



**INSTITUTO POLITÉCNICO NACIONAL**

ESCUELA SUPERIOR DE CÓMPUTO



# Desarrollo de Sistemas Distribuidos

“Tarea 1: Sistema Distribuido que verifica si un número es primo ”

**Grupo: 4CV13**

**Alumno:**

**Godinez Morales Mario Sebastian**

**Profesor:**

**Pineda Guerrero Carlos**

# Introducción

Los sistemas distribuidos son una clase de sistemas informáticos compuestos por múltiples dispositivos conectados entre sí a través de una red de comunicaciones que trabajan juntos para realizar una tarea común. Estos sistemas permiten a las aplicaciones funcionar de manera eficiente y escalable, aprovechando el poder de procesamiento y almacenamiento de múltiples dispositivos en lugar de depender de un solo servidor centralizado.

Se utilizan en una amplia variedad de aplicaciones que van desde redes sociales, juegos en línea hasta sistemas de control de tráfico aéreo y plataformas de comercio electrónico. Los sistemas distribuidos también son fundamentales en el campo de la inteligencia artificial y el aprendizaje automático donde son de mucha ayuda al momento de entrenar modelos complejos en grandes conjuntos de datos.

Sin embargo, los sistemas distribuidos presentan desafíos únicos en términos de seguridad, confiabilidad y escalabilidad. La coordinación y comunicación entre los dispositivos debe ser cuidadosamente diseñada para garantizar que la tarea se realice de manera efectiva y sin errores. Además los sistemas distribuidos deben ser capaces de manejar fallos en cualquier punto de la red sin interrumpir la tarea en curso.

## Ventajas:

- Escalabilidad: los sistemas distribuidos pueden escalar fácilmente agregando nuevos dispositivos a la red, lo que permite manejar una mayor cantidad de trabajo sin disminuir el rendimiento.
- Tolerancia a fallos: en un sistema distribuido, si un dispositivo falla, otros dispositivos pueden asumir su función para garantizar que la tarea se complete de manera efectiva.
- Redundancia: la duplicación de datos y dispositivos en un sistema distribuido puede mejorar la tolerancia a fallos y la recuperación ante desastres.
- Eficiencia: en un sistema distribuido, los dispositivos pueden trabajar en paralelo para realizar tareas de manera más eficiente que en un sistema centralizado.
- Mejora en el rendimiento: en un sistema distribuido, se pueden utilizar dispositivos especializados para tareas específicas, lo que puede mejorar el rendimiento general del sistema.

## Desventajas:

- Complejidad: la naturaleza distribuida de los sistemas puede aumentar la complejidad del diseño y la implementación del sistema.
- Coordinación: la coordinación entre dispositivos puede ser un desafío en un sistema distribuido, ya que los dispositivos pueden trabajar en diferentes momentos y con diferentes datos.
- Seguridad: la seguridad en un sistema distribuido puede ser un desafío, ya que múltiples dispositivos deben ser protegidos contra amenazas internas y externas.
- Costos: la implementación y el mantenimiento de un sistema distribuido pueden ser costosos, ya que se requieren múltiples dispositivos y recursos de red.

- Latencia: la latencia en las comunicaciones entre dispositivos puede afectar el rendimiento del sistema y la calidad de la experiencia del usuario.

## Desarrollo

Para el desarrollo de esta práctica primero empecé programando el servidor B que es el intermediario entre el cliente y las instancias de los servidores A, el servidor empieza preguntando a qué puerto debe conectarse, en mi caso lo deje en el puerto 5,000.

