



Instituto Politécnico  
Nacional  
Escuela Superior de  
Cómputo



Alumno: Cisneros Gonzalez Miguel  
Angel

Grupo:4CV12

Tarea No.1

Obtener Numero Pi

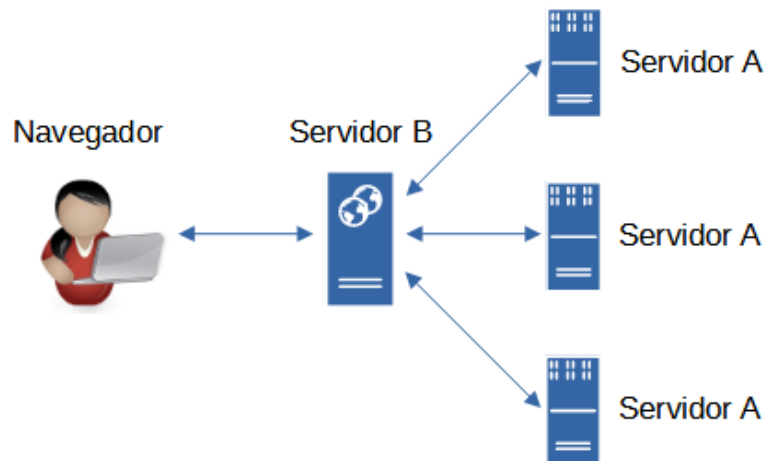
Materia: Desarrollo De Sistemas  
Distribuidos

Nombre del Profesor:  
Pineda Guerrero Carlos

En esta primera tarea el objetivo era obtener el numero Pi por medio de una aproximación de la siguiente función :

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103+26390k)}{(k!)^4 396^{4k}}$$

Por medio de esta estructura de sistema distribuido



Esto era requerido con tres maquinas virtuales dos de ellas independientes y una de ella que alojara el servidor A y el servidor B

El servidor A su tarea era :

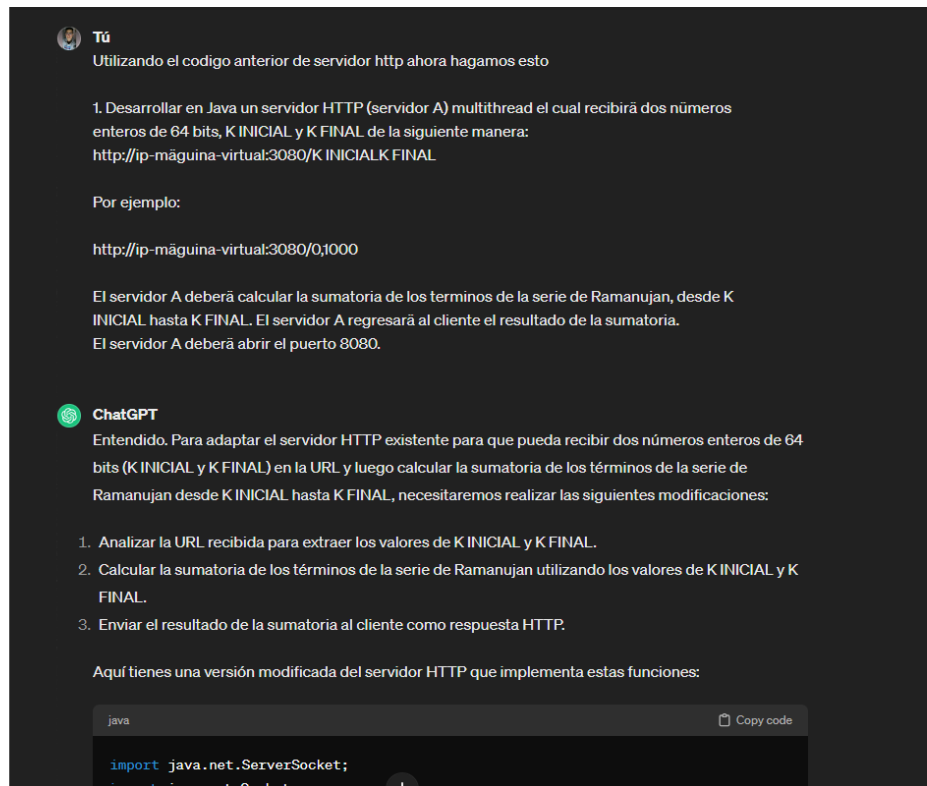
El servidor A debe calcular la sumatoria de los términos de la serie de Ramanujan, desde K INICIAL hasta K FINAL. El servidor A regresara al cliente el resultado de la sumatoria.

