



Clase 11. Programación Backend

Websockets



OBJETIVOS DE LA CLASE

- Comprender la diferencia entre HTTP y WebSocket.
- Integrar WebSocket a nuestro proyecto de Express.
- Generar la inicialización sobre el cliente para conectarse al servidor mediante WebSocket

CRONOGRAMA DEL CURSO

Clase 10



Pug & Ejs

Clase 11



Websockets

Clase 12



**Aplicación chat con
websocket**

Repasando...

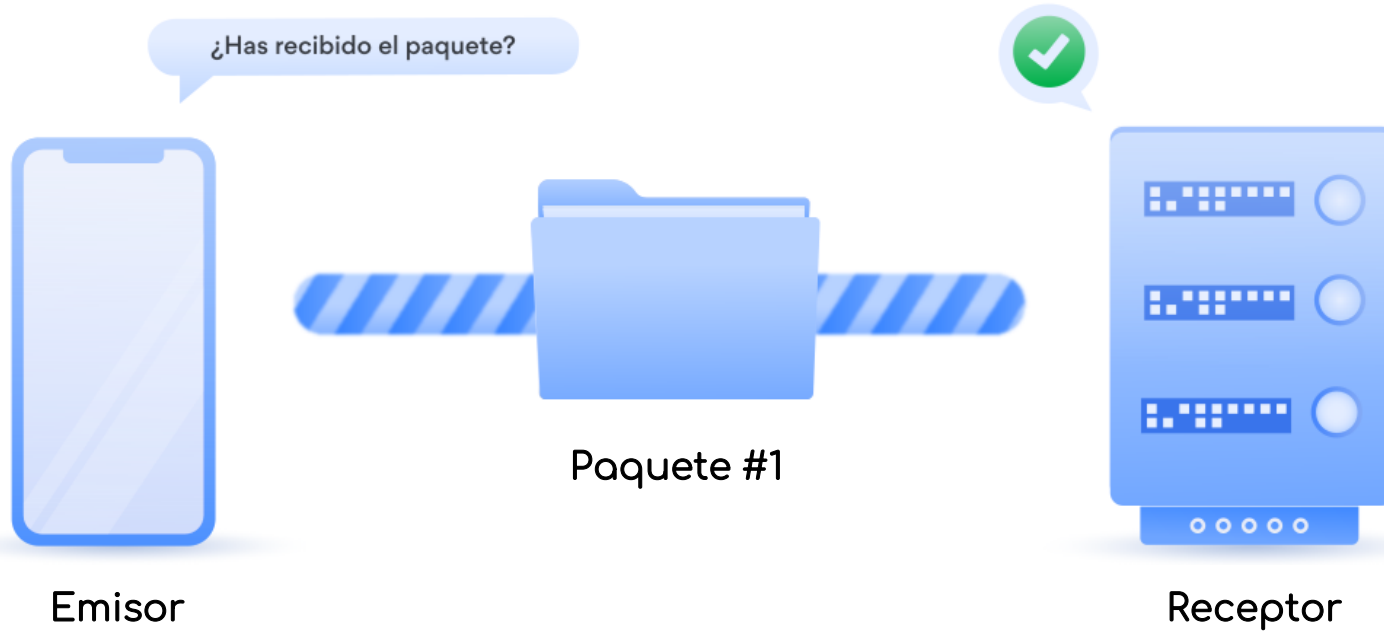
- Recordemos que los **protocolos** son **conjuntos de normas** para **formatos de mensaje** y **procedimientos** que permiten a las máquinas y los programas de aplicación **intercambiar información**.
- Cada máquina implicada en la comunicación debe seguir estas normas para que el sistema principal de recepción pueda **interpretar el mensaje**.

Protocolos de comunicación

- El Protocolo **TCP** o **Transfer Control Protocol** consiste en un **acuerdo estandarizado** sobre el que se realiza la **transmisión de datos** entre los participantes de una **red informática**.
- Los programas que forman redes de datos en una **red de ordenadores** emplean el **protocolo TCP** para crear conexiones entre sí, de forma que se pueda **garantizar el flujo de datos entre las partes**.
- A través de este protocolo se asegura que los **datos lleguen a su destino en el mismo orden que se transfirieron y sin errores**.
- El protocolo TCP da soporte al protocolo **HTTP**.

Protocolo TCP

Cómo funciona TCP



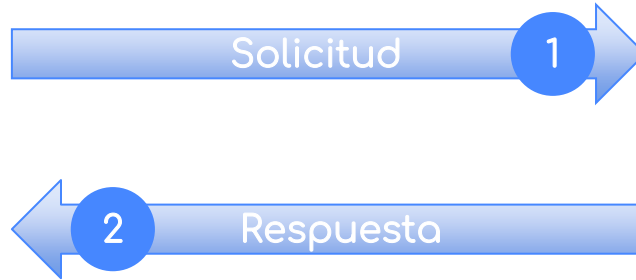
- **HTTP** significa **Hyper Text Transfer Protocol** y es un **protocolo de comunicación** que define la manera en que se **comunican** un dispositivo **cliente** y un **servidor** conectados a través de la **web**.
- El protocolo se basa en un **esquema de petición-respuesta**.
- Existen clientes que realizan **solicitudes de transmisión de datos**, y un **servidor** que **atiende** estas solicitudes.