Una vez que se tiene el puerto, ponemos al servidor a escuchar o esperar una conexión con un for infinito, una vez que se establece la conexión creamos los flujos de entrada y salida de datos que nos van a permitir hacer el intercambio de información entre el cliente y servidor, posteriormente pedimos el número al cliente envía un número al servidor B para saber si es primo o no, y lo leemos en el servidor B con el flujo entrada.readLong. Como lo muestra el siguiente bloque de código en la imagen 1:

```

public static void main(String[] args) throws Exception {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Introduzca el puerto para el servidor B");
    int puertob = scanner.nextInt();
    ServerSocket servidor = new ServerSocket(puertob); // Creamos un socket y lo ligamos al puerto 5000
    System.out.println("Esperando conexión"); // Mostramos mensaje en espera de una conexión
    for (;;) { // El servidor se queda escuchando indefinidamente
        Socket conexion = servidor.accept(); // Aceptamos la conexión
        System.out.println("Conexión establecida"); // Se notifica al usuario que se ha establecido una conexión
        /* Entrada y salida de datos */
        DataOutputStream salida = new DataOutputStream(conexion.getOutputStream());
        DataInputStream entrada = new DataInputStream(conexion.getInputStream());
        long k; // Declaramos el parámetro k
        long numero = entrada.readLong(); // Leemos el número recibido
        k = numero / 3; // Según la definición k = n/3
        System.out.println("Se ha recibido el número: " + numero);
        System.out.println("Enviando a los servidores A");
        /* Se crean los sockets para los servidores A */
        Socket conexionA1 = new Socket("localhost", 5001); // lo ligamos al puerto 5001
        Socket conexionA2 = new Socket("localhost", 5002); // lo ligamos al puerto 5002
        Socket conexionA3 = new Socket("localhost", 5003); // lo ligamos al puerto 5003
        /* Mandamos los intervalos a los servidores como se especificó en la tabla */
        Worker A1 = new Worker(conexionA1, numero, numeroInicial: 2, k); // Instancia 1
    }
}

```

Terminal Output:

```

Instancia.
1234567811, 411522604, 823045206
Enviado desde el servidor A: NO DIVIDE
Instancia.
1234567811, 823045207, 1234567810
Enviado desde el servidor A: NO DIVIDE
Esperando conexión...
PS C:\Users\marlo\OneDrive\Documents\Sistemas Distribuidos\Tarea1>

