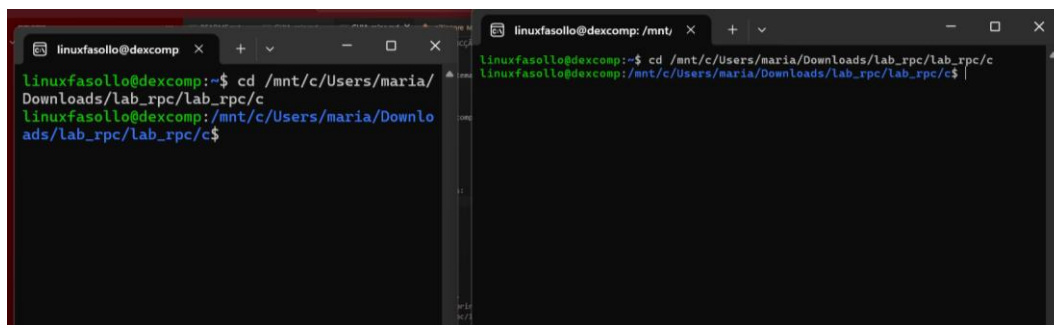
### 2.1 Teste de integridade da entrada de dados

O foco foi garantir que o cliente só insira a opção possíveis no menu interativo e números inteiros para os operandos, conforme o tipo de dado definido no protocolo (int).

- **Rejeição de Caracteres:** Entradas não numéricas (ex: a) são rejeitadas, e o menu é reexibido.
- **Proibição de Decimais:** Entradas com ponto ou vírgula (ex: 4.5 ou 4,5) são detectadas. O sistema lê o número inteiro (4), detecta o caractere extra (ponto ou vírgula) no *buffer* e rejeita a entrada, prevenindo que o servidor receba dados parciais ou inconsistentes com o formato int.

## 3. Passo a passo de compilação e execução

1. Preparação e acesso ao diretório: serão necessários dois terminais abertos. Em ambos os terminais, navegue até o diretório que contém os arquivos do código fonte.



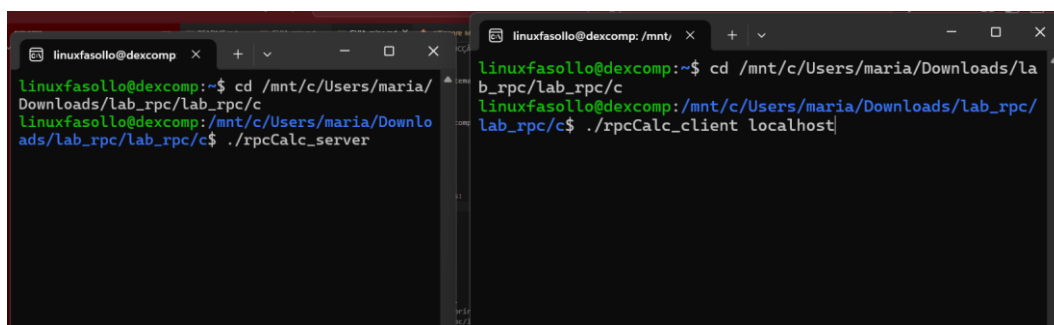
The image shows two terminal windows side-by-side. The left window shows the user navigating to the directory `/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c`. The right window shows the user navigating to the directory `/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c`.

```
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c$

linuxfasollo@dexcomp: /mnt/ $ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c$
```

2. No terminal 1, execute o programa do servidor com o comando `./rpcCalc_server`

3. No terminal 2, execute o programa do cliente, passando o endereço do servidor como argumento (localhost) com o comando `./rpcCalc_client localhost`.



The image shows two terminal windows side-by-side. The left window shows the user running the server program `./rpcCalc_server`. The right window shows the user running the client program `./rpcCalc_client localhost`.

```
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c$ ./rpcCalc_server

linuxfasollo@dexcomp: /mnt/ $ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c$ ./rpcCalc_client localhost
```

4. Após a execução correta dos comandos, o menu interativo aparecerá no terminal do cliente.

```
linuxfasollo@dexcomp: /mnt/ × + v
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/c/lab_rpc/c
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/c/lab_rpc/c$ ./rpcCalc_client localhost

      CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: |
```

5. Execução dos testes

- Soma e Subtração

```
linuxfasollo@dexcomp: /mnt/ × + v
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c$ ./rpcCalc_client localhost

      CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 1
Digite o primeiro numero [x]: 5
Digite o segundo número [y]: 8
Resultado do servidor: 13

      CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 2
Digite o primeiro numero [x]: 9
Digite o segundo número [y]: 3
Resultado do servidor: 6

      CLIENTE RPC
---- CALCULADORA ----
```

