Instituto Politécnico Nacional

Escuela Superior de Cómputo

**Sistemas Distribuidos**

**PRACTICA 2:**

**Un cliente/Un servidor**

Alumno:

|  |  |
| --- | --- |
| Meza Díaz Arturo Rodrigo | 2022630251 |

Grupo: 7CM1

Maestro: Carreto Arellano Chadwick

Fecha de realización: 3 de marzo de 2025

Fecha de entrega: 3 de marzo de 2025

Índice.

[1. Antecedentes. 3](#_Toc190811586)

[2. Planteamiento del problema. 3](#_Toc190811587)

[3. Propuesta de solución. 3](#_Toc190811588)

[4. Materiales y métodos. 3](#_Toc190811589)

[5. Desarrollo de solución. 4](#_Toc190811590)

[6. Resultados. 5](#_Toc190811591)

[7. Conclusiones. 6](#_Toc190811592)

# Antecedentes.

En un entorno de gimnasio, es común que los clientes registren sus rutinas de entrenamiento en un sistema central. Este sistema debe ser capaz de manejar las solicitudes de los clientes de manera secuencial, sin necesidad de concurrencia. Es posible implementar un sistema cliente-servidor simple que procese las solicitudes una por una.

Java proporciona herramientas para la comunicación entre cliente y servidor a través de sockets, lo que permite implementar un sistema secuencial sin la complejidad de los hilos.

# Planteamiento del problema.

En un gimnasio, los clientes envían sus rutinas de entrenamiento (ejercicio, series, repeticiones y peso) a un servidor central que las registra. Sin un sistema adecuado, la gestión de estas rutinas puede volverse ineficiente, especialmente si no se requiere concurrencia.

Se requiere implementar un sistema que permita a los clientes enviar sus rutinas de entrenamiento de manera secuencial, asegurando que cada una sea registrada correctamente y sin interferencias.

# Propuesta de solución.

Para resolver este problema, se implementará un sistema de gestión de gimnasio basado en un modelo cliente-servidor aislado. El servidor recibirá las rutinas de entrenamiento de los clientes de manera secuencial y las registrará en una lista. Así mismo, cada cliente se conectará al servidor, enviará su rutina y recibirá una confirmación.

# Materiales y métodos.

Lenguaje:

* Java.

Librerías utilizadas:

* **java.net.Socket**: Para la comunicación entre cliente y servidor.
* **java.io.BufferedReader** y **java.io.PrintWriter**: Para la entrada y salida de datos.
* **java.util.List**: Para almacenar las rutinas de entrenamiento.

Métodos implementados:

* **enviarRutina(String rutina)**: Método del cliente para enviar la rutina al servidor.
* **recibirRutina()**: Método del servidor para recibir y registrar las rutinas.

# Desarrollo de solución.

El código desarrollado implementa un servidor que recibe rutinas de entrenamiento de los clientes de manera secuencial. Cada cliente se conecta al servidor, envía su rutina y recibe una confirmación. Por último, el servidor registra las rutinas en una lista y las imprime en la consola.

**ENLACE DE REPOSITORIO:**

Codigo del Servidor:



Codigo del Cliente:

A screen shot of a computer program

AI-generated content may be incorrect.

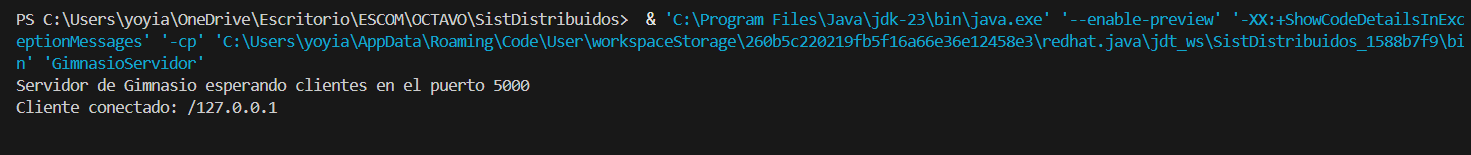
# Resultados.

Para verificar el correcto funcionamiento del código, se ejecutó el programa y se observaron los siguientes resultados mostrados a continuación:

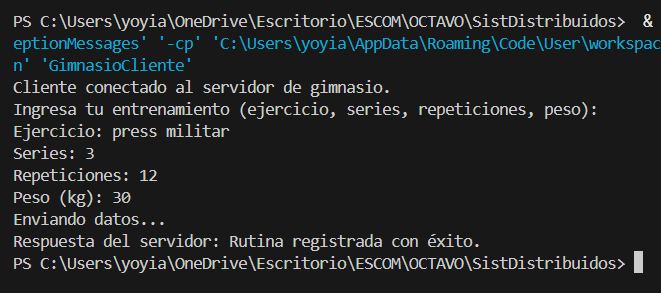
1. Cada cliente envió su rutina de entrenamiento al servidor.
2. El servidor recibió y registró las rutinas en una lista.
3. Cada rutina fue procesada de manera secuencial.
4. El servidor confirmó la recepción de cada rutina.

La salida en consola es:

**Salida del Servidor:**

****

**Salida del Cliente:**



**Salida final del Servidor:**

**A screen shot of a computer

AI-generated content may be incorrect.**

Esto demuestra que el cliente y el servidor procesan y confirman correctamente la o las rutinas.

# Conclusiones.

Se logró implementar correctamente un sistema de gestión de gimnasio utilizando un cliente y un servidor. La solución permitió gestionar las rutinas de entrenamiento de manera secuencial, registrando cada una sin interferencias. Al no utilizar hilos, y manejar un único cliente y servidor, el sistema es más simple y adecuado para entornos donde no se requiere concurrencia.

Este modelo puede extenderse en futuras implementaciones para incluir características avanzadas, como la persistencia de datos en una base de datos o la gestión de varios gimnasios.