

Introdução

Este projeto tem como propósito a simulação de um sistema distribuído que replica as dinâmicas de transferências externas entre clientes e instituições financeiras. O ambiente simulado é composto por três elementos principais: o código do Cliente em Python, o código do Banco em Java e o código do Banco Central em C++. Cada componente desempenha um papel crucial na simulação da complexa interação entre diferentes entidades no contexto financeiro.

O Código do Cliente, elaborado em Python, é a interface inicial das transações, representando a origem das transferências. Com o uso da biblioteca `requests`, este código gera solicitações aleatórias de transferência para diversas instituições financeiras, simbolizando a diversidade de operações possíveis. Ao realizar requisições HTTP para o serviço do Banco, processa as respostas e exibe os resultados, proporcionando uma visão prática e dinâmica da interação entre clientes e o sistema financeiro distribuído.

O Código do Banco, desenvolvido em Java com base no framework `Spark`, opera como um intermediário entre os clientes e o sistema centralizado do Banco Central. Essencial para o projeto, este código coordena efetivamente a execução das transferências externas, escolhendo a instituição financeira de destino com base no cliente ID. Adicionalmente, realiza chamadas a uma API externa simulada, representando a obtenção de informações do cliente de destino junto ao Banco Central. A robustez deste componente é crucial para garantir a execução adequada das transações e o registro preciso de todos os detalhes envolvidos.

O Código do Banco Central, implementado em C++, constitui o núcleo do sistema distribuído simulado. Funcionando como a autoridade central na regulação das transferências, o Banco Central coordena as interações entre clientes e instituições financeiras. Ao receber solicitações de transferência externa, utiliza `threads` para simular a execução concorrente das instituições financeiras, proporcionando uma atmosfera realista ao ambiente simulado. A comunicação com as instituições financeiras ocorre via HTTP, utilizando a biblioteca `cpp-httpplib`. Este componente é crucial para verificar a capacidade do sistema em processar solicitações, coordenar transferências simultâneas e responder eficazmente a exceções.

A implementação destes códigos reflete a interconexão e interdependência necessárias para simular, de maneira controlada e realista, as operações de transferência em um ambiente distribuído. Nas seções subsequentes, cada componente será detalhado, abordando suas funcionalidades, implementações específicas e os resultados obtidos nos testes aplicados.

Código Cliente

O código do cliente em Python simula transferências externas para o serviço do Banco. Ele apresenta duas classes principais: `Cliente`, que representa um cliente com ID e saldo, e `ClienteDestino`, que contém informações sobre o destinatário da transferência. A função `realizar_transferencia` é responsável por simular uma transferência externa, enviando uma requisição GET para a URL do serviço do Banco e tratando a resposta para extrair informações sobre o destinatário. A função `gerar_chave_destino` gera uma chave de destino aleatória. A função principal `main` executa loops simulando transferências para diferentes instituições e clientes, utilizando a função de transferência. O código é modular, utilizando bibliotecas como `requests`, `random` e `string` para manipulação de dados e comunicação remota. Em resumo, o código é uma implementação simples para simular transferências externas, oferecendo flexibilidade na escolha de clientes, valores e destinatários durante a simulação.

Ambiente de Testes:

- Linguagem de Programação: Python
- Biblioteca Utilizada: Requests

Objetivo dos Testes:

- Confirmar se o cliente pode realizar transferências externas.
- Verificar como o cliente reage a situações específicas durante as transferências.

Estratégia de Teste:

- Simulação de transferências para diferentes instituições e clientes.
- Monitoramento das respostas para garantir que as transferências ocorram conforme o esperado.
- Testes para situações em que o servidor retorna códigos de erro.
- Manipulação de exceções durante a transferência.
- Transferências Bem-Sucedidas:
- Todos os casos de transferência foram concluídos sem erros, e o cliente exibiu corretamente os detalhes da transferência.

Resultados:

- Todas as transferências foram bem-sucedidas.
- Os detalhes das transferências foram apresentados corretamente.

