

# Banco de dados replicados na edge, uma abordagem híbrida

## Grupo 4

Ben Hur Faria Reis

Filipe Silveira Chaves

Thiago Monteles de Sousa

Felipe Aguiar Costa

João Paulo Rocha

## Introdução

Em sistemas distribuídos, a consistência é um elemento-chave para garantir que as réplicas de dados em diferentes nós estejam sincronizadas e apresentem uma visão coerente do estado global. A replicação de bancos de dados oferece maior disponibilidade e escalabilidade, mas pode introduzir desafios relacionados à consistência dos dados. Duas abordagens principais são adotadas para lidar com esse problema: consistência forte e consistência eventual. A consistência forte assegura que todas as leituras refletem imediatamente as últimas operações de escrita, enquanto a consistência eventual permite que as réplicas se tornem consistentes em algum momento futuro. A escolha entre essas abordagens depende das necessidades específicas do sistema e dos requisitos do usuário.

Diante desse contexto, o presente projeto tem como objetivo criar um banco de dados replicado que ofereça a possibilidade de escolha entre a consistência forte e a consistência eventual, proporcionando maior flexibilidade e adaptabilidade ao sistema. Através da aplicação de princípios de sistemas distribuídos, fundamentos de arquiteturas e paradigmas de comunicação, pretende-se alcançar uma solução eficiente e confiável para a replicação de dados. Além disso, o projeto explorará o uso de técnicas como os Conflict-free Replicated Data Types (CRDT) para garantir a consistência eventual e o algoritmo Raft para assegurar a consistência forte. Dessa forma, busca-se proporcionar aos usuários a capacidade de selecionar a abordagem mais adequada às suas necessidades e ao contexto de uso do banco de dados replicado.

## Revisão bibliográfica

Diversas técnicas de consistência são capazes de serem utilizadas aplicando conceitos de consistência forte ou eventual. Para isso, foi realizada uma busca por referências que possibilitem a fundamentação teórica desses conceitos e auxiliem na construção final do projeto.

Um dos artigos selecionados é "[Conflict-free Replicated Data Types: An Overview](#)" de Nuno Preguiça (2018). Ele introduz a abordagem da consistência eventual, permitindo divergência temporária de estados entre os nós, mas disponibilizando mecanismos de convergência determinística. O artigo apresenta conceitos de sincronização de estados de dados com consistência eventual e diversas semânticas de concorrência utilizando CRDTs, como registradores, contadores, conjuntos, listas, mapas e modelos de sincronização baseados em estados ou operações.

Outro artigo relevante é "[DSON: JSON CRDT Using Delta-Mutations For Document Stores](#)" de Rik Rinberg et al. (2022). Ele propõe uma nova abordagem para resolver o problema de sincronização de dados em sistemas de armazenamento de documentos distribuídos, utilizando CRDTs em um formato baseado em JSON chamado DSON.

Além disso, o trabalho intitulado "[Consistency Models of NoSQL Databases](#)" de Miguel Diogo (2019) analisa e compara a implementação do modelo de consistência em cinco bancos de dados NoSQL, tanto eventual quanto forte. Através dessas comparações, o autor define os melhores métodos e abordagens para a implementação de cada tipo de consistência, identificando vantagens e desvantagens em diferentes bancos de dados.

Outro artigo relevante é "[Strongly consistent replication for a bargain](#)" de Krikellas (2022). Nesse estudo, o autor analisa e compara duas técnicas escaláveis que buscam garantir segurança e desempenho para sistemas de banco de dados utilizando consistência forte. Mais uma vez, são apresentados métodos e abordagens para a implementação desse tipo de consistência em diferentes contextos.

Por fim, a revisão bibliográfica é finalizada com o artigo "[Consenso em Sistemas Distribuídos: RAFT vs CRDTs](#)" de Donald Hamnes (2019). Nessa análise comparativa, o autor examina os dois protocolos de consenso mais utilizados em consistência eventual (CRDTs) e forte (RAFT). O estudo demonstra conceitos teóricos sobre esses protocolos, possíveis implementações e realiza simulações para garantir a usabilidade e eficácia dos mesmos.

Com isso foi possível obter as informações necessárias para a escolha dos protocolos de consenso eventual e forte que serão utilizados neste trabalho.

## Referencial Teórico

### 1. Fundamentos de Sistemas Distribuídos relacionados ao Projeto (background):

Nesta seção, serão descritos os princípios de sistemas distribuídos que serão aplicados no projeto, bem como os fundamentos das arquiteturas e estilos arquiteturais relevantes para a replicação do banco de dados.

