Capa de presentación:

Nos va a permitir que dos extremos (dispositivos totalmente distintos) se entiendan. Los lleva a un modelo estándar para que todos se entiendan.

Tiene los datos de como se manda la información, si se manda encriptada, como se encriptó, desencriptó.

Capa de Aplicación:

Nos va a permitir comunicarnos con la red

Hay muchos protocolos. Cada uno de estos define su forma de trabajar, como se manda el mensaje, que formato, etc.

Hay 4 modelos:

* Mainframe (dumb client)
* Cliente/Servidor: El servidor en estado “listening”
* Peer to Peer (P2P):
* Hibrido: Puede haber servidores o P2P, es una mezcla

Atraves de TCP/IP podemos identificar un equipo. Solo identificar el equipo, no lo que pasa dentro.

El port o puerto nos indica a que servicio o proceso va dirigido tal mensaje.

En la parte de aplicación tenes el servidor y la aplicación web.

5 Valores:

* Ip local
* puerto local
* ip receptor
* puerto receptor
* protocolo

Protocolo HTTP:

Formato de URL: protocol://[user:pass@]host:[port]/[path]

Aplica el modelo Cliente/Servidor

Es un servidor sin estado, no se acuerda lo que hace.

Los puertos 0 a 1023, son puertos privilegiados

Los puertos 1024 en adelante son puertos no privilegiados

Clientes: browsers o navegadores

Servidores: NGINX, Apache, etc.

Client REQUEST:

* La primer línea es el header, request.
* Desde la segunda línea en adelante se cuentan los headers que tienen un formato: “VALOR: CONTENIDO”. La cantidad de headers que hay son los que siguen ese formato

Server REQUEST:

* La primer línea de estado.

Respuestas:

* HTTP/ 200 OK
* HTTP/ 301 Moved Permanently
* HTTP/ 400 Bad Request
* HTTP/ 403 Access Forbidden
* HTTP/ 404 Not Found
* HTTP/ 405 Method Not Allowed
* HTTP/ 500 Internal Server Error (CGI Error)
* HTTP/ 501 Method Not Implemented

Metodos:

* GET: Para obtener un valor. Puede enviar información, pero no demasiada. No espera recibir datos en body.
* HEAD: Idéntico al GET pero no me devuelve el objeto. Solo devuelve la cabecera
* POST: Esta pensado para enviar datos al servidor. Lo envia dentro del body.
* PUT: Sube un objeto desde el cliente al servidor.
* DELETE: Permite borrar un archivo dentro del servidor.
* LINK, UNLINK: Permite establecer relaciones entre distintos objetos del servidor.

La cabecera HOST en la versión 1.1 es obligatoria. Puede aparecer en la 1.0 pero no es obligatoria.

HTTPS:

En https nos conectamos por default al puerto 443, en cambio en http es el 80.

El servicio HTTPS es el mismo que el HTTP, pero la información se envia cifrada.

El cliente quiere hacer un get, se abre el TCP, y antes de que devuelva la respuesta, se encripta todo