2. Especificaciones de Comunicación

Para la comunicación entre los nodos, se utilizará **gRPC** debido a su alta eficiencia y soporte para la transmisión de datos en sistemas distribuidos. gRPC es un marco de comunicación basado en **Protocol Buffers** que permite definir interfaces mediante un archivo .proto y generar automáticamente el código para cliente y servidor.

1. **Cliente (Proceso 1):**
    - Este nodo será una aplicación que enviará solicitudes de lectura y escritura.
    - Las solicitudes de **escritura** siempre se envían al líder del clúster, mientras que las solicitudes de **lectura** pueden ser atendidas por los nodos seguidores (followers).
    - Se comunicará con el proxy mediante gRPC.
2. **Proxy (Proceso 2):**
    - Actuará como intermediario entre el cliente y los nodos de datos.
    - Su función principal es redirigir las solicitudes de los clientes hacia el nodo líder para operaciones de escritura y hacia los seguidores para operaciones de lectura.
    - Cuando el líder falla, el proxy debe ser notificado de la nueva elección de líder y redirigir las solicitudes al nuevo nodo líder.
    - Utiliza gRPC para comunicarse tanto con el cliente como con los nodos de datos.
3. **Nodos de datos (Procesos 3, 4 y 5):**
    - **Líder** (Proceso 3): Recibe las solicitudes de escritura y las replica en los seguidores.
    - **Seguidores** (Procesos 4 y 5): Mantienen réplicas del estado de la base de datos y atienden solicitudes de lectura.
    - Todos los nodos se comunican entre sí usando gRPC para coordinar la replicación de datos y la elección de líder.

Planos de Datos y Control

1. **Plano de datos:**
    - El plano de datos maneja la replicación de las operaciones entre los nodos de datos.
    - El líder recibe las solicitudes de los clientes y coordina las actualizaciones de datos en los seguidores.
2. **Plano de control:**
    - El plano de control es responsable de gestionar la elección de líder.

- Cuando un nodo detecta la falta de comunicación del líder, inicia el proceso de elección mediante la votación entre nodos utilizando gRPC.

3. Diseño del Sistema

El sistema distribuido estará compuesto por cinco instancias EC2 en AWS, cada una con un rol específico:

1. **Cliente (Instancia 1):**
    - Interfaz para los usuarios para enviar solicitudes de lectura y escritura.
    - Enviar solicitudes a través del proxy.
2. **Proxy (Instancia 2 -NGINX):**
    - Punto de entrada para las solicitudes del cliente.
    - Redirige las solicitudes de escritura al líder actual y las de lectura a cualquier seguidor disponible.
    - Mantiene información actualizada sobre qué nodo es el líder del clúster.
3. **Nodos de Datos (Instancias 3, 4, y 5):**
    - Nodo líder (proceso 3): Gestiona las solicitudes de escritura y coordina la replicación.
    - Nodos seguidores (procesos 4 y 5): Replican los datos del líder y atienden las solicitudes de lectura.
    - Los nodos de datos están sincronizados mediante el algoritmo de consenso Raft.

Arquitectura

La arquitectura sigue un patrón maestro-seguidor (líder-seguidor):

- **Cliente → Proxy → Nodo Líder → Nodos Seguidores**
- El líder coordina la replicación de logs y asegura la consistencia de los datos.
- Los seguidores sirven las operaciones de lectura y participan en el proceso de elección de líder en caso de fallo del líder.

Cliente-Servidor (Cliente y Proxy)

- **Cliente**: El cliente actúa como la interfaz de usuario que envía solicitudes de lectura y escritura al sistema.
- **Proxy**: El proxy es un intermediario que se comunica con los nodos del sistema distribuido. Redirige las solicitudes de escritura al líder y las de lectura a los seguidores, manteniendo al cliente abstraído de los detalles internos del sistema.