Mientras que el servidor B hace:

El servidor B dividirá el intervalo [0,3000] en tres intervalos, entonces el servidor B se conectara a tres instancias del servidor A. El servidor B enviara, a cada instancia del servidor A, el intervalo [K inicial ,K FINAL] correspondiente.

El servidor B recibirá la suma parcial obtenida por cada instancia del servidor A. El servidor B sumará las tres sumas parciales, multiplicara el resultado por  $2\sqrt{2}/9801$  y obtendrá el reciproco. Entonces el servidor B regresara al navegador el valor aproximado de PI asi obtenido.

Para esto utilizamos Chat GPT con los siguientes pront:



El cual nos arrojo este código :

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.math.BigDecimal;

class ServidorHTTP {
    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        public void run() {
            try {
                BufferedReader entrada = new BufferedReader(new
                InputStreamReader(conexion.getInputStream()));
                PrintWriter salida = new PrintWriter(conexion.getOutputStream());

                // Leer la solicitud del cliente
                String solicitud = entrada.readLine();
                System.out.println(solicitud);
```

```

// Extraer K INICIAL y K FINAL de la URL
String[] partes = solicitud.split("/");
if (partes.length < 2) {
    salida.println("Solicitud no válida");
    return;
}

String[] numeros = partes[1].split(",");
if (numeros.length != 2) {
    salida.println("Solicitud no válida");
    return;
}

long kInicial = Long.parseLong(numeros[0]);
long kFinal = Long.parseLong(numeros[1]);

// Calcular la sumatoria de la serie de Ramanujan
BigDecimal sumatoria = calcularSumatoriaRamanujan(kInicial, kFinal);

// Enviar el resultado al cliente
salida.println("Resultado de la sumatoria: " + sumatoria);
salida.flush();
} catch (Exception e) {
    System.err.println(e.getMessage());
} finally {
    try {
        conexion.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}
}

private BigDecimal calcularSumatoriaRamanujan(long kInicial, long kFinal) {
    BigDecimal sumatoria = BigDecimal.ZERO;
    for (long k = kInicial; k

```

Como vi que estaba mal realice otro pront:



Tú

y si ocupamos big decimal ya que long se desborda como quedaria



ChatGPT

Claro, podemos utilizar `BigDecimal` para manejar números enteros grandes sin preocuparnos por el desbordamiento. Aquí te dejo el código modificado para usar `BigDecimal` en lugar de `long`:

```
java Copy code

import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.math.BigDecimal;

class ServidorHTTP {
    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        public void run() {
            try {
                BufferedReader entrada = new BufferedReader(new InputStreamReader(conexion.getInputStream()));
                PrintWriter salida = new PrintWriter(conexion.getOutputStream());

                // Leer la solicitud del cliente
                String solicitud = entrada.readLine();
                System.out.println(solicitud);
            }
        }
    }
}
```

El cual obtuvimos esta respuesta :

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.math.BigDecimal;
```

```
class ServidorHTTP {
    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        public void run() {
            try {
                BufferedReader entrada = new BufferedReader(new
InputStreamReader(conexion.getInputStream()));
                PrintWriter salida = new PrintWriter(conexion.getOutputStream());

                // Leer la solicitud del cliente
```

```

String solicitud = entrada.readLine();
System.out.println(solicitud);

// Extraer K INICIAL y K FINAL de la URL
String[] partes = solicitud.split("/");
if (partes.length < 2) {
    salida.println("Solicitud no válida");
    return;
}

String[] numeros = partes[1].split(",");
if (numeros.length != 2) {
    salida.println("Solicitud no válida");
    return;
}

BigDecimal kInicial = new BigDecimal(numeros[0]);
BigDecimal kFinal = new BigDecimal(numeros[1]);

// Calcular la sumatoria de la serie de Ramanujan
BigDecimal sumatoria = calcularSumatoriaRamanujan(kInicial, kFinal);

// Enviar el resultado al cliente
salida.println("Resultado de la sumatoria: " + sumatoria);
salida.flush();
} catch (Exception e) {
    System.err.println(e.getMessage());
} finally {
    try {
        conexion.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}
}

private BigDecimal calcularSumatoriaRamanujan(BigDecimal kInicial, BigDecimal kFinal)
{
    BigDecimal sumatoria = BigDecimal.ZERO;
    for (BigDecimal k = kInicial; k.compareTo(kFinal) <= 0; k = k.add(BigDecimal.ONE)) {
        BigDecimal a = factorial(4 * k.intValueExact()).multiply(BigDecimal.valueOf(1103 +
26390 * k.intValueExact()));
        BigDecimal b =
factorial(k.intValueExact()).pow(4).multiply(BigDecimal.valueOf(396).pow(4 *
k.intValueExact()));
        BigDecimal termino = a.divide(b, 100, BigDecimal.ROUND_HALF_EVEN);
        sumatoria = sumatoria.add(termino);
    }
}

