1. Metodologia de Implementação

A implementação do projeto foi realizada utilizando a linguagem **Python** e a biblioteca **gRPC**. A escolha pelo gRPC se deu por ser uma estrutura de RPC moderna, de alto desempenho e de código aberto, que utiliza o **Protobuf** (Protocol Buffers) como mecanismo para serializar dados estruturados.

O fluxo de desenvolvimento para ambas as atividades (Calculadora e Minerador) seguiu as etapas padronizadas do gRPC:

Definição do Contrato de Serviço: Primeiramente, foi criado um arquivo.proto para cada atividade. Este arquivo é a "planta baixa" da comunicação, onde foram definidos os serviços (os nomes das funções remotas) e as estruturas das mensagens (os dados de requisição e resposta).

Geração de Código (Stubs): O compilador do gRPC (grpc_tools.protoc) foi utilizado para ler o arquivo.proto e gerar automaticamente os arquivos-base em Python. Esses arquivos contêm os "stubs" do cliente (que abstraem a chamada de rede) e as classes "servicer" do servidor (que servem como esqueleto para a lógica de negócios).

Implementação do Servidor: Uma classe de servidor foi criada, herdando da classe "servicer" gerada. A lógica de negócios de cada função remota (ex: o cálculo matemático da calculadora ou a validação do hash do minerador) foi implementada dentro desta classe. O servidor foi configurado para rodar em um *thread pool* e escutar em uma porta de rede específica, aguardando conexões.

Implementação do Cliente: O script do cliente foi desenvolvido para se conectar ao endereço e porta do servidor. O cliente utiliza o "stub" gerado para fazer chamadas de função remotas como se fossem funções locais. Uma interface de usuário baseada em menu de console foi criada para permitir a interação do usuário.

Gerenciamento de Estado (Minerador): Para a Atividade 2, o servidor foi projetado para manter uma tabela interna de transações e seus estados (desafio, solução, vencedor). Mecanismos de controle de concorrência (locks) foram utilizados para garantir que o estado da tabela permanecesse íntegro, especialmente durante a submissão de soluções por múltiplos clientes simultâneos.

2. Testes Realizados

Para validar o funcionamento das aplicações, foi montado um ambiente de teste local, onde o servidor era executado em um terminal e um ou mais clientes eram executados em terminais separados.

Atividade 1: Calculadora

Teste de Operações Básicas: Foram testadas as quatro operações (soma, subtração, multiplicação e divisão) com valores numéricos válidos para garantir que o resultado matemático era corretamente calculado no servidor e retornado ao cliente.

Teste de Caso de Borda (Erro): Foi realizado um teste específico de divisão por zero. O objetivo era verificar se o servidor capturaria a exceção matemática e se o gRPC comunicaria esse erro de volta ao cliente, que por sua vez deveria exibir uma mensagem amigável em vez de travar.

Atividade 2: Minerador de Criptomoedas

Teste de Funções "Get": Foi validado se um cliente conseguia obter corretamente os dados do servidor (ID da transação atual, desafio, status, etc.).

Teste de Mineração (Cliente Único): Um único cliente foi executado. Foi usada a opção "Mine" para buscar o desafio, resolvê-lo localmente (força bruta) e submeter a solução. O teste verificou se o servidor aceitou a solução como válida.

Teste de Concorrência (Múltiplos Clientes): Este foi o teste principal. O servidor foi iniciado, seguido por dois clientes (Cliente A e Cliente B). Ambos os clientes receberam a ordem de minerar o mesmo desafio simultaneamente. O objetivo era verificar se o servidor lidaria corretamente com a "corrida": o primeiro a submeter a solução válida seria declarado o vencedor e o segundo cliente (mesmo com a solução correta) seria notificado de que o desafio já havia sido resolvido.

Teste de Integridade de Estado: Após o teste de concorrência, o servidor foi consultado novamente. Verificou-se se o Winner da transação anterior foi corretamente registrado e se o servidor havia gerado um novo desafio para a próxima transação, permitindo a continuidade da mineração.

3. Resultados Encontrados

Os testes confirmaram que as duas aplicações funcionaram conforme o esperado.

Resultados da Calculadora:

As operações básicas retornaram resultados matemáticos corretos.

No teste de divisão por zero, o servidor identificou o erro, e o cliente recebeu a notificação de "INVALID_ARGUMENT", exibindo a mensagem "Divisão por zero não é permitida." Isso valida o mecanismo de propagação de erros do gRPC.

Resultados do Minerador:

As funções de consulta (getters) retornaram os dados corretos do estado inicial do servidor.

O teste de cliente único foi bem-sucedido, com o servidor aceitando a solução e se preparando para o próximo desafio.

O teste de concorrência foi o mais conclusivo:

O Cliente A, que submeteu a solução primeiro, recebeu a resposta de "Solução VÁLIDA".

O Cliente B, que submeteu logo em seguida, recebeu a resposta "Desafio JÁ FOI SOLUCIONADO".

O terminal do servidor registrou a solução do Cliente A como vencedora e imediatamente criou uma nova transação com um novo desafio.

Os resultados demonstram que o servidor gRPC foi capaz de gerenciar o estado da aplicação e lidar com requisições concorrentes de forma robusta e correta, conforme especificado pelos requisitos do laboratório.