Descrição dos princípios de sistemas distribuídos: Serão explorados os princípios de replicação e tolerância a falhas, fundamentais para a criação de um banco de dados replicado.

### **Descrição dos fundamentos de arquiteturas de sistemas distribuídos e dos estilos arquiteturais:**

Serão aplicados dois estilos arquiteturais - cliente-servidor e peer-to-peer - para a replicação do banco de dados. Onde no modelo cliente-servidor, os dispositivos (clientes) solicitam serviços ou recursos a um servidor central. O servidor é responsável por fornecer as informações ou funcionalidades solicitadas pelos clientes. Já no modelo peer-to-peer, os dispositivos se comunicam diretamente entre si, formando uma rede descentralizada. Cada dispositivo conectado à rede, chamado de "nó", pode atuar tanto como cliente quanto como servidor, permitindo compartilhamento de recursos e troca de informações entre os pares.

Considerando esses conceitos, no projeto em questão as chamadas de serviços externos aos nós do cluster do banco de dados seguirão o esquema cliente-servidor, enquanto a comunicação interna de replicação dos dados no cluster será no estilo peer-to-peer, sem a necessidade de um ponto central no cluster.

### **Descrição dos fundamentos de paradigmas de comunicação em sistemas distribuídos:**

Em protocolos de comunicação alguns dos mais utilizados é o Remote Procedure Call (RPC), que é definido como um paradigma de comunicação utilizado em sistemas distribuídos para permitir que processos em diferentes dispositivos se comuniquem e executem procedimentos em outros dispositivos remotos. Em vez de lidar diretamente com detalhes de comunicação de rede, o RPC permite chamadas de procedimentos em dispositivos remotos da mesma forma que chamadas locais, tornando a comunicação entre processos transparente e facilitando a interação entre diferentes partes de um sistema distribuído. Já o protocolo HTTP facilita a comunicação externa entre os clientes e os nós do cluster, onde pode ser utilizado um esquema request-reply do HTTP que permite os clientes realizem solicitações (requests) aos nós do cluster e recebam respostas (replies) contendo os dados solicitados. Isso proporciona uma interface padronizada para a manipulação externa dos dados armazenados no banco de dados replicado.

Seguindo esses paradigmas o projeto em questão usará como comunicação interna entre os nós do cluster será realizada utilizando o paradigma de comunicação RPC (Remote Procedure Call). Já a comunicação externa entre os clientes e os nós do cluster utilizará o protocolo HTTP com o esquema request-reply, permitindo a manipulação externa dos dados com tratamento transparente de replicação e consistência.

## **2. Descrição sobre robustez em sistemas distribuídos:**

Nesta parte do referencial teórico, serão apresentados conceitos relacionados à garantia de robustez no contexto de sistemas distribuídos os tipos de consistência bem como seus conceitos.

A consistência em um sistema distribuído refere-se à propriedade de que os dados armazenados em diferentes nós do sistema estejam sempre de acordo entre si. Em uma consistência forte um nível mais rigoroso de garantia de consistência é requisitado. Isso significa que, em qualquer momento, todos os nós do sistema veem exatamente os mesmos dados, independentemente de qual nó eles estão acessando. Para alcançar a consistência forte, os sistemas geralmente utilizam mecanismos de bloqueio e sincronização rigorosos, o que pode resultar em um desempenho mais lento, especialmente em cenários de alta concorrência.

Uma das técnicas mais comuns para garantir a consistência forte é o uso de algoritmos de consenso, como o algoritmo Raft. Nesses algoritmos, os nós coordenam-se para determinar um único líder que é responsável por coordenar e aplicar todas as operações no sistema. As operações são replicadas em todos os nós e só são confirmadas após o líder garantir que a maioria dos nós as tenha processado com sucesso. Essa abordagem oferece alta consistência, mas pode ser mais suscetível a problemas de disponibilidade se o líder falhar ou se a rede entre os nós estiver instável.

Uma outra modalidade de consistência é o modelo eventual, a qual é um método menos rigoroso de consistência em sistemas distribuídos. Nesse caso, os nós podem ter visões diferentes dos dados em um determinado momento, mas, com o tempo, os dados convergem para um estado consistente em todos os nós. Isso significa que, após um período de tempo suficiente e sem novas alterações, todos os nós eventualmente terão as mesmas informações.