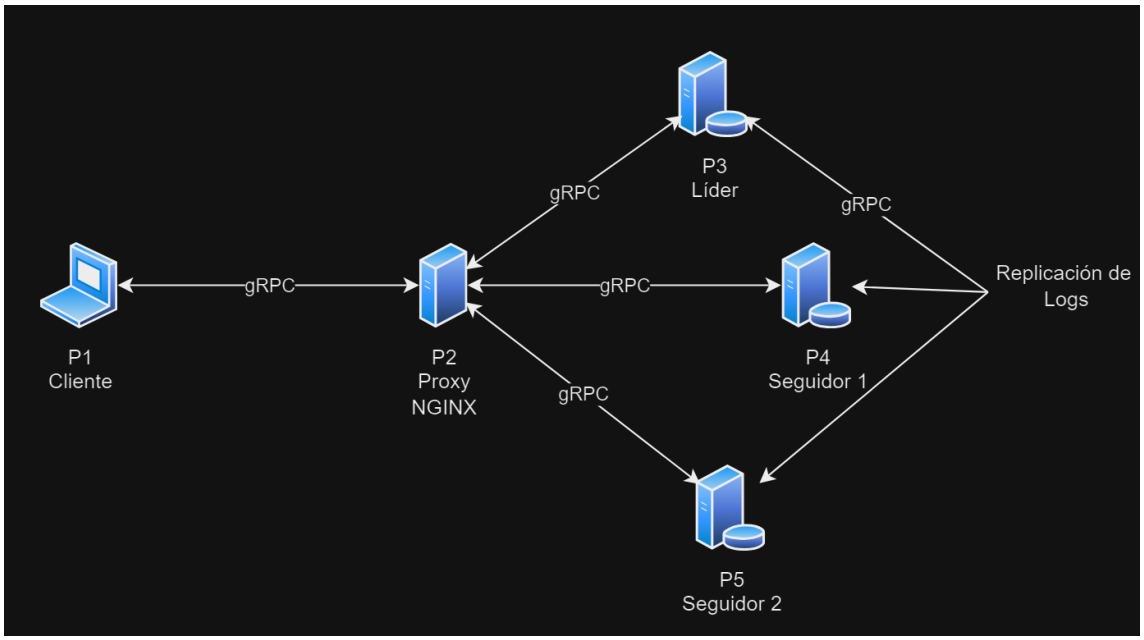
Arquitectura en N-Layer (N Capas)

- El proyecto también tiene una separación clara entre el **plano de control** y el **plano de datos**, lo que introduce una arquitectura en capas. El plano de control maneja la

coordinación entre los nodos (como la elección de líder), mientras que el plano de datos se ocupa de las solicitudes de clientes y la replicación de información.

Diagrama de Arquitectura

1. Cliente hace solicitud a través del Proxy.
2. Proxy identifica el líder actual y redirige las solicitudes de escritura.
3. El líder coordina la replicación de datos con los seguidores.
4. Si el líder falla, se inicia un proceso de elección para seleccionar un nuevo líder entre los seguidores.



Flujo de Solicitudes:

1. **Cliente → Proxy (NGINX)**:
   - El cliente envía solicitudes al proxy, sin conocer la estructura interna del sistema (quién es líder o seguidor).
2. **Proxy → Nodos (Líder o Seguidores)**:
   - El proxy redirige las solicitudes de escritura al líder y las de lectura a los seguidores.
   - Cuando el líder cambia, el proxy es actualizado con la nueva dirección del líder.
3. **Nodos (Líder y Seguidores)**:
   - Solo los nodos mantienen el **log replicado** para asegurar que las operaciones de escritura sean consistentes y replicadas entre todos los seguidores.

3. Tecnologías Seleccionadas

Para el desarrollo de este proyecto se utilizarán las siguientes herramientas y tecnologías:

1. **Lenguaje de programación: Java**

- Java es robusto, tiene un ecosistema extenso y es ampliamente utilizado en sistemas distribuidos.

2. **Maven**
    - Maven será el gestor de dependencias y construcción del proyecto. Se utilizarán dependencias para gRPC y alguna librería compatible con Raft.
3. **gRPC**
    - gRPC será utilizado para las comunicaciones entre los nodos, ya que permite definir interfaces claras y seguras mediante archivos .proto.
4. **Raft (librería compatible con Maven)**
    - Para implementar el algoritmo Raft, una opción recomendada es utilizar la librería **Copycat** (https://atomix.io/copycat/), la cual es compatible con Java y Maven. Copycat proporciona una implementación sencilla y eficiente de Raft para la replicación de logs y la elección de líderes en sistemas distribuidos.
5. **AWS EC2**
    - El proyecto se ejecutará en cinco instancias de EC2:
        - Una instancia para el **cliente**.
        - Una instancia para el **proxy**.
        - Tres instancias para los **nodos de datos** (líder y seguidores).
    - Cada instancia tendrá configuraciones de seguridad y red apropiadas para permitir la comunicación mediante gRPC.
6. **NGINX**
    - Ofrecerá balanceo de carga y tolerancia a fallos al gestionar la redirección de las solicitudes gRPC a los nodos correctos.