- Multiplicação e Divisão

```

linuxfasollo@dexcomp: /mnt/ x + v - □ x
Escolha a operação desejada: 2
Digite o primeiro numero [x]: 9
Digite o segundo número [y]: 3
Resultado do servidor: 6

CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 3
Digite o primeiro numero [x]: 4
Digite o segundo número [y]: 8
Resultado do servidor: 32

CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 4
Digite o primeiro numero [x]: 10
Digite o segundo número [y]: 2
Resultado do servidor: 5

CLIENTE RPC
---- CALCULADORA ----

```

6. Validação de erros: o programa não aceita caracteres diferentes e números que não sejam inteiros.

```

linuxfasollo@dexcomp: /mnt/ x + v -
Resultado do servidor: 5

CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 2.3
Opção inválida. Apenas números inteiros são aceitos para a opção.

CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: ,
Opção inválida. Tente novamente.

CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 6
Opção inválida. Tente novamente.

CLIENTE RPC
---- CALCULADORA ----

```

```

linuxfasollo@dexcomp: /mnt/ x + v - □ x
5. Sair
Escolha a operação desejada: 6
Opção inválida. Tente novamente.

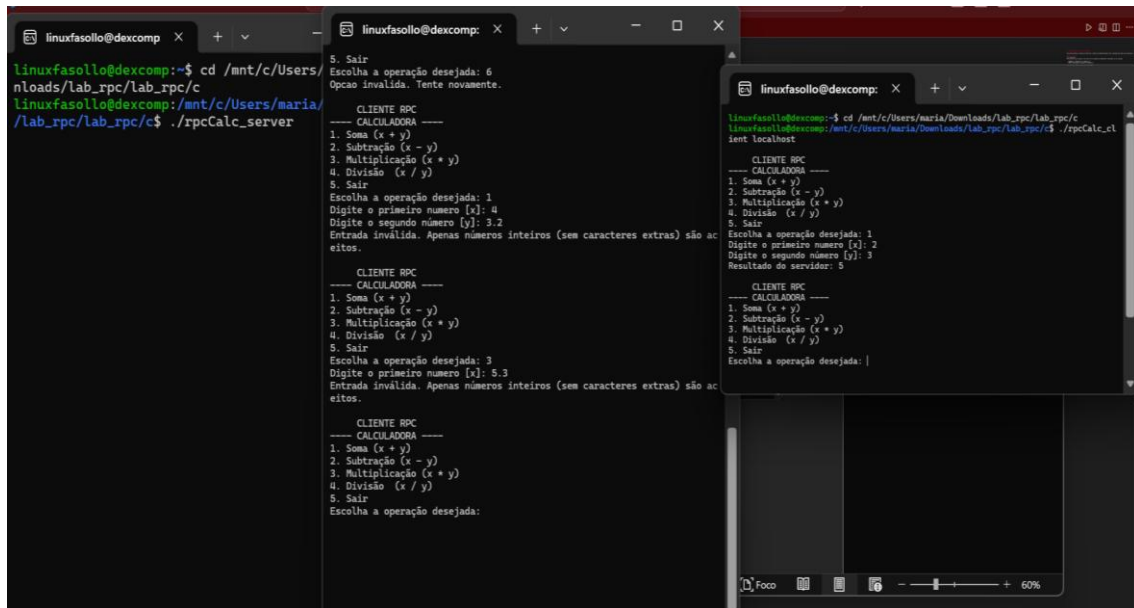
CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 1
Digite o primeiro numero [x]: 4
Digite o segundo número [y]: 3.2
Entrada inválida. Apenas números inteiros (sem caracteres extras) são aceitos.

CLIENTE RPC
---- CALCULADORA ----
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 3
Digite o primeiro numero [x]: 5.3
Entrada inválida. Apenas números inteiros (sem caracteres extras) são aceitos.

CLIENTE RPC
---- CALCULADORA ----

```

## 7. Exemplificando servidor atuando com dois clientes simultâneos:



The image shows three terminal windows illustrating a server-client interaction for a calculator application. The server is running on the left, and two clients are running on the right.