```

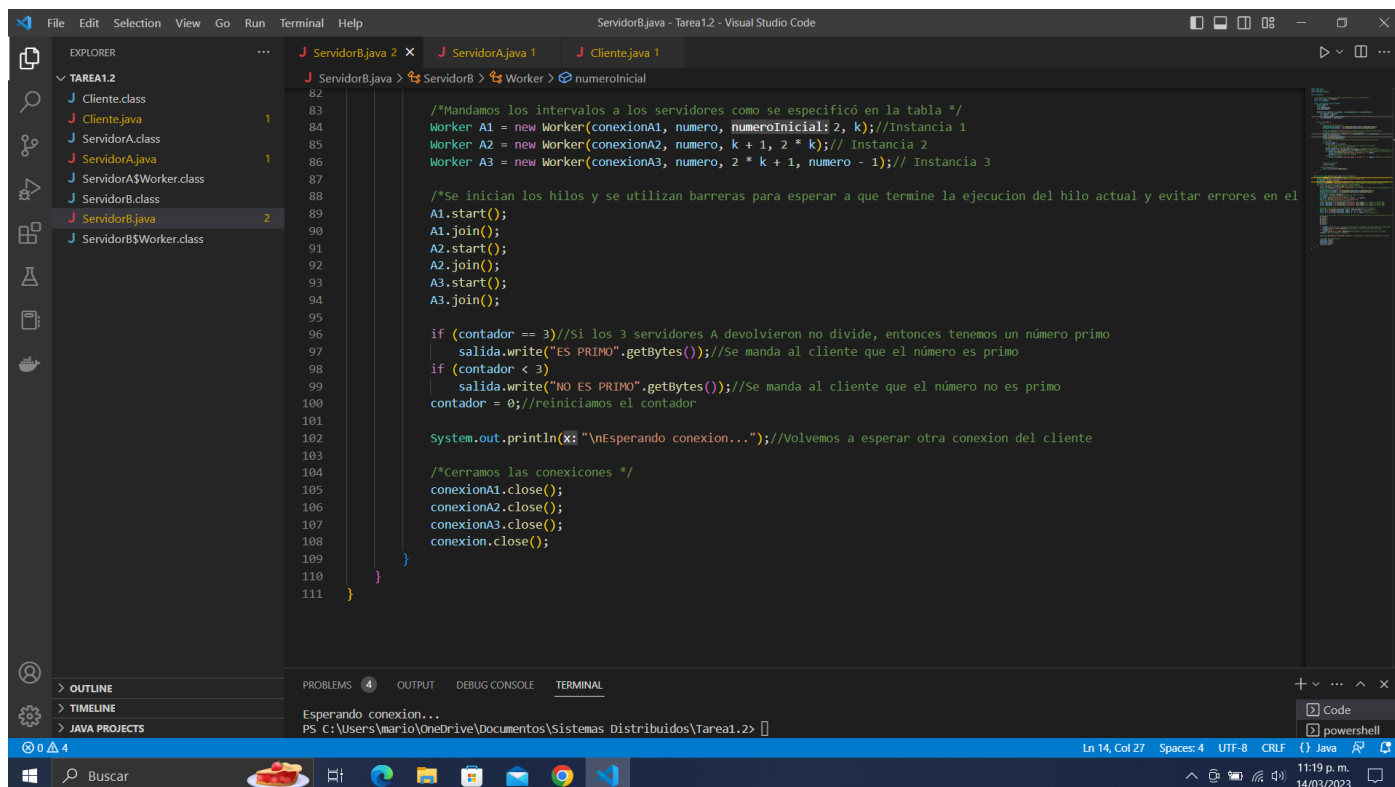
Imagen 1

Una vez recibimos el número lo mostramos en consola y creamos los sockets de las 3 instancias de los servidores A que están ligados a los puertos 5,001 , 5,002 y 5,003 respectivamente.

Los servidores A recibirán intervalos de números de acuerdo a la siguiente tabla:

Intervalo	NÚMERO INICIAL	NÚMERO FINAL
1	2	K
2	K+1	2*K
3	2*K+1	NÚMERO-1

Para mandar dichos intervalos, se creó un constructor que tiene 4 parámetros que son conexion, numeroInicial y numeroFinal dentro de la clase Worker, dicha clase extiende la clase Thread lo cual nos permite ejecutar en hilos y para mandar los intervalos a los servidores A simplemente mandamos a llamar a la clase y creamos 3 constructores con sus respectivos valores de acuerdo a la tabla mostrada como lo muestra la imagen 2:



```
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111

/*Mandamos los intervalos a los servidores como se especificó en la tabla */
Worker A1 = new Worker(conexionA1, numero, numeroInicial: 2, k); // Instancia 1
Worker A2 = new Worker(conexionA2, numero, k + 1, 2 * k); // Instancia 2
Worker A3 = new Worker(conexionA3, numero, 2 * k + 1, numero - 1); // Instancia 3

/*Se inician los hilos y se utilizan barreras para esperar a que termine la ejecución del hilo actual y evitar errores en el
A1.start();
A1.join();
A2.start();
A2.join();
A3.start();
A3.join();

if (contador == 3) // Si los 3 servidores A devolvieron no divide, entonces tenemos un número primo
    salida.write("ES PRIMO".getBytes()); // Se manda al cliente que el número es primo
if (contador < 3)
    salida.write("NO ES PRIMO".getBytes()); // Se manda al cliente que el número no es primo
contador = 0; // reiniciamos el contador

System.out.println("\nEsperando conexion..."); // Volvemos a esperar otra conexion del cliente

/*Cerramos las conexiones */
conexionA1.close();
conexionA2.close();
conexionA3.close();
conexion.close();
}
```

