UNIDAD 4: Node.js

1. [UNIDAD 4: Introducción](https://learn.nextu.com/mod/page/view.php?id=3540&pid=P_WEBDEV_V2)

Estudiaremos la escritura de servidores, desde cero, en node.js, el ecosistema JavaScript del lado del servidor que es destacado por su capacidad de procesar grandes cantidades de datos en tiempos muy bajos.

Los objetivos del Programa que corresponden a esta Unidad son:

* Usar metodologías sincrónicas y asincrónicas para gestionar datos desde el servidor.

Los objetivos del Curso 6 que corresponden a esta Unidad son:

* Describir el funcionamiento de la ejecución sincrónica y asincrónica
* Definir qué es una API y cómo funcionan los Servicios Web
* Implementar AJAX
* Comprender qué es y cómo funciona Node.js

1. Lección 1: Qué es Node.js y cómo funciona
   1. [Qué es Node.js](https://learn.nextu.com/mod/lesson/view.php?id=3541&pid=P_WEBDEV_V2)

Qué es Node.js?

Node.js es un entorno de ejecución multiplataforma, de código abierto, totalmente asíncrono, con una entrada y salida de datos basada en una arquitectura orientada a eventos.  
  
Node.js es JavaScript pero del lado del servidor, utiliza la máquina virtual de Google Chrome (V8) para la interpretación y ejecución del código de javascript. Node fue desarrollado con la misión de crear programas y servidores web de alto rendimiento y fácilmente escalables.  
  
Node.js es ideal para crear servidores que manejen muchas conexiones simultáneas y una concurrencia de usuarios/datos bastante alta.

¿Qué es V8?

V8 es el motor de JavaScript creado por Google para su navegador Chrome que se encarga de interpretar y ejecutar el código. Es un intérprete bastante poderoso con una característica especial, OPEN-SOURCE, es decir de código abierto, y cualquier persona puede descargar el código fuente de este intérprete y hacerlo funcionar en una aplicación. Node usa la máquina virtual V8 y le da otro propósito para usarlo en el servidor.

¿Por qué no utilizar el mismo lenguaje que usted emplea en el cliente para programar el servidor ?

Fue una de las preguntas con las que se inició la idea, la posibilidad de crear una arquitectura que contemplara el lenguaje Javascript tanto del lado del cliente como del servidor.

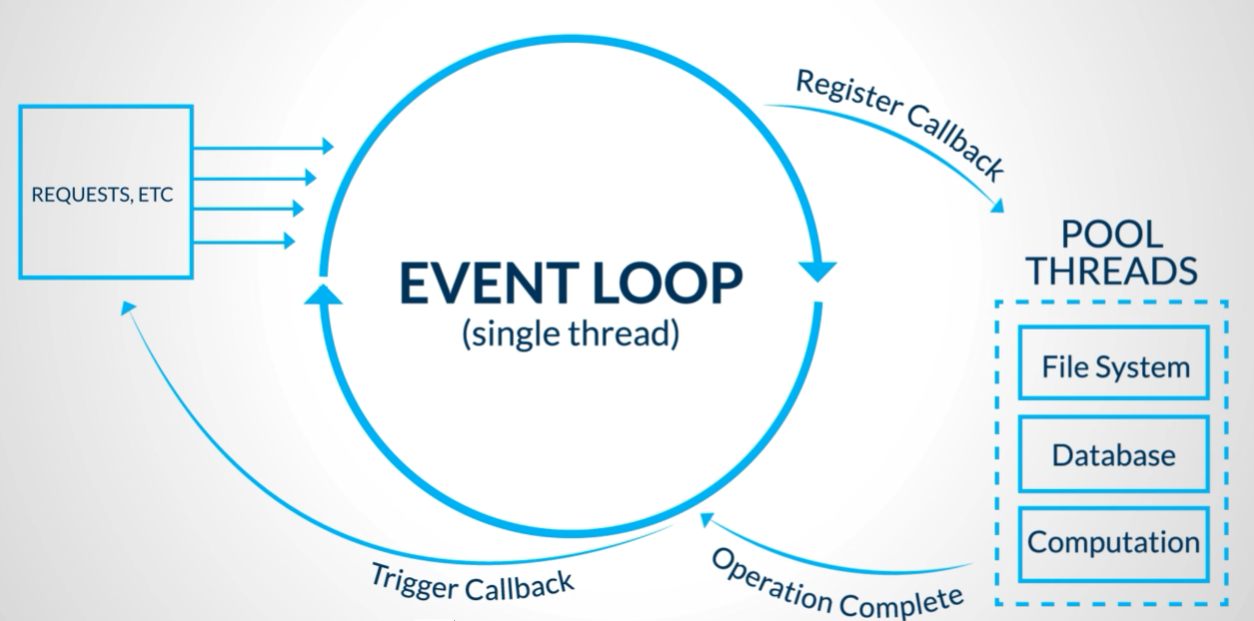
Programación orientada a eventos:

Hemos escuchado que existen varios paradigmas de programación muy populares como POO(Programación Orientada a Objetos), el cual es muy común encontrarlo en la mayoría de lenguajes más populares de la actualidad.  
  
Node usa el paradigma de Programación Orientada a Eventos. Al igual que en el lado del cliente estamos a la espera de acciones del usuario y sus correspondientes eventos, en el lado del servidor no es nada diferente, aunque no se están presionando botones ni llenando formularios, en otros niveles están sucediendo eventos, como establecimiento de conexiones, envío y recepción de datos y todas las demás actividades para las que un servidor debe estar en capacidad de responder.

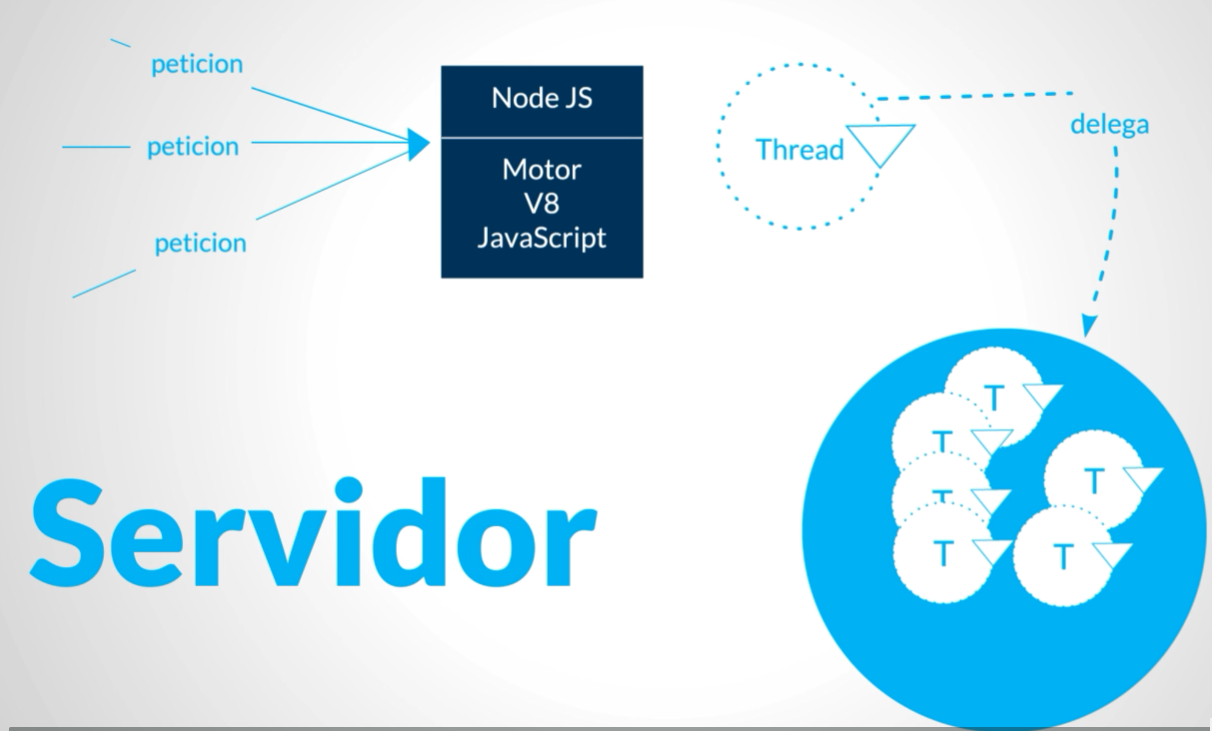
* 1. Cómo funciona Node.js

Eventos 🡪 Asíncrono

Trabaja con un único hilo de ejecución, las tareas y procesos es de un modelo llamado evento Loop.