**Terminal 1 (Server):**

```
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/c
linuxfasollo@dexcomp:~/Downloads/lab_rpc/lab_rpc/c$ ./rpcCalc_server
```

**Terminal 2 (Client 1):**

```
linuxfasollo@dexcomp:~$ ./rpcCalc_client localhost
```

**Terminal 3 (Client 2):**

```
linuxfasollo@dexcomp:~$ ./rpcCalc_client localhost
```

The server output shows the menu and user input for the first client:

```
5. Sair
Escolha a operação desejada: 6
Opcao invalida. Tente novamente.

CLIENTE RPC
--- CALCULADORA ---
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 1
Digite o primeiro número [x]: 4
Digite o segundo número [y]: 3.2
Entrada inválida. Apenas números inteiros (sem caracteres extras) são aceitos.
```

The server output shows the menu and user input for the second client:

```
CLIENTE RPC
--- CALCULADORA ---
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: 3
Digite o primeiro número [x]: 5.3
Entrada inválida. Apenas números inteiros (sem caracteres extras) são aceitos.
```

The server output shows the menu and user input for the third client:

```
CLIENTE RPC
--- CALCULADORA ---
1. Soma (x + y)
2. Subtração (x - y)
3. Multiplicação (x * y)
4. Divisão (x / y)
5. Sair
Escolha a operação desejada: |
```

## ATIVIDADE 2 – Minerador de Criptomoedas usando gRPC

### 1. Metodologia de Implementação – gRPC, Python

Esse relatório detalha a metodologia, teste e resultados do projeto Minerador, simulando um sistema de mineração distribuída utilizando a arquitetura gRPC com Python, seguindo o enunciado do projeto, uso de SHA-1 para o desafio e implementação das chamadas remotas.

#### 1.1 Protocolo e Comunicação

- Definição do Contrato: O arquivo `.proto` define o serviço (api) e as estruturas de dados que controlam o fluxo de trabalho de mineração.
- Geração de Stubs: A ferramenta `protoc` (em conjunto com `grpc_tools`) gera os módulos Python (`_pb2.py` e `_pb2_grpc.py`). Estes *stubs* gerenciam a serialização e a comunicação de rede, permitindo que o foco da engenharia fosse direcionado à lógica de concorrência.

#### 1.2 Lógica do servidor

O servidor gerencia o estado global para garantir a integridade da disputa:

- Estado Global: O servidor mantém um estado global (`SERVER_STATE`) que rastreia a transação atual (`current_transaction_id`) e um histórico de vencedores.
- Proof-of-Work (PoW): A validação é baseada no algoritmo SHA-1. Uma solução é considerada válida apenas se o *hash* gerado iniciar com um número de zeros igual ao `challenge_value` (nível de dificuldade atual).
- Gerenciamento de Concorrência (`submitChallenge`): Esta função é crítica para a integridade. O servidor aceita a primeira solução válida para o ID de transação pendente, registra o `clientID` vencedor e imediatamente incrementa o ID da transação. Soluções subsequentes enviadas para o ID já resolvido são rejeitadas com um código de erro específico.

#### 1.3 Lógica do Cliente

O cliente simula o papel de um minerador real.

- Função 'mine': O cliente implementa a função de mineração, que consulta o desafio atual, itera sobre valores incrementais, calcula o *hash* para cada nonce e submete a primeira solução válida encontrada ao servidor.
- Disputa: O uso de múltiplos clientes com IDs únicos (ex: 100 e 200) simula a disputa real pela recompensa.



## **2. Testes e Considerações**

Os testes focaram em validar a integridade do estado do servidor e sua capacidade de lidar corretamente com a alta concorrência no evento de vitória.

### **2.1 Resultado 1: Lógica de vitória**

- Teste: dois clientes com diferentes IDs (ex: 100 e 200) executam a opção 6 (MINE) ao mesmo tempo.
- Resultado: apenas um cliente (o vencedor) obteve o retorno de sucesso. O outro cliente recebe a mensagem informando que o desafio já foi resolvido.

O teste foi bem-sucedido porque o servidor reconheceu apenas um dos clientes como vencedor. O cliente perdedor foi imediatamente notificado com uma mensagem de erro, indicando que o desafio já havia sido resolvido.

### **2.2 Resultado 2: Dificuldade Progressiva**

Os testes confirmaram a consistência do estado global do servidor ao longo de múltiplas transações.

- Teste: Observar o `challenge_value` e o histórico de vencedores após cada transação resolvida.
- Resultado Validado: O servidor incrementou o `challenge_value` a cada transação resolvida, simulando o aumento de dificuldade que ocorre em redes reais de mineração. Isso validou que o servidor mantém o estado (`SERVER_STATE`) de forma consistente entre as chamadas RPC concorrentes.