```

```

        BigDecimal sqrt2 = BigDecimal.valueOf(Math.sqrt(2));
        BigDecimal resultado =
BigDecimal.ONE.divide(sqrt2.multiply(BigDecimal.valueOf(9801)), 100,
BigDecimal.ROUND_HALF_EVEN).multiply(sumatoria);
        return BigDecimal.ONE.divide(resultado, 100,
BigDecimal.ROUND_HALF_EVEN).divide(BigDecimal.valueOf(4), 100,
BigDecimal.ROUND_HALF_EVEN);
    }

    private BigDecimal factorial(int n) {
        BigDecimal result = BigDecimal.ONE;
        for (int i = 2; i <= n; i++) {
            result = result.multiply(BigDecimal.valueOf(i));
        }
        return result;
    }
}

public static void main(String[] args) throws Exception {
    ServerSocket servidor = new ServerSocket(8080);

    for (;;) {
        Socket conexion = servidor.accept();
        new Worker(conexion).start();
    }
}
}

```

Después de esto y pelearme con chat gpt decidir seguir la tarea en copilot entonces use los estos pront:

 **Usted**

ya vi donde esta el error mira se supone que el servidor A deberá calcular la sumatoria de los terminos de la serie de Ramanujan, desde K INICIAL hasta K FINAL. El servidor A regresará al cliente el resultado de la sumatoria. y nosotros lo que hacemos es utilizar al final es esto

```

        BigDecimal result = BigDecimal.ONE.divide(sqrt2.multiply(BigDecimal.valueOf(9801)), 100, BigDecimal.ROUND_HALF_UP).multiply(suma);
        return BigDecimal.ONE.divide(result, 100, BigDecimal.ROUND_HALF_UP).divide(BigDecimal.valueOf(4), 100, BigDecimal.ROUND_HALF_UP);
    }
}

```

```

// Método para calcular el factorial
public static BigDecimal factorial(int n) {
    BigDecimal result = BigDecimal.ONE;
    for (int i = 2; i <= n; i++) {
        result = result.multiply(BigDecimal.valueOf(i));
    }
    return result;
}
}

```

} cosa que no debíamos de hacer si ves? esto solo se hace en el servidor B

| Para calcular una aproximación de PI, el servidor B dividirá el intervalo [0,3000] en tres intervalos, entonces el servidor B se conectará a tres instancias del servidor A. El servidor B enviara, a cada Instancia del servidor A, el intervalo [K inicial,K FINAL] correspondiente. ;

El servidor B recibirá la suma parcial obtenida por cada instancia del servidor A. El servidor B sumará las tres sumas parciales, multiplicará el resultado por  $2 \cdot \sqrt{2} / 9801$  y obtendrá el recíproco. Entonces el servidor B regresará al navegador el valor aproximado de PI así obtenido. ;



Y obtuve la siguiente respuesta:

Entiendo tu punto. Parece que el servidor A está haciendo más cálculos de los que debería. Según tu descripción, el servidor A solo debería calcular la sumatoria de los términos de la serie de Ramanujan desde `kInicial` hasta `kFinal` y devolver ese resultado. El cálculo final para obtener la aproximación de PI debería hacerse en el servidor B.

Aquí está el código modificado del servidor A que solo calcula la sumatoria de los términos de la serie de Ramanujan:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.math.BigDecimal;
import java.net.ServerSocket;
import java.net.Socket;

public class ServidorA2 {
    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        public void run() {
            try {
                BufferedReader entrada = new BufferedReader(new
InputStreamReader(conexion.getInputStream()));
                PrintWriter salida = new PrintWriter(conexion.getOutputStream());

                String[] partesNumeros = entrada.readLine().split("");
                int kInicial = Integer.parseInt(partesNumeros[0]);
                int kFinal = Integer.parseInt(partesNumeros[1]);

                // Calcular la suma de la serie de Ramanujan
                BigDecimal suma = calcularSumaRamanujan(kInicial, kFinal);

                // Enviar la suma al cliente
                salida.println(suma);
                salida.flush();
            } catch (Exception e) {
                System.err.println(e.getMessage());
            } finally {
                try {
                    conexion.close();
                } catch (Exception e) {
                    System.err.println(e.getMessage());
                }
            }
        }
    }
}
```