Librería llamada libuv 🡪 múltiples hilos asíncronos

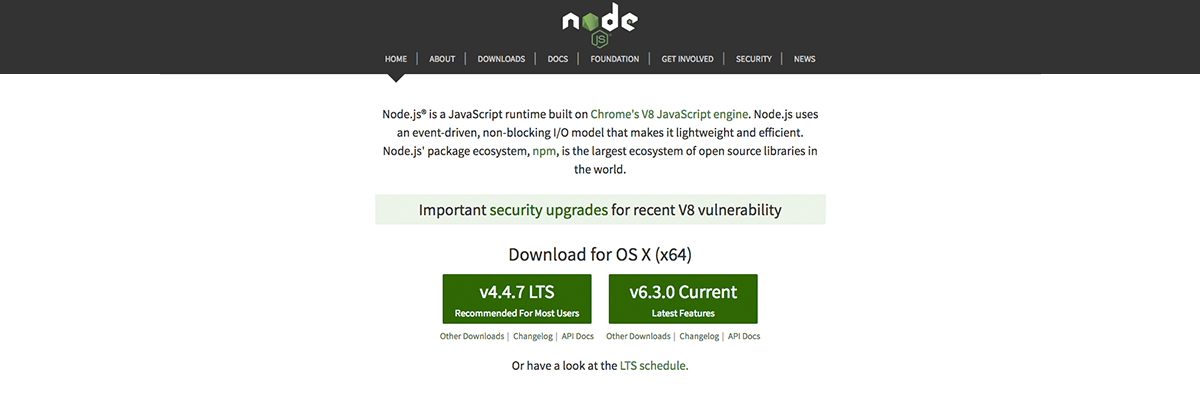


* 1. Descarga e instalación

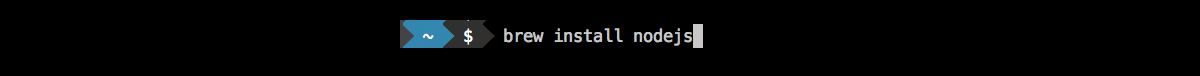
Descarga e Instalación de Node.js

Para tener Node.js en nuestra máquina, es necesario descargar el binario e instalarlo.  
  
Si estás en MacOS existen varias formas:

Utiliza Brew (el administrador de paquetes para MacOS), abre la terminal y digita:



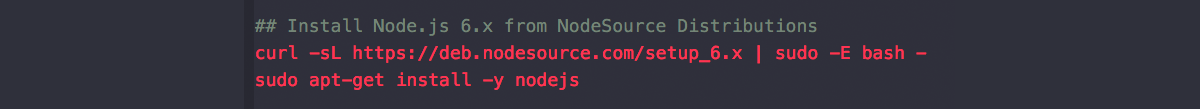
Otra forma es ir directamente a la página oficial: [https://nodejs.org](https://nodejs.org/en/) y descargar alguna de las versiones disponibles:



Actualmente existen dos versiones de Node.js disponibles, la versión 4.x es LTS (Long Term Support), es decir, una versión que estará estable por mucho tiempo y no sufrirá cambios drásticos que puedan romper el ecosistema, es la versión más recomendada para la mayoría de usuarios.  
  
La versión 6 es en la cual se incorporan y se prueban las nuevas características y funcionalidades y es posible que sufra muchos cambios y corrección de errores. Esto puede causar que dejen de funcionar algunas propiedades. No se recomienda para aplicaciones en producción.  
  
Si estás en Windows, para instalar Node.js debes descargar el archivo ejecutable desde la página oficial, aceptar los términos y condiciones y hacer clic en siguiente hasta finalizar la instalación.  
  
Si estás en linux - Debian o algún derivado es porque eres un usuario un poco más avanzado, la recomendación que hacemos es que instales Node.js via PPA, agregando el repositorio de NodeSource y luego descargando via apt-get.



Ahora sólo falta verificar que Node.js esté correctamente instalado en la máquina, para lo que debes ingresar a una terminal y digitar “node --version”:



La Terminal te devuelve la versión de node que está instalada.

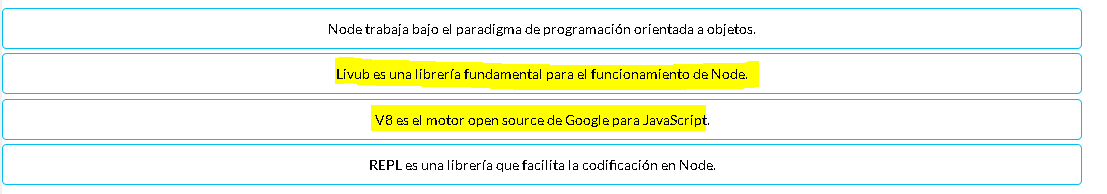
* 1. Node.js REPL

Que es un REPL – Readable Print to Loop

* 1. Actividad

Tiempo de pensar

En esta lección aprendiste que node es un entorno de ejecución multiplataforma que permite realizar gran cantidad de acciones del lado del servidor. Además, usa el lenguaje JavaScript lo cual es muy conveniente para los desarrolladores ya que no hay diferencia entre el lenguaje usado en el front-end y el usado en el back-end.  
  
Instrucciones:  
  
Lee con atención las opciones de respuesta, las cuales contienen algunas afirmaciones relacionadas con Node. Selecciona las 2 afirmaciones verdaderas.



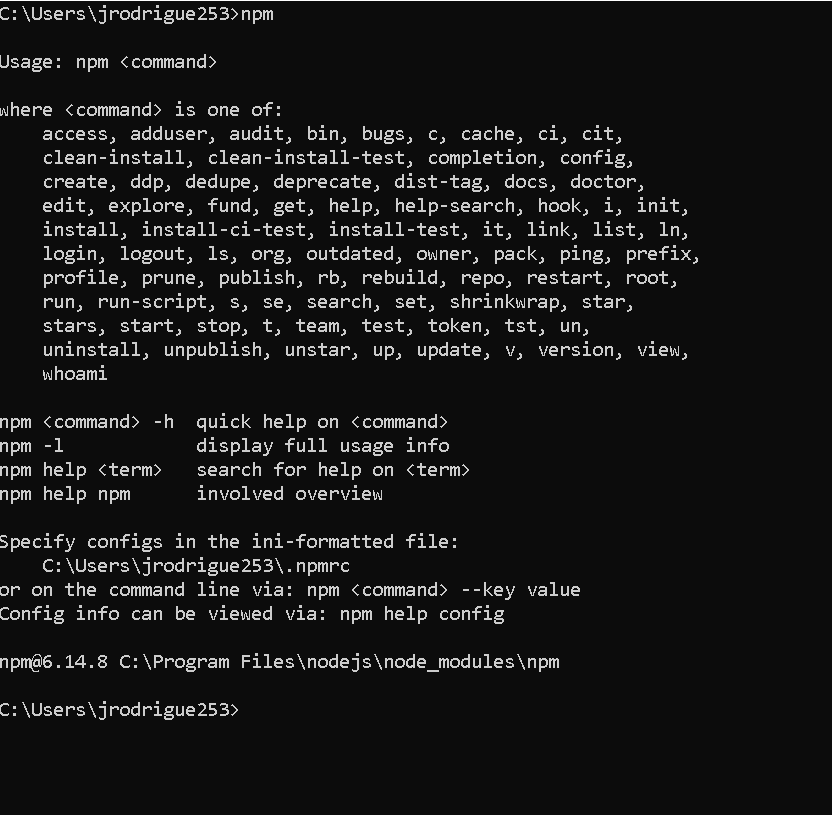
1. Lección 2: Node Package Manager (NPM)
   1. Qué es NPM

