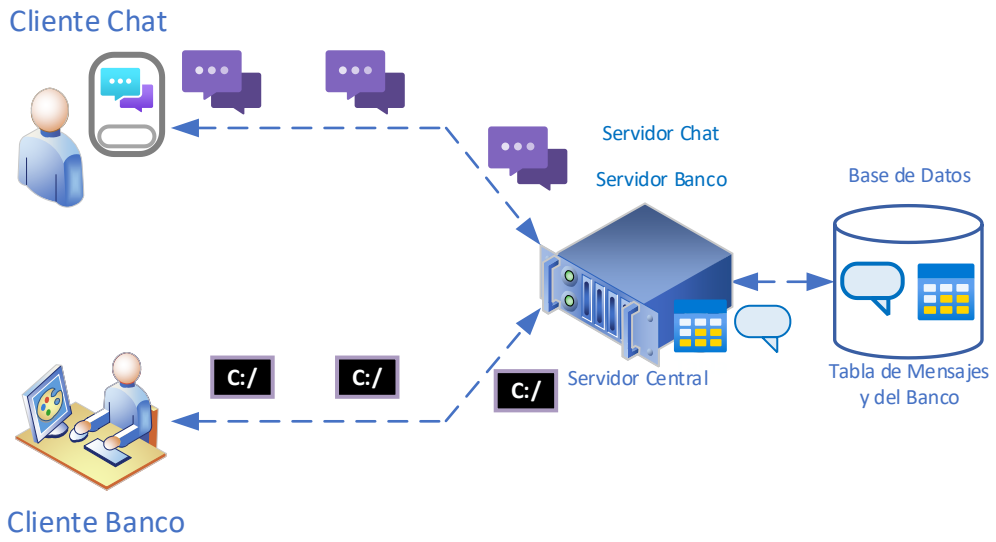
Escuela de Ciencias de la Computación
Parcial 2025-II
CC4P1 Programación Concurrente y Distribuida

Se tiene un sistema chat y consola para atención al cliente para transacciones de pagos y de deudas.



Sistema de Chat y del Banco tiene su base de datos

Donde cada cliente se conecta con su id al servidor mediante mensajes en chat, y tienen las siguientes actividades:

El **cliente Chat** solicita al **servidor Chat**:

- Entorno tipo chat con UI.
- Consulta vía texto de saldos, transacciones y préstamos.

También cada cliente Banco se comunica mediante terminar con el Servidor banco para diferentes consultas.

El **cliente Banco** solicita al **servidor Banco**:

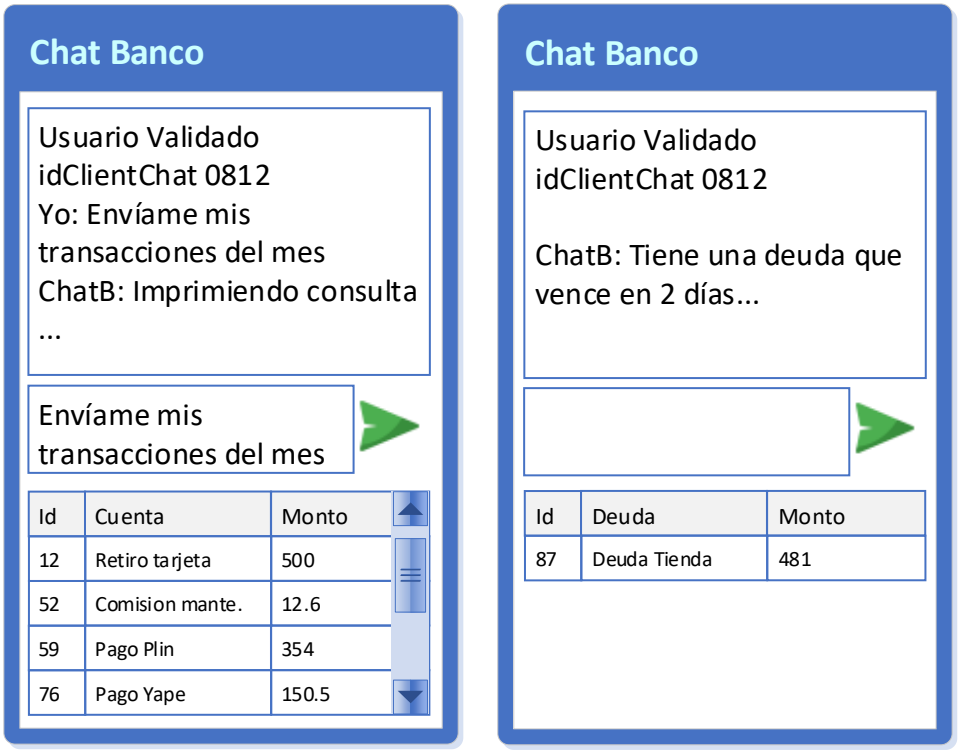
- Al inicio crear 10000 cuentas con monto dinero inicial
- Entorno vía comandos.
- CRUD necesario de cuentas, transacciones y prestamos, depende de su desarrollo.

para consultar sus saldos, y el servidor envía a sus clientes.