Imagen 2

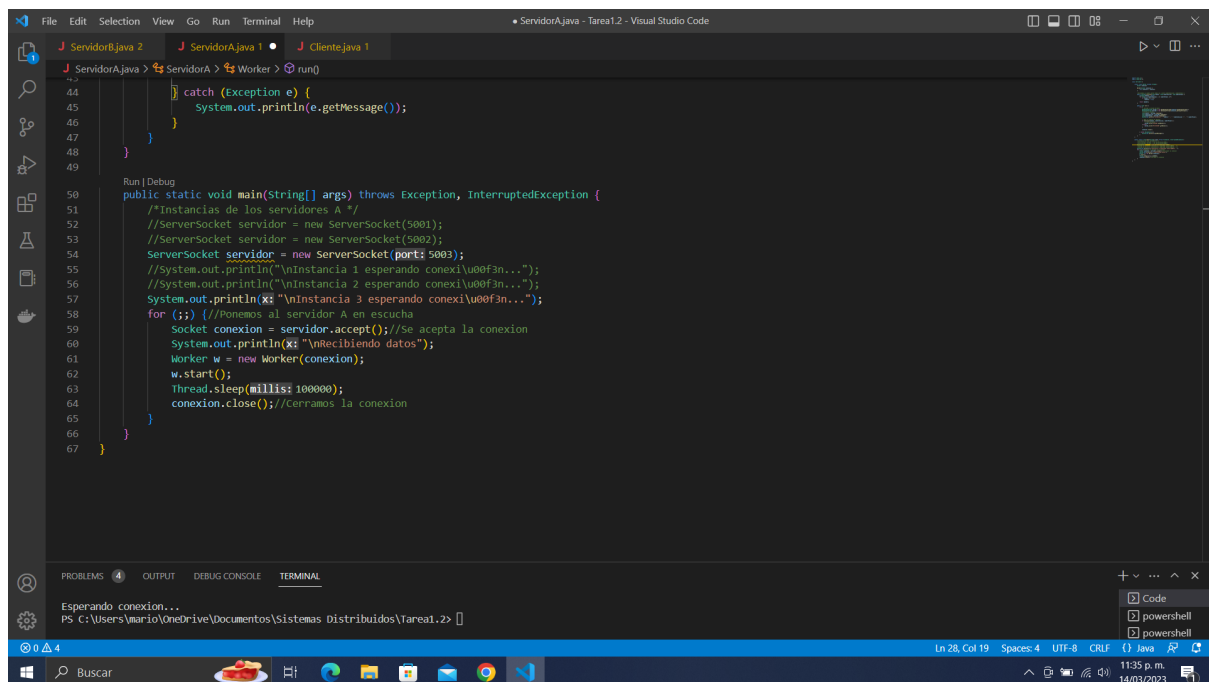
Posteriormente como se observa en la imagen utilizamos barreras para esperar a que se termine la ejecución del hilo actual para poder continuar con el siguiente y así evitar errores en el procesamiento.

Finalmente del servidor A (explicado a continuación) recibiremos Strings que dicen “DIVIDE” o “NO DIVIDE”, al final se contarán la cantidad de strings y si las 3 son no divide entonces tendremos un número primo, caso contrario será compuesto.

Para la siguiente parte se programó el servidor A, primero se creó la clase principal en donde asignamos el número de puerto del servidor, una vez lo tenemos ligado al puerto se pone el servidor A en escucha y una vez que se tenga una conexión por parte del servidor B, empezamos con la lectura de los datos utilizando los flujos de entrada y salida de datos, esto es vamos a leer los parámetros que mandamos en el servidor B que es el número a verificar, numeroInicial y numeroFinal.

Una vez tenemos guardados los datos en las variables número, numeroInicial, numeroFinal. Se programó una función llamada división, que su función es dividir el número proporcionado por el cliente entre otro número n que pertenece al intervalo entre el número inicial y final. Si al final del proceso, ningún número divide al número que proporcionó el cliente, la función devolverá una bandera booleana en estado de false y entonces mediante un if el servidor A devolverá al servidor B la cadena "NO DIVIDE", caso contrario la función devolverá la bandera en estado de true y el Servidor A devolverá la cadena "DIVIDE".