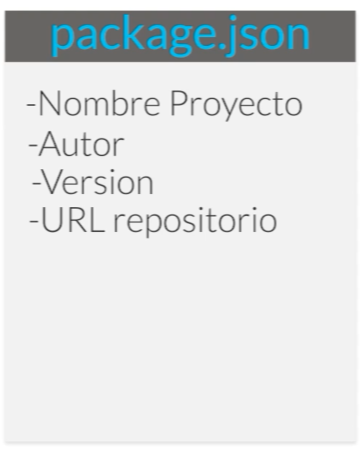
¿Qué es NPM?

No es conocido, a excepción de sus creadores, el verdadero significado de las siglas N.P.M. Lo que sí sabemos es que es un maravilloso gestor de paquetes y dependencias para JavaScript, diseñado para funcionar en la línea de comandos. Es un sistema que viene integrado con Node.js y se instala en el mismo momento que se instala Node en la máquina.  
  
Casualmente N.P.M. se acomoda perfectamente al nombre de Node Package Manager y así es como se conoce entre la comunidad de desarrolladores JavaScript. N.P.M. es una plataforma de código abierto, en la que miles de programadores de todo el mundo pueden crear librerías o paquetes como son conocidos en el ecosistema Node.js y publicarlos para que puedan ser descargados y usados por otras personas.  
  
En palabras concretas, N.P.M. es un gestor de módulos y paquetes para Node.js en el cual los desarrolladores crean, comparten y reutilizan módulos en las aplicaciones.  
  
N.P.M. cuenta con una cantidad de paquetes increíbles, los cuales nos ayudan a resolver muchos de los problemas que nos encontramos dia dia como desarrolladores Node.js

3.2 Iniciar con NPM

Npm en la línea de comandos



etc

3.3 Paquetes propios y crear un servidor simple en Node

Sumamente modular

Incluir un módulo:

Require(‘http’)

Instalar paquetes

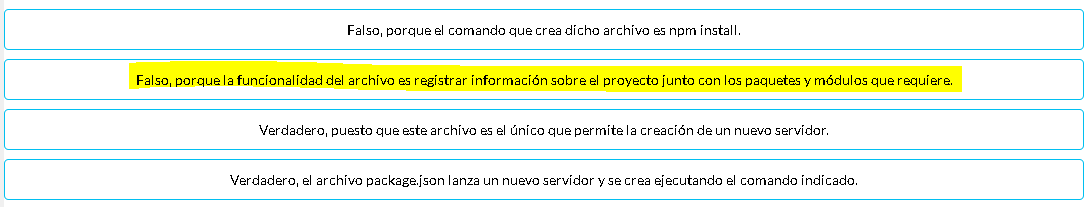


3.4 Paquetes NPM

3.5 Actividad

Tiempo de pensar

En esta lección tuviste la oportunidad de aprender sobre NPM y su función de administración de paquetes para Node. Saber usar esta herramientas en los proyectos de desarrollo implementados en Node, es fundamental puesto que en la mayoría de ocasiones tendrás la necesidad de usar paquetes de terceros en tu código.  
  
Instrucciones:  
  
Lee el enunciado a continuación y luego selecciona la respuesta correcta que indique si dicho enunciado es falso o verdadero y por qué.  
  
Enunciado:  
  
El archivo package.json se crea al ejecutar el comando “npm init” en el directorio del proyecto y sirve para lanzar un nuevo servidor.

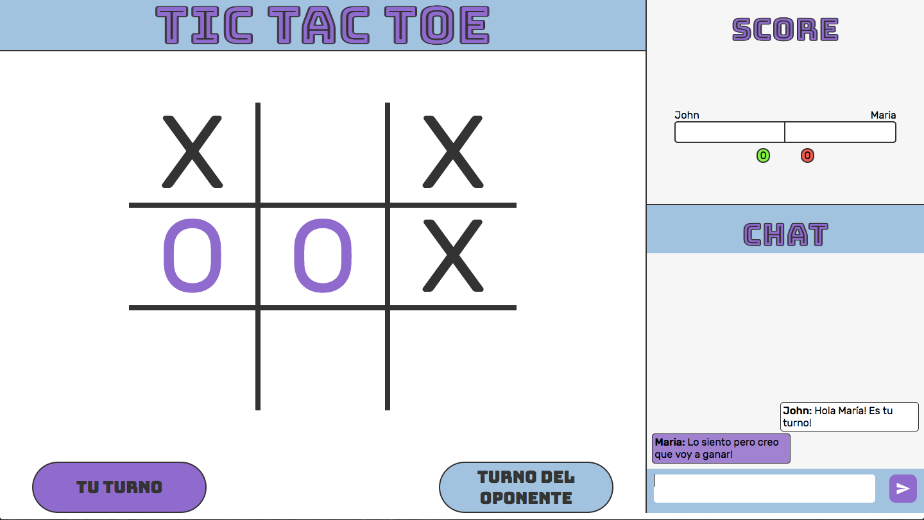


3.6 Ejercicio de Codificación

HORA DE  
**codificar**

45 MINUTOS

Node Package Manager es una herramienta muy útil al momento de crear proyectos web. Al realizar desarrollos del lado del servidor, podrás notar que siempre es necesario usar paquetes y módulos externos que complementan las funcionalidades a desarrollar. Adicionalmente NPM facilita la estructuración de los directorios y archivos que componen el proyecto, dando así orden y modularidad al desarrollo.  
  
En este ejercicio iniciaremos con el desarrollo de un juego en línea de Tic Tac Toe, que completaremos con los ejercicios de las siguientes lecciones. De este modo, la solución de cada ejercicio se convierte en la base sobre la cual debes trabajar en el siguiente. El front-end del proyecto ya está preparado y tu misión es desarrollar el Back-End con Node.js, haciendo uso de las herramientas que irás aprendiendo en esta unidad.



Contenido

Para el desarrollo de este ejercicio debes descargar el archivo base llamado codBase.zip. En él encontrarás una carpeta llamada src/ en la cual se encuentran los directorios client/ y server/, los cuales contienen los archivos del front-end y del back-end respectivamente. Adicionalmente encuentras el archivo scripts.txt que necesitarás para el desarrollo de este ejercicio.