Sistema Chat Mensajes

El cliente del Sistema de Chat de Mensajes es solamente interfaz gráfica tipo chat.

Tiene una interfaz gráfica simple con textedit, button, list, textviwe, label.



Tiene una Base de Datos de Mensajes y del Banco´

Respecto las tablas del sistema de mensajes, puede contener las siguientes tablas y atributos dependiendo su diseño. El sistema Chat Mensajes, puede tener las siguientes tablas con los siguientes atributos dependiendo su desarrollo.

ClienteChat	id_cliente_chat (PK), nombre, fecha_conexion
MensajesChat	id_mensaje (PK), id_cliente_chat (FK), mensaje, respuesta, fecha_envio, tipo_mensaje
AvisosServidor	id_aviso (PK), id_cliente (FK), tipo_aviso, contenido, fecha_envio

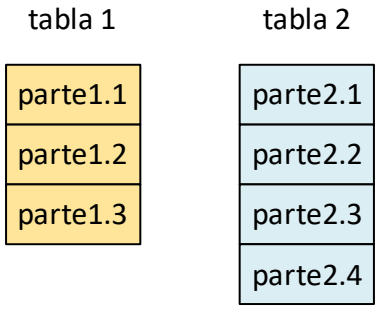
El cliente del Sistema de Banco es con comandos de manera de terminal.

Respecto las tablas del sistema del banco, el Sistema de Pagos, se pueden hacer transacciones de cuenta a cuenta, y también transferir pagos de préstamos y actualizar la tabla de préstamos. El sistema Banco, puede tener las siguientes tablas y atributos dependiendo su desarrollo.

Cuentas	id_cuenta (PK), id_cliente, saldo, fecha_apertura
Transacciones	id_transaccion (PK), id_cuenta (FK), tipo, monto, fecha
Prestamos	id_prestamo (PK), id_cliente, monto, monto pendiente, estado, fecha_solicitud

Estructura de las Tablas Datos Distribuida

Las tablas estarán en un archivo de formato por ejemplo txt. El sistema ahora tendrá tablas principales, todas las tablas particionadas en tres o más partes y replicadas tres veces en varios nodos trabajadores.



Al inicio el Se creará la tabla con miles de cuentas con sus clientes. También se tendrá una opción de arqueo de la tabla cuenta donde se sumará el dinero de todas las cuentas, para verificar el dinero del total no sufran cambios.

Flujo del Sistema Distribuido

Clientes

Hay dos tipos de clientes:

- Cliente Chat
  El cliente Chat en un cliente con Interfaz Gráfica, Pueden realizar múltiples solicitudes de datos de cuentas de manera escrita las transacciones, estados de préstamos, así como las deudas.

- Cliente Banco
  El cliente Banco un cliente tipo terminal, pueden realizar múltiples transacciones con un delay aleatorio para simular concurrencia real con cientos de transacciones. Envían solicitudes de consulta o transferencia al servidor central.

