Tema 3

-Internet: Red de Redes// Conjunto descentralizado de redes de comunicación heterogéneas interconectadas que utilizan el modelo TCP/IP.

-Dirección IP: Identificador único de cada dispositivo dentro de una red. *Tipos: pública / privada + estática / dinámica.*

- Class A: de 0.0.0.0 a 127.255.255.255
- Class B: de 128.0.0.0 a 191.255.255.255
- Class C: de 192.0.0.0 a 223.255.255.255
- Class D: de 224.0.0.0 a 239.255.255.255
- Class E: de 240.0.0.0 a 255.255.255.255

-**Puertos de comunicaciones** : Sirven para distinguir las distintas conexiones según el protocolo que utilicen. Tipos:

- Puertos conocidos
- Puertos registrados
- Puertos dinámicos

-Modelo cliente / servidor: Compuesto por: Un servidor / varios servidores que ofrece/n una serie de servicios y unos clientes que acceden a dichos servicios a través de la red. Tipos:

- 1 nivel > cliente y servidor en el mismo equipo.
- 2 niveles > modelo tradicional (un servidor + varios clientes).
- 3 niveles > servidores: aplicación + datos.

-La clase InetAddress representa direcciones IP.

- static InetAddress getLocalHost(). Devuelve la ip del localhost.
- static InetAddress getByName(String host). Devuelve la ip del hostname pasado como parámetro.
- Static InetAddress[] getAllByName(String host). Devuelve un array con todas las direcciones que puede tener asociadas un host.
- String getHostAddress(). Devuelve la ip en forma de cadena.

-SOCKETS TCP: Son un mecanismo de comunicación que permite que los procesos en una máquina se comuniquen con los procesos en otra máquina a través de una red. Son confiables con acuse de recibo. Establece conexión previa. Tiene mecanismos de control de flujo. Garantiza el orden y la recepción del envío de paquetes. Es menos rápido.

-SOCKETS UDP: A diferencia del protocolo TCP, la recepción en destino y el orden de llegada de los datos NO están garantizados. No se establece una conexión previamente al inicio de la comunicación. Se comienza el envío y recepción directamente. Se usan cuando se requiere una entrega rápida, en lugar de una garantizada.

-Optimizar sockets: Sus objetivos son atender múltiples peticiones simultáneamente, usamos hilos. Monitorizar tiempos de respuesta, el tiempo que el servidor necesita para procesar petición cliente + enviar datos. La transmisión es el tiempo que necesitan mensajes hasta llegar a destino. La seguridad.

Tema 4:

-Red informática: sus funciones son compartir información y recursos. Tiene como objetivo mejorar el rendimiento de la organización.

-Servicio en red: es un programa que proporciona una determinada funcionalidad o utilidad al sistema. Están basados en protocolos (normas y reglas).

-Protocolos estándar: Tienen una capa aplicación que define los protocolos de alto nivel. El protocolo de nivel de aplicación define explícitamente el formato de los mensajes que se intercambian entre cliente y servidor, así como las posibles secuencias en las que estos pueden ser enviados y recibidos.

-Protocolos básicos:

- FTP > Transferencia de archivos.
- HTTP > Transferencia de hipertexto.
- SMTP > Transferencia de correo electrónico.
- DNS* > Resolución de nombres (nombre dominio - IP).
- TFTP > Transferencia de archivos trivial

-Comunicación entre capas: Los protocolos de aplicación se comunican con el nivel de transporte mediante un API, denominada API Socket, y que en el caso de Java viene implementada mediante las clases del paquete java.net vistos en la unidad anterior.

- API bajo nivel:

- Direcciones > InetAddress.
- Sockets > Socket, ServerSocket y DatagramSocket

-API alto nivel:

- URL.
- Conexiones > URLConnection,..

-Localizador de Recursos Uniforme. Utilizando la clase URL, se puede establecer una conexión con cualquier recurso que se encuentre en Internet o en nuestra Intranet. Una vez establecida la conexión se puede obtener el contenido del recurso en el cliente. Podríamos descargar el contenido de cualquier recurso, independientemente de su protocolo.