```

    }
}
}

```

```

public static void main(String[] args) throws Exception {
    // Puerto fijo para el servidor A
    int puerto = 8081;
    ServerSocket servidor = new ServerSocket(puerto);

    for (;;) {
        Socket conexion = servidor.accept();
        new Worker(conexion).start();
    }
}

// Método para calcular la suma de la serie de Ramanujan
public static BigDecimal calcularSumaRamanujan(int inicio, int fin) {
    BigDecimal suma = BigDecimal.ZERO;
    BigDecimal a, b;

    for (int k = inicio; k <= fin; k++) {
        a = factorial(4 * k).multiply(BigDecimal.valueOf(1103 + 26390 * k));
        b = factorial(k).pow(4).multiply(BigDecimal.valueOf(396).pow(4 * k));
        suma = suma.add(a.divide(b, 100, BigDecimal.ROUND_HALF_UP));
    }

    return suma;
}

// Método para calcular el factorial
public static BigDecimal factorial(int n) {
    BigDecimal result = BigDecimal.ONE;
    for (int i = 2; i <= n; i++) {
        result = result.multiply(BigDecimal.valueOf(i));
    }
    return result;
}
}

```

Para después solo terminar con el servidor B y probarlo localmente:

En este código modificado, el método `calcularSumaRamanujan` solo calcula la sumatoria de los términos de la serie de Ramanujan y devuelve ese resultado. El cálculo final para obtener la aproximación de PI se hará en el servidor B. Espero que esto sea lo que estabas buscando. Si tienes más preguntas, no dudes en preguntar. ¡Estoy aquí para ayudar! 😊 user ahora dame el código del servidor B modificado completo assistant' Por supuesto, aquí está el código completo del servidor B modificado. Los valores de las iteraciones inicial y final se determinan en el bucle `for` dentro del método `run` de la clase `Worker`.

Me dio este código:

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;

```

```

import java.math.BigDecimal;
import java.net.ServerSocket;
import java.net.Socket;

public class ServidorB {
    public static void main(String[] args) throws Exception {
        // Puerto fijo para el servidor B
        int puerto = 80;
        ServerSocket servidor = new ServerSocket(puerto);

        for (;;) {
            Socket conexion = servidor.accept();
            new Worker(conexion).start();
        }
    }

    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        public void run() {
            try {
                BufferedReader entrada = new BufferedReader(new
InputStreamReader(conexion.getInputStream()));
                PrintWriter salida = new PrintWriter(conexion.getOutputStream());

                String req = entrada.readLine();
                System.out.println(req);

                for (;;) {
                    String encabezado = entrada.readLine();
                    System.out.println(encabezado);
                    if (encabezado.equals("")) {
                        break;
                    }
                }
            }

            // Dividir el intervalo [0,3000] en tres intervalos
            int intervalo = 3000 / 3;
            BigDecimal sumaTotal = BigDecimal.ZERO;

            // Direcciones IP y puertos de las máquinas virtuales con los servidores A
            String[] ipsServidoresA = {"localhost", "localhost", "localhost"};
            int[] puertosServidoresA = {8080, 8081, 8082};

```

```

// Conectar a las tres instancias del servidor A
for (int i = 0; i < 3; i++) {
    int kInicial = i * intervalo;
    int kFinal = (i + 1) * intervalo - 1;

    // Enviar el intervalo [K inicial,K FINAL] al servidor A
    Socket socketA = new Socket(ipsServidoresA[i], puertosServidoresA[i]);
    PrintWriter salidaA = new PrintWriter(socketA.getOutputStream());
    BufferedReader entradaA = new BufferedReader(new
InputStreamReader(socketA.getInputStream()));

    salidaA.println(kInicial + "," + kFinal);
    salidaA.flush();

    // Recibir la suma parcial del servidor A
    BigDecimal sumaParcial = new BigDecimal(entradaA.readLine());
    sumaTotal = sumaTotal.add(sumaParcial);

    socketA.close();
}

// Calcular la aproximación de PI
BigDecimal pi =
BigDecimal.ONE.divide(sumaTotal.multiply(BigDecimal.valueOf(2)).multiply(BigDecimal.valueOf(Math.sqrt(2))).divide(BigDecimal.valueOf(9801), 100, BigDecimal.ROUND_HALF_UP)),
100, BigDecimal.ROUND_HALF_UP);

// Enviar la aproximación de PI al cliente
salida.println("HTTP/1.1 200 OK");
salida.println("Content-type: text/plain");
salida.println();
salida.println(pi);
salida.flush();
} catch (Exception e) {
    System.err.println(e.getMessage());
} finally {
    try {
        conexion.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}
}
}
}
}