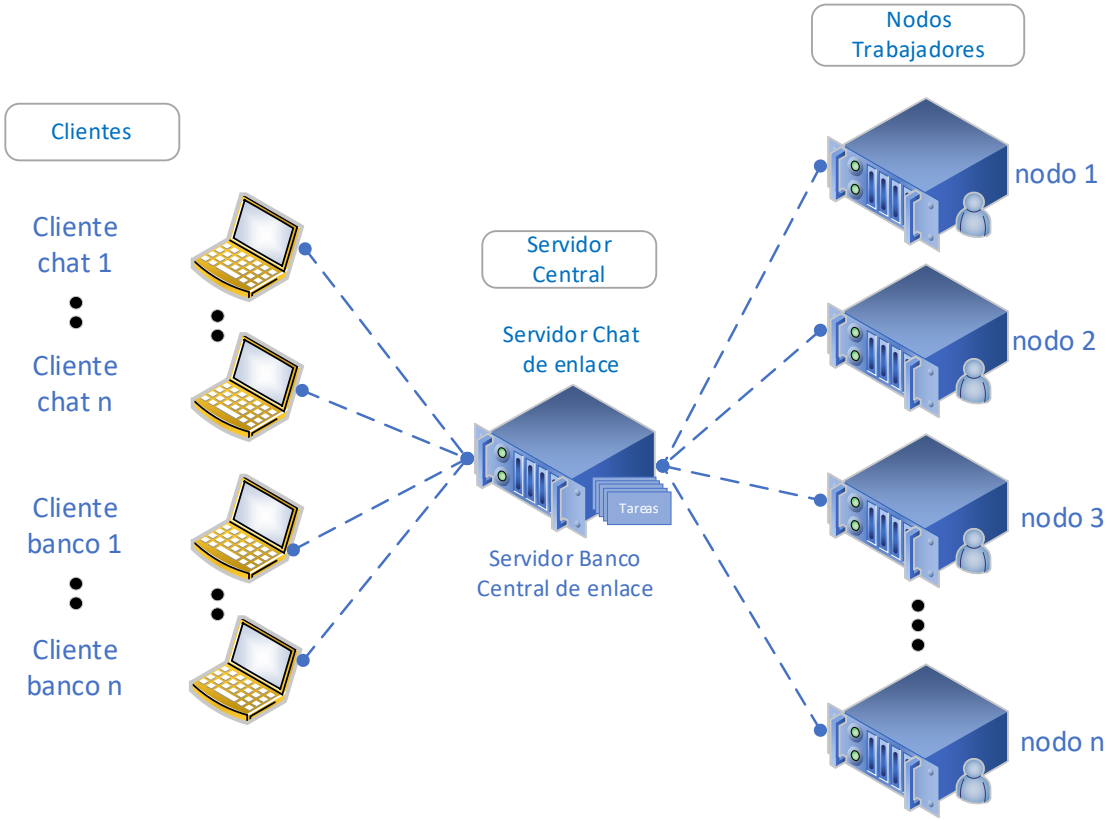
Servidor Central

El Servidor central ejecuta de manera de comandos sin UI, y procesa toda la información usando threads de manera concurrente. Tiene dos servidores para el cliente chat y para el cliente servidor. El servidor central actúa como coordinador. Recibe solicitudes de

clientes chat o clientes del banco asigna operaciones a nodos trabajadores disponibles para la réplica de sus tablas. Si un nodo trabajador falla, reenvía la operación a otro nodo con la réplica del dato. Si un nodo trabajador falla reenvía la operación a otro nodo. Cuando el nodo trabajador termina su tarea, luego lo deriva al servidor central el resultado, y a su vez el servidor central envía el resultado al cliente chat o al cliente banco, como la confirmación o no.

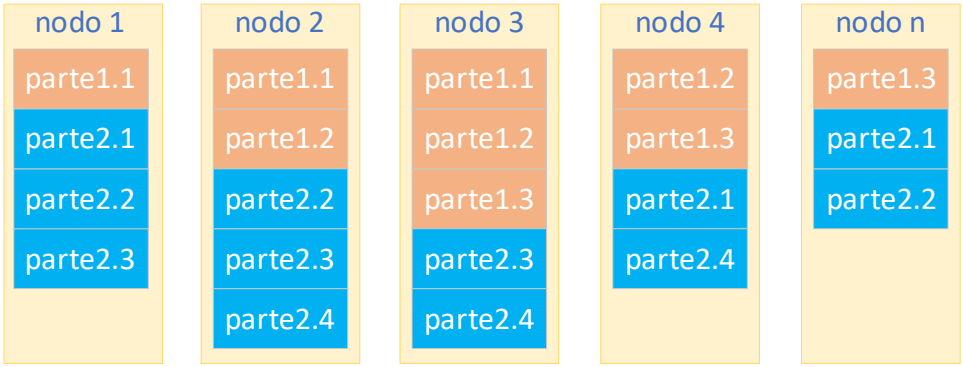
Nodos Trabajadores

El nodo procesa las consultas o transacciones usando todos núcleos que tienen mediante threads. Cada trabajador disponible, verifica cada transacción de lectura o escritura, como la existencia de saldo, prestamos, deudas, etc. antes de ejecutar transferencias y da los resultados al servidor central.