La siguiente imagen muestra el bloque de código del Servidor A:



```
File Edit Selection View Go Run Terminal Help
ServidorA.java - Tarea1.2 - Visual Studio Code

J ServidorA.java 2 J ServidorA.java 1 J Cliente.java 1
J ServidorA.java > J ServidorA > Worker > run()
44      catch (Exception e) {
45          System.out.println(e.getMessage());
46      }
47  }
48  }
49  }

Run | Debug
50 public static void main(String[] args) throws Exception, InterruptedException {
51     /*Instancias de los servidores A */
52     //ServerSocket servidor = new ServerSocket(5001);
53     //ServerSocket servidor = new ServerSocket(5002);
54     ServerSocket servidor = new ServerSocket(port; 5003);
55     //System.out.println("\nInstancia 1 esperando conexi\u00f3n...");
56     //System.out.println("\nInstancia 2 esperando conexi\u00f3n...");
57     System.out.println(x; "\nInstancia 3 esperando conexi\u00f3n...");
58     for (;;) { //Ponemos al servidor A en escucha
59         Socket conexion = servidor.accept(); //se acepta la conexion
60         System.out.println(x; "\nRecibiendo datos");
61         Worker w = new Worker(conexion);
62         w.start();
63         Thread.sleep(millis; 100000);
64         conexion.close(); //Cerramos la conexion
65     }
66 }
67 }
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

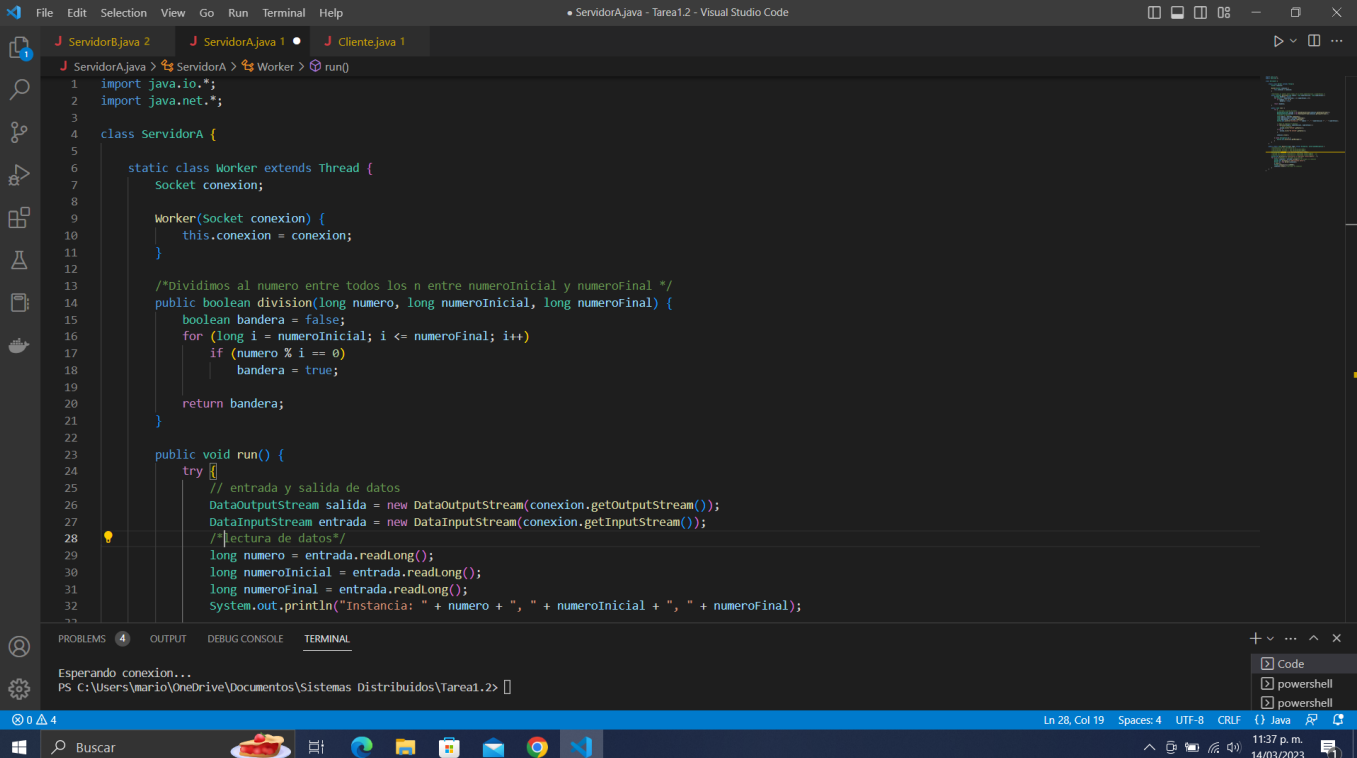
Esperando conexion...  
PS C:\Users\maria\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>

Ln 28, Col 19 Spaces: 4 UTF-8 CRLF Java

11:35 p.m. 14/03/2023

Imagen 3

La siguiente imagen muestra la función division:



```
1  import java.io.*;
2  import java.net.*;
3
4  class ServidorA {
5
6      static class Worker extends Thread {
7          Socket conexion;
8
9          Worker(Socket conexion) {
10             this.conexion = conexion;
11         }
12
13         /*Dividimos al numero entre todos los n entre numeroInicial y numeroFinal */
14         public boolean division(long numero, long numeroInicial, long numeroFinal) {
15             boolean bandera = false;
16             for (long i = numeroInicial; i <= numeroFinal; i++)
17                 if (numero % i == 0)
18                     bandera = true;
19
20             return bandera;
21         }
22
23         public void run() {
24             try {
25                 // entrada y salida de datos
26                 DataOutputStream salida = new DataOutputStream(conexion.getOutputStream());
27                 DataInputStream entrada = new DataInputStream(conexion.getInputStream());
28                 /*Lectura de datos*/
29                 long numero = entrada.readLong();
30                 long numeroInicial = entrada.readLong();
31                 long numeroFinal = entrada.readLong();
32                 System.out.println("Instancia: " + numero + ", " + numeroInicial + ", " + numeroFinal);
33             } catch (IOException e) {
34                 e.printStackTrace();
35             }
36         }
37     }
38 }
```

Imagen 4

Una vez se tienen las 3 respuestas de los servidores A, en el servidor B compara las cadenas, si empiezan con NO entonces significa que ningún número  $n$  dividió al número del cliente y con ayuda de un contador al final contamos si tenemos en total 3 casos en donde no se dividió, si se cumple el caso entonces tenemos un número primo, en caso contrario tenemos un número compuesto.

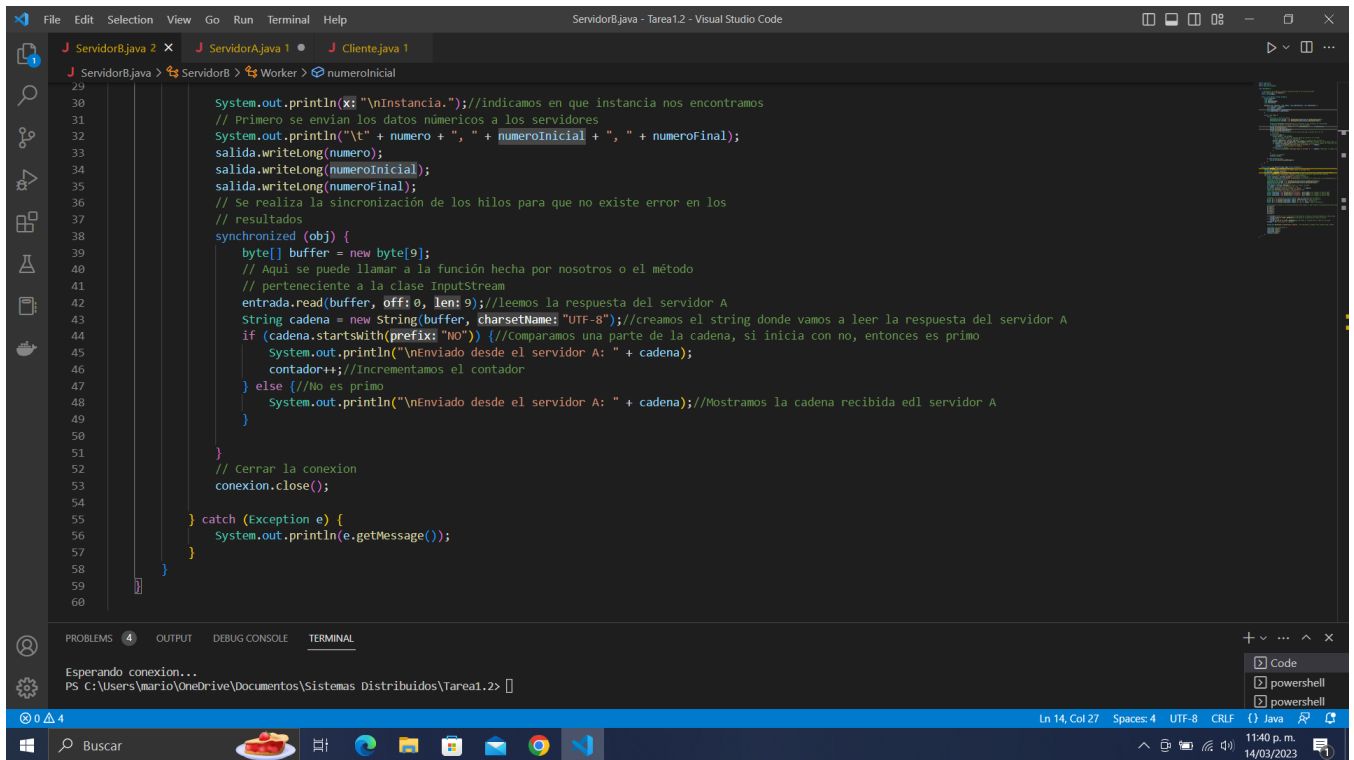


Imagen 5

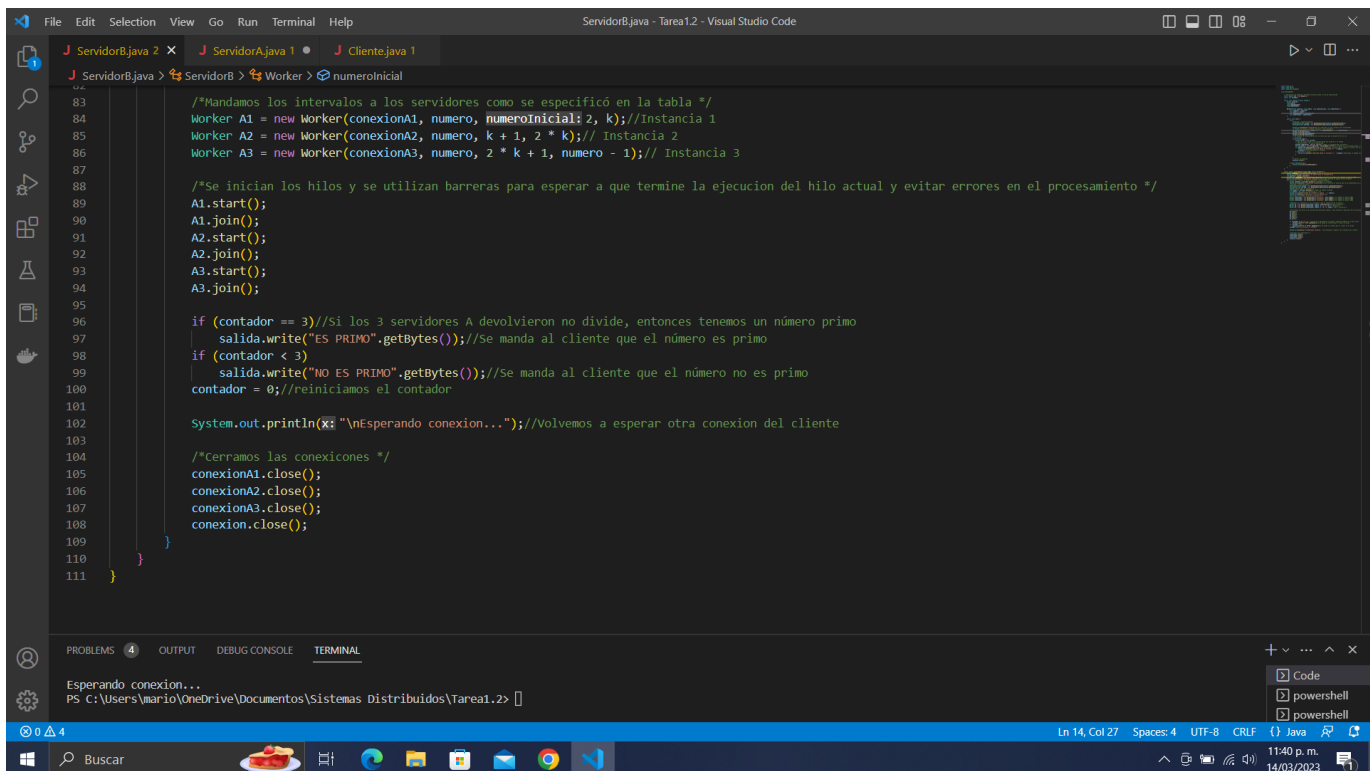


Imagen 6

# Resultados

Probando para el número de ejemplo en la tarea 1 igual a 1234567811 tenemos:

