Transações Bancárias Distribuídas

Lara Esquivel[1], **Lucas de Paiva**[1]

[1]Universidade Estadual de Feira de Santana (UEFS)

Resumo. *Este artigo descreve o desenvolvimento de uma aplicação distribuída de transações bancárias, que tem como objetivo oferecer uma solução alternativa ao PIX para países que não possuem um Banco Central. Ao decorrer da leitura, serão descritas as escolhas feitas para a resolução dos problemas encontrados em relação a concorrência e sincronização das transações, bem como os resultados alcançados sobre o produto final desenvolvido.*

1. Introdução

Nos últimos anos, o avanço da tecnologia tem revolucionado a forma como os clientes interagem com os serviços bancários. Em todo o mundo, os dispositivos móveis se tornaram ferramentas indispensáveis para realizar movimentações financeiras de forma rápida e conveniente. No Brasil, a criação do sistema de pagamentos instantâneos PIX e os significativos investimentos dos bancos em aplicativos móveis impulsionaram um aumento impressionante de 75% nas transações financeiras por mobile banking em 2021, em comparação ao ano anterior, de acordo com a Pesquisa Febraban sobre Tecnologia Bancária.

O PIX não apenas promoveu a inclusão de milhões de brasileiros que não possuíam cartões de crédito nas formas eletrônicas de pagamento, mas também simplificou a vida daqueles que tradicionalmente utilizavam métodos como boletos, cartões, dinheiro em espécie e cheques. O Relatório de Economia Bancária do Banco Central do Brasil revelou que, apenas um ano após o lançamento do PIX, mais de 100 milhões de pessoas já haviam realizado pelo menos uma transação utilizando essa ferramenta inovadora. O sucesso notável da tecnologia brasileira despertou o interesse de outros países, que têm observado com atenção a experiência do Brasil na área.

No entanto, alguns países enfrentam um desafio peculiar: a ausência de um banco central para coordenar e regular as transações financeiras. Isso torna a tarefa de desenvolver um sistema semelhante ao PIX mais complexa, uma vez que não podem recorrer aos recursos centralizados normalmente utilizados para controlar as transações. Apesar desse obstáculo, o governo de um desses países está empenhado em oferecer aos seus cidadãos um sistema eficiente que permita a criação de contas bancárias, facilitando pagamentos, depósitos e transferências de valores, mesmo sem a presença de um banco central.

Diante desse desafio, o governo de um país fictício solicitou aos bancos uma solução conjunta para o problema. Como resultado, os bancos se uniram em um consórcio bancário e contrataram uma equipe de especialistas em sistemas distribuídos para propor e implementar essa solução inovadora. A principal exigência para essa solução distribuída é permitir que os clientes, a partir de qualquer banco, possam realizar transações atômicas sobre o dinheiro em contas particulares ou conjuntas, inclusive envolvendo mais de duas

contas. Além disso, é essencial garantir que a comunicação entre os servidores dos bancos seja estabelecida de forma segura, evitando movimentações financeiras além do saldo disponível em cada conta e prevenindo o ”duplo gasto” do dinheiro.

Neste artigo, abordaremos em detalhes o desenvolvimento dessa solução distribuída, adaptada às necessidades do país que não possui um banco central. Exploraremos as possíveis abordagens tecnologias que podem ser implementadas para alcançar esses objetivos.

2. Desenvolvimento

2.1. API do banco

A API de banco foi desenvolvida em Python, uma linguagem amplamente utilizada na área de desenvolvimento de software. A escolha do framework Flask permitiu a criação de uma API robusta, flexível e de fácil implementação. O Flask fornece recursos para criar rotas e manipular solicitações HTTP, tornando-o uma escolha ideal para a construção de serviços web eficientes e escaláveis.