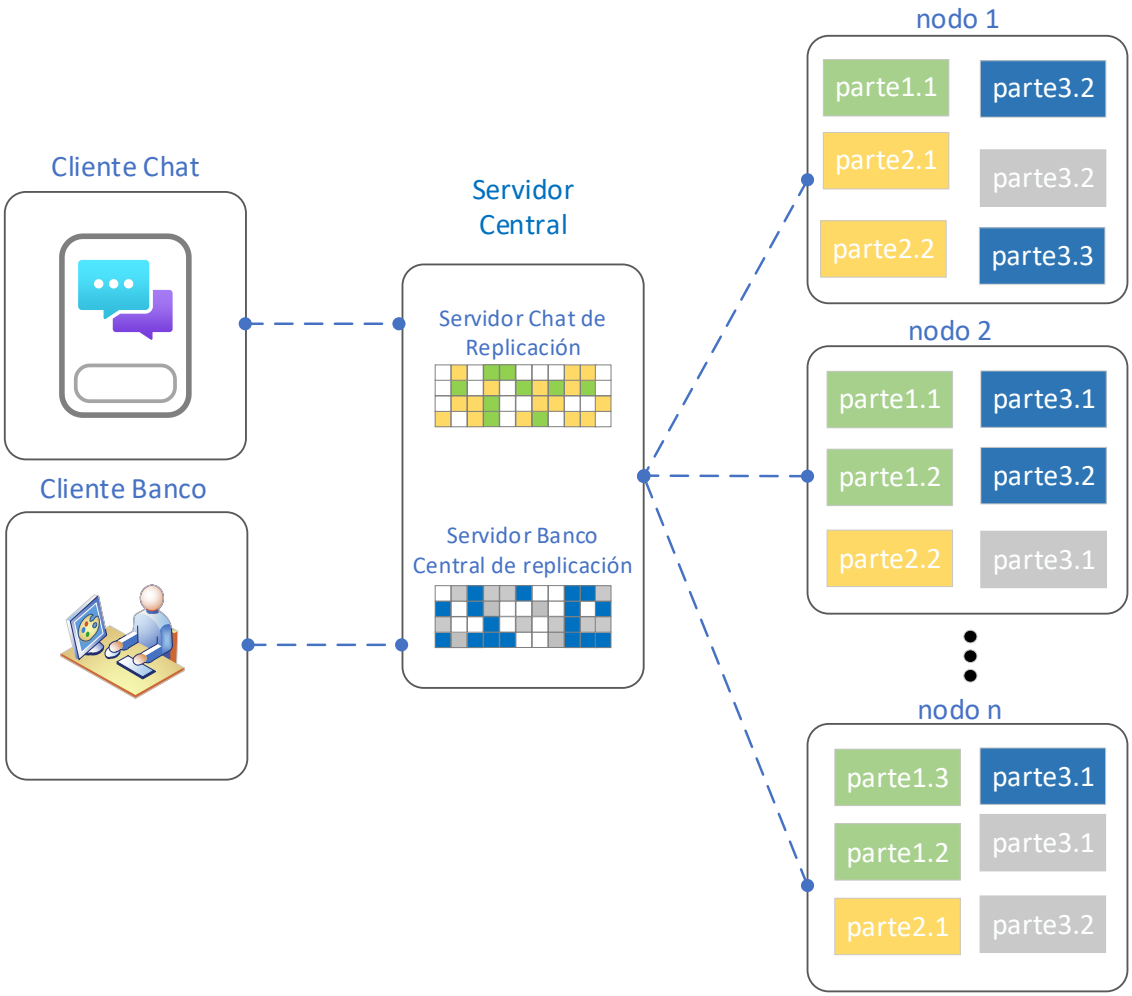


Se almacenan las particiones de las tablas en los nodos trabajadores. Cada tabla se divide en 3 o más partes y se replica 3 veces en diferentes nodos. Ejecutan transacciones de lectura y actualización según lo asignado por el servidor central.

Nodos Trabajadores



El nodo central tiene la ubicación de las replicas de las particiones de todas las tablas de tal manera que si algún nodo se cae el sistema sigue funcionando en otro nodo con su replica.



