

## Crear un chat con el uso de Node JS y Mongodb

### Manual

*Hernández Martínez César Diego*

Versión: 0524

Fecha: 09/06/2021

[0524]

Queda prohibido cualquier tipo de explotación y, en particular, la reproducción, distribución, comunicación pública y/o transformación, total o parcial, por cualquier medio, de este documento.

## ÍNDICE

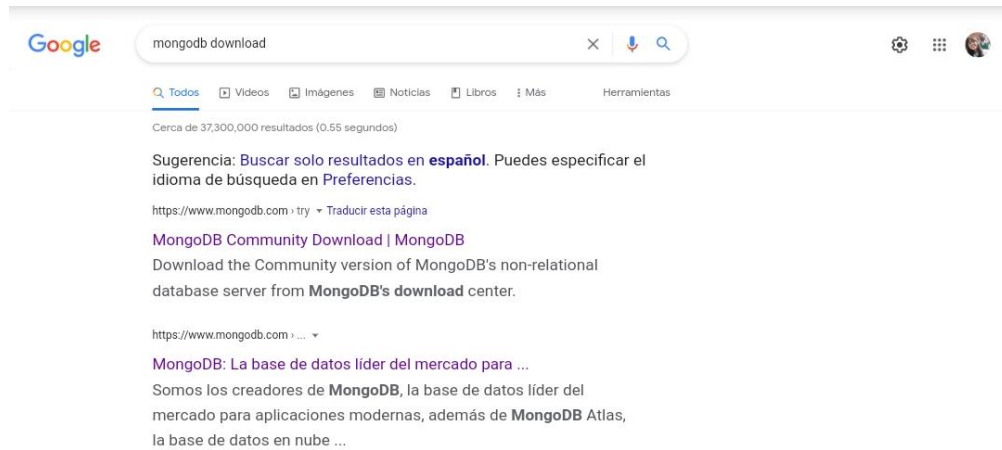
1 INSTALACION DE MONGODB.....	4
1.1 Descargar mongodb .....	4
1.2 Instalar mongodb .....	5
2 INSTALACION DE NODE JS. ....	6
2.1 Modificar la plantilla principal.....	6
2.2 Modificar informacion .....	7
3 CREACION DE CARPETAS DE NUESTRO PROYECTO.....	8
4 INSTALACION DE NODE JS Y DEMAS DEPENDENCIAS .....	9
4.1 Navegar hasta nuestro proyecto .....	9
4.2 Instalar JSON .....	10
4.3 Solucionar error de JSON .....	11
4.4 Instalar express.....	11
5 CREACION DE NUESTRO SERVIDOR.....	12
5.1 Importar librerías .....	12
5.2 Creación de variables .....	12
5.3 Llamar a mongoose .....	12
5.4 Conexión a base de datos .....	12
5.5 Llamada al funcionamiento de la base de datos.....	13
5.6 Creando archivos estadísticos .....	13
5.7 Iniciando el servidor.....	14
6 CREAR ARCHIVO SOCKETS .....	15
6.1 Crear archivo .....	15
6.2 Registrar usuarios.....	16
6.3 Mandar mensajes y mensajes privados .....	16
6.4 Desconexión de usuarios.....	17
7 CREAR Y CONFIGURAR EL ARCHIVO CHAT .....	18
7.1 Crear archivo chat .....	18
7.2 Configurar archivo .....	18
8 FUNCIONAMIENTO DE LA APP .....	19
8.1 Configurar archivo .....	19
8.2 Creación de eventos.....	19
9 CREAR EL DISEÑO DE NUESTRO CHAT .....	21
9.1 Crear archivo .....	21
9.2 Poner etiqueta JQuery .....	21
9.3 Poner etiqueta fontawesome.....	22
9.4 Etiqueta Bootstrap .....	23
10 DAR DISEÑO .....	25
10.1 Configurar body.....	25

11 CREAR ARCHIVO DEL ESTILO .....	27
11.1 Crear css .....	27
11.2 Editar archivo .....	27
12 CREAR MAS PAGINAS PARA UN MEJOR DISEÑO .....	28
12.1 Crear página usando Bootstrap.....	28
13 PROBAR NUESTRA PAGINA.....	30
14 SUBIR NUESTRA PAGINA A INTERNET .....	33
14.1 Registrarse en páginas.....	33

# 1 INSTALACION DE MONGODB

## 1.1 Descargar Mongo db.

- Para instalar mongodb debemos ir a la página oficial basta con poner en el buscador mongodb download y dar doble clic sobre el primer resultado que nos da.