Protocolo HTTP

Cómo funciona HTTP



Cliente



Servidor



***¿Alguna pregunta hasta
ahora?***

WebSocket



WebSocket

- **Websocket** es un **protocolo de red basado en TCP** que establece cómo deben intercambiarse datos entre redes.
- Es un protocolo **fiable** y **eficiente**, utilizado por prácticamente todos los clientes.
- El protocolo TCP **establece conexiones** entre **dos puntos finales** de comunicación, llamados **sockets**.
- De esta manera, el **intercambio de datos** puede producirse en las **dos direcciones**.

Websocket

- En las **conexiones bidireccionales**, como las que crea Websocket, se **intercambian datos en ambas direcciones al mismo tiempo**.
- La ventaja de usar Websocket es **acceder de forma más rápida** a los datos.
- Websocket permite una **comunicación directa y en tiempo real** entre una aplicación web y un servidor Websocket.

Websocket

Cómo funciona WEBSOCKET



- Websocket permitió por primera vez **acceder** a una **web** de **forma dinámica** en **tiempo real**.
- Basta con que el **cliente establezca** una **conexión** con el **servidor**, que se confirma mediante el llamado apretón de manos o **WebSocket Protocol Handshake**.
- Con él, **el cliente envía** al **servidor** todos los **datos de identificación** necesarios para el **intercambio de información**.

WebSocket: principios

Cómo funciona WEBSOCKET



- Para iniciar el intercambio con WebSocket el **cliente envía una solicitud**, al igual que en el clásico HTTP. Sin embargo, la **conexión** se establece **mediante TCP** y **permanece abierta** tras el handshake entre el cliente y el servidor.
- El nuevo esquema URL de WebSocket para las páginas web mostradas se define con el **prefijo ws** en lugar de http. El prefijo que corresponde a una conexión segura es **wss**, de forma análoga a https.

Detalle de intercambio de datos

Solicitud Cliente



```
1 GET ws://localhost:3000/websocket/?idCliente=1664165749796&EIO=4&transport=websocket&sid=0vuks5u2AFS-Gez_AAAC HTTP/1.1
2 Host: localhost:3000
3 Connection: Upgrade
4 Pragma: no-cache
5 Cache-Control: no-cache
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36
7 Upgrade: websocket
8 Origin: http://localhost:3000
9 Sec-WebSocket-Version: 13
10 Accept-Encoding: gzip, deflate, br
11 Accept-Language: en-US,en;q=0.9,es-US;q=0.8,es;q=0.7
12 Sec-WebSocket-Key: KQABW194jqWf0Z+VzjjI5g==
13 Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
```

Respuesta Servidor



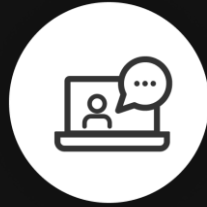
```
1 HTTP/1.1 101 Switching Protocols
2 Upgrade: websocket
3 Connection: Upgrade
4 Sec-WebSocket-Accept: Y0rtQz0M3PmS+QHx/fAjG80e+gY=
```

- Para establecer **conexiones de forma rápida**. Por ejemplo: **chats de asistencia técnica, tickers de noticias** o de **actualizaciones de bolsa en directo, servicios de mensajería instantánea y juegos en tiempo real**.
- Websocket también resulta muy útil en las **redes sociales** para establecer **conexiones en directo** con otras personas, así como para enviar y recibir **mensajes instantáneos**. Permite obtener **altas velocidades de transmisión** y **limitar los tiempos de latencia**.

***¿Para qué se
utiliza
WebSocket?***

Resumen

- Websocket **no es** un **sustituto total** de **HTTP**, pero puede usarse como **canal** de comunicación **eficiente y bidireccional** siempre que se necesite dar o recibir **información** en **tiempo real**.
- El protocolo **Websocket** está muy vinculado con el desarrollo de **HTML5**: un intento de hacer la **web más rápida**, más **dinámica** y más **segura**.
Permite a las aplicaciones web reaccionar mucho más rápido que con la comunicación HTTP convencional. Sin embargo, esto no significa que haya que reemplazar el protocolo tradicional: a pesar de la existencia de Websocket, **HTTP sigue siendo un estándar clave** en Internet.