-Protocolo FTP (File Transfer Protocol). Clase principal: FTPClient. Es un protocolo de red de la capa de aplicación, que funciona sobre TCP/IP y arquitectura cliente/servidor. En cada sesión FTP se utilizan dos canales o conexiones:

- *Canal de Control* que se utiliza para enviar y recibir comandos de control entre el cliente y el servidor. Puerto 21 en el servidor, y dinámico del lado del cliente. Flujo de datos bidireccional. El cliente envía comandos y el servidor responde con mensajes de confirmación o error. Es el primer canal que se abre en una sesión FTP, para permitir al cliente conectarse y comenzar a enviar comandos.
- *Canal de Datos:* Es el canal usado para la transferencia de archivos propiamente dicha. Puerto dinámico del lado del cliente. Normalmente, el puerto del control + 1. El puerto en el servidor puede ser el puerto 20 o un puerto dinámico, dependiendo de si es FTP activo o pasivo.

-Protocolo SMTP:

■*Comando Misión HELO o EHLO* (servidor o dominio) Se utiliza para abrir una sesión SMTP con el servidor o un dominio específico
■*MAIL FROM: I*ndicamos la dirección de origen, que debe estar gestionada por el servidor
■*SMTP RCPT TO:* Dirección de destino
■*DATA* Con este comando se indica el inicio del contenido del mensaje. Se termina con un punto como único texto en una línea.
■*QUIT* Para desconectar del servidor SMTP

-Cliente SMTP con Java: hay varias formas de programarlo, mediante sockets o mediante librerías o paquetes opcionales de la API de Java.

-PROGRAMACIÓN SERVIDORES: El servidor debe poder atender a multitud de peticiones que pueden ser concurrentes en el tiempo > hilos. Importante optimizar el tiempo de respuesta del servidor > monitorizar. Si vamos a programar servicios basados en protocolos existentes del nivel de aplicación, es necesario conocer y aplicar los detalles funcionales del protocolo

-Servidor HTTP: necesitamos conocer bien los detalles del protocolo para su programación. Sobre qué protocolo de la capa transporte funciona, quién habla primero: cliente o servidor, qué datos envía y recibe cada parte (Cabeceras) y Cómo reacciona cada parte a cada valor de cabecera.

Tema 5:

-Los objetivos de seguridad que debe proporcionar una aplicación con comunicaciones seguras son los siguientes:

■Confidencialidad.Garantiza que los datos transmitidos por la aplicación sólo van a estar disponibles para las entidades autorizadas a acceder a dicha información.
■Integridad. Garantizar que los datos originales no han sido modificados o alterados por alguna entidad no autorizada durante su transmisión.
■Autenticación. Asegurar que la entidad emisora es quien dice ser.
■No repudio. Asegurar que las acciones de un usuario han sido realizadas exactamente por dicho usuario
■Autorización. Asegurar que una entidad puede realizar una acción en concreto.

-Ataques:

■ Activos: alteran o crean una nueva comunicación (suplantación, modificación,…) [fáciles de detectar].
■ Pasivos: no alteran la comunicación (escucha o monitorización) [difíciles de detectar] > cifrado información

-Excepción: evento que ocurre durante la ejecución de un programa e interrumpe el flujo normal de las instrucciones. Tipos:

■*Error* > Problemas muy graves que suelen ser no recuperables (no se suelen tratar).
■*Exception* > Errores que no son críticos (pueden ser tratados y continuar la ejecución)

-Tipos de comprobación:

- ■*matches()* > todo el elemento sigue el patrón.
- ■*find()* > todo o parte del elemento sigue el patrón

-[DRAE] Criptografía: Arte de escribir con clave secreta o de un modo enigmático > “escritura secreta".

-Cifrado simétrico (clave privada) : misma llave para cifrar y descifrar. Ventaja > algoritmo rápido. Inconveniente > distribución claves por canal inseguro

-Cifrado asimétrico (clave pública): llaves diferentes para cifrar y descifrar. Ventaja > elimina problemas de distribución claves. Inconvenientes > algoritmo más lento / “Man in the Middle”.

- ■Clave pública (se puede compartir y distribuir con nuestros interlocutores): permite cifrar.
- ■Clave privada (personal e intransferible): descifra los mensajes cifrados con la clave pública.

-Funciones HASH. Convierte uno o varios elementos de entrada a una función en otro elemento. Los más usados son MD5 y SHA.

-Técnicas criptográficas:

- ■Resumen de mensajes, basado en funciones HASH. Un algoritmo de resumen toma como entrada un mensaje de longitud variable y lo convierte en un resumen de longitud fija, cuyas principales características son: siempre debe proporcionar la misma salida, debe ser aleatorio y unidireccional.
- ■Firmas digitales. Son el equivalente digital de las firmas personales. Se basan en criptografía de clave pública y resumen de mensajes o funciones HASH.