¡Les damos la bienvenida!

¿Comenzamos?

Esta clase va a ser

- grabada

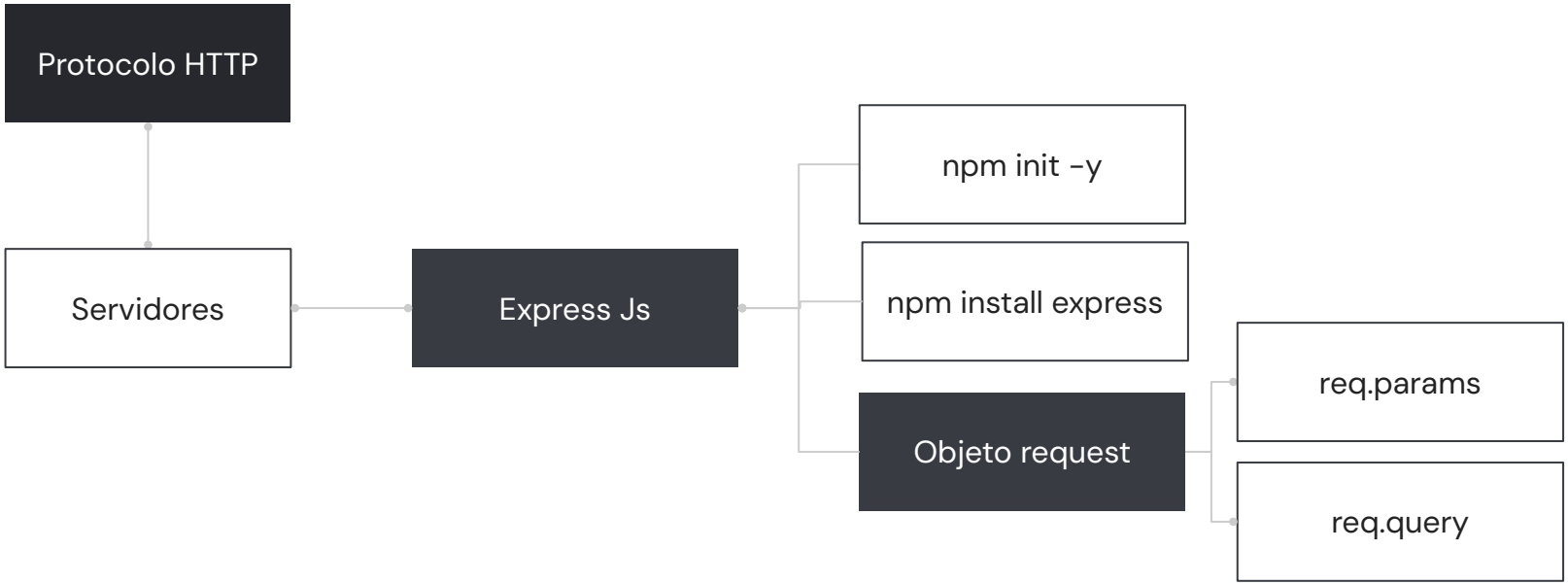
Clase 06. DESARROLLO AVANZADO DE BACKEND

Servidores Web

Objetivos de la clase

- **Entender** el protocolo HTTP
- **Aprender** a crear un servidor utilizando el módulo nativo http
- **Crear** un servidor de express
- **Profundizar** sobre las peticiones GET de un servidor express.

MAPA DE CONCEPTOS



Entendiendo el protocolo HTTP

HTTP

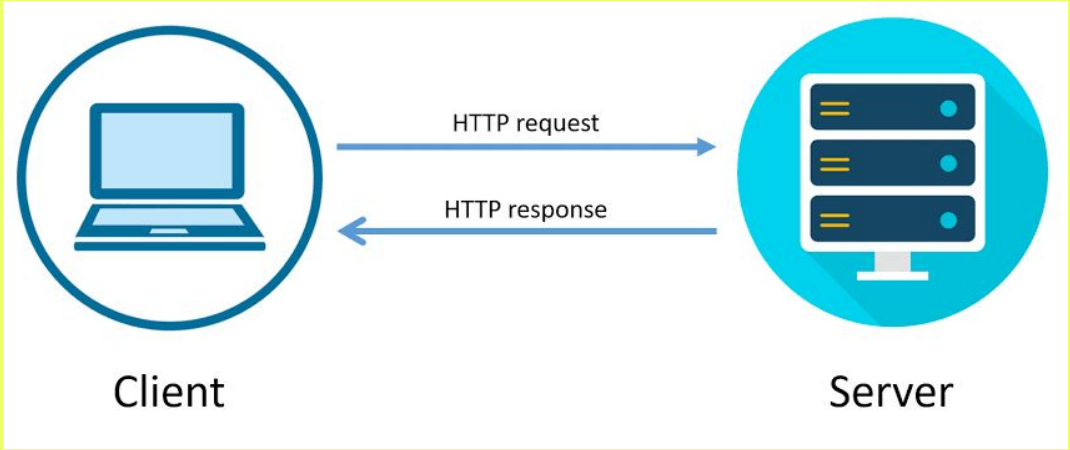
HTTP (Hyper Text Transfer Protocol) refiere a un **protocolo**, el cual es un conjunto de reglas que permite la comunicación entre dos o más sistemas. Gracias a este protocolo, las computadoras saben comunicarse entre sí y permiten comunicarse con servidores para obtención de datos. (lo podemos ver en todas las páginas que visitamos)

CODERHOUSE

HTTP

¿Cómo funciona?

Se basa en un modelo de **petición - respuesta**. De manera que el **cliente** tiene que hacer una petición de información, entonces **el servidor** se encarga de responder con dicha información.



¿Y si el servidor se apaga?

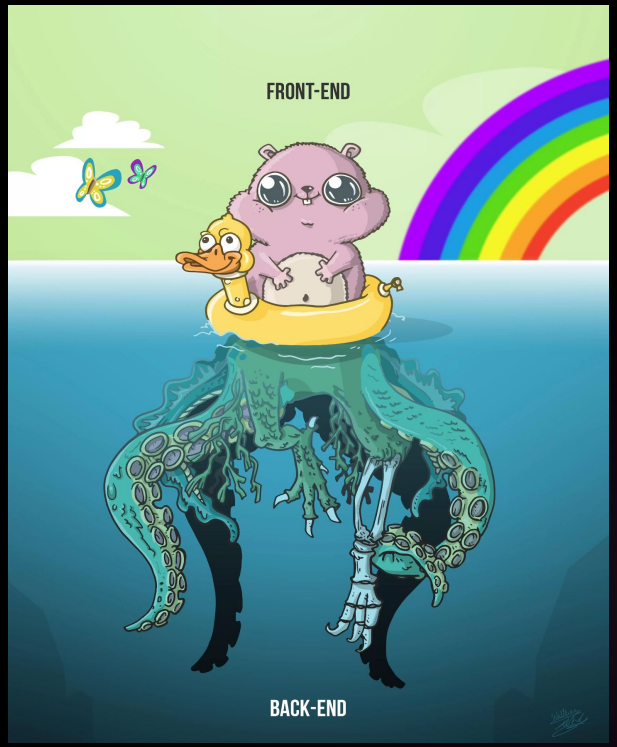
Sabemos que un programa se inicia, hace sus operaciones, y luego finaliza.

¿Qué tiene de diferente el servidor? ¿Cómo puede recibir peticiones en diferentes períodos de tiempo sin finalizar? 🤔

¡Importante!

El cliente siempre es quien hace las peticiones (requests) y el servidor siempre será quien hace las respuestas (responses) . Cuando hacemos frontend, somos clientes. ¡En este curso te tocará ser el servidor y devolver respuestas!

PARA RECORDAR



Mientras que el **FrontEnd** se centra en la experiencia del usuario, **¡que esta sea lo más presentable y amigable posible!**

...el **BackEnd** tiene una estructura interna para que todo funcione correctamente. **¡Que no lo vea el usuario!**

Instalar nodemon de manera global

Como nuestro servidor se mantiene escuchando todo el tiempo, los cambios que hagamos en el código no se reflejan automáticamente. Por ello **tenemos que apagar el servidor y levantarlo nuevamente**.

Sólo así podemos visualizar los cambios que hagamos en el código (así sea un simple console.log()

Nodemon nos permitirá **reiniciar automáticamente** el servidor en cuanto detecta que hay cambios en el código. De esta manera, podemos concentrarnos en el código, sin tener que realizar el reinicio manual cada vez que queremos ver algo.

Para instalarlo, basta correr el comando:

```
npm install -g nodemon
```

Ejemplo en vivo

- ✓ Crear un servidor con el módulo nativo de nodejs “http”. Setear una respuesta que contenga el mensaje “¡Mi primer hola mundo desde backend!”
- ✓ El servidor debe escuchar en el puerto 8080. (Correr con nodemon)
- ✓ Probar el servidor desde el navegador.
- ✓ Hacer algún cambio en código y corroborar que se reinicie automáticamente.

Servidor con Express js

Ejemplo de una consulta en Express

- ✓ Estructurar un servidor basado en express, el cual escuche peticiones en el puerto 8080
- ✓ Realizar una función para el método GET en la ruta ‘/saludo’, el cual responderá con “¡Hola a todos, pero ahora desde express!”
- ✓ Ejecutar con nodemon y probar en el navegador el endpoint generado

ACTIVIDAD EN CLASE

Otras respuestas con express

Crear un proyecto basado en express js, el cual cuente con un servidor que escuche en el puerto 8080. Además de configurar los siguientes endpoints:

- ✓ El endpoint del método GET a la ruta '/bienvenida' deberá devolver un html con letras en color azul, en una string, dando la bienvenida.
- ✓ El endpoint del método GET a la ruta '/usuario' deberá devolver un objeto con los datos de un usuario falso: {nombre, apellido,edad, correo}

Break

¡10 minutos y volvemos!

Objeto request

req.params

req.params

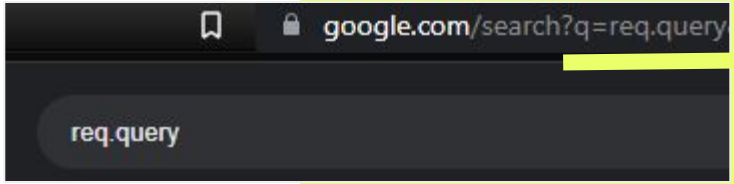
Se utiliza cuando necesitamos obtener elementos dinámicos desde la ruta que está llamando el cliente.
Para poder definir un “parámetro” dentro de la ruta a trabajar, basta con colocar el símbolo de dos puntos **:** antes del parámetro, de esta manera, express reconoce que queremos que ese elemento sea dinámico.

req.query

req.query

Como su nombre lo indica, query refiere a las múltiples consultas que se pueden hacer a un determinado endpoint, basta conque en la url coloquemos el símbolo **?** , entonces express reconocerá que hay que meter información al objeto **req.query** para poder utilizarlo en el endpoint.

Cuando buscamos algo en nuestro navegador, llamamos a un endpoint haciendo un determinado query.



CODERHOUSE

¿Qué diferencia hay con params?

La principal diferencia que hay entre req.params y req.query, es que en req.query puedo meter la cantidad de consultas que yo así desee, ya que las queries no vienen inmersas en la ruta, sino que son un elemento aparte.

Así, si desconozco el número de cosas que se van a consultar en mi ruta, la mejor opción es utilizar **queries**, mientras que, si sólo necesito un número específico y reducido de parámetros, habría que optar por **params**

Al final, no hay una mejor que otra, sirven para casos diferentes e incluso podemos utilizar ambas en la misma consulta.

¿Preguntas?

Opina y valora esta clase

Muchas gracias.