Programación Distribuida

Integrantes: Katherine Muñoz Sánchez

Jonathan Guzmán Zambonino

La programación distribuida se refiere a la práctica de desarrollar software que se ejecuta en múltiples dispositivos o servidores interconectados para lograr un objetivo común.

Ejemplo de programación distribuida.

https://github.com/kathyLizMu/programacionDistribuida.git

El código seleccionado es de un Sistema Distribuido Simple para el manejo de Cuentas Bancarias, utilizando arquitectura cliente-servidor, con comunicación vía sockets TCP/IP.

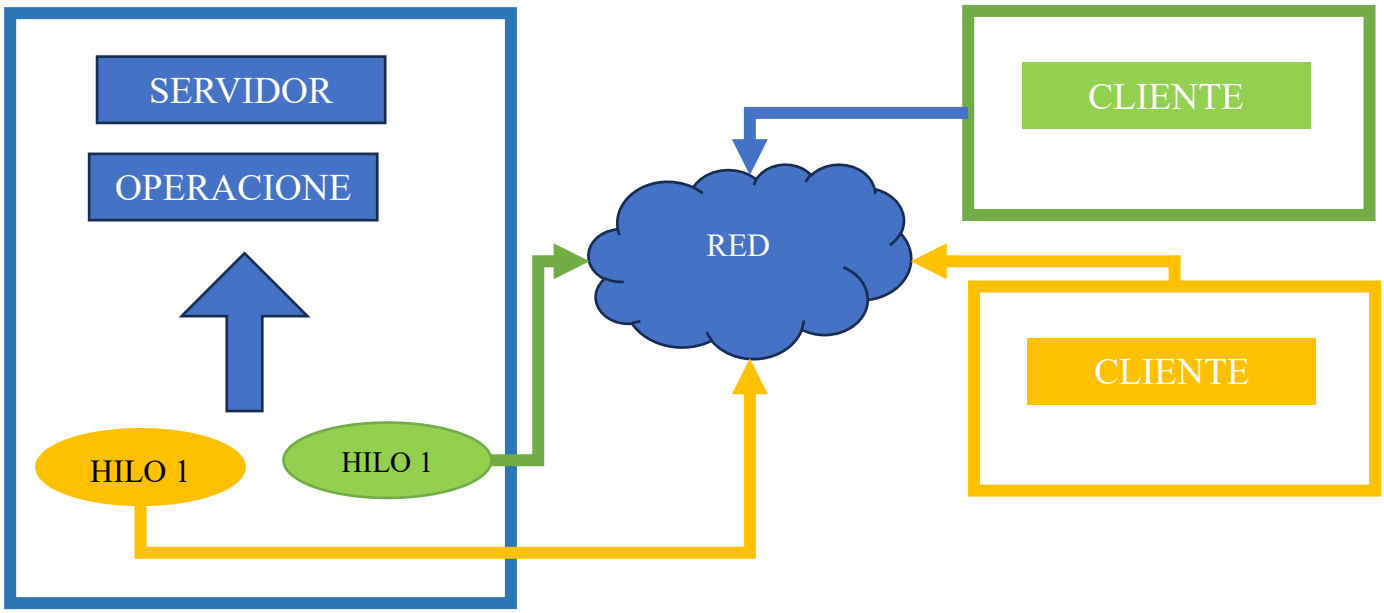
Operaciones

- Acceso
- Consulta de saldo
- Retiro
- Depósito
- Transferencia

Arquitectura

Cliente-servidor, donde el servidor central maneja las operaciones bancarias y los clientes interactúan con el servidor para realizar transacciones. Tanto el cliente como el servidor son procesos separados que se ejecutan de manera independiente.

Los clientes se conectan al servidor y solicitar las operaciones deseadas. Cada cuenta bancaria tendrá una cuenta corriente vinculada a un número de ID y al respectivo nombre del cliente



Servidor

Cada conexión TCP/IP a los clientes se maneja en un subproceso separado, para permitir que el servidor atienda a múltiples clientes simultáneamente.

Almacenamiento

El servidor es el encargado de gestionar las cuentas bancarias y las operaciones realizadas por las mismas. Para ello, guarda cada cuenta en un archivo JSON, siendo el nombre del archivo el número de ID del usuario.

Concurrencia mediante hilos

El código utiliza hilos (threads) para manejar múltiples conexiones de clientes simultáneamente. Cada conexión TCP/IP se maneja en un subproceso separado, lo que permite que el servidor atienda a varios clientes al mismo tiempo.