### **2.3 Limitação de Recursos e Custo Computacional**

Uma observação crítica feita durante a fase de testes foi a escala do custo operacional imposto pelo algoritmo PoW.

- O projeto demonstrou que a dificuldade de mineração cresce exponencialmente. Para manter a estabilidade do sistema e evitar a sobrecarga do ambiente de teste, o nível máximo de desafio (`MAX_CHALLENGE`) foi limitado a 6 zeros.
- Essa restrição foi necessária porque tentar processar o teste com um nível de dificuldade de 7 zeros resultou em falhas operacionais da máquina.

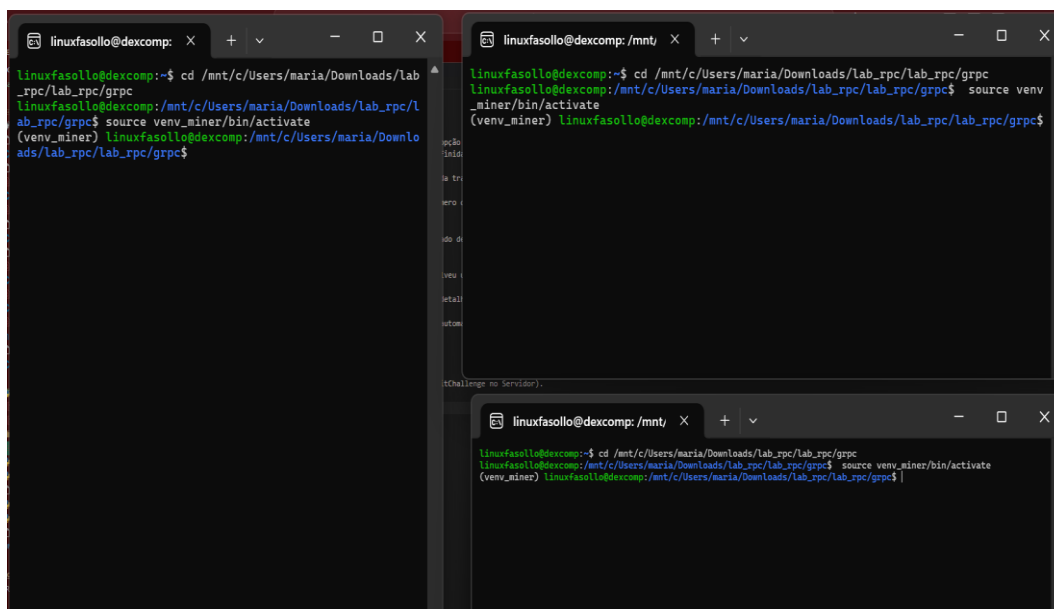
### 3. Passo a passo de compilação e execução

Pré-requisitos: O usuário deve garantir que possui o Python 3 e as ferramentas necessárias para construir o ambiente gRPC.

- Instalação de ferramentas:  
*sudo apt install python3 python3-pip protobuf-compiler python3-venv*
- Criação do ambiente virtual (Venv) dentro da pasta com código fonte:  
*python -m venv venv\_miner*

1. Preparação e ativação do ambiente: Serão necessários três terminais abertos no ambiente Linux/WSL.

2. Navegue até a pasta onde contém o diretório grpc do projeto e ative o ambiente virtual com o comando `source venv_miner/bin/activate`.



The image shows three terminal windows in a Linux/WSL environment. The top-left terminal shows the user navigating to the directory `/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc` and activating the virtual environment `venv_miner`. The top-right terminal shows the same user in the same directory, activating the virtual environment. The bottom terminal shows the user in the same directory, activating the virtual environment. The output of the commands is visible in each terminal.

```
linuxfasollo@dexcomp: ~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc
linuxfasollo@dexcomp: /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$ source venv_miner/bin/activate
(venv_miner) linuxfasollo@dexcomp: /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$
```

```
linuxfasollo@dexcomp: ~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc
linuxfasollo@dexcomp: /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$ source venv_miner/bin/activate
(venv_miner) linuxfasollo@dexcomp: /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$
```

```
linuxfasollo@dexcomp: /mnt/ X + v
linuxfasollo@dexcomp: ~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc
linuxfasollo@dexcomp: /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$ source venv_miner/bin/activate
(venv_miner) linuxfasollo@dexcomp: /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$
```

3. No terminal 1, inicie o servidor do Minerador com o comando `python mine_grpcCalc_server.py`

4. No terminal 2, inicie o primeiro concorrente, com o comando `python grpcCalc_client.py localhost:50051 100` (podendo modificar o último valor, que é referente ao ID do cliente).

