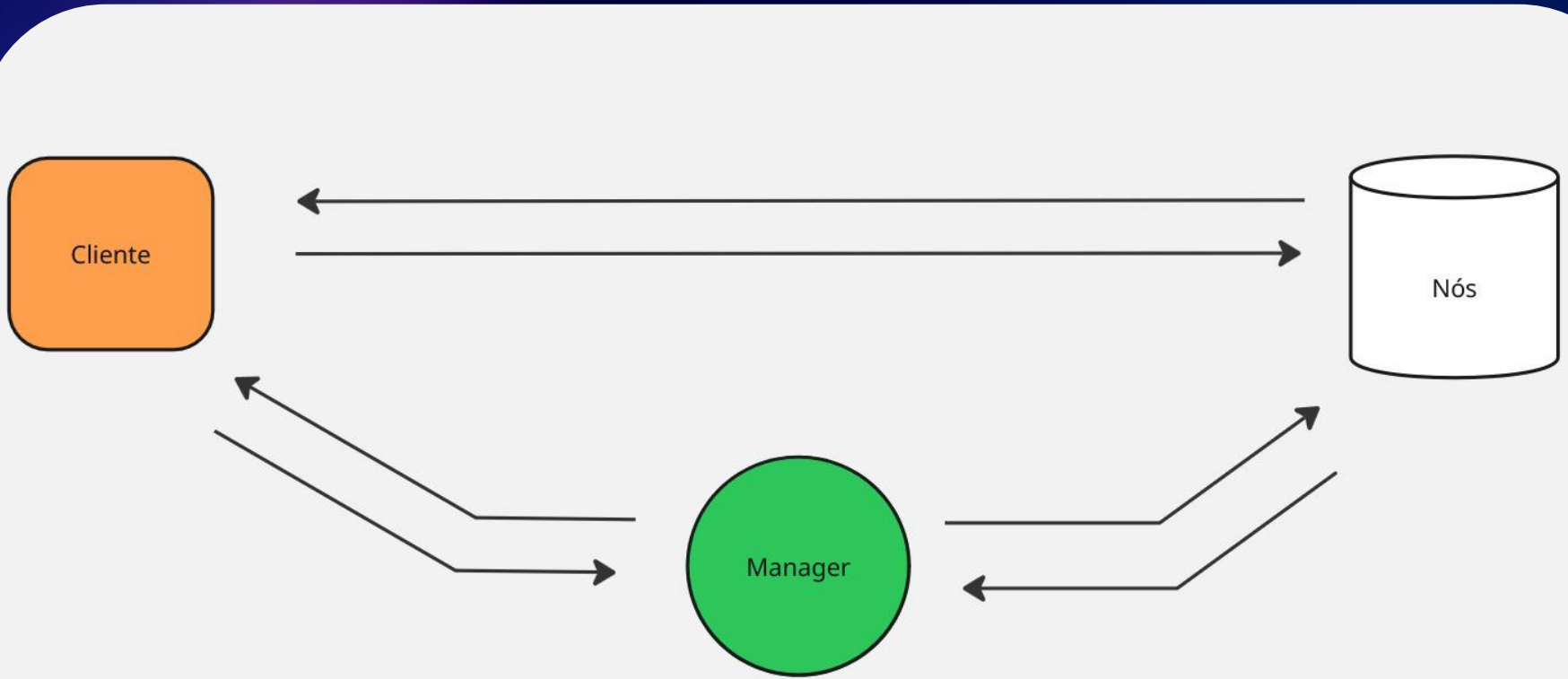
Big File System

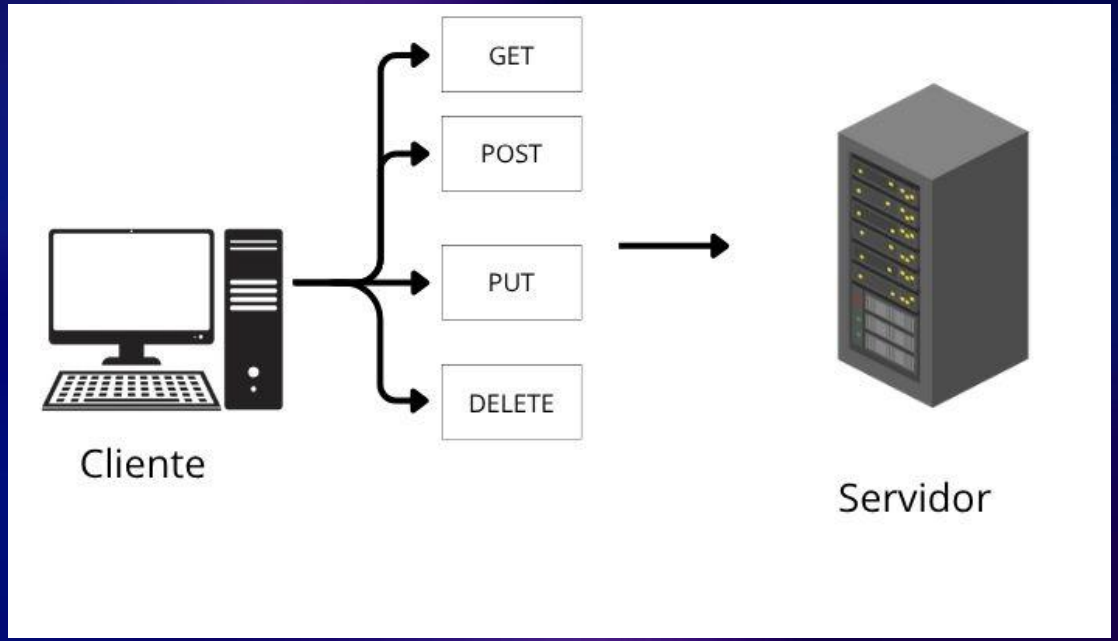
Arquitetura



Protocolos

- **Flask:** APIs REST para comunicação entre cliente, manager e nós.
- **Pika:** Integração com RabbitMQ para mensagens assíncronas.
- **Requests:** Comunicação HTTP simples entre os componentes.
- **RabbitMQ:** Middleware de mensagens que garante alta disponibilidade e coordenação eficiente.Garante a parte assíncrona do sistema.