[Descargar archivo base](https://s3.amazonaws.com/nextu-content-production/Desarrollador_Web/06_BackEnd/Ejercicios/WEB16S_C6U4L2_Ejercicio1/solucion/codBase.zip)

Instrucciones

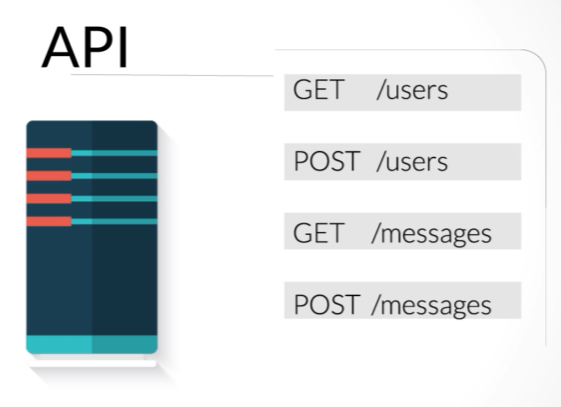
1. Crear una carpeta general en donde guardarás todo el proyecto y en su interior pega la carpeta src/ del código base.
2. Ejecutar el comando npm init desde la terminal, sobre la carpeta recién creada para generar el archivo package.json.
   * Instalar, usando npm, las siguientes dependencias:
   * babel-preset-es2015
   * babelify
   * browserify
   * express
   * jquery
   * socket.io
   * socket.io-client
   * watchify
3. Abrir el archivo package.json y en la propiedad scripts añadir las propiedades y valores indicados en el archivo scripts.txt. Recuerda usar el formato json para escribirlos adecuadamente.
4. Importar, en el archivo index.js dentro del directorio server, los siguientes módulos:
   * http
   * express
   * socket.io
5. Definir el puerto que usará el servidor y crear una aplicación express, en el mismo archivo del paso anterior.
6. Indicar a la aplicación express que los archivos estáticos están en la carpeta public.
7. Ejecutar el comando npm install para que se instalen todas las dependencias necesarias que aún no se hayan instalado.
8. Ejecutar el comando npm start y verificar que no han habido errores.  
     
   Si vas a desarrollar el proyecto en Windows ten presente que debes instalar:
   * babel-cli de manera global, usando el comando: npm install -g babel-cli
   * browserify de manera global, usando el comando: npm install -g browserify

Buenas Prácticas

Al momento de incluir módulos y paquetes para el desarrollo de las funcionalidades de un sistema, es muy recomendable separar los paquetes que se van a usar en la fase de desarrollo de los paquetes que se usarán en el proceso de producción, debido a que ésto evitará la instalación e inclusión de paquetes innecesarios en ambos ámbitos, y optimizando la carga del servidor.

1. Lección 3: Crear una aplicación Node.js
   1. Explicación y estructura del proyecto

Arquitectura: Ejemplo de un chat simple



Utilizando un JSON, lo más conveniente es usar una DB.

Estructura:



* 1. Construyendo la API
  2. Almacenamiento de los datos

/server/storage/index.js

Modulo filesystem que es parte de NodeJS

SaveData

getData

\_\_dirname 🡪 Función de NodeJS que almacena la ruta del directorio actual

Operador Ternario, ejemplo

var dataPath = dataType == 'users' ?

\_\_dirname + path.join('/data/users.json'):

\_\_dirname + path.join('/data/messages.json')



Promises 🡪 son una de las características de JS para el manejo de tareas asíncronas, es una forma de controlar los tiempos de ejecución.

Reject() 🡪 Cuando hay error

Resolve() 🡪 Cuando resuelve

* 1. Creando el frontend



Para detectar cuando un usuario envía un mensaje y renderizarlo en el HTML vamos a necesitar una serie de Funciones (en tiempo real), a través de la tecla enter o mediante el botón enviar:

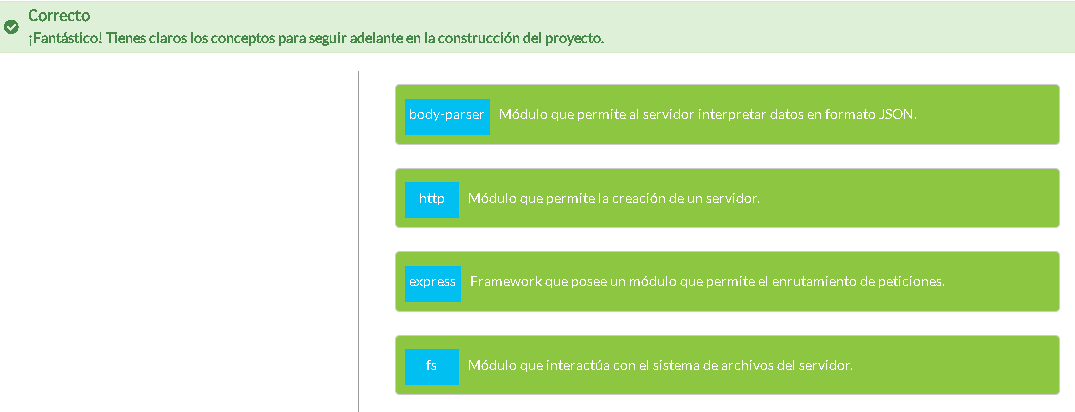
watchMessages()

renderMessages()

* 1. Integrando el front y back-end
  2. Actividad

Tiempo de pensar

En lo que llevas de esta lección has aprendido mucho. Hemos iniciado el desarrollo de un proyecto con Node y has podido conocer nuevos conceptos como las promesas y los enrutadores. Hasta ahora, hemos utilizado una serie de módulos tanto propios de Node como externos y es muy importante conocer claramente para qué sirve cada uno de ellos.  
  
Instrucciones:  
  
A continuación, verás a la izquierda una columna con un conjunto de módulos utilizados en Node para el desarrollo del proyecto que hemos iniciado y a su derecha la definición de cada módulo. Relaciónalos según corresponda:



* 1. Ejercicio de codificación

HORA DE  
**codificar**

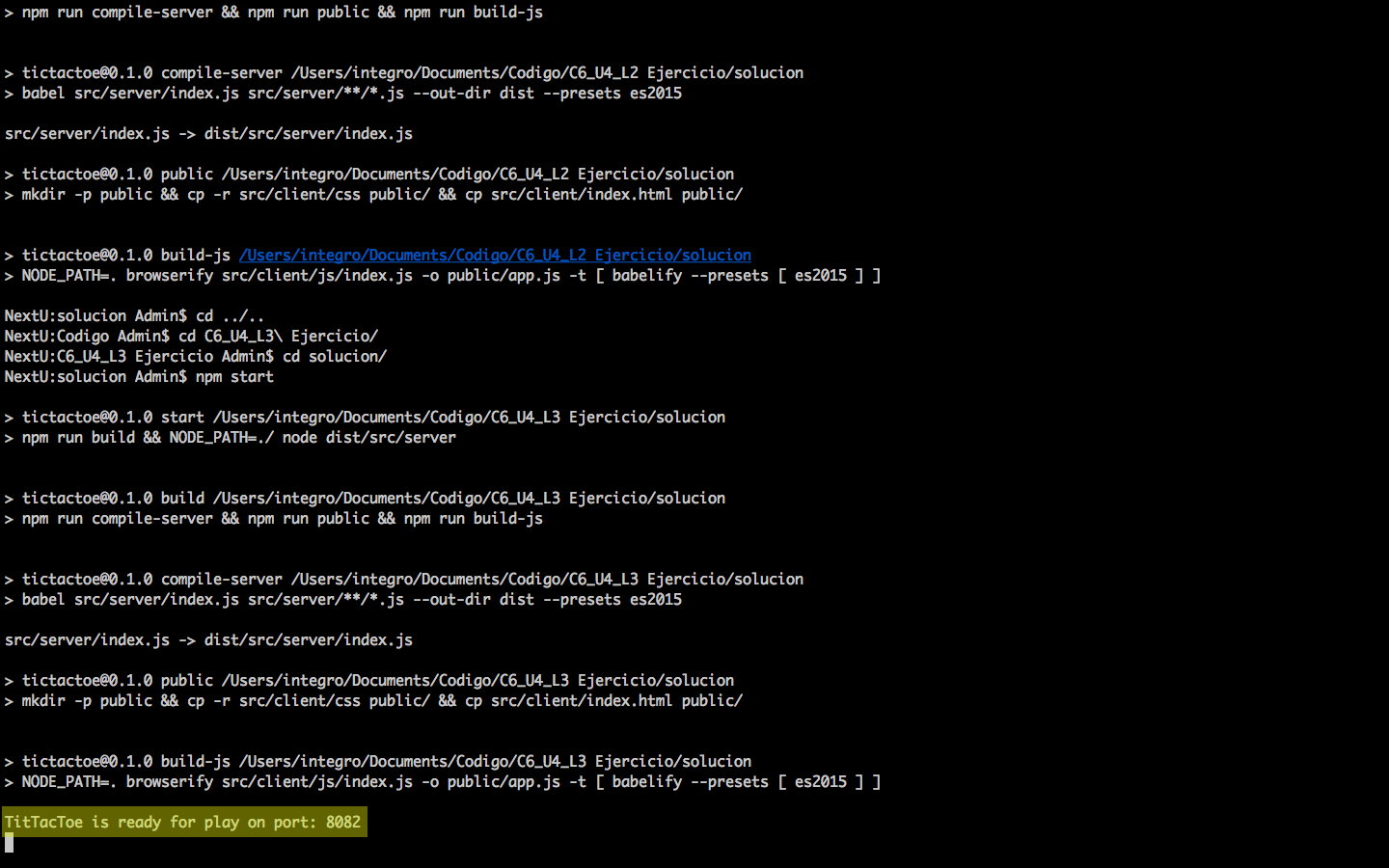
Usar Node.js como un entorno de ejecución, trae consigo muchas ventajas y permite crear aplicaciones que interactúan entre clientes y servidor de una manera muy fácil y rápida. Además es posible hacer uso de una gran cantidad de módulos propios o paquetes externos para complementar y añadir funcionalidades al proyecto. De esta manera es preciso conocer cómo se inicializan y usan los elementos de dichos módulos, para sacar su mejor provecho.  
  
En este ejercicio continuarás el desarrollo del videojuego en línea Tic Tac Toe, para lo cual necesitas el proyecto que iniciaste en el ejercicio de la lección anterior. En esta ocasión debes completar la preparación del entorno del servidor para poder iniciarlo y que esté listo para atender las peticiones.

Contenido

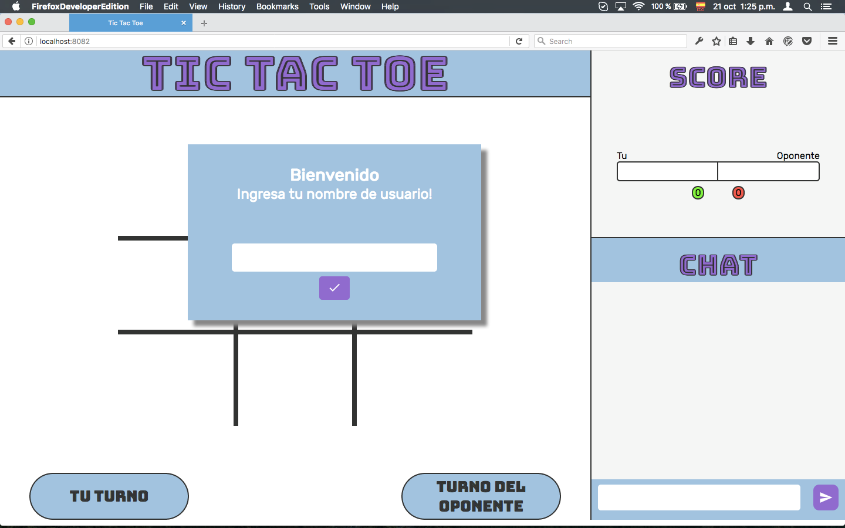
Para el desarrollo de este ejercicio debes continuar con el resultado del ejercicio de la lección anterior.

Instrucciones

1. Crea un servidor http basado en una aplicación express en el archivo **index.js** ubicado en la carpeta **server**.
2. Instancia un objeto de la librería Socket.io para que corra sobre el servidor creado en el paso anterior.
3. Ejecuta el servidor para que escuche por el puerto definido en la variable creada en el ejercicio de la lección anterior.
4. Imprime en consola, al lanzar el servidor, un mensaje que indique que éste se ha inicializado y que está escuchando por el puerto correspondiente.
5. Ejecuta el comando **npm start** sobre el directorio del proyecto y verificar que el servidor haya sido iniciado correctamente.



1. Verifica el funcionamiento correcto del servidor accediendo desde el navegador a localhost y el puerto definido. Debes acceder a la siguiente página:



Buenas Prácticas

Indicar el puerto por el cual correrá la aplicación es un proceso de mucho cuidado al que se debe prestar la atención merecida. En este ejercicio hemos definido el puerto en una variable dentro del código por facilidad y mejor comprensión del concepto del puerto y el servidor. Sin embargo, es muy recomendado que el puerto sea asignado desde una variable de entorno y no desde el mismo código que ejecuta el servidor, ésto con el fin de aumentar la seguridad del sistema y evitar que posibles atacantes conozcan fácilmente el puerto por el corre la aplicación.

1. Lección 4: Realtime
   1. Qué es el Realtime y su filosofía

¿Qué es el Real-Time?

En el ecosistema del desarrollo web existe una tendencia conocida como las aplicaciones real-time es decir que funcionan en tiempo real y no necesitas hacer click en botones ni ejecutar acciones para que se vean reflejados los cambios o se reciban las nuevas actualizaciones.  
  
El concepto de real time o tiempo real se basa en usar un conjunto de tecnologías y herramientas que nos permitan abrir un canal de comunicación entre el cliente y el servidor y que éste se mantenga abierto en todo momento, entonces no tenemos que preocuparnos por estar preguntándole al servidor si tiene nueva información que entregarnos, sino que cuando el servidor tiene nuevos datos se los notifica al cliente y viceversa.  
  
Los sistemas en tiempo real se caracterizan por sus tiempos de respuesta, que son inmediatos. Entre las aplicaciones más comunes que funcionan en tiempo real están los chats, notificaciones, mensajería, monitoreo de datos y procesos.

Filosofía de una aplicación Real-Time tipo CHAT

Imaginate a 3 personas conectadas al mismo sitio web, desde cualquier lugar del mundo. El sitio web es una sala de chat y las personas se encuentran en la misma conversación.  
  
Retomando el tema del modelo basado en eventos, supongamos que cada vez que un usuario envía un mensaje se emite un evento llamado “mensaje”.  
  
Entonces, todos los usuarios en línea están suscritos a ese evento: mensaje.  
  
Si pensamos en el flujo de la información, el usuario envía el mensaje y se emite el evento, éste llega al servidor y el servidor notifica a todos los usuarios suscritos que hay nueva información.  
  
Todo este proceso sucede en tiempo real, sin necesidad de que los otros usuarios estén enviando peticiones al servidor para preguntarle por nuevos mensajes que posiblemente hayan emitido otros usuarios.  
  