5. No terminal 3, inicie o segundo concorrente com o comando `python grpcCalc_client.py localhost:50051 200` (podendo alterar o valor “200”).

6. Após a execução correta dos comandos, o menu interativo do Minerador aparecerá nos terminais dos clientes e o Servidor já irá gerar o primeiro challenge.

```
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc
linuxfasollo@dexcomp:~/lab_rpc/grpc$ source venv_miner/bin/activate
linuxfasollo@dexcomp:~/lab_rpc/grpc$ python mine_grpcCalc_server.py
NOVO DESAFIO GERADO: ID 0, ZEROS 2
Servidor gRPC Minerador rodando na porta 50051...

linuxfasollo@dexcomp:~/lab_rpc/lab_rpc/grpc$ python mine_grpcCalc_client.py
CLIENTE INICIADO: ID ÚNICO = 100. Conectando a localhost:50051...

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção:
```

## 7. Execução dos testes

7.1 Os dois usuários consultam o ID Pendente, sendo 0 o atual, já que não houve nenhuma mineração executada.

```
linuxfasollo@dexcomp:~/lab_rpc/lab_rpc/grpc$ python mine_grpcCalc_client.py
CLIENTE INICIADO: ID ÚNICO = 100. Conectando a localhost:50051...

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 1
ID da Transação Atual Pendente: 0

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 1
ID da Transação Atual Pendente: 0
```

7.2 Consulta valor do desafio (atual x não atual)

```
linuxfasollo@dexcomp:~/lab_rpc/lab_rpc/grpc$ python mine_grpcCalc_client.py
CLIENTE INICIADO: ID ÚNICO = 100. Conectando a localhost:50051...

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 2
ID da Transação para consultar o Desafio: 0
Desafio associado ao ID 0: 2 zeros à esquerda.

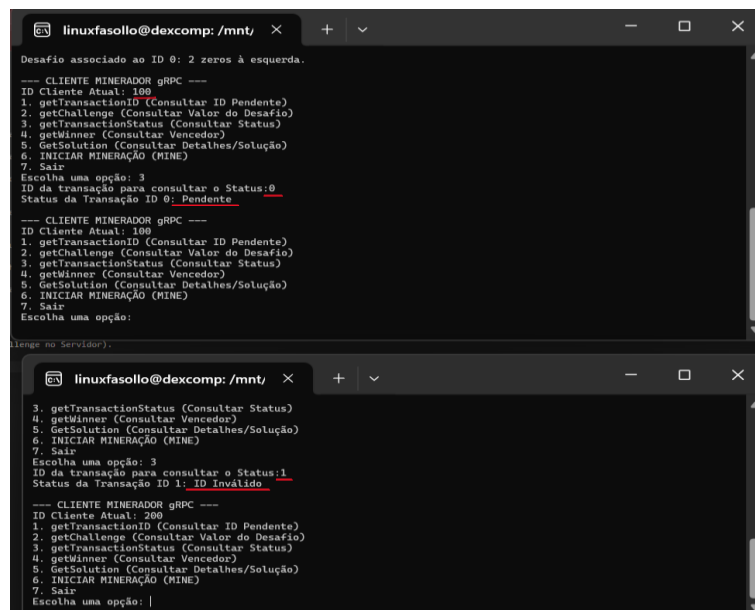
--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
```

```
linuxfasollo@dexcomp:~/lab_rpc/lab_rpc/grpc$ python mine_grpcCalc_client.py
CLIENTE INICIADO: ID ÚNICO = 200. Conectando a localhost:50051...

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 2
ID da Transação para consultar o Desafio: 1
Erro: ID 1 inválido. Retorno: -1

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: |
```

### 7.3 Consultar status (status pendente para ID do desafio atual que ainda não foi solucionado x status inválido para ID do desafio que ainda não foi gerado).

