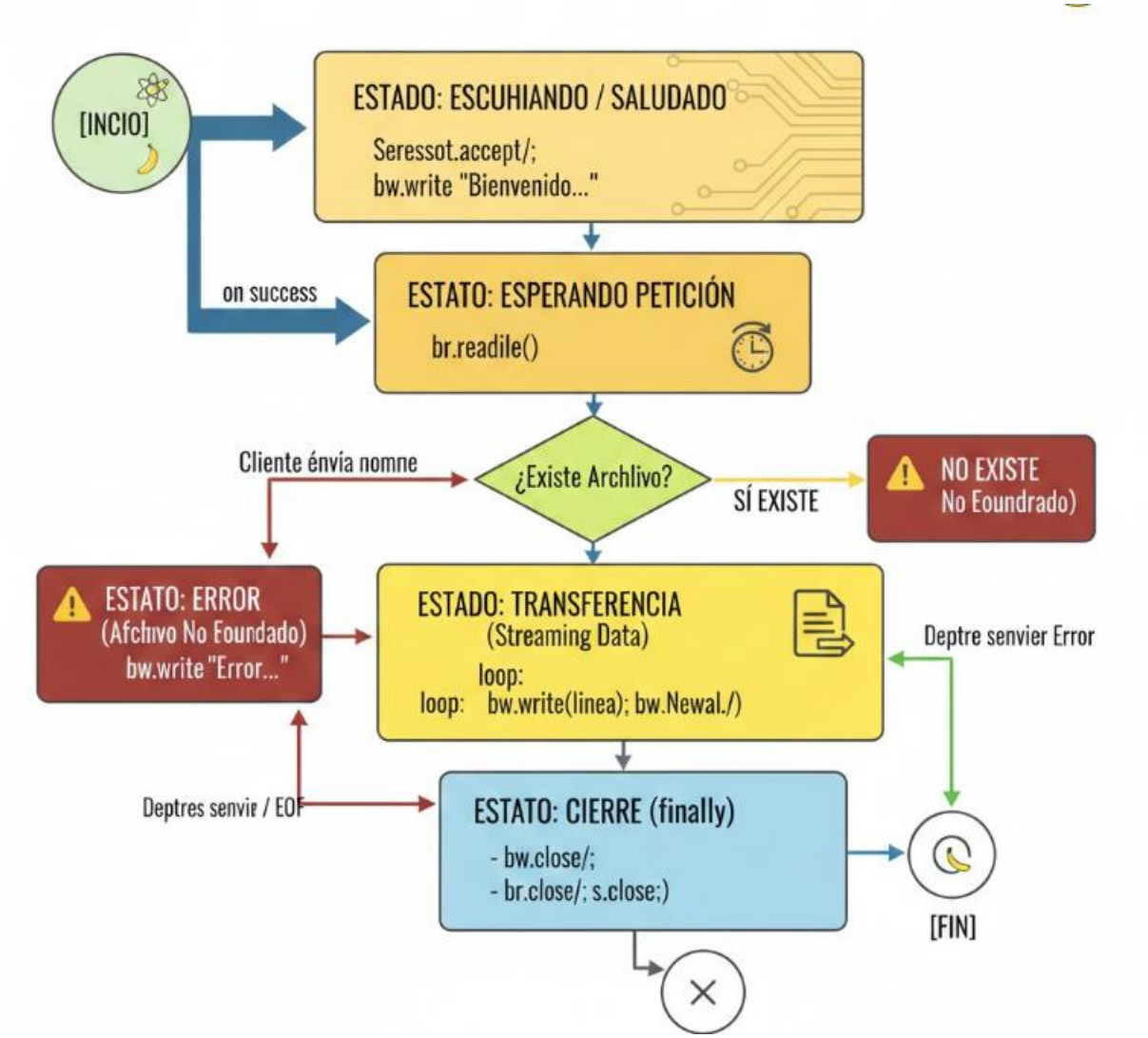
ESTADOS



Para representar visualmente el funcionamiento de tu **Servidor**, el diagrama de estados es ideal porque muestra cómo el hilo cambia de "humor" según lo que recibe del cliente.