```

Después de obtener los códigos y verificarlos que hicieran bien su trabajo es decir probarlos localmente procedí a realizar el mismo ejercicio pero en Azure de la siguiente manera:

### Primero cree las maquinas virtuales 3:

Crear una máquina virtual - Mi...Curso Desarrollo de Sistemas...SIS05-4CV12: Tarea 1. Sistem...SIS05-4CV12: Tarea 1. Cre...

portal.azure.com/#create/MicrosoftVirtualMachine-AIIM

Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción \*

Azure for Students

Grupo de recursos \*

(Nuevo) Tarea1

Crear nuevo

Detalles de instancia

Nombre de máquina virtual \*

T1-201863137195-1

Región \*

(US) East US 2

Opciones de disponibilidad

Zona de disponibilidad

Zona de disponibilidad \*

Zonas 2

Ahora puede seleccionar varias zonas. Si selecciona varias zonas, se creará una VM por zona. Más información >

Tipo de seguridad

Estandar

Imagen \*

Ubuntu Server 20.04 LTS - x64 gen. 2

Ver todas las imágenes | Configurar la generación de máquinas virtuales

Esta imagen es compatible con características de seguridad adicionales. Haga clic aquí para cambiar a la versión de inicio seguro.

Arquitectura de VM

Arm64

☒ x64

Ejecución de Azure Spot con descuento

☐

Anterior

Siguiente: Discos >

Revisar y crear

Enviar comentarios

Crear una máquina virtual - Mi...Curso Desarrollo de Sistemas...SIS05-4CV12: Tarea 1. Sistem...SIS05-4CV12: Práctica 1. Cre...

portal.azure.com/#create/MicrosoftVirtualMachine-AIIM

Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Datos básicos

Discos

Redes

Administración

Supervisión

Opciones avanzadas

Etiquetas

Revisar y crear

Creé una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Azure Marketplace o use una imagen personalizada propia. Complete la pestaña Conceptos básicos y, después, use Revisar y crear para aprovisionar una máquina virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración. Más información >

Es posible que esta suscripción no sea apta para implementar máquinas virtuales de ciertos tamaños en determinadas regiones.

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción \*

Azure for Students

Grupo de recursos \*

Tarea1

Crear nuevo

Detalles de instancia

Nombre de máquina virtual \*

T1-201863137195-2

Región \*

(US) East US 2

Opciones de disponibilidad

Zona de disponibilidad

Zona de disponibilidad \*

Zonas 2

Ahora puede seleccionar varias zonas. Si selecciona varias zonas, se creará una VM por zona. Más información >

