**ARQUITECTURA CLIENTE-SERVIDOR, SOCKETS, TIPOS DE SOCKETS**

**TEMA 03 – SOCKETS**

La arquitectura cliente-servidor es un modelo de computación que divide las tareas en dos roles principales: el cliente y el servidor. En este modelo, un cliente envía solicitudes a un servidor y el servidor responde a esas solicitudes. Es fundamental en muchas aplicaciones modernas y en la infraestructura de internet. Los clientes son usuarios o aplicaciones que solicitan servicios, y los servidores son programas o sistemas que proporcionan esos servicios.

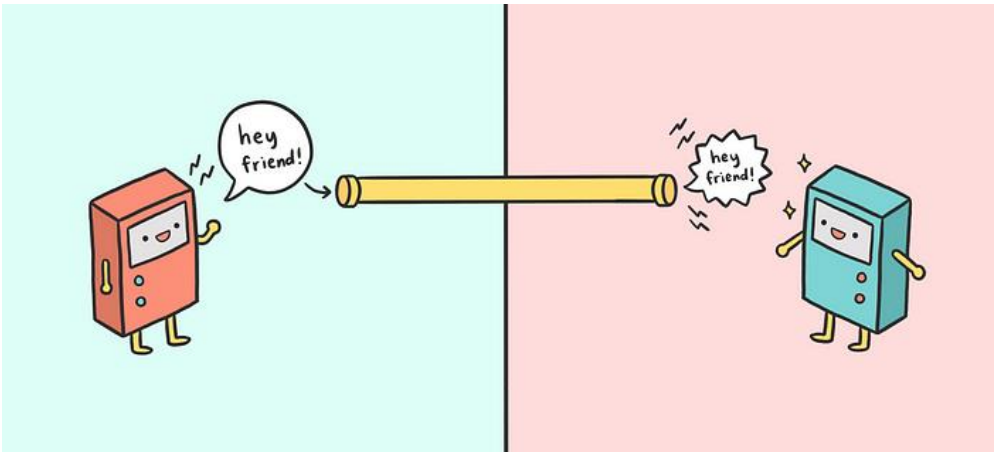
**COMPONENTES PRINCIPALES:**

**1. Cliente:** Un cliente es una aplicación o un sistema que solicita un servicio o recurso específico. En un contexto de red, un cliente es cualquier dispositivo o programa que solicita servicios de otro dispositivo o programa, generalmente un servidor.

**2. Servidor:** Un servidor es una aplicación o un sistema que proporciona servicios o recursos a los clientes. Los servidores están siempre en espera de las solicitudes de los clientes y responden a esas solicitudes.

**CONCEPTO DE SOCKET:**

Un socket es cada uno de los extremos de un canal de comunicación, entre cliente y servidor. Ya sea en el caso de que cliente y servidor estén en la misma máquina o en máquinas diferentes.




Los sockets son una API (Interfaz de Programación de Aplicaciones) que proporciona una interfaz para la comunicación entre procesos, ya sea en la misma máquina o en máquinas diferentes a través de una red. En el contexto de la arquitectura cliente-servidor:

- Los servidores crean sockets y se ponen en modo de escucha, esperando las solicitudes entrantes de los clientes.

- Cuando un cliente quiere acceder a un servicio proporcionado por un servidor, el cliente crea un socket para establecer la conexión con el servidor.
- Después de que la conexión se establece correctamente, el cliente envía solicitudes al servidor a través de su socket.
- El servidor recibe las solicitudes a través de su socket, procesa las solicitudes y envía las respuestas de vuelta al cliente a través del mismo socket.
- Cuando la comunicación ha terminado, ya sea porque se completó la tarea o porque el cliente decidió desconectarse, ambas partes cierran sus sockets.

**¿QUÉ ES EL DOMINIO DE UN SOCKET?**

En el contexto de la programación de sockets, un **dominio de socket**, también conocido como **familia de direcciones** (Address Family, AF), especifica la gama de direcciones que pueden ser asignadas a un socket. Los sistemas operativos proporcionan diferentes dominios de socket para admitir diferentes tipos de comunicación entre procesos. Algunos de los dominios de socket más comunes son:

1. AF_INET (IPv4): Este dominio de socket se utiliza para la comunicación a través de IPv4. Los sockets que utilizan este dominio tienen direcciones IP que constan de cuatro números decimales separados por puntos, como 192.168.1.1. Los servidores y clientes que se comunican a través de Internet a menudo utilizan sockets IPv4.

2. AF_INET6 (IPv6): Similar a AF_INET, pero para IPv6. IPv6 utiliza direcciones IP más largas y tiene una capacidad de dirección mucho mayor que IPv4. Con la creciente adopción de IPv6, este dominio de socket es cada vez más importante.

3. AF_UNIX (Unix Domain Sockets): Este dominio de socket se utiliza para la comunicación entre procesos en la misma máquina. En lugar de utilizar direcciones IP y números de puerto, los sockets de dominio Unix se asocian con archivos en el sistema de archivos. Esto permite la comunicación local eficiente entre procesos en una misma máquina.

4. AF_NETLINK (Netlink Sockets): Se utiliza para la comunicación entre el kernel y los procesos del usuario en sistemas Linux. Es utilizado para una variedad de propósitos, incluyendo la comunicación entre el kernel y las utilidades del sistema.

5. AF_BLUETOOTH (Bluetooth Sockets): Este dominio de socket se utiliza para la comunicación a través de conexiones Bluetooth en dispositivos que admiten esta tecnología.

6. AF_PACKET (Packet Sockets): Se utiliza para enviar o recibir paquetes de red a nivel de interfaz de red. Es utilizado para aplicaciones que necesitan acceder a tramas de red sin procesamiento adicional por parte del kernel.