**NODE.JS**

* Node.js es un entorno en tiempo de ejecución multiplataforma de código abierto para la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono y basado en el motor V8 de Google.

**HTTP**

* Métodos HTTP:
  + GET: Recoger información del servidor. (SELECT)
  + POST: Añadir información al servidor. (INSERT)
  + PUT: Reemplazar información en el servidor. (UPDATE)
  + PATCH: Actualizar parte de la información.(UPDATE - WHERE)
  + DELETE: Eliminar información del servidor. (DELETE)
  + OPTIONS: Pedir información sobre métodos (saber si podemos ejecutar alguno de los métodos anteriores).
* **Las cabeceras** serán el envío al servidor de cómo queremos hacer la petición.
  + **Cabeceras Content-Type:** Me va a indicar el tipo de contenido que tengo
    - Text/html
    - Text/css
    - Application/javascript
    - Image/jpeg
    - Application/json
    - Application/xml
  + **Cabeceras Content-Lenght:** Me va a informar como de largo es el contenido
* Los estados son números que indica el estado de la petición:
  + 2XX: Todo ha ido bien.
  + 3XX: La petición se ha redirigido.
  + 4XX: Errores del cliente.
  + 5XX: Ha habido un error al procesar la petición.
* **El cuerpo de la petición** es la información en sí que queremos enviar, editar o que el servidor nos devuelva y mi cuerpo irá de acuerdo con el content-type definido en mi cabecera, es decir, si definí application/json pues mi cuerpo será en forma JSON
* **Las queries** van a permitirte añadir información extra a los datos que queramos enviarle al servidor. La estructura de la query será,
  + Añadir ? al final de la URL
  + Nombre=valor
  + Separados con &
  + Ejemplo: <https://www.youtube.com/watch?v=sRkw968M2zY>
* Node utiliza el manejador de paquetes NPM (Node Package management) allí instalamos todas las librerías que necesitemos en nuestro proyecto
* Para Node.js usamos una librería llamada **Express** que me servirá como servidor en mi BackEnd
* La forma en que Nod trae modulos de otros sitios es: **const express = require('express')** funciona como import…
* En el response hay varios tipos de respuesta: Vacía, plana, con datos y estructurada
* Para desarrollar los errores mas organizado, se puede crear un archivo apartado en una carpeta network, esto me permitirá llevar control de mis mensaje satisfactorios o erróneos al cliente de la aplicación
* El archivo routes.js será encargado de gestionar todas las rutas dentro de mi aplicación, este archivo lo llama server.js

**WEBSOCKETS**

* El protocolo Websocket o wss:// crea un túnel de información entre el usuario y el servidor el cual se quedará abierto hasta que el servidor y/o el cliente cierre la conexión para pedir información en tiempo real.
* Un ejemplo es la conexión de Uber cuando transmite información del carro por donde va, o los mensajes de whatsapp, todo lo que se tenga que usar en tiempo real debe ser con sockets

**PASOS INICIAR PROYECTO NPM**

* **Npm init** para iniciar el proyecto
* **Npm install express** esto indica cual será el servidor que usaremos para Node
* **npm install -g nodemon** esta instrucción instalara nodemon en nuestro proyecto, esto hará que la consola se refresque automáticamente cada vez que suceda un nuevo cambio en mi aplicación, para ello, iniciamos el proyecto con nodemon server
* **npm install mongoose** esta instrucción me sirve para modelar mis datos de Mongo
* **npm install socket.io** me permitirá trabajar con el protocolo de websocket
* **npm i cors** este paquete sirve para hacer las cabeceras http más restrictivas