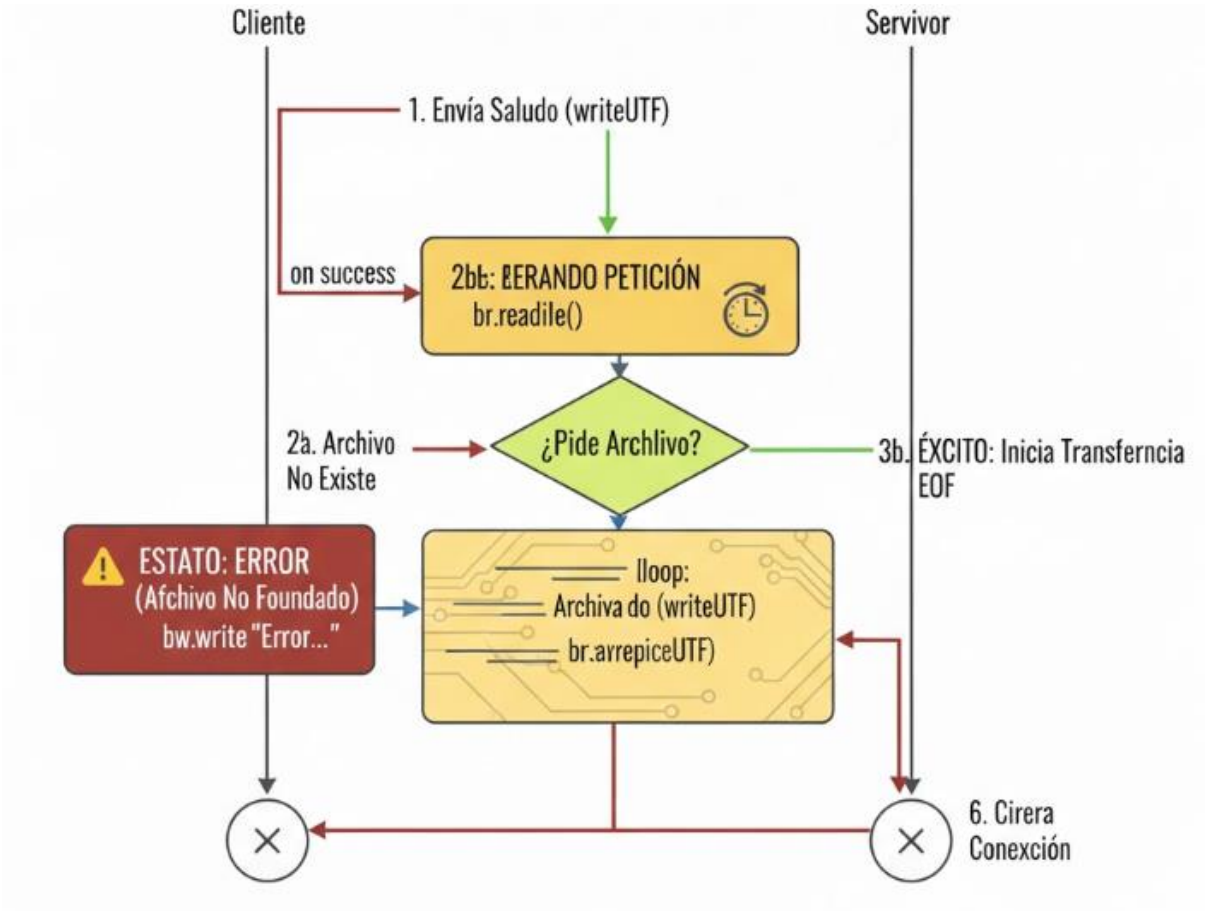
Aquí tienes la representación visual de los estados por los que pasa tu clase `Servidor` desde que nace hasta que muere:

Explicación de los Estados Visualizados

1. **Estado: INICIO (Start)**
    a. **Evento:** El `ServerSocket` acepta la conexión y el hilo principal llama a `start()`.

    b. **Acción:** Se reserva memoria para el objeto `Servidor` y se preparan los buffers (`br` y `bw`).
2. **Estado: SALUDANDO (Greeting)**
    a. **Acción:** El servidor envía la cadena de bienvenida.
    b. **Transición:** Una vez enviado, pasa automáticamente a esperar.
3. **Estado: ESPERANDO PETICIÓN (Waiting for Request)**
    a. **Acción:** El hilo se queda bloqueado (pausado) en la línea `br.readLine()`. No consume CPU, solo espera datos del cliente.
    b. **Evento:** El cliente envía el nombre del archivo.
4. **Estado: VALIDANDO ARCHIVO (Validating)**
    a. **Lógica:** Se ejecuta el `if (f.exists())`.
    b. **Decisión:** * Si **No existe**: Salta al estado de **ERROR**.
        i. Si **Existe**: Abre el `FileReader` y pasa a **TRANSFERENCIA**.
5. **Estado: TRANSFERENCIA (Streaming Data)**
    a. **Bucle:** Lee una línea del disco -> la envía al socket -> repite hasta el final del archivo.
    b. **Transición:** Al terminar el archivo, envía la marca `"EOF"`.
6. **Estado: FINALIZACIÓN / CIERRE (Closing)**
    a. **Acción:** Se ejecuta el bloque `finally`. Se cierran los flujos de datos y el socket.
    b. **Resultado:** El hilo termina su ejecución y es eliminado por el recolector de basura de Java.

TRANSACCIONES



```
int estado=1

do{
   switch(estado){

   case 1:
      flujo_salida.writeUTF("Introduce comando (ls/get/exit)");
      comando=flujo_entrada.readUTF();

      if(comando.equals("ls")){
System.out.println("\tEl cliente quiere ver el contenido del directorio");
    // Muestro el directorio

estado=1;
break;
      }else
         if(comando.equals("get")){
   // Voy al estado 3 para mostrar el fichero
   estado=3;
   break;
         }else
       estado=1;
       break;

   case 3://voy a mostrar el archivo
      flujo_salida.writeUTF("Introduce el nombre del archivo");
      String fichero =flujo_entrada.readUTF();
      // Muestor el fichero

      estado=1;
```

```
         break;
      }

      if(comando.equals("exit")) estado=-1;
   }while(estado!=-1);
```

1. **Tiempo de Procesamiento:** Mide el tiempo que el servidor pasa interactuando con la lógica inicial y esperando la respuesta del cliente. Es un tiempo "muerto" donde la CPU suele estar esperando al usuario o al disco.
2. **Tiempo de Transacción:** Mide la velocidad del flujo de datos. Si el archivo es muy grande o la red es lenta, este valor subirá.

Resumen de unidades

- **NanoSegundos (10^{-9} s):** Ideal para medir procesos internos de CPU.
- **MiliSegundos (10^{-3} s):** Es lo que solemos usar para logs de red, ya que la latencia de Internet rara vez baja de 1ms.

```
@Override
public void run() {
  try {
    // --- INICIO PROCESAMIENTO ---
    long inicioProcesamiento = System.nanoTime();

    bw.write("Bienvenido, ¿que fichero quieres?");
    bw.newLine();
    bw.flush();

    String nombreArchivo = br.readLine();
    File f = new File(nombreArchivo);

    long finProcesamiento = System.nanoTime();
    // ----------------------------

    if (!f.exists()) {
      bw.write("Error: Archivo no encontrado");
      bw.newLine();
      bw.flush();
    } else {
      // --- INICIO TRANSACCIÓN (Envío de datos) ---
      long inicioTransaccion = System.nanoTime();
```

```
      try (BufferedReader fr = new BufferedReader(new FileReader(f))) {
        String linea;
        while ((linea = fr.readLine()) != null) {
          bw.write(linea);
          bw.newLine();
        }
        bw.write("EOF");
        bw.newLine();
        bw.flush();
      }

      long finTransaccion = System.nanoTime();
      // ----------------------------------------

      // Cálculos (Convertimos a milisegundos para que sea legible)
      double tiempoProc = (finProcesamiento - inicioProcesamiento) / 1_000_000.0;
      double tiempoTrans = (finTransaccion - inicioTransaccion) / 1_000_000.0;

      System.out.println("\n--- ESTADÍSTICAS DEL HILO ---");
      System.out.println("Tiempo de Procesamiento: " + tiempoProc + " ms");
      System.out.println("Tiempo de Transacción: " + tiempoTrans + " ms");
      System.out.println("-----------------------------\n");
    }
  } catch (IOException ex) {
    System.err.println("Error: " + ex.getMessage());
  } finally {
    try { if (s != null) s.close(); } catch (IOException e) { e.printStackTrace(); }
  }
```

```
}
```