2.1.1. Recursos da API

1. Lista de Transações: A API de banco oferece suporte a uma lista de transações, permitindo o registro e o rastreamento de todas as atividades financeiras realizadas. Essa funcionalidade é essencial para manter um histórico detalhado das transações realizadas pelos clientes e garantir a transparência das operações.
2. Lista de Clientes: Além das transações, a API também possui uma lista de contas, que podem ser particular ou individual. Ela armazena informações importantes sobre cada indivíduo ou dupla que utiliza os serviços bancários. Essa funcionalidade facilita a gestão de clientes e fornece uma base sólida para o gerenciamento de contas, saldos e atividades financeiras relacionadas.
3. Relógio de Lamport: A implementação do relógio de Lamport pela API de banco permite a sincronização e o rastreamento adequados das transações. Esse mecanismo é utilizado para estabelecer uma ordenação parcial dos eventos, permitindo que os bancos mantenham uma visão consistente das transações em todo o sistema. O uso do relógio de Lamport contribui para a integridade dos dados e a coerência das operações.
4. Integração entre Bancos: Uma funcionalidade destacada da API de banco é a capacidade de solicitar transações entre diferentes instituições bancárias. Essa integração é possível graças ao design modular e ao suporte de comunicação entre as APIs dos bancos. Ao permitir que um banco solicite a execução de uma transação a outro banco, a API facilita a interação e a colaboração entre instituições financeiras.

2.1.2. Funcionalidades de Transações Automatizadas

1. Criação de Cliente: A API oferece a funcionalidade de criar clientes, permitindo o registro e a inclusão de novos usuários no sistema bancário. Essa funcionalidade

é essencial para a expansão da base de clientes e a disponibilização dos serviços financeiros a um público mais amplo
2. Consulta de Transação: Os clientes têm a capacidade de consultar suas transações através da API de banco. Essa funcionalidade permite que os usuários acessem informações atualizadas sobre suas atividades financeiras, incluindo saldos, depósitos, transferências e outras transações relevantes.
3. Depósito e Transferência: A API oferece suporte a depósitos e transferências de fundos entre contas. Os clientes podem realizar depósitos em suas próprias contas ou efetuar transferências para outras contas dentro do mesmo banco ou entre diferentes instituições. Essas operações são essenciais para a movimentação de fundos e a realização de transações financeiras.
4. Transações Automatizadas: Uma característica notável da API de banco é a capacidade de realizar transações automáticas periodicamente. Essas transações podem ser configuradas para ocorrerem a cada um ou alguns segundos, de acordo com os requisitos do sistema. Essa funcionalidade automatizada oferece comodidade aos clientes e permite a execução de operações regulares sem intervenção manual.

2.2. Sincronização

Considerando a ausência de um banco central, a sincronização adequada é um elemento crítico para o desenvolvimento de um sistema distribuído eficiente. Nesse contexto, a utilização do Relógio de Lamport é uma abordagem viável para garantir a sincronização entre os diversos componentes do sistema.

O Relógio de Lamport é uma técnica de marcação de eventos em um sistema distribuído que permite ordenar as operações executadas pelos diferentes nós. Ele baseia-se na noção de tempo lógico e é capaz de estabelecer uma ordem parcial entre os eventos, mesmo que não seja possível obter um consenso global sobre a ordem precisa.

Ao adotar o Relógio de Lamport, cada evento de transação que um banco recebe, é marcado com um carimbo de tempo lógico que reflete na contagem de transações acumuladas até o momento. Isso permite que os diferentes nós tenham conhecimento de qual banco tem mais transações pendentes, permitindo que a prioridade seja dada a este banco.

A solução distribuída baseada no Relógio de Lamport envolveria a integração desse mecanismo de sincronização nos sistemas dos bancos participantes. Cada banco do sistema teria seu próprio relógio de Lamport, que seria atualizado à medida que as transações fossem chegando. Os relógios contém um vetor em que cada posição é referente ao contador do relógio cada banco. Por exemplo, o Banco de ID=0 terá o relógio na posição 0 do vetor. Quando uma transação é recebida pelo banco, o relógio de Lamport é utilizado para marcar o evento e estabelecer sua posição temporal. Além disso, cada banco só poderá incrementar o relógio da própria posição do vetor.

A sincronização entre os relógios de Lamport dos diferentes nós pode ser alcançada por meio da troca de mensagens contendo os carimbos de tempo. Nesse problema, por exemplo, um banco A com suas transações pendentes em uma fila solicita aos outros bancos permissão para poder executá-las. Os bancos receptores, por sua vez, recebem o relógio do banco A e fazem uma comparação com seu próprio relógio, com o objetivo de verificar se o banco A tem mais urgência.