Para alcançar a consistência eventual, são utilizados mecanismos de replicação que permitem que os nós operem de forma assíncrona e independente. Uma abordagem comum é a utilização de Conflict-free Replicated Data Types (CRDTs). Os CRDTs são projetados para garantir que as operações de atualização dos dados sejam comutativas e associativas, o que permite que os nós combinem as alterações de forma independente, sem gerar conflitos. À medida que as atualizações se propagam entre os nós, eles eventualmente convergem para um estado consistente.

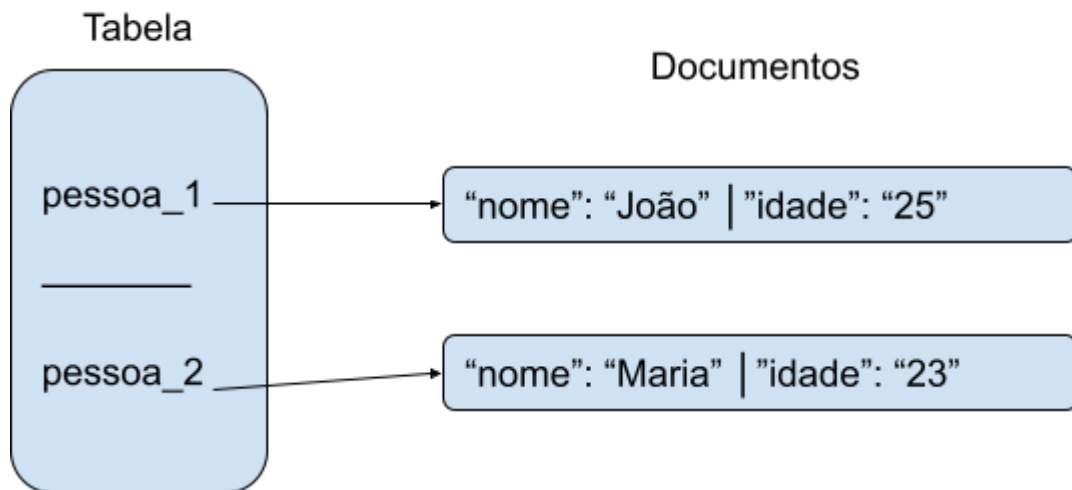
A consistência eventual é especialmente útil em sistemas que valorizam a disponibilidade e a tolerância a falhas, uma vez que os nós podem continuar a operar mesmo que a comunicação entre eles seja interrompida temporariamente. No entanto, pode haver uma janela de tempo em que os dados estão inconsistentes, o que pode ser aceitável em alguns cenários de aplicação, como em redes sociais ou sistemas de colaboração, onde pequenas discrepâncias temporárias não são críticas.

## **Metodologia:**

Nesta seção, será descrito como o projeto será implementado e quais etapas serão seguidas para atingir os objetivos propostos. A metodologia abrangerá a criação das tabelas no banco de dados, a especificação da consistência (forte ou eventual) para cada tabela, as operações de deleção, inserção, modificação e obtenção de registros nas tabelas, bem

como a utilização dos princípios, arquiteturas e paradigmas de comunicação descritos no referencial teórico.