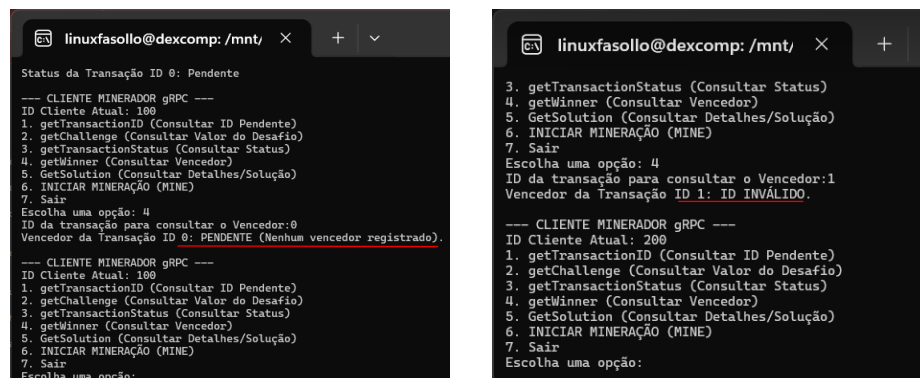


```
linuxfasollo@dexcomp: /mnt/
Desafio associado ao ID 0: 2 zeros à esquerda.

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 3
ID da transação para consultar o Status:0
Status da Transação ID 0: Pendente

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção:
```

### 7.4 Consultar vencedor (status pendente para ID do desafio atual que ainda não foi solucionado x status inválido para ID do desafio que ainda não foi gerado).



```
linuxfasollo@dexcomp: /mnt/
Status da Transação ID 0: Pendente

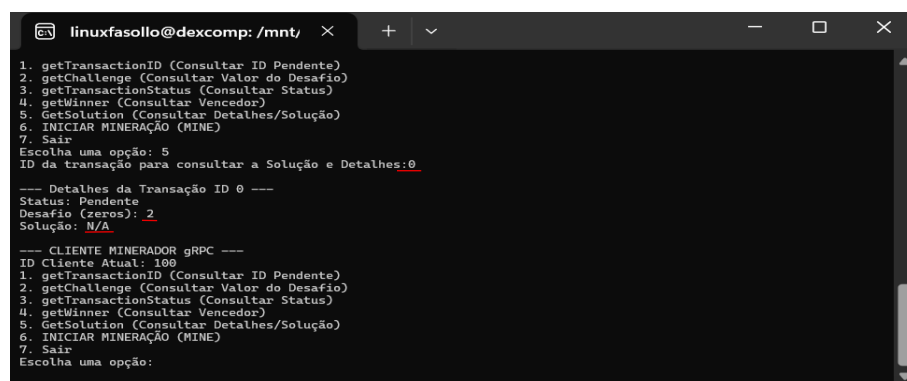
--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 4
ID da transação para consultar o Vencedor:0
Vencedor da Transação ID 0: PENDENTE (Nenhum vencedor registrado).

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção:
```

```
linuxfasollo@dexcomp: /mnt/
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 4
ID da transação para consultar o Vencedor:1
Vencedor da Transação ID 1: ID INVÁLIDO.

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção:
```

### 7.5 Consultar detalhes e solução. O desafio do ID 0 é um desafio de 2 zeros e está pendente, portanto, não há solução.



```
linuxfasollo@dexcomp: /mnt/
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 5
ID da transação para consultar a Solução e Detalhes:0

--- Detalhes da Transação ID 0 ---
Status: Pendente
Desafio (zeros): 2
Solução: N/A

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção:
```

7.5 O desafio do ID 1 ainda não foi gerado, portanto, não há valor de desafio nem ID existente.

```
linuxfasollo@dexcomp: /mnt/
7. Sair
Escolha uma opção: 5
ID da transação para consultar a Solução e Detalhes:1

--- Detalhes da Transação ID 1 ---
Status: ID Inválido
Desafio (zeros): 0
Solução: ID INVÁLIDO

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: |
```

## 7.6 INICIAR MINERADOR

Cliente 100 rodou o minerador, o sistema informa que houve solução para o challenge ID 0 pelo cliente respectivo e informa que um Novo desafio já foi gerado.

```
linuxfasollo@dexcomp:~$ cd /mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc
linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$ source venv_miner/bin/activate
(venv_miner) linuxfasollo@dexcomp:/mnt/c/Users/maria/Downloads/lab_rpc/lab_rpc/grpc$ python mine.grpcCalc_server.py
Servidor gRPC Minerador rodando na porta 50051...
NOVO DESAFIO GERADO: ID 0, ZEROS 2
NOVO DESAFIO GERADO: ID 1, ZEROS 3
SUCESSO! Solução aceita para ID 0 pelo ClientID 100.

linuxfasollo@dexcomp: /mnt/
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 6

--- INICIANDO MINERAÇÃO ---
1. TransactionID atual: 0
2. Desafio (zeros à esquerda): 2
3. Mineração multithread em andamento (8 threads)...
4. Solução encontrada (Local): Client-100-100-0
SUCESSO! A solução foi aceita. Você (ID 100) ganhou a recompensa e um novo desafio foi gerado!

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 100
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: |

linuxfasollo@dexcomp: /mnt/
7. Sair
Escolha uma opção: 5
ID da transação para consultar a Solução e Detalhes:1

--- Detalhes da Transação ID 1 ---
Status: ID Inválido
Desafio (zeros): 0
Solução: ID INVÁLIDO

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: |
```