```
C:\Windows\system32\cmd.exe - java ServidorB
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive
C:\Users\mario\OneDrive>cd Documentos
C:\Users\mario\OneDrive\Documentos>cd Sistemas Distribuidos
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos>cd Tarea 1.2
El sistema no puede encontrar la ruta especificada.
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos>cd Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac ServidorB.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java ServidorB
Introduzca el puerto para el servidor B

C:\Windows\system32\cmd.exe - java ServidorA
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac ServidorA.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java ServidorA
Instancia 1 esperando conexión...

C:\Windows\system32\cmd.exe - java ServidorA
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac ServidorA.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java ServidorA
Instancia 2 esperando conexión...

C:\Windows\system32\cmd.exe - java Cliente
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac Cliente.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java Cliente
Instancia 3 esperando conexión...
```

Como se muestra en la imagen, tenemos al servidor B pidiendo el puerto, y las instancias de los servidores A esperando la conexión, posteriormente vamos a ligar el servidor B al puerto 5000 y a introducir el número 1234567811 en el cliente:

```
C:\Windows\system32\cmd.exe - java ServidorB
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive
C:\Users\mario\OneDrive>cd Documentos
C:\Users\mario\OneDrive\Documentos>cd Sistemas Distribuidos
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos>cd Tarea 1.2
El sistema no puede encontrar la ruta especificada.
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos>cd Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac ServidorB.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java ServidorB
Introduzca el puerto para el servidor B
5000
Esperando conexion
Conexion establecida

C:\Windows\system32\cmd.exe - java ServidorA
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac ServidorA.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java ServidorA
Instancia 1 esperando conexión...

C:\Windows\system32\cmd.exe - java ServidorA
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac ServidorA.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java ServidorA
Instancia 2 esperando conexión...

C:\Windows\system32\cmd.exe - java Cliente
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>javac Cliente.java
C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2>java Cliente
Introduzca un numero para verificar su primalidad:
1234567811
```

Como se muestra en la imagen, tenemos al servidor B pidiendo el puerto, y las instancias de los servidores A esperando la conexión, posteriormente vamos a ligar el servidor B al puerto 5000 y a introducir el número 1234567811 en el cliente



Posteriormente obtenemos como resultado:

```
C:\Windows\system32\cmd.exe - java ServidorB
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario>javac ServidorA.java
C:\Users\mario>java ServidorA

Instancia 1 esperando conexión...

Recibiendo datos
Instancia: 1234567811, 2, 411522603

C:\Windows\system32\cmd.exe - java ServidorA
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario>javac ServidorA.java
C:\Users\mario>java ServidorA

Instancia 3 esperando conexión...

Recibiendo datos
Instancia: 1234567811, 823045207, 1234567810

C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario>javac Cliente.java
C:\Users\mario>java Cliente
Introduzca un numero para verificar su primalidad:
1234567811
ES PRIMO
```

Como se observa en la terminal del cliente es un número primo

Ahora probando con el número 1,000,000,007 tenemos:

```
C:\Windows\system32\cmd.exe - java ServidorB
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario>javac ServidorA.java
C:\Users\mario>java ServidorA

Instancia 2 esperando conexión...

Recibiendo datos
Instancia: 1234567811, 411522604, 823045206

Recibiendo datos
Instancia: 1000000007, 333333336, 666666670

C:\Windows\system32\cmd.exe - java ServidorA
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario>javac ServidorA.java
C:\Users\mario>java ServidorA

Instancia 3 esperando conexión...

Recibiendo datos
Instancia: 1234567811, 823045207, 1234567810

C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mario>cd OneDrive\Documentos\Sistemas Distribuidos\Tarea1.2
C:\Users\mario>javac Cliente.java
C:\Users\mario>java Cliente
Introduzca un numero para verificar su primalidad:
1234567811
ES PRIMO

Introduzca un numero para verificar su primalidad:
1000000007
ES PRIMO
```

Como se observa en la terminal del cliente, también es primo.

## **Conclusiones**

Los sistemas distribuidos son una solución para superar las limitaciones de los sistemas centralizados, permiten escalar y aumentar el rendimiento del sistema de manera eficiente, tolerar fallos, mejorar la redundancia y la seguridad de los datos. Sin embargo, también presentan desafíos en cuanto a la complejidad del diseño, la coordinación entre dispositivos, la seguridad, el costo y la latencia. Para implementar un sistema distribuido exitoso, es necesario abordar estos desafíos de manera efectiva y comprender las limitaciones y beneficios de este enfoque. En la tarea 1 podemos ver que es muy útil al momento de procesar número muy grandes debido a que se balancea la carga en mas maquinas y no solo en una lo que nos permite realizar cálculos rápidamente