Taller de Programación

Guía 7

Estudiosos:

Juan David Rincon Muñoz

Julian Eduardo Lozano Rios

Universidad Manuela Beltran

Ingeniería de software

Olga Lucia Roa

11 de noviembre de 2024

Sesión 1

1. What is Distributed Computing?

Distributed computing is a model where computing resources are spread across multiple systems, often over a network, to work together toward a common goal. In this setup, tasks are divided into smaller parts and allocated to different computers (nodes), which communicate and collaborate to solve problems or process data. This approach enables systems to share resources and handle larger or more complex tasks than would be possible with a single system.

2. What are Sockets Used For?

Sockets are endpoints for communication between two machines over a network. They provide a way for applications to communicate, either within the same machine or across different machines. A socket connection involves an IP address and port number, where data can be sent or received. There are two main types:

TCP sockets: Offer reliable, connection-oriented communication.
UDP sockets: Provide a faster, connectionless communication but without guaranteed delivery.

3. What is the Difference Between UDP and TCP?

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are both protocols for sending data over the internet, but they operate differently:

TCP:

Reliable, connection-oriented protocol.

Ensures data packets are delivered in order and without errors.

Used for applications where accuracy is more important than speed (e.g., web browsing, file transfer).

UDP:

Faster, connectionless protocol.

Does not guarantee order or delivery.

Suitable for applications where speed is essential, and some data loss is acceptable (e.g., live video streaming, gaming).

4. What is RMI and JNDI? And How Are They Related to Sockets?

RMI (Remote Method Invocation): A Java API that allows objects on different JVMs (Java Virtual Machines) to communicate and invoke each other's methods. It enables distributed object communication, with Java handling networking details through sockets. Essentially, RMI abstracts socket-based communication for remote procedure calls.

JNDI (Java Naming and Directory Interface): A Java API that provides a unified way to access different directory and naming services (e.g., LDAP). It allows applications to look up resources and services by name, often used in distributed systems to locate remote objects (like those used in RMI).

Relation to Sockets: RMI relies on sockets for the underlying network communication, although it abstracts these details from the developer. JNDI, on the other hand, doesn't directly rely on sockets but helps manage distributed resources in applications that may use sockets for networking.

5. What is a Web Service?

A web service is a standardized way for applications to communicate over the internet using HTTP/HTTPS. It allows different applications, often written in different languages or running on different systems, to interact. Web services typically use XML (SOAP) or JSON (REST) to structure the exchanged data, making them platform-independent. This enables seamless communication and data exchange, supporting integration between systems in distributed computing.

Preguntas Orientadoras.

1. Aprendizajes obtenidos y relación con el futuro profesional

Conocimiento de Redes y Protocolos de Comunicación: Al desarrollar un chat en línea con sockets, comprendimos cómo funcionan los protocolos TCP y UDP y su aplicación en diferentes escenarios de comunicación. Este conocimiento es clave en el campo profesional, especialmente en áreas como la ciberseguridad, la administración de redes y el desarrollo de aplicaciones distribuidas.

Programación Concurrente y Gestión de Conexiones: Trabajar con sockets para desarrollar un chat implica gestionar múltiples conexiones de usuarios de forma simultánea, lo que requiere el uso de hilos o procesos en paralelo. Esta habilidad es crucial en el desarrollo de sistemas que deben manejar múltiples usuarios o dispositivos, como aplicaciones de comercio electrónico, sistemas bancarios o servicios en la nube.

Resolución de Problemas y Debugging en Tiempo Real: Durante el desarrollo del chat en línea, surgieron problemas como la pérdida de mensajes o conexiones interrumpidas, lo que nos llevó a mejorar nuestras habilidades de debugging y solución de problemas en tiempo real. Estas competencias son vitales en el ámbito profesional, especialmente en situaciones de producción donde resolver errores rápidamente es fundamental para mantener la disponibilidad del sistema.

2. Dificultades y estrategias de solución

Dificultad: La mayor dificultad fue lograr una comunicación fluida y confiable entre los clientes y el servidor, especialmente en lo que respecta al manejo de múltiples conexiones de manera estable.

Estrategias de Solución:

Implementación de Hilos: Usamos programación multihilo para permitir que el servidor maneje múltiples usuarios de manera simultánea sin bloquear las conexiones entrantes.

Pruebas de Conexión y Recuperación de Errores: Realizamos pruebas de conexión constantes y añadimos un mecanismo para manejar reconexiones en caso de caídas temporales de los clientes, para minimizar la pérdida de mensajes.

Uso de Logs y Consola para Debugging: Implementamos mensajes de log para rastrear el flujo de datos y detectar en qué momento se producían errores. Esto facilitó identificar cuellos de botella y optimizar la velocidad de respuesta del servidor.

Este proyecto no solo permitió comprender conceptos técnicos, sino que también fomenta habilidades de trabajo en equipo, ya que todos colaboramos en la solución de los problemas y en la revisión del código, lo cual es muy valioso para el trabajo en entornos de desarrollo profesional.

github:

En este repositorio se encuentra la guia:

https://github.com/juandatiner/Guias-Taller-de-Programacion

youtube:

https://youtu.be/NOsF1WLgR4s?si=LJ9k_TWE11CqvfWK