Respostas a Erros do Servidor:

- Testes realizados com o servidor retornando códigos de erro (como 404, 500). Verificação da capacidade do cliente de lidar com essas situações e fornecer mensagens informativas.

- O cliente tratou adequadamente os códigos de erro.
- Mensagens informativas foram exibidas em caso de falha.

Tratamento de Exceções:

- Testes realizados com falhas simuladas durante a transferência. Verificação de como o cliente responde a exceções e exibe mensagens de erro.
- O cliente respondeu corretamente a exceções simuladas.
- Mensagens de erro compreensíveis foram exibidas.

Conclusões:

- O código do cliente demonstrou ser eficaz na realização de transferências simuladas. O tratamento de erros e exceções foi implementado de maneira adequada.

Código Banco:

O código do Banco em Java implementa um serviço de transferência externa entre instituições financeiras. A classe principal é Banco, que possui métodos para inicializar clientes, realizar transferências e obter informações sobre o destinatário. A classe Cliente representa um cliente com ID e saldo, e a classe ClienteDestino contém informações sobre o destinatário da transferência.

O serviço é iniciado na porta 4567, utilizando o framework Spark para lidar com as requisições HTTP. A rota /FazerTransferencia é mapeada para a realização de transferências, onde são obtidos parâmetros da requisição, como clienteld, valorTransferencia e chaveDestino. A instituição financeira de destino é escolhida com base no clienteld, e a transferência é realizada, descontando o valor do saldo do cliente de origem.

O método obterClienteDestino faz uma chamada HTTP para o serviço do Banco Central (<http://localhost:5288/Bacen>) para obter informações sobre o destinatário com base na chave de destino. A resposta é processada para extrair dados como o nome da instituição e o número da conta. Em caso de sucesso, é retornado um objeto ClienteDestino.

Ambiente de Testes:

- Linguagem de Programação: Java
- Framework Utilizado: Spark

Objetivo dos Testes:

- Confirmar se o banco pode processar solicitações de transferência externa corretamente.
- Verificar a capacidade do banco de lidar com situações específicas, como chamadas de API externa e exceções.

Estratégia de Teste:

- Simulação de solicitações de transferência externa para clientes de diferentes instituições financeiras.
- Verificação da capacidade do banco de obter informações do cliente de destino por meio de chamadas à API externa.
- Testes para situações em que ocorrem exceções durante o processamento da transferência.
- Transferências Bem-Sucedidas:
- Todos os casos de transferência foram concluídos sem erros, e o banco exibiu corretamente os detalhes da transferência.

Resultados:

- Todas as transferências foram bem-sucedidas.

- Os detalhes das transferências foram registrados corretamente.

Obtenção de Informações do Cliente de Destino:

- Testes realizados para garantir que o banco seja capaz de obter informações do cliente de destino por meio de chamadas à API externa.
- O banco conseguiu obter informações do cliente de destino para todas as transferências.
- As informações foram corretamente processadas e utilizadas durante a transferência.

Tratamento de Exceções:

- Testes realizados com falhas simuladas durante o processamento da transferência. Verificação de como o banco responde a exceções e exibe mensagens de erro.
- O banco respondeu adequadamente a exceções simuladas.
- Mensagens de erro compreensíveis foram exibidas em caso de falha.

Conclusões:

- O código do banco demonstrou ser eficaz no processamento de transferências externas simuladas. A capacidade de obter informações do cliente de destino por meio da chamada à API externa foi verificada.

Código Banco Central:

O código do Banco Central em C++ implementa um servidor para lidar com solicitações de transferência externa entre diferentes instituições financeiras. O serviço é iniciado na porta 8080, utilizando a biblioteca `httpplib` para criar um servidor HTTP. A classe principal é `BancoCentral`, que possui um mapa de instituições financeiras representado pela classe `InstituicaoFinanceira`.