Notación de las Queries

El sistema soporta tipos de transacciones por ejemplo tomar como base de:

Query 1: Consulta de cuenta sobre el saldo de una cuenta específica.

```
OPERACIÓN: CONSULTAR_CUENTA
PARÁMETROS: ID_CUENTA
VALIDACIÓN: EXISTE CUENTA con ID_CUENTA
ACCIONES:
- BUSCAR CUENTA por ID_CUENTA
- EXTRAER DATOS: ID_CLIENTE, SALDO, FECHA_APERTURA
SALIDA: DETALLE_CUENTA / ERROR (si la cuenta no existe)
```

Ejemplo de consulta: CONSULTAR_CUENTA (101) → Retorna: 1500.00

Query 2: Transferencia entre cuentas, realiza una transferencia de fondos entre dos cuentas, validando saldo previo.

```
OPERACIÓN: TRANSFERIR_CUENTA
PARÁMETROS: ID_CUENTA_ORIGEN, ID_CUENTA_DESTINO, MONTO
VALIDACIÓN:
- EXISTEN ambas cuentas (origen y destino)
- SALDO de CUENTA_ORIGEN ≥ MONTO
ACCIONES:
- RESTAR MONTO del SALDO de la CUENTA_ORIGEN
- SUMAR MONTO al SALDO de la CUENTA_DESTINO
- CREAR registro en la tabla TRANSACCIONES con:
   - ID_CUENTA_ORIGEN
   - ID_CUENTA_DESTINO
   - TIPO = "Transferencia"
   - MONTO
   - FECHA
SALIDA: CONFIRMACIÓN / ERROR
```

Ejemplo de operación: TRANSFERIR_CUENTA (101, 102, 500.00) → Retorna: Confirmación

Se verifica que la cuenta 101 tenga al menos 500.00 antes de proceder.

Query 3: Consulta del estado de pago de préstamo, se utiliza para consultar el estado actual y el monto pendiente de los préstamos asociados a una cuenta específica, permitiendo al cliente conocer su nivel de deuda.

```
OPERACIÓN: ESTADO_PAGO_PRESTAMO
PARÁMETROS: ID_CUENTA
VALIDACIÓN:
- EXISTE CUENTA con ID_CUENTA
- EXISTEN PRÉSTAMOS asociados a la cuenta
ACCIONES:
- BUSCAR todos los PRÉSTAMOS vinculados al ID_CUENTA
- PARA CADA PRÉSTAMO:
  - CONSULTAR MONTO_TOTAL
  - CONSULTAR MONTO_PAGADO acumulado
  - CALCULAR MONTO_PENDIENTE = MONTO_TOTAL – MONTO_PAGADO
  - DETERMINAR ESTADO:
    - Si MONTO_PENDIENTE = 0 → "Cancelado"
```

```
    - Si MONTO_PENDIENTE > 0 y fecha límite no vencida → "Activo"
    - Si MONTO_PENDIENTE > 0 y fecha límite vencida → "Vencido"
SALIDA:
- LISTADO de PRÉSTAMOS con:
  - ID_PRESTAMO
  - MONTO_TOTAL
  - MONTO_PAGADO
  - MONTO_PENDIENTE
  - ESTADO_ACTUAL
- ERROR (si no hay préstamos asociados a la cuenta)
```

Ejemplo de operación donde se quiere saber el estado de pago del préstamo: ESTADO_PAGO_PRESTAMO (107) → Retorna: Listado de Prestamos

LP = lenguaje de programación.

Desarrollo de los Cliente y el Servidor Central (LP1) y Nodos "n" (LP2), donde se tienen que cumplir con los siguientes casos:

- Caso 1 donde LP1 = LP2 (ejem se ejecute como mínimo jdk 8) (todo el sistema distribuido paralelo funciona con java).
- Caso 2 donde LP1 <> LP2, donde los(s) Nodo(s) está en otro lenguaje que no sea Java.
- Caso 3 los que tienen grupo de tres o más agregar otros lenguajes de programación en Nodo(s) adicional(es) (ejem LP3, LP4, etc.) se toma en cuenta la cantidad de lenguajes de programación.

Desarrollar en un cluster de datos para desplegar el programa.

Desarrollando el servidor en internet o en la nube (opcional).

Describir la arquitectura diseñada.

Describir el diagrama de protocolo.

No usar websocket, no usar socket io, no usar algún middleware, no usar otras librerías de conexión u otros protocolos ya trabajados por un tercero.

Usar las Apis, librerías o sdk estándares del lenguaje de programación base.

Explicar el Desarrollo del programa.

Subir en Univirtual.

Se consultará una pregunta en su cluster o en otro cluster de clase.

Hacer pruebas donde los clientes puedan enviar cientos de transacciones regulando con un delay aleatorio entre las transacciones, para saturar el sistema y ver sus límites.

Hacer notas de recordatorio para su exposición.

Comprimido consta

- Subir códigos fuentes de en extensión en el LP1 y LP2 (o LP3), según sea el caso.
- PDF Informe.
- PDF Presentación.
- Evaluar el desempeño con el mayor número de nodos en red con graficas.

Parte 1