- **Serialização:**Json.

Rotas REST



GET /list → Lista os arquivos disponíveis.

POST /upload_request → O **cliente pede ao manager** uma lista de nós disponíveis para **enviar (fazer upload) um novo arquivo.**

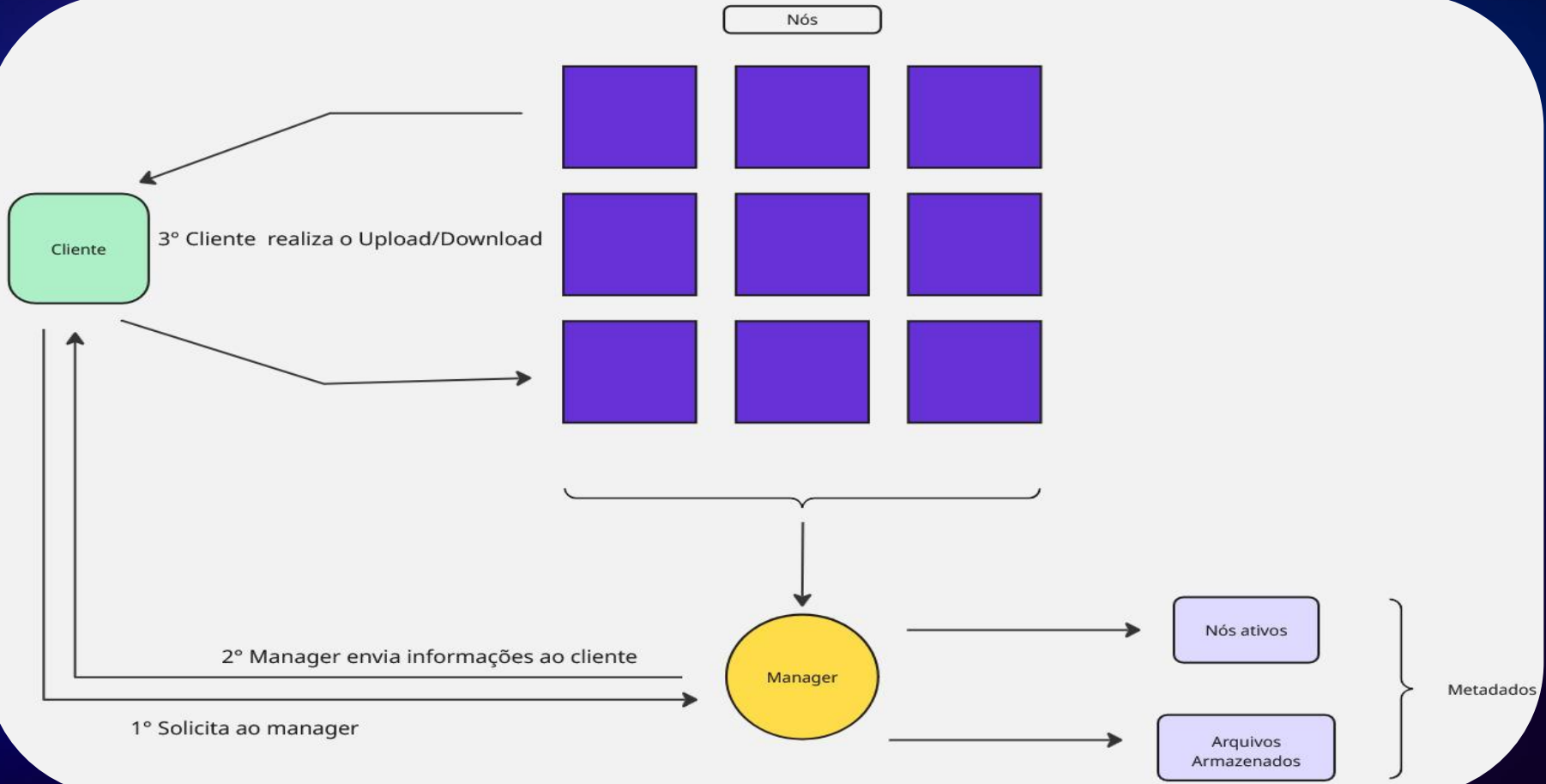
GET /download_location/<filename> → O **cliente pede ao manager** a localização (URL dos nós) de **onde pode baixar (fazer download) um arquivo específico.**