Entretanto, a sincronização acontece da seguinte forma: suponha que existam os bancos A, B e C. Quando o banco A faz todo o processo de permissão para conseguir executar suas transações pendentes, e conclui todas, ele vai atualizar o relógio de todos os outros bancos. Entretanto, os outros bancos não atualizarão o relógio do banco A de volta, para não perderem a contagem de suas próprias transações. Dessa forma, a prioridade sempre será dada para o banco com a maior quantidade de transações.

A Figura 1 Exemplifica as etapas do algoritmo para que a fila de transações de um banco possam ocorrer com sucesso:

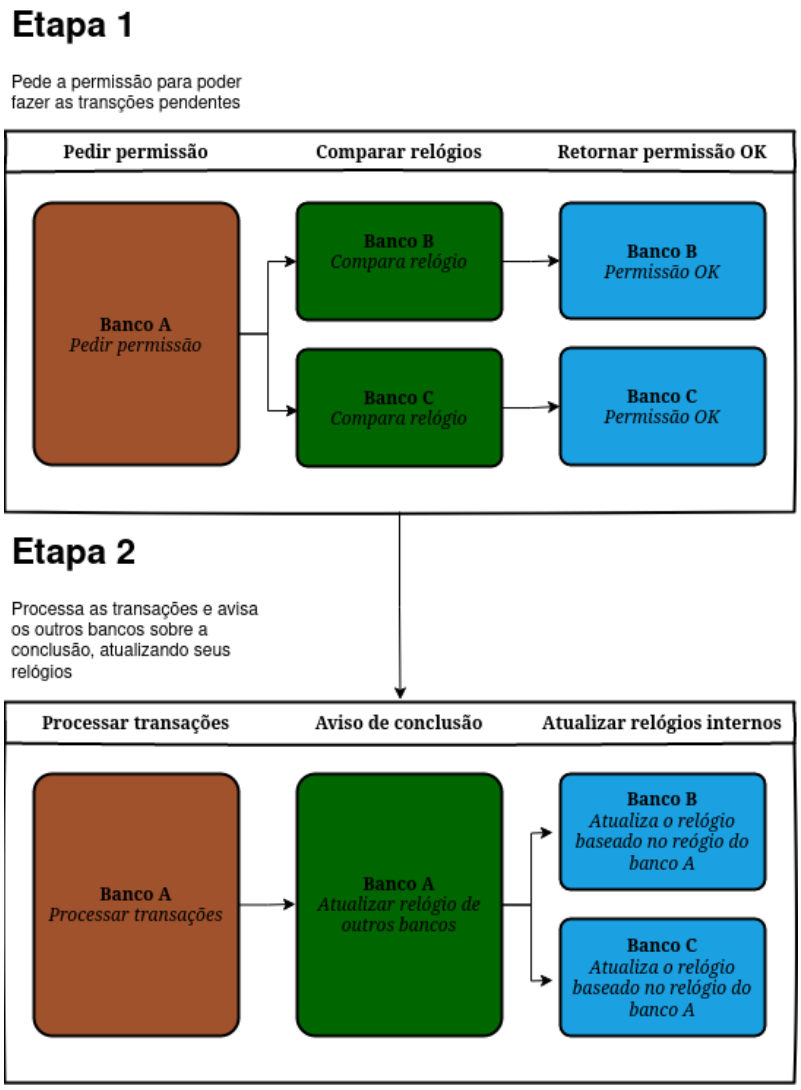

Figure 1. Sequência do algoritmo utilizando o relógio

Caso um dos bancos não não autorizem a permissão para o Banco A executar as transações, significa que aquele banco tem uma fila de transações maior no momento e tem prioridade.

Supondo que num primeiro momento, tenhamos a seguinte distribuição dos relógios dos bancos:

	Relógio		
	v[0]	v[1]	v[2]
A	5	3	1
B	2	4	1
C	2	3	1

Table 1. Os relógios desde a última sincronização estão dessa maneira. O banco A irá pedir permissão para executar sua fila de transações.