Aunque enviar peticiones, por ejemplo, cada segundo sea una opción válida y el envío de mensajes puede hacerse, esto sería malo a nivel de rendimiento ya que enviar tantas peticiones tan repetidamente puede causar que nuestra máquina se ponga un poco más lenta, aparte de esto, al segundo de intervalo entre cada petición debemos sumarle el tiempo que se demora en responder el servidor entre cada petición, uno o dos segundos adicionales. Por otro lado, si hubieran enviado varios mensajes en el intervalo del segundo todos los mensajes llegaron juntos y se renderizaron al mismo tiempo esto podría causar un efecto pésimo para la experiencia del usuario.  
  
Lo que se espera de este tipo de aplicaciones es que el flujo de datos y la comunicación entre los diferentes usuarios sea instantánea, sin intervalos, sin ejecutar acciones para pedir actualizaciones, muy transparente para el usuario y ante todo muy eficiente.  
  
En los proximos videos agregaremos la funcionalidad realtime a nuestro chat para poder comunicarnos con las otras personas.

* 1. WebSockets y Socket.io

Websocket 🡪 Son una tecnología que permite una comunicación constante entre el navegador del usuario (cliente) y el navegador, permitiendo enviar datos y recibir sin la necesidad de estar consultando

Socket.io 🡪 es una poderosa API de websockets para Node.js, donde la recepción y envío de datos se controlan por eventos, un modelo muy parecido al funcionamiento natural de NodeJs

Esta librería posee dos versiones:

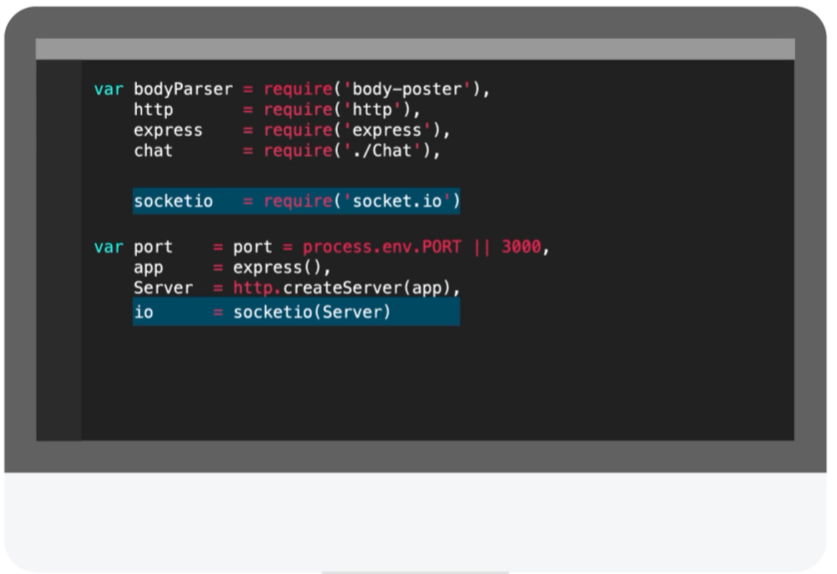
* Cliente: socket.io-client
* Servidor: socket.io

<http://socket.io> 🡪documentación completa y demos, entre otras cosas.

Función global llamada io. La cuál emite un evento llamado connection al servidor, si el servidor está bien configurado y está atendiendo a este evento, la conexión entre un lado a otro se realizará satisfactoriamente



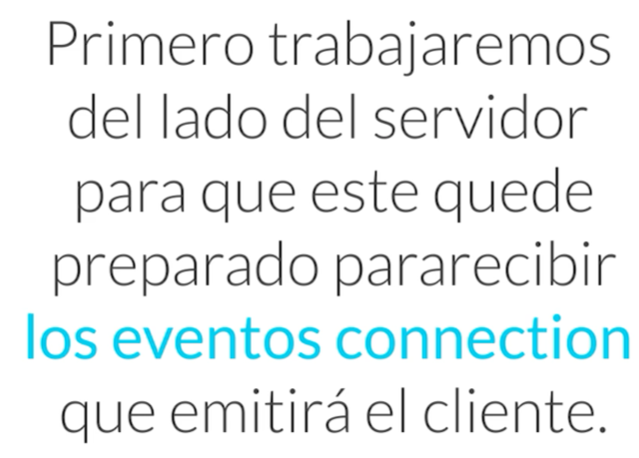
En el servidor debemos incluir la librería





El socket tiene un ID único

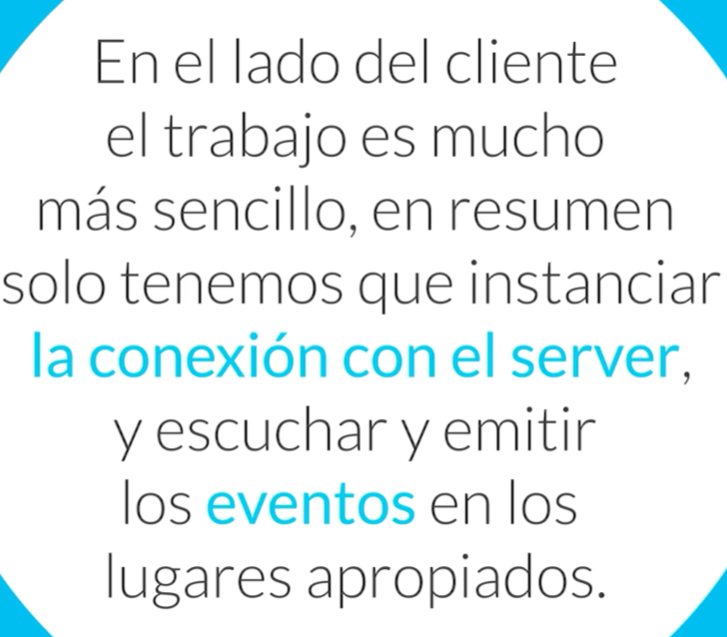
* 1. Agregar la funcionalidad Realtime al chat



Están presentes 3 eventos en un chat en el servidor:

* Cuando se uno un usuario 🡪 userJoin()
* Cuando se envía un nuevo mensaje 🡪 otroMessage()
* Y cuando un usuario abandona la conexión 🡪 Evento disconnect() 🡪 Por la librería cuando la conexión finaliza

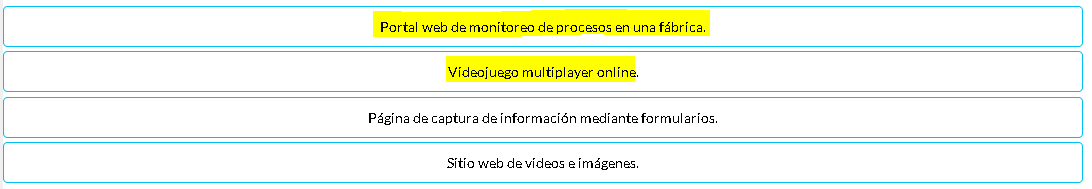
En el lado del cliente:



* 1. Actividad 1

Tiempo de pensar

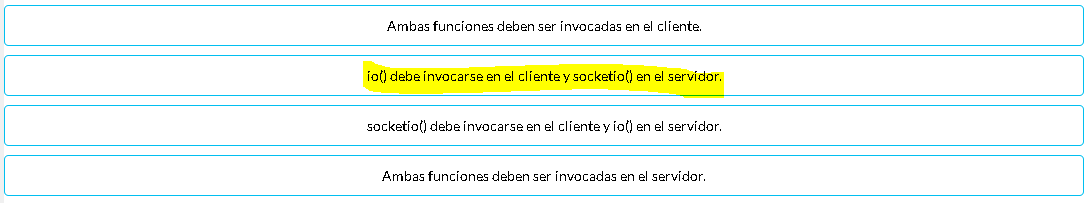
En los recursos de esta lección aprendiste el concepto de Realtime y cuán importante es saber utilizarlo en proyectos como el Chat que estamos desarrollando.  
  