A classe `InstituicaoFinanceira` contém informações sobre uma instituição, como porta e nome. Cada instituição é inicializada no método `iniciarServico` do `Banco Central`. Quando uma solicitação de transferência externa é recebida na rota `/transferenciaExterna`, o servidor extrai os detalhes da transferência do corpo da requisição e identifica a instituição de destino com base na chave da instituição. Em seguida, é chamado o método `fazerTransferencia` da instituição financeira correspondente.

A instituição financeira realiza a transferência interna, e o resultado é retornado como uma resposta HTTP JSON contendo o status e a mensagem. Caso a instituição de destino não seja encontrada, é retornada uma resposta de erro.

Além disso, o código simula a execução de processos em threads separadas para cada instituição financeira. Isso pode incluir lógicas específicas de cada instituição. O programa principal também aguarda a conclusão de todas as threads antes de encerrar.

Em resumo, o código do Banco Central implementa um servidor para processar solicitações de transferência externa entre instituições financeiras, com suporte a múltiplas instituições e execução assíncrona em threads para simular processos distintos. A modularidade e a flexibilidade do código permitem fácil expansão para lidar com mais instituições financeiras.

Ambiente de Testes:

- Linguagem de Programação: C++
- Bibliotecas Utilizadas: `cpp-httpplib`
- Threads: Utilizadas para simular o funcionamento simultâneo de múltiplas instituições financeiras

Objetivo dos Testes:

- Confirmar se o Banco Central pode processar solicitações de transferência externa corretamente.
- Verificar a capacidade do Banco Central de coordenar transferências com instituições financeiras simultaneamente.
- Garantir que o código respeite a concorrência e funcione corretamente quando várias instituições financeiras são simuladas em threads diferentes.

Estratégia de Teste:

- Simulação de solicitações de transferência externa para diferentes instituições financeiras.
- Verificação da coordenação entre o Banco Central e as instituições financeiras simuladas em threads.
- Testes para situações em que ocorrem exceções durante o processamento da transferência.
- Avaliação da capacidade do código de funcionar corretamente em um ambiente concorrente.

Transferências Bem-Sucedidas:

- Todos os casos de transferência foram concluídos sem erros, e o Banco Central coordenou efetivamente a transferência com as instituições financeiras simuladas.

Resultados:

- Todas as transferências foram bem-sucedidas.
- O Banco Central coordenou corretamente a execução simultânea de transferências com múltiplas instituições financeiras.

Tratamento de Exceções:

- Testes realizados com falhas simuladas durante o processamento da transferência. Verificação de como o Banco Central responde a exceções e exibe mensagens de erro.
- O Banco Central respondeu adequadamente a exceções simuladas.
- Mensagens de erro compreensíveis foram exibidas em caso de falha.

Concorrência e Threads:

- Testes realizados para garantir que o Banco Central funcione corretamente quando múltiplas instituições financeiras são simuladas em threads separadas.
- O Banco Central demonstrou funcionar corretamente em um ambiente concorrente.
- As instituições financeiras foram simuladas em threads separadas, e a coordenação ocorreu sem problema.

Conclusão

A conclusão deste projeto se mostrou coerente para o sistema distribuído, evidenciando as interações complexas entre os códigos do Cliente, do Banco e do Banco Central. A jornada pelos códigos permitiu explorar diversos aspectos, desde a origem das transações até o núcleo regulador do Banco Central.

O Código do Cliente revelou-se como o ponto de partida para as transferências, cumprindo eficientemente seu papel na geração de solicitações simuladas. A modularidade e flexibilidade deste código facilitaram a simulação de diversas operações e a interação dinâmica com o sistema distribuído. A capacidade de tratar exceções e fornecer mensagens informativas demonstrou a robustez necessária para lidar com situações diversas durante as transferências.

O Código do Banco, desenvolvido em Java, destacou-se como um intermediário crucial, orquestrando as transferências e realizando chamadas simuladas à API externa do Banco Central. Sua habilidade em coordenar eficazmente as operações entre as instituições financeiras, escolhendo a instituição de destino com base no cliente ID, contribuiu significativamente para a simulação realista. A capacidade de obter informações do cliente de destino por meio da chamada à API externa foi validada com sucesso, demonstrando a eficácia desse componente na execução de suas funções.