### 7.6.1 Cliente 2 consulta ganhador

```
linuxfasollo@dexcomp: /mnt/
--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 4
ID da transação para consultar o Vencedor:0
🏆 VENCEDOR: Client ID 100 para Transação ID 0!

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: |
```

### 7.6.2 Cliente verifica status do novo desafio gerado

```
linuxfasollo@dexcomp: /mnt/
🏆 VENCEDOR: Client ID 100 para Transação ID 0!

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 1
ID da Transação Atual Pendente: 1

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: |
```

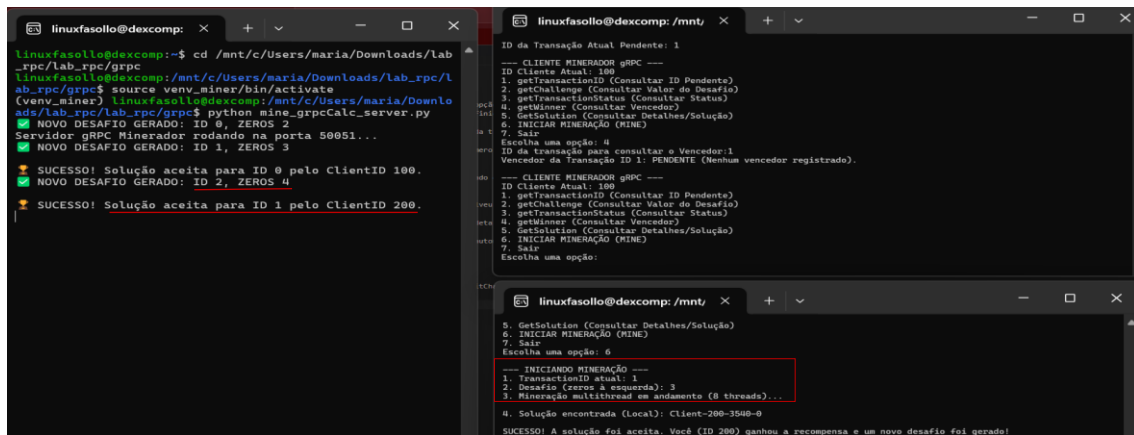
### 7.6.3 Cliente 2 verifica status do ID 0

```
linuxfasollo@dexcomp: /mnt/

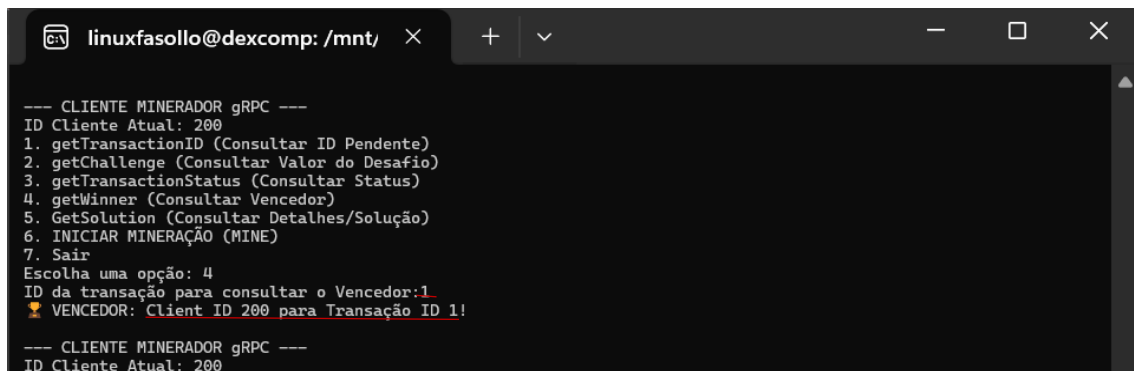
--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
Escolha uma opção: 3
ID da transação para consultar o Status:0
Status da Transação ID 0: Resolvido

--- CLIENTE MINERADOR gRPC ---
ID Cliente Atual: 200
1. getTransactionID (Consultar ID Pendente)
2. getChallenge (Consultar Valor do Desafio)
3. getTransactionStatus (Consultar Status)
4. getWinner (Consultar Vencedor)
5. GetSolution (Consultar Detalhes/Solução)
6. INICIAR MINERAÇÃO (MINE)
7. Sair
```

7.7 Cliente 200 rodou o minerador, o sistema informa que houve solução para o challenge ID 1 pelo cliente respectivo e informa que um Novo desafio já foi gerado.