El acceso a estos archivos se realiza simultáneamente, utilizando el módulo threading.Lock para garantizar que solo un subproceso tenga acceso al archivo a la vez.

La coordinación de los hilos y la gestión segura de los recursos compartidos son elementos clave en la programación distribuida. Es importante para permitir que varios clientes se conecten y realicen operaciones al mismo tiempo sin bloquear el servidor.

Cliente

Comunicación

El cliente es responsable de enviar solicitudes al servidor y recibir respuestas. Para ello, utiliza el módulo **socket** para conectarse al servidor y enviar las solicitudes. Permitiendo la transferencia de datos a través de la red, por lo que las máquinas pueden estar en ubicaciones físicas diferentes.

Protocolo

La comunicación entre el cliente y el servidor se realiza a través de mensajes de texto, cada uno de los cuales representa una operación. Las clases que representan los mensajes están en el módulo pixson.recursos.protocolo, donde cada clase representa un mensaje.

Ejemplo de mensaje de solicitud de transferencia del 10.5 de la cuenta 123456789 al 987654321, en el tiempo lógico 10:

t:10|op:4|origen_rg:1111111111|destino_rg:987654321|valor:10.5

Reloj lógico

Los sistemas distribuidos son concurrentes. Esto implica que cada componente es autónomo y ejecuta tareas concurrentes. De este modo debe haber una sincronización y coordinación de dichas tareas y a su vez, aparecen los problemas de la concurrencia como deadlocks2 y comunicación no confiable.

Debida a la comunicación por medio de mensajes, se dificulta la precisión con la que cada componente debe sincronizar su reloj, apareciendo el concepto de latencia

Con los relojes lógicos no es necesario contar con una sincronización exacta de los relojes, sino que es más relevante contar con un orden de ocurrencia de los eventos. De este modo, se define la relación sucedió antes, que permite determinar que, si dos procesos se encuentran relacionados, un evento de uno de ellos suceda antes que otro.

Se implementó un reloj lógico, basado en el algoritmo de Lamport, para identificar el orden de las operaciones. El cliente y el servidor comienzan con el reloj lógico puesto a cero, y cada operación incrementa el reloj lógico en 1.

Todos los mensajes enviados por el cliente y el servidor tienen un sello con el valor del reloj lógico actualizado. Cuando se recibe un nuevo mensaje, el reloj lógico del receptor se actualiza al valor más alto entre el reloj lógico y la marca de tiempo del mensaje recibido.

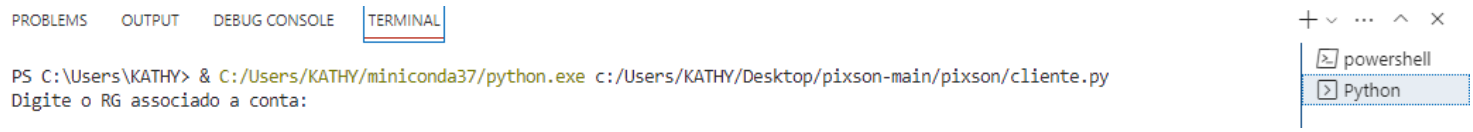
La utilización de un reloj lógico, en este caso basado en el algoritmo de Lamport, es una técnica común en sistemas distribuidos para ordenar eventos y operaciones en un entorno en el que múltiples componentes están ejecutándose de manera concurrente y en diferentes máquinas.

El Servidor entrega un numero a cada hilo que desea acceder a este y les atiende de uno en uno por orden de llegada. Una vez que toca le toca su turno, el proceso ingresa a su etapa crítica y después de realizada esta, el servidor le libera y atiende al siguiente proceso. En el algoritmo de Lamport se permite que varios hilos obtengan el mismo número. En este caso, se aplica un algoritmo de desempate, que garantiza que sólo un hilo entra en sección crítica