O Código do Banco Central, o epicentro regulador, emergiu como a entidade capaz de coordenar as operações simultâneas entre instituições financeiras, estabelecendo assim a essência distribuída do sistema simulado. A implementação de threads para simular processos concorrentes das instituições financeiras e a resposta eficaz a exceções fortaleceram a confiança na capacidade do Banco Central de desempenhar seu papel central de regulamentação.

Num contexto mais amplo, a interconexão harmoniosa entre esses códigos reflete a essência da arquitetura distribuída. A análise dos resultados dos testes confirma a eficácia do sistema em processar transferências simuladas, lidando com situações adversas e mantendo a integridade das operações. A capacidade de coordenar processos concorrentes foi comprovada, destacando a robustez do sistema distribuído simulado.

Em última análise, este projeto proporcionou uma exploração profunda das dinâmicas de um sistema distribuído simulado, destacando tanto os desafios quanto as eficácias inerentes a essa arquitetura. O conhecimento adquirido ao longo dessa jornada contribui para uma compreensão mais sólida dos princípios fundamentais e das complexidades envolvidas na construção e manutenção de sistemas distribuídos na prática.

Avaliação Geral do Sistema

O sistema distribuído simulado, composto pelos códigos do Cliente em Python, do Banco em Java e do Banco Central em C++, representa uma solução robusta e complexa para a simulação de transferências externas entre entidades financeiras. Este ambiente distribuído foi meticulosamente projetado para funcionar em sistemas operacionais distintos, aumentando significativamente a complexidade e desafiando a integração em ambientes heterogêneos.

Ao ser implementado em Python, o Cliente serviu como ponto de origem das transferências, operando em um ambiente operacional diferente do Banco, que foi desenvolvido em Java. Esta diversidade de sistemas operacionais, que por sua vez representa uma condição realista em muitos cenários, adiciona uma camada adicional de complexidade à simulação. Os resultados dos testes indicaram que o Cliente foi capaz de se conectar eficazmente ao Banco, estabelecendo a robustez do sistema em ambientes variados.

O código do Banco, além de coordenar as transferências, teve que lidar com a heterogeneidade dos sistemas operacionais, mantendo sua eficiência em um contexto desafiador. O sucesso na realização das transferências e na obtenção de informações do Cliente de Destino, mesmo em ambientes operacionais diferentes, destaca a capacidade deste componente de superar desafios complexos.

O Banco Central, operando em C++, desempenhou um papel crucial na coordenação e regulamentação das operações simultâneas. O uso de threads para simular instituições financeiras concorrentes, combinado com a execução em sistemas operacionais diversos, contribuiu para uma simulação mais próxima da realidade. O sistema não apenas gerenciou com sucesso a complexidade introduzida por diferentes sistemas operacionais, mas também coordenou transferências em uma rede compartilhada, evidenciando a adaptabilidade e resiliência do código.

Além disso, a simulação foi conduzida em uma rede única, a qual representou um cenário realista de instituições financeiras operando em uma infraestrutura compartilhada. A comunicação eficiente entre os componentes em uma rede comum valida ainda mais a robustez do sistema, proporcionando uma visão mais holística das interações distribuídas.

Em conclusão, o sistema distribuído simulado não apenas abordou a complexidade inerente às transferências externas entre entidades financeiras, mas também enfrentou desafios adicionais, como a diversidade de sistemas operacionais e o compartilhamento de uma rede. A eficácia do sistema em operar em ambientes heterogêneos e coordenar operações distribuídas destaca sua resiliência e versatilidade em cenários do mundo real. Este projeto oferece uma visão aprofundada não apenas sobre a simulação de operações financeiras distribuídas, mas também

sobre a adaptação a ambientes operacionais diversos, reforçando sua aplicabilidade em contextos complexos.