Instrucciones:  
  
Selecciona dos respuestas correctas a la pregunta que se formula a continuación en relación con Realtime.  
  
Pregunta:  
  
Realtime consiste en crear un canal de comunicación entre el cliente y el servidor que permanece abierto pendiente de cualquier cambio. De acuerdo con lo anterior, ¿Cuáles son ejemplos de aplicaciones Realtime?



* 1. Actividad 2

Tiempo de pensar

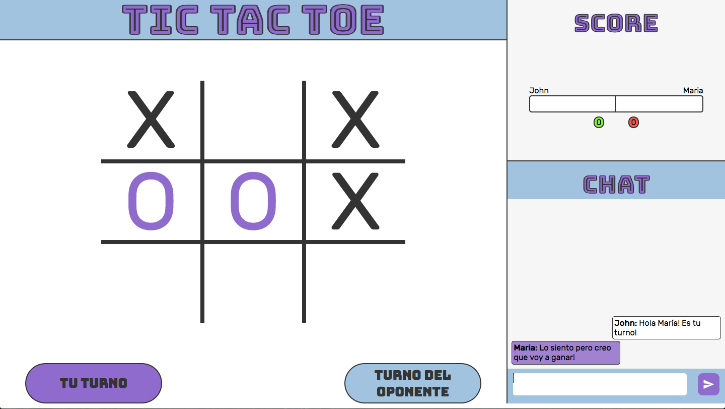
En este video conociste la librería Socket.io y el concepto de los WebSockets. Como pudiste observar, los WebSockets son la pieza fundamental que permite establecer puertos de comunicación contínua sin necesidad de hacer peticiones, lo que hace posible las aplicaciones Realtime.  
  
Instrucciones:  
  
Lee con atención la pregunta que se presenta a continuación y elige la respuesta correcta.  
  
Pregunta:  
  
Dos funciones fundamentales de la librería Socket.io para establecer un canal de comunicación son io() y socketio(). ¿Dichas funciones deben invocarse en el cliente o en el servidor?



* 1. Ejercicio de codificación

HORA DE  
**codificar**

60 MINUTOS

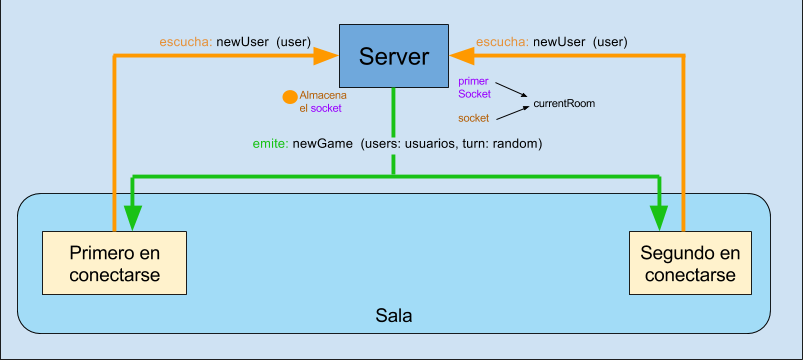
La conectividad en el mundo de la red, ha permitido la ejecución de que tienen efectos sobre varios clientes conectados a un mismo servidor. Este fenómeno se conoce como Realtime o ejecuciones en tiempo real, y presentan una amplia posibilidad de formas en que pueden dar un apoyo enorme al desarrollo de portales y sistemas de información web. Librerías como Socket.io dan vida a este concepto y permiten al desarrollador mediante su API de fácil uso, implementar ejecuciones en tiempo real.  
  
En este ejercicio continuarás el desarrollo del videojuego en línea Tic Tac Toe, para lo cual necesitas el proyecto que iniciaste en el ejercicio de la lección anterior. En esta ocasión debes terminar el proyecto, dotándolo de todas sus funcionalidades y conectividad en tiempo real.  
  


Contenido

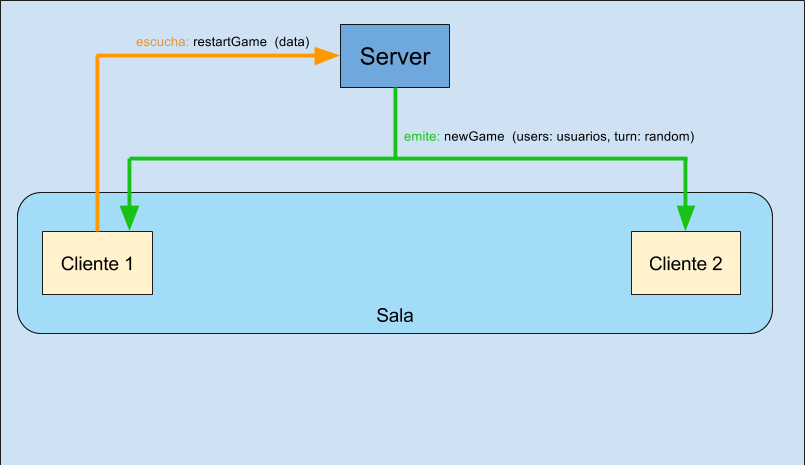
Para el desarrollo de este ejercicio debes continuar con el resultado del ejercicio de la lección anterior.

Instrucciones

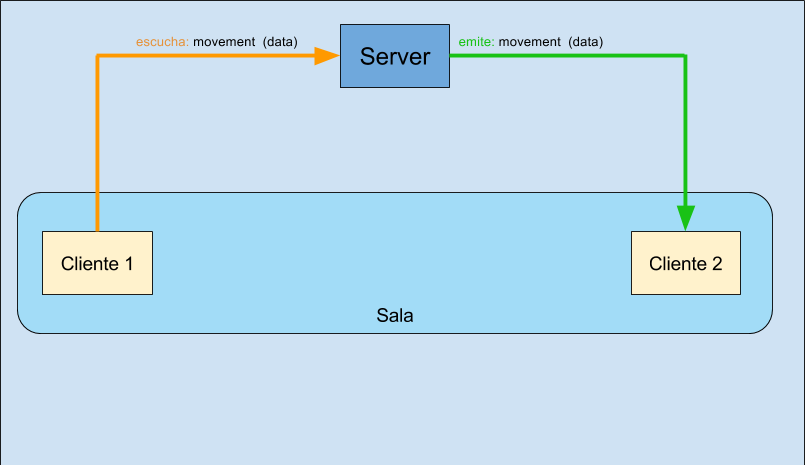
1. Disponer el objeto socket del servidor que se instanció en el ejercicio anterior, para que escuche las conexiones entre los posibles clientes y el servidor.
2. Mostrar un mensaje en consola, cuando se establezca una conexión, indicando que existe una nueva conexión al juego, junto con la propiedad id del objeto socket para mostrar el id del cliente que está enlazado.
3. Escuchar el evento ‘newUser’ que recibe un objeto como parámetro con el nombre del usuario que se ha conectado. Verificar si este usuario es el primero que se conecta al servidor, y si lo es, almacenar el socket por el que se ha conectado y el nombre del usuario en variables. Si el usuario que emite este evento, es el segundo en conectarse, se debe crear una variable llamada currentRoom que concatene los nombres de los dos usuarios conectados, esta variable será el identificador de la sala entre los dos usuarios. Se deben unir los sockets de ambos usuarios a la sala y por último emitir un evento ‘newGame’ enviando un arreglo con los nombres de los dos usarios en la propiedad users, y un número aleatorio entre 1 y 2 en la propiedad turn.