	Relógio		
	v[0]	v[1]	v[2]
A	5	3	1
B	2	4	1
C	2	3	1

Table 2. O banco A vai enviar seu relógio para o Banco B fazer a comparação com seu relógio. A comparação é feita da seguinte maneira: nos números em azul, será subtraído 5-2=3, pois A tem a posição v[0] já atualizada em relação a mesma posição no Banco B o que significa que A tem 3 transações na fila. Já nos números em vermelho, a subtração será 4-3=1, pois B tem a posição v[1] já atualizada em relação a mesma posição no Banco A o que significa que o Banco B tem 1 transação apenas na fila. Logo, o banco B retornará uma confirmação para o Banco A poder fazer a transação.

	Relógio		
	v[0]	v[1]	v[2]
A	5	3	1
B	2	4	1
C	2	3	1

Table 3. O banco A vai enviar seu relógio para o Banco C fazer a comparação com seu relógio. A comparação é feita da seguinte maneira: nos números em azul, será subtraído 5-2=3, pois A tem a posição v[0] já atualizada em relação a mesma posição no Banco C o que significa que A tem 3 transações na fila. Já nos números em verde, a subtração será 1-1=0, pois C tem a posição v[2] já atualizada em relação a mesma posição no Banco A, o que significa que o Banco C não tem nenhuma transação na fila. Logo, o banco C retornará uma confirmação para o Banco A poder fazer a transação.

	Relógio		
	v[0]	v[1]	v[2]
A	5	3	1
B	5	4	1
C	5	3	1

Table 4. Quando o Banco A concluir todas as suas transações, ele enviará o seu relógio para os outros dois bancos atualizarem. Como um banco só pode atualizar apenas a posição correspondente ao seu ID, apenas as posições v[0] dos outros bancos serão atualizadas, que estão destacadas em azul.

3. Conclusão

A API de banco desenvolvida em Python com o framework Flask demonstra uma solução versátil e eficiente para gerenciar transações, clientes e integração entre diferentes instituições financeiras. Com funcionalidades como criação de clientes, consulta de transações, depósitos, transferências e transações automatizadas, a API oferece uma base sólida para a construção de sistemas bancários confiáveis e escaláveis. Sua implementação própria do relógio de Lamport contribui para a consistência das operações e a integridade dos dados, garantindo uma experiência financeira segura e eficaz.

Além das funcionalidades abordadas, é importante destacar a importância da concorrência e sincronização proporcionadas pelo relógio de Lamport para o sistema bancário. A implementação desse mecanismo de sincronização garante que as transações sejam ordenadas corretamente, evitando conflitos e inconsistências nos registros financeiros. Essa sincronização é fundamental para garantir a integridade e a confiabilidade do sistema como um todo.

Outro aspecto relevante é a necessidade de implementação de um front-end para permitir a interação dos usuários com o sistema bancário. Embora a API forneça a base para o gerenciamento das transações e clientes, um front-end amigável e intuitivo facilitará a utilização dos serviços pelos clientes, fornecendo uma interface intuitiva para realizar operações financeiras de forma eficiente.

Para uma evolução futura do sistema, é possível considerar a implementação de integração com um banco de dados para armazenamento seguro e escalável das informações financeiras. Além disso, a substituição do relógio de Lamport por outras formas avançadas de sincronização pode ser explorada para aprimorar ainda mais o desempenho e a confiabilidade do sistema, levando em conta os avanços tecnológicos e as necessidades específicas do ambiente bancário.

Por fim, é fundamental considerar a implementação de medidas de segurança robustas para proteger o sistema bancário como um todo. Isso pode envolver a criptografia dos dados, a autenticação dos usuários, a detecção de atividades suspeitas e outras medidas de segurança para garantir a privacidade e a proteção das informações financeiras dos clientes. A segurança é um aspecto crítico para o sucesso e a confiança do sistema bancário, e sua implementação adequada deve ser uma prioridade em todas as etapas do desenvolvimento e da evolução do sistema.

4. References

[Tanembaum and Steen 2016], [Coulouris 2012]

References

Coulouris, G. F. (2012). *Distributed systems : concepts and design*. Boston:Addison-Wesley, 3rd edition.

Tanembaum, A. S. and Steen, M. V. (2016). *Distributed systems : principles and paradigms*. Maarten Van Steen, 2nd edition.