DELETE /remove/<filename> → Cliente solicita a remoção de um arquivo.

Requisitos Funcionais

1. Listar
   Comando: ls
2. Remover
   Comando: rm <exemplo.png>
3. Upload
   Comando: cp C:\Users\user\Downloads\exemplo.png remote:
4. Download
   Comando: cp remote:exemplo.png C:\Users\user\Downloads\exemplo.png

Fluxo


Nós
Cliente
3° Cliente realiza o Upload/Download
2° Manager envia informações ao cliente
1° Solicita ao manager
Manager
Nós ativos
Arquivos Armazenados
Metadados

Client

- Transfere os dados diretamente com os nós.
- Divide os arquivos em chunks.
- Remonta os chunks após o download.

Manager

- Armazena os metadados e informações sobre os nós
- Verifica os nós ativos
- Recria réplicas perdidas em outros nós ativos

Nós

- Armazena os arquivos
- Envia para o Manager quando o arquivo foi salvo

Requisitos Não Funcionais

Replicação

- **Fator de Réplica Configuável:**
  O usuario pode definir o fator de réplica.
  Padrão: 2 cópias por arquivo.
- **Replicação assincrona:**
  A replicação ocorre em segundo plano, sem interferir no upload e download.

Data Sharding

- **Divisão em Chunks:**
  Arquivos grandes são automaticamente divididos em blocos (chunks) de tamanho configurável (ex.: 4MB).
- **Indice centralizado:**
  A localização de cada chunk e de suas réplicas é mantida no Manager através de um índice de metadados.
- **Cabeçalho:**
  Cada chunk tem um cabeçalho com informações sobre o chunk como: Indice do Chunk, nome do arquivo e total de chunks.

Tolerância a falhas

- **Monitoramento via Heartbeats:**
  O manager recebe sinais periódicos (heartbeats) dos nós a cada 5 segundos.
- **Timeout:**
  Se um nó não enviar nenhum sinal em 15 segundos ele é considerado inativo.
- **Redirecionamento automático:**
  No caso da falha de um nó , o manager redireciona as operações para suas réplicas, e mantém o fator de réplica 2 com outro nó ativo.
- **Monitoramento:**
  O sistema monitora continuamente os nós ativos e garante que o fator de réplica seja preservado, criando novas réplicas sempre que necessário.

Escalabilidade

- **Escalabilidade horizontal:**
  Adição de novos nós aumenta a capacidade total do sistema (armazenamento + desempenho).

- **Balanceamento de carga:**
  O Manager distribui arquivos (chunks) entre os nós disponíveis, balanceando a carga.

- **Transferência direta Cliente ↔ Nós:**
  Uploads e downloads não passam pelo Manager, evitando gargalos e permitindo maior paralelismo.

- **Alta vazão de leitura/escrita:**
  Leitura e escrita simultânea em diferentes chunks melhora o desempenho com o crescimento da rede.

- **Adição dinâmica de nós:**
  O sistema pode reconhecer novos nós e utilizá-los automaticamente para novos uploads e réplicas.

Pontos negativos

- **Manager centralizado**
  A arquitetura atual centraliza o manager e pode se tornar um gargalo com muitos clientes conectados ou alto volume de operações simultâneas.
  - Possível solução:
    - Replicação do manager: Aumentaria a robustez do sistema, criando múltiplas instâncias do manager.
- **Sem controle de concorrencia**
  Vários clientes podem escrever no mesmo arquivo.
  - Possível solução:
    - Implementar um sistema de locks e controle de versão.
- **Limitação do flask**
  O servidor padrão não é eficiente sob alta carga, podendo gerar gargalos.
- Possível solução:
  - Substituir o flask pelo Gunicorn por exemplo.
- **Falta de autenticação**
  O sistema atual não tem autenticação para upload,download e remoção de arquivos.
- **Metadados temporarios**
  Atualmente ele armazena os metadados na memória, gerando perda das informações sobre eles ao reiniciar o manager.

Obrigado