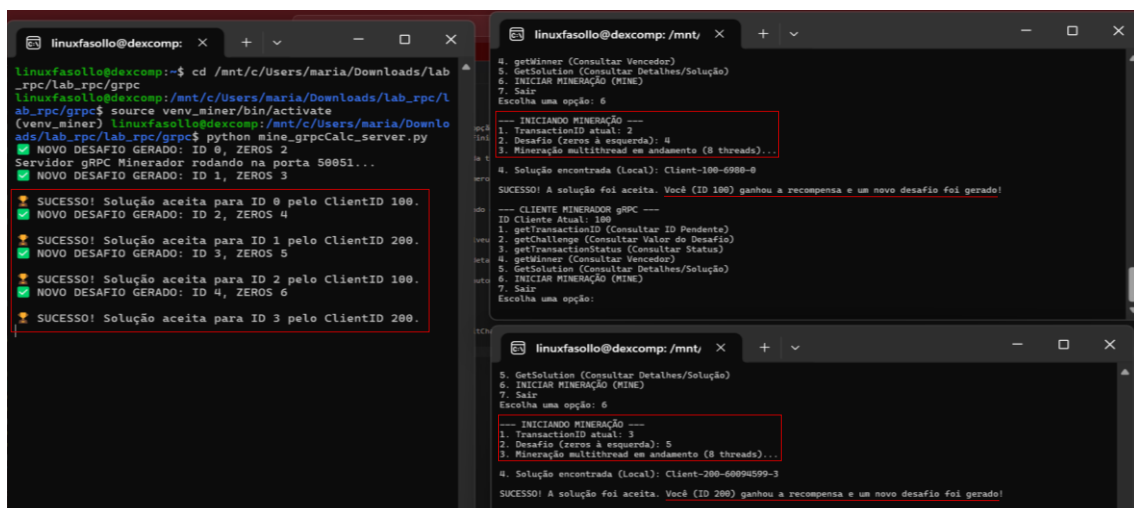


### 7.7.1 Cliente 2 verifica ganhar do ID 1



### 7.8 Os dois clientes mineram:

- O cliente 100 soluciona o desafio do ID 2, de nível 4.
- O servidor gera novo desafio: ID 3 de nível 5.
- O cliente 200 soluciona o desafio do ID 3, de nível 5.
- O sistema gera novo desafio: ID 4, nível 6.



## **Conclusão**

O desenvolvimento da Calculadora RPC em C e do Minerador PoW em Python proporcionou uma visão prática e comparativa das metodologias de desenvolvimento em sistemas distribuídos. O projeto demonstrou que a escolha da arquitetura é fundamental para atender aos requisitos específicos de cada aplicação. Enquanto o RPC em C se provou eficaz para a comunicação de baixo nível, exigindo rigoroso controle sobre a formatação de dados, e o gRPC em Python demonstrou ser superior para implementar sistemas modernos e de alto desempenho que exigem complexas lógicas de concorrência e troca de dados estruturados.

A Calculadora RPC (C), utilizando o método tradicional, exigiu um rigoroso foco na integridade dos dados. Os testes confirmaram a eficácia da lógica de validação do cliente em garantir que apenas números inteiros fossem enviados pela rede, provando que, mesmo em arquiteturas mais antigas, é possível manter a confiabilidade da comunicação. Em contraste, o Minerador (gRPC/Python) abordou desafios modernos de concorrência. A arquitetura gRPC permitiu focar diretamente no problema central: a proteção do estado do servidor. O sucesso do teste de disputa, onde apenas um cliente pôde registrar a vitória, validou a solução implementada contra a condição de disputa, assegurando que a regra de ter um vencedor por transação fosse mantida.

A conclusão integrada é que ambos os projetos foram bem-sucedidos em validar suas lógicas centrais, fornecendo uma base sólida para a compreensão das decisões de engenharia necessárias na construção de aplicações distribuídas.