Capturas

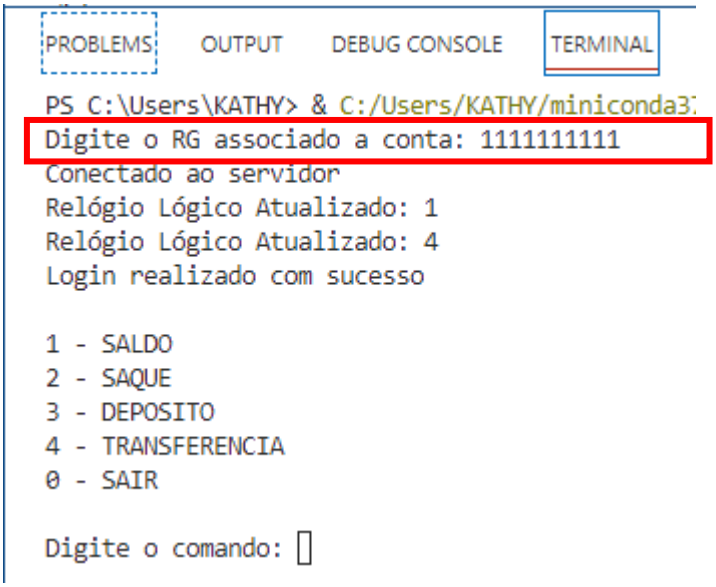
Iniciar servidor

Iniciar cliente

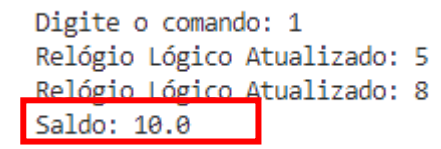


Operaciones

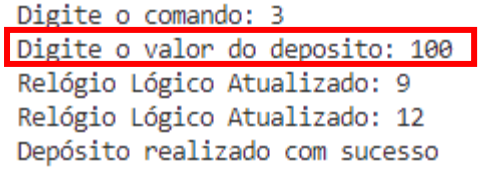
Acceso a la cuenta 1111111111



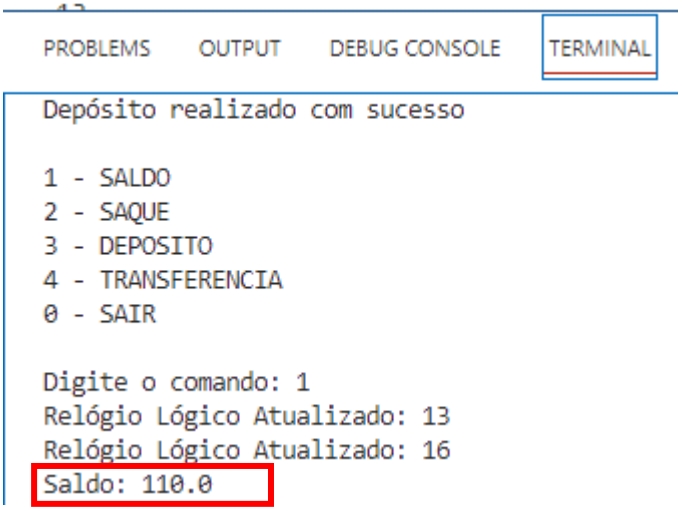
Consulta de saldo



Deposito de $100



Consulta de lo depositado



Retiro de $15

```
Digite o comando: 2
Digite o valor do saque: 15
Relógio Lógico Atualizado: 17
Relógio Lógico Atualizado: 20
Saque realizado com sucesso
```

Transferencia de $20 a la cuenta 0000000000

```
Digite o comando: 4
Digite o RG do destinatário: 0000000000
Digite o valor da transferência: 20
Relógio Lógico Atualizado: 21
Relógio Lógico Atualizado: 24
Transferência realizada com sucesso
```

Consulta de saldo final

```
Digite o comando: 1
Relógio Lógico Atualizado: 25
Relógio Lógico Atualizado: 28
Saldo: 75.0
```

Desconexión del cliente

Parte servidor

```
Servidor iniciado na porta 5000
Aguardando conexão...
Novo cliente conectado ('127.0.0.1', 59329)
Relógio Lógico Atualizado: 2
Relógio Lógico Atualizado: 3
Relógio Lógico Atualizado: 6
Relógio Lógico Atualizado: 7
Relógio Lógico Atualizado: 10
Relógio Lógico Atualizado: 11
Relógio Lógico Atualizado: 14
Relógio Lógico Atualizado: 15
Relógio Lógico Atualizado: 18
Relógio Lógico Atualizado: 19
Relógio Lógico Atualizado: 22
Relógio Lógico Atualizado: 23
Relógio Lógico Atualizado: 26
Relógio Lógico Atualizado: 27
Cliente desconectado
```