1. Diferencias entre sistemas centralizados y distribuidos (y descentralizados)

Sistemas centralizados:

- Todos los datos y el procesamiento se controlan desde un único servidor o autoridad central.
- Ventajas: control fácil, simplicidad en administración.
- Desventajas: punto único de fallo, problemas de escalabilidad, posibles cuellos de botella.

Sistemas distribuidos:

- Componentes separados físicamente que se comunican a través de una red para funcionar de manera coordinada.
- Ventajas: escalables, tolerantes a fallos, equilibran cargas de trabajo y mejoran el rendimiento.
- Desafíos: más complejos de gestionar, coordinación y sincronización más compleja.

Sistemas descentralizados:

- No hay una autoridad central; el control está distribuido entre varios nodos independientes.
- Ventajas: mayor resiliencia, transparencia, tolerancia a fallos.
- Desafíos: coordinación compleja, posibles vulnerabilidades.

Contexto histórico y ejemplos:

- Centralizados: computación tradicional con mainframes, cliente-servidor, banca tradicional.
- Distribuidos: computación en la nube, microservicios, bases de datos distribuidas.
- Descentralizados: blockchain, redes peer-to-peer.

2. Conceptos: Sockets, HTTP y APIs REST

Sockets:

- Un socket es un punto final abstraído de comunicación entre procesos, indicando direcciones IP, puertos y protocolo (por ejemplo, TCP/IP)
- Permiten la comunicación cliente-servidor y son la base para intercambio de datos fiable y ordenado

HTTP y APIs REST:

- HTTP (Hypertext Transfer Protocol) es el protocolo estándar para solicitudes y respuestas entre cliente y servidor en la web.
- APIs REST son un estilo arquitectónico que utiliza HTTP para realizar operaciones CRUD utilizando métodos estándar (GET, POST, etc.) y códigos de respuesta.
- REST enfatiza sistemas stateless: cada petición es independiente y no mantiene estado entre solicitudes.

REST API vs WebSocket:

- REST API: comunicación bajo demanda (request/response), sin mantener conexión abierta
- WebSocket: canal bidireccional persistente que permite intercambio en tiempo real; la conexión permanece abierta entre cliente y servidor
- REST es ideal para operaciones estándar y microservicios, mientras que WebSocket es mejor para datos en tiempo real