Tipo de seguridad

Estandar

Imagen \*

Ubuntu Server 20.04 LTS - x64 gen. 2

Ver todas las imágenes | Configurar la generación de máquinas virtuales

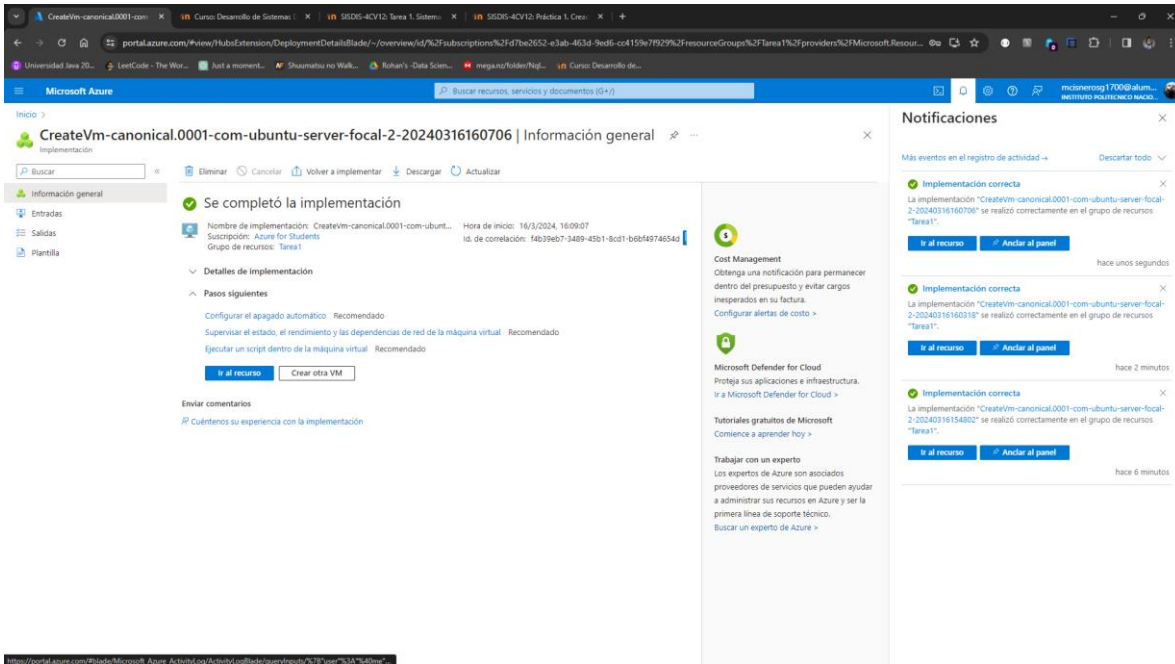
Anterior

Siguiente: Discos >

Revisar y crear

Enviar comentarios

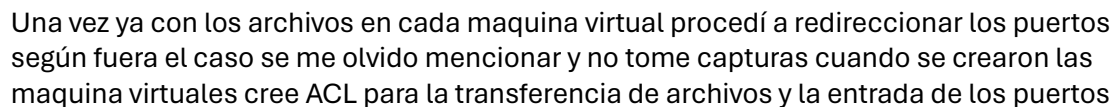
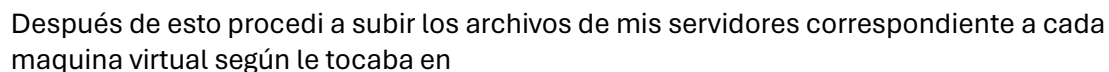
Aquí verifique que las tres maquinas virtuales estuvieran creadas correctamente en la pantalla de notificaciones



Después lo que hice fue conectarme por ssh a cada una de las maquinas e instalar el jdk de java primero utilizando sudo apt update

```
miky@T1-201863137195-1: ~  
See "man sudo_root" for details.  
  
miky@T1-201863137195-1:~$ sudo apt update  
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease  
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]  
Get:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]  
Get:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]  
Get:5 http://azure.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]  
Get:6 http://azure.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]  
Get:7 http://azure.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]  
Get:8 http://azure.archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]  
Get:9 http://azure.archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]  
Get:10 http://azure.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]  
Get:11 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3156 kB]  
Get:12 http://azure.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [504 kB]  
Get:13 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [2755 kB]  
Get:14 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [385 kB]  
Get:15 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1175 kB]  
Get:16 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [282 kB]  
Get:17 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [25.7 kB]  
Get:18 http://azure.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [26.1 kB]  
Get:19 http://azure.archive.ubuntu.com/ubuntu focal-updates/multiverse Translation-en [7768 B]  
Get:20 http://azure.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 c-n-f Metadata [620 B]  
Get:21 http://azure.archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [45.7 kB]  
Get:22 http://azure.archive.ubuntu.com/ubuntu focal-backports/main Translation-en [16.3 kB]  
Get:23 http://azure.archive.ubuntu.com/ubuntu focal-backports/main amd64 c-n-f Metadata [1420 B]  
Get:24 http://azure.archive.ubuntu.com/ubuntu focal-backports/restricted amd64 c-n-f Metadata [116 B]  
Get:25 http://azure.archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [25.0 kB]  
Get:26 http://azure.archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [16.3 kB]
```

Sudo apt install openjdk-8-jdk esto es cada una de las 3 maquinas virtuales





```
miky@T1-20186317195:~$ curl -v http://74.209.93.176/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
Sec-Purpose: prefetch;prerender
Purpose: prefetch
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;vmb3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: es-419,es-US;q=0.9,en;q=0.8,en;q=0.7

GET /pi HTTP/1.1
Host: 74.209.93.176
User-Agent: curl/8.4.0
Accept: */*

GET /favicon.ico HTTP/1.1
Host: 74.209.93.176
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://74.209.93.176/pi
Accept-Encoding: gzip, deflate
Accept-Language: es-419,es-US;q=0.9,en;q=0.8,en;q=0.7

miky@T1-20186317195:~$

update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole
(jconsole) in auto mode
Setting up libgail-common:amd64 (2.24.32-4ubuntu1) ...
Setting up ubuntu-mono (19.04-4ubuntu3) ...
Processing triggers for systemd (245.4-4ubuntu3.23) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed, done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3ubuntu0.4) ...
miky@T1-20186317195-1:~$ mkdir servidorA
miky@T1-20186317195-1:~$ cd servidorA/
miky@T1-20186317195-1:~/servidorA$ nano ServidorA.java
miky@T1-20186317195-1:~/servidorA$ javac ServidorA.java
miky@T1-20186317195-1:~/servidorA$ java ServidorA
*Only@T1-20186317195-1:~/servidorA$ java ServidorA
*Only@T1-20186317195-1:~/servidorA$ java ServidorA
miky@T1-20186317195-1:~/servidorA$ java ServidorA

miky@T1-20186317195:~$

done.
done.
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3ubuntu0.4) ...
miky@T1-20186317195-1:~$ nano java ServidorA3.java
miky@T1-20186317195-1:~$ cd ServidorA3/
miky@T1-20186317195-1:~/ServidorA3$ nano ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ javac ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ java ServidorA3
*Only@T1-20186317195-1:~/ServidorA3$ sudo iptables -A PREROUTING -t nat -p tcp --dport 80 -j REDIRECT --to-port 8081
miky@T1-20186317195-1:~/ServidorA3$ javac ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ java ServidorA3

Use "fg" to return to nano.
[1] Stopped nano java ServidorA3.java
miky@T1-20186317195-1:~$ nano java ServidorA3.java
miky@T1-20186317195-1:~$ cd ServidorA3/
miky@T1-20186317195-1:~/ServidorA3$ nano ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ javac ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ java ServidorA3
*Only@T1-20186317195-1:~/ServidorA3$ sudo iptables -A PREROUTING -t nat -p tcp --dport 80 -j REDIRECT --to-port 8082
miky@T1-20186317195-1:~/ServidorA3$ javac ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ java ServidorA3
```