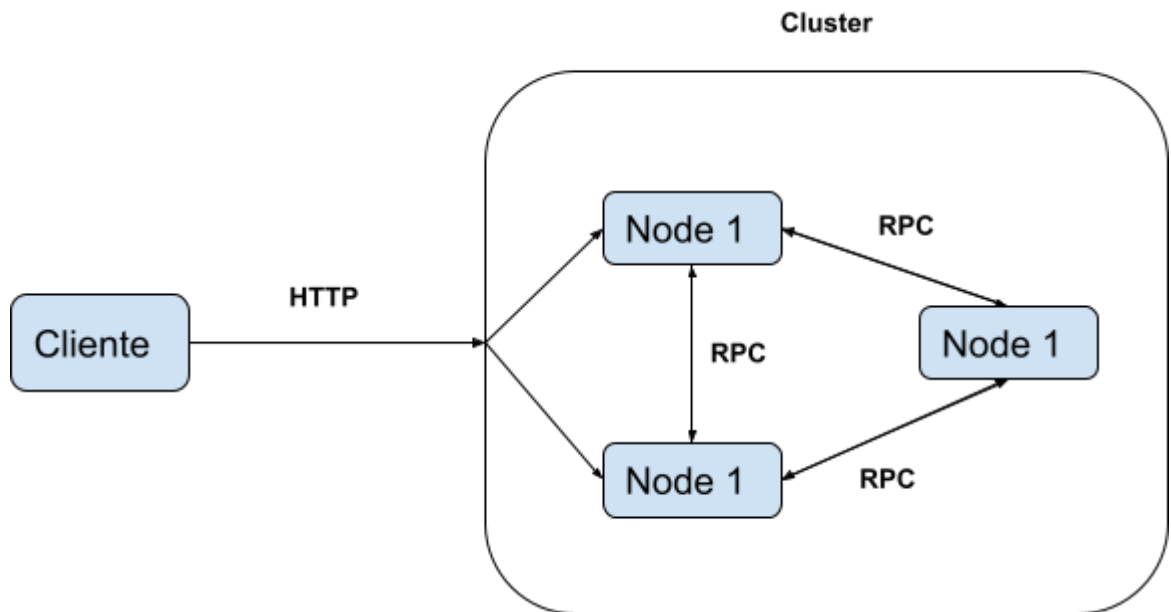
### Design dos dados



Para o design dos dados será utilizado tabelas no banco de dados, onde cada tabela é uma coleção de documentos, cada documento tem um ID do tipo string, cada tabela pode ser configurada para ter ou consistência forte (RAFT) ou eventual (CRDT). Além disso, os documentos irão guarda um mapa chave-valor, as chaves e os valores strings, as chaves podem ser arbitrárias, não tendo um esquema fixo para os dados.

### Implementação das operações

Serão implementadas as operações de deleção, inserção, modificação e obtenção de registros nas tabelas, de acordo com a consistência especificada para cada uma. Os princípios de replicação e tolerância a falhas serão aplicados para garantir a efetividade da replicação do banco de dados.



Como demonstrado na figura acima, serão utilizados os estilos arquiteturais cliente-servidor e peer-to-peer, onde as chamadas de serviços externos e a comunicação interna de replicação seguirão os estilos cliente-servidor e peer-to-peer, respectivamente. Utilização do paradigma de comunicação RPC e protocolo HTTP, tendo a comunicação entre os nós do cluster e os clientes será feita utilizando RPC e o protocolo HTTP com esquema request-reply.

A comunicação HTTP terá como possíveis interações:

**PUT /<table>**

Cria uma tabela (indica no corpo se ela vai ter consistência forte ou eventual)

**DELETE /<table>**

Cria uma tabela (indica no corpo se ela vai ter consistência forte ou eventual)

**GET /<table>/<doc id>**

Retorna o mapa de chaves e valores de um documento

**PUT /<table>/<doc id> (body: keys/values)**

Substitui o documento inteiro, apagando outras chaves não especificadas nesta chamada

**PATCH /<table>/<doc id> (body: keys/values)**

Substitui valores dentro de um documento, mantendo chaves já existentes que não foram especificadas nesta chamada

**DELETE /<table>/<doc id>**

Deleta um documento

Por fim será realizado a Integração dos conceitos de CRDT e algoritmo Raft para garantir a consistência eventual e a consistência forte, respectivamente, os conceitos de CRDT e o algoritmo Raft serão integrados à implementação do projeto.

### Situação atual do projeto (17/07/2023)

	Maio	Junho	Julho	Agosto
<b>Revisão bibliográfica</b>	X	X		
<b>Apresentação interno de métodos de consistência</b>	X	X		
<b>Codificação</b>		X	X	X

X concluído

X em processo

Para concluir o projeto teremos as seguintes etapas:

1. Revisão bibliográfica: (TODOS)
2. Elaboração de apresentações internas sobre métodos de consistência (TODOS)
3. Decisão sobre os métodos de consistência (TODOS)
4. Elaboração de um esquema arquitetural de comunicação e dados. (TODOS)
5. Codificação de um banco de dados replicado (Felipe Aguiar, Ben Hur)
6. Codificação do protocolo CRDT (Felipe Aguiar, Ben Hur)
7. Codificação do protocolo RAFT (Felipe Chaves, João Paulo)
8. Codificação de um visualizador dos estados dos nós (Thiago, Ben Hur)
9. Codificação do agregador final dos protocolos e banco de dados replicado (Thiago)

<b>Status</b>	<b>Etapas</b>	<b>observação</b>
<b>Concluída</b>	1 ao 4	
<b>Em andamento (com algum código já feito)</b>	6,7,8	todos possuem já algum nível de implementação
<b>Em breve</b>	9	Etapa final