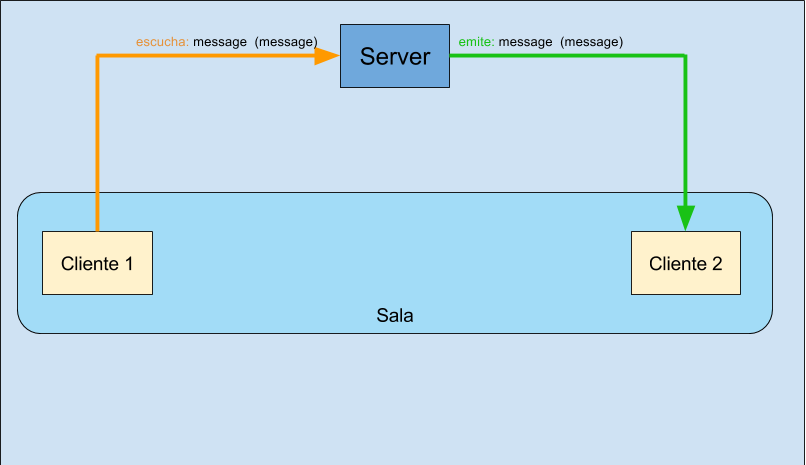
1. Escuchar el evento ‘restartGame’ que recibe un objeto que contiene la propiedad ‘users’ que corresponde a un vector con los nombres de los jugadores en la sala; y emitir a ambos miembros el evento ‘newGame’ enviando un objeto con un arreglo que contenga los nombres de los dos usuarios en la sala en una propiedad llamada users, y un número aleatorio entre 1 y 2 en una propiedad llamada turn.



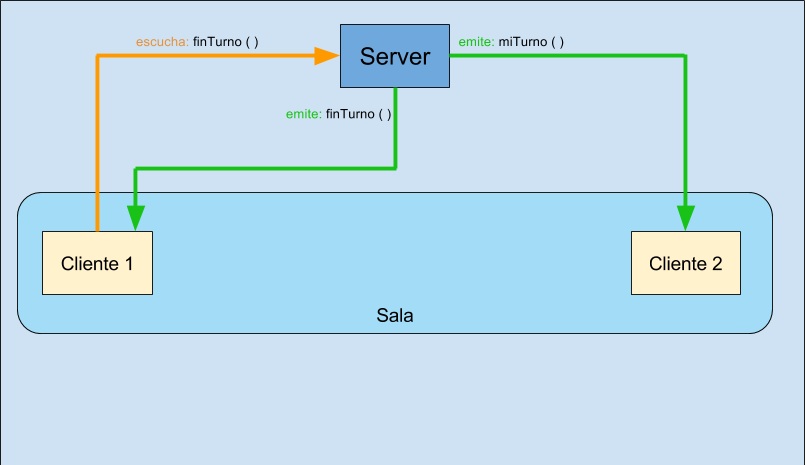
1. Escuchar el evento ‘movement’ que recibe un objeto como parámetro y emitir el evento ‘movement’, junto el objeto data recibido, sólo al otro miembro de la sala que no originó dicho evento en principio.



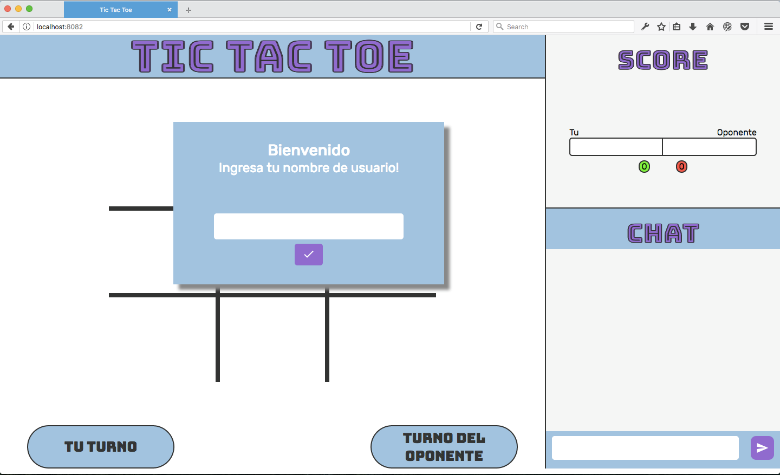
1. Escuchar el evento ‘message’ que recibe una cadena de caracteres con un mensaje y emitir el evento ‘message’ junto con la cadena recibida al otro usuario que no envió el evento.



1. Escuchar el evento ‘finTurno’ el cual no recibe ningún parámetro. Emitir el evento ‘finTurno’ al mismo socket que envió del que se envió el evento en principio; y emitir el evento ‘miTurno’ al otro usuario.



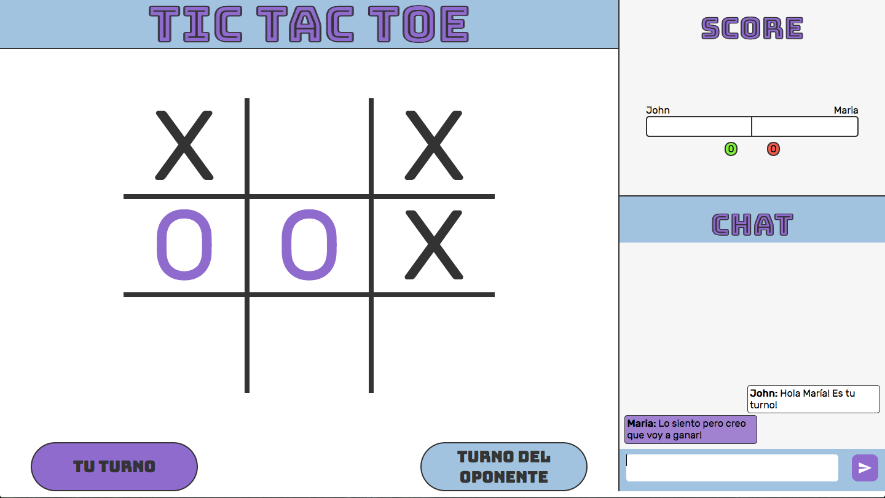
1. Verifica el funcionamiento correcto del videojuego iniciando el servidor con el comando npm start y luego conectándose desde el navegador al loclahost por el puerto que especificaste en el código. Dile a un amigo que se encuentre en tu misma red wi-fi que ingrese desde su computadora al navegador y digite la dirección IP de tu computador seguida del puerto que especificaste para correr el servidor.  
     
   Conectado desde mi máquina:



Conectado desde la máquina de un amigo:



1. Una vez conectados ambos clientes al servidor, digiten sus nombres y verifica el correcto funcionamiento del sistema jugando una partida de Tic Tac Toe con tu amigo. Además, prueba que el chat funcione correctamente y envíale un mensaje a tu amigo.



Buenas Prácticas

Al utilizar la librería Socket.io puedes tener salas de muchos usuarios para emitir y escuchar eventos. Sin embargo, es muy importante que cada sala sea creada mediante un identificador único el cual puedes generar dinámicamente mediante el id de los usuarios que ingresan a la conexión; esto te permitirá crear salas de un número controlado de usuarios, sin que se repitan sus identificadores y tengas problemas con la emisión y recepción de eventos.

1. Lección 5: ECMAScript 6
   1. [let, const](https://learn.nextu.com/mod/lesson/view.php?id=3545&pid=P_WEBDEV_V2)
   2. Arrow functions, lexical this
   3. Parametros por defecto, template string
   4. Clases, modulos, object/array destructuring
   5. Reescribiendo el proyecto en ES6
   6. Actividad
   7. Ejercicio de codificación
2. UNIDAD 4: Prueba