Ya con los puertos redirigidos pase a compilar y ejecutar cada servidor

```
miky@T1-20186317195:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
games:x:4:65534:games:/usr/games:/usr/sbin/nologin
miky:x:1000:1000:miky:/home/miky:/bin/bash

System load: 0.01 Processes: 106
Usage of /: 7.4% of 28.89GB Users logged in: 1
Memory usage: 35% IPv4 address for eth0: 10.0.0.4
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

20 updates can be applied immediately.
18 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Mar 16 22:22:51 2024 from 189.217.75.16
miky@T1-20186317195-1:~$ cd ServidorB/
miky@T1-20186317195-1:~/ServidorB$ nano ServidorB.java
miky@T1-20186317195-1:~/ServidorB$ javac ServidorB.java
miky@T1-20186317195-1:~/ServidorB$ java ServidorB

miky@T1-20186317195:~$

Setting up libgtk2.0-bin (2.24.32-4ubuntu1) ...
Setting up openjdk-8-jdk:amd64 (8u392-ga-1-20.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/appletviewer to provide /usr/bin/appletviewer (appletviewer) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Setting up libgail-common:amd64 (2.24.32-4ubuntu1) ...
Setting up ubuntu-mono (19.04-4ubuntu3) ...
Processing triggers for systemd (245.4-4ubuntu3.23) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed, done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3ubuntu0.4) ...
miky@T1-20186317195-1:~$ mkdir ServidorA2
miky@T1-20186317195-1:~$ cd ServidorA2/
miky@T1-20186317195-1:~/ServidorA2$ nano ServidorA2.java
miky@T1-20186317195-1:~/ServidorA2$ javac ServidorA2.java
miky@T1-20186317195-1:~/ServidorA2$ java ServidorA2

miky@T1-20186317195:~$

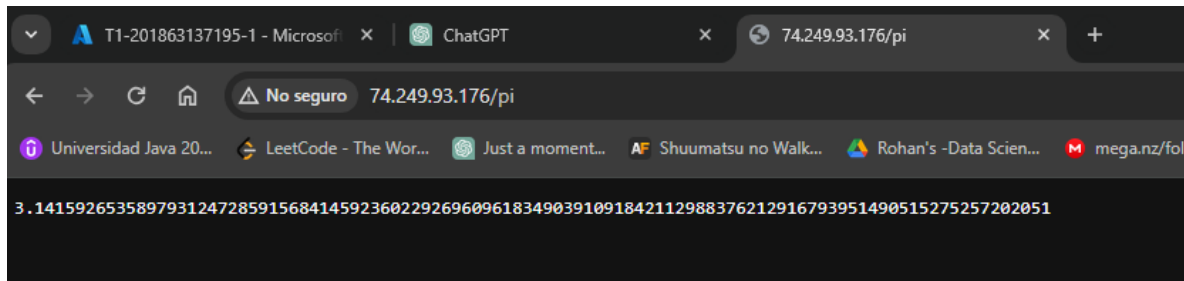
Setting up libgtk2.0-bin (2.24.32-4ubuntu1) ...
Setting up openjdk-8-jdk:amd64 (8u392-ga-1-20.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/appletviewer to provide /usr/bin/appletviewer (appletviewer) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Setting up libgail-common:amd64 (2.24.32-4ubuntu1) ...
Setting up ubuntu-mono (19.04-4ubuntu3) ...
Processing triggers for systemd (245.4-4ubuntu3.23) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed, done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3ubuntu0.4) ...
miky@T1-20186317195-1:~$ mkdir ServidorA3
miky@T1-20186317195-1:~$ nano java ServidorA3.java
miky@T1-20186317195-1:~$ cd ServidorA3/
miky@T1-20186317195-1:~/ServidorA3$ nano ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ javac ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ java ServidorA3

Use "fg" to return to nano.
[1]+ Stopped nano java ServidorA3.java
miky@T1-20186317195-1:~$ nano java ServidorA3.java
miky@T1-20186317195-1:~$ cd ServidorA3/
miky@T1-20186317195-1:~/ServidorA3$ nano ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ javac ServidorA3.java
miky@T1-20186317195-1:~/ServidorA3$ java ServidorA3
```

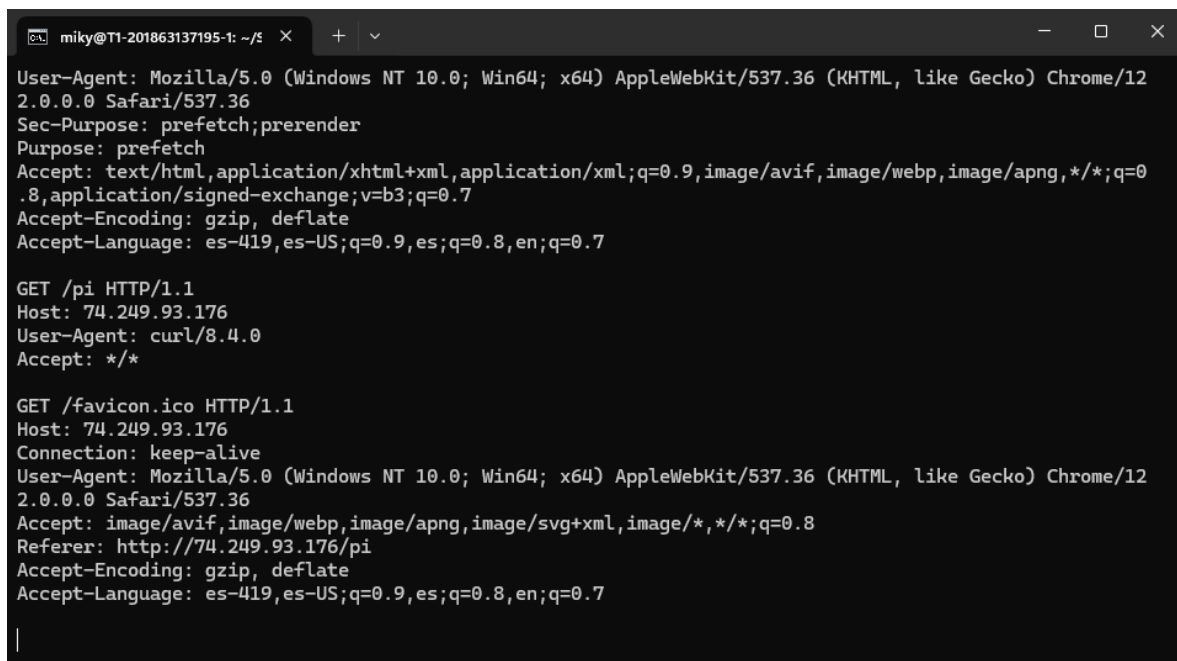
Ahora solo quedo probar que funciona en el navegador emitiendo la petición

Dirección ip de la maquina virtual 1/pi

Este fue mi resultado y si vemos la ventana de la maquina 1 del servidor b nos apareció esto



Este fue lo que aparecía en el servidor B después de realizar el resultado



Conclusión:

La tarea 1 se me hizo un poco complicada ya que no tenia muchos conocimientos de la nube y de todo lo de servidores pero me sirvió demasiado para entender prácticamente todas las clases que hasta el momento vimos ahora entiendo como es que se suben y se maneja mejor la nube y como es que se desarrolla un pequeño sistema distribuido es fascinante que esto puede crecer exponencialmente es decir muchísimo y es fascinante ver como puedes probar y hacer cosas ya no desde lo local.

\*\*\*\*\*Como nota debido a un error que me manda por que el servidor según tenia el mismo puerto y no me acepta el 80 lo que hice fue hacer la conexión mediante el puerto 1025 y redireccionar el puerto 80 al 1025 subo el archivo sin modificar del servidorB pero eso fue lo que hice para que jalara y funcionara\*\*\*