TALLER DE PROGRAMACIÓN

GUÍA 7

MIGUEL ANGEL ZAMBRANO

HUGO ANDRES FORERO

SAMUEL ANTONIO TARAZONA

TALLER DE PROGRAMACIÓN

UNIVERSIDAD MANUELA BELTRAN

2024

Actividad de Trabajo Autónomo

1. **What is distributed computing?**
    - Distributed computing is a model where multiple computers work together to solve a problem or perform tasks. These computers can be in different locations but are connected through a network, allowing them to share resources, data, and processing power. Distributed computing enables systems to function as a single entity, which improves performance, scalability, and fault tolerance.
2. **What are Sockets used for?**
    - Sockets provide a way to establish a communication link between two devices over a network. In Java, sockets allow applications to send and receive data across the network. By creating sockets on both the client and server sides, two applications can connect and exchange information, such as in chat applications or networked games.
3. **What is the difference between UDP and TCP?**
    - **TCP (Transmission Control Protocol)**: TCP is a connection-oriented protocol, meaning it establishes a reliable connection between the sender and receiver before data transmission. TCP ensures data packets are delivered in order and without errors by managing flow control, acknowledgments, and retransmissions if needed.
    - **UDP (User Datagram Protocol)**: UDP is a connectionless protocol, which does not establish a reliable connection. It sends data packets without checking if they are received or delivered in the correct order. UDP is faster and more efficient than TCP but lacks reliability, making it suitable for applications where speed is crucial, such as streaming or online gaming.
4. **What is RMI and JNDI? And how are they related to Sockets?**
    - **RMI (Remote Method Invocation)**: RMI is a Java API that enables objects on different JVMs (Java Virtual Machines) to invoke each other's methods. It allows a Java application to interact with objects over a network, as if they were on the same machine, by using remote references.
    - **JNDI (Java Naming and Directory Interface)**: JNDI provides a directory service to Java applications, allowing them to locate resources and objects by name, such as databases or EJBs (Enterprise JavaBeans). JNDI is commonly used in distributed applications to find and access resources.
    - Both RMI and JNDI rely on sockets for network communication. RMI uses sockets to establish connections between remote objects, while JNDI may use sockets indirectly to connect to naming or directory services over a network.
5. **What is a Web Service?**
    - A Web Service is a standardized way for applications to communicate over the internet, often using HTTP. Web services enable applications built on different platforms to interact, exchange data, and perform operations using common protocols like XML, SOAP, or REST. They are widely used for integrating systems across organizational boundaries and are accessible via the web regardless of the platform or language used.

Actividad de Trabajo Autónomo (Video)

Aprendizajes obtenidos:

- **Conexión y comunicación entre procesos:** Aprendí cómo establecer conexiones entre procesos independientes usando sockets, un conocimiento útil para crear aplicaciones de red en el futuro, como sistemas cliente-servidor o aplicaciones de mensajería.
- **Manejo de puertos y direcciones IP:** Comprendí la importancia de las direcciones IP y los puertos para identificar puntos finales en la comunicación. Esto será esencial en mi carrera para implementar conexiones seguras y bien configuradas en redes distribuidas.
- **Diferencia entre sockets activos y pasivos:** Entendí cómo se diferencian los sockets según el rol del servidor o del cliente, lo que ayuda en la configuración de sistemas de red, permitiendo desarrollar aplicaciones donde el servidor pueda gestionar múltiples conexiones.

Dificultad principal:

- **Problema:** La mayor dificultad fue configurar correctamente los sockets en el servidor y el cliente para lograr una comunicación efectiva, especialmente en el manejo de excepciones y errores de conexión.
- **Solución:** La solución fue investigar más sobre el manejo de excepciones en Java y probar configuraciones adicionales. También me apoyé en ejemplos de código y documentación para entender mejor la estructura y el flujo de los sockets.

Estrategias de solución:

- Realizar pruebas incrementales: Fui ejecutando pruebas paso a paso para identificar errores de configuración en el proceso de conexión.
- Consultar documentación adicional: Revisé la documentación de Java y recursos externos para aclarar conceptos de la implementación de sockets.
- Pedir ayuda a compañeros o al instructor: Conversar sobre los problemas con otros me dio perspectivas nuevas y consejos prácticos para resolver los errores.

Código Socket:

https://drive.google.com/drive/u/0/folders/1ZHlmFp7d1xCXJ5Jf2HCSE0a75RSiGBrb