***¿Alguna pregunta hasta
ahora?***



BREAK

¡5/10 MINUTOS Y VOLVEMOS!



socket.io

- Socket.IO es una **biblioteca de JavaScript** para **aplicaciones web en tiempo real**. Permite la comunicación **bidireccional** en **tiempo real** entre **servidores** y **clientes web**.
- Tiene dos partes:
 - Una **biblioteca del lado del cliente** que se **ejecuta en el navegador**.
 - Una **biblioteca del lado del servidor** para **Node.js**.
- Ambos componentes tienen una **API** casi idéntica. Al igual que Node.js, está impulsado por **eventos**.

***¿Que es
socket.io?***

- Socket.IO **utiliza principalmente** el protocolo **Websocket** proporcionando la misma interfaz.
- Se puede **usar como** un **contenedor** para Websocket aunque proporciona muchas más funciones, incluida la transmisión a múltiples sockets, el almacenamiento de datos asociados con cada cliente y E/S asíncronas.
- Se puede instalar con **npm**.

socket.io: características

- **Fiabilidad:** Las conexiones se establecen incluso en presencia de:
 - Proxies y balanceadores de carga.
 - Firewall personal y software antivirus.
- **Soporte de reconexión automática:**

A menos que se le indique lo contrario, un cliente desconectado intentará siempre volver a conectarse, hasta que el servidor vuelva a estar disponible.

socket.io:
características

- **Detección de desconexión:** Se implementa un mecanismo de heartbeat, lo que permite que tanto el servidor como el cliente sepan cuando el otro ya no responde.
- **Soporte binario:** Se puede emitir cualquier estructura de datos serializable, que incluye:
 - ArrayBuffer y Blob en el navegador
 - ArrayBuffer y Buffer en Node.js

socket.io:
características

Por fin 🤔... vamos al IDE... 🖥️



SERVIDOR CON WEBSOCKET

Vamos a practicar lo aprendido hasta ahora

SERVIDOR CON WEBSOCKET - 1

Desafío
generico



- 1) Desarrollar un servidor basado en express que tenga integrado Websocket. Con cada conexión de cliente, el servidor debe emitir por consola en mensaje: '¡Nuevo cliente conectado!'

Tiempo: 5 minutos

CODER HOUSE

SERVIDOR CON WEBSOCKET - 2

Desafío
generico



- 2) Sobre la estructura anteriormente creada, agregar en la vista de cliente un elemento de entrada de texto donde al introducir texto, el mensaje se vea reflejado en todos los clientes conectados en un párrafo por debajo del input.

El texto debe ser enviado caracter a caracter y debe reemplazar el mensaje previo.

SERVIDOR CON WEBSOCKET - 3

Desafío
generico



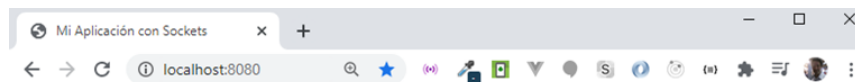
- 3) Basado en el ejercicio que venimos realizando, ahora los mensajes enviados por los clientes deberán ser almacenados en el servidor y reflejados por debajo del elemento de entrada de texto cada vez que el usuario haga un envío. La estructura de almacenamiento será un array de objetos, donde cada objeto tendrá la siguiente estructura:

```
{ socketid: (el socket.id del que envió el mensaje), mensaje: (texto enviado) }
```

- Cada cliente que se conecte recibirá la lista de mensajes completa.
- Modificar el elemento de entrada en el cliente para que disponga de un botón de envío de mensaje.
- Cada mensaje de cliente se representará en un renglón aparte, anteponiendo el socket id.

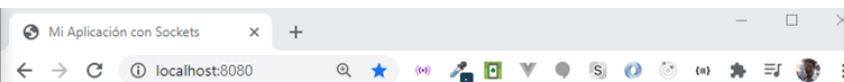
Tiempo: 15 minutos

CODER HOUSE



Cliente Websockets

SocketId: gSOwhnuxTvC1Nsf6AAAF -> Mensaje: hola
SocketId: CaaecnmWUlh_8Vq5AAAH -> Mensaje: Que tal
SocketId: gSOwhnuxTvC1Nsf6AAAF -> Mensaje: bien



Cliente Websockets

SocketId: gSOwhnuxTvC1Nsf6AAAF -> Mensaje: hola
SocketId: CaaecnmWUlh_8Vq5AAAH -> Mensaje: Que tal
SocketId: gSOwhnuxTvC1Nsf6AAAF -> Mensaje: bien

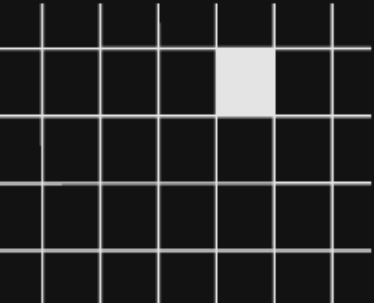
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Websocket
 - Socket.IO
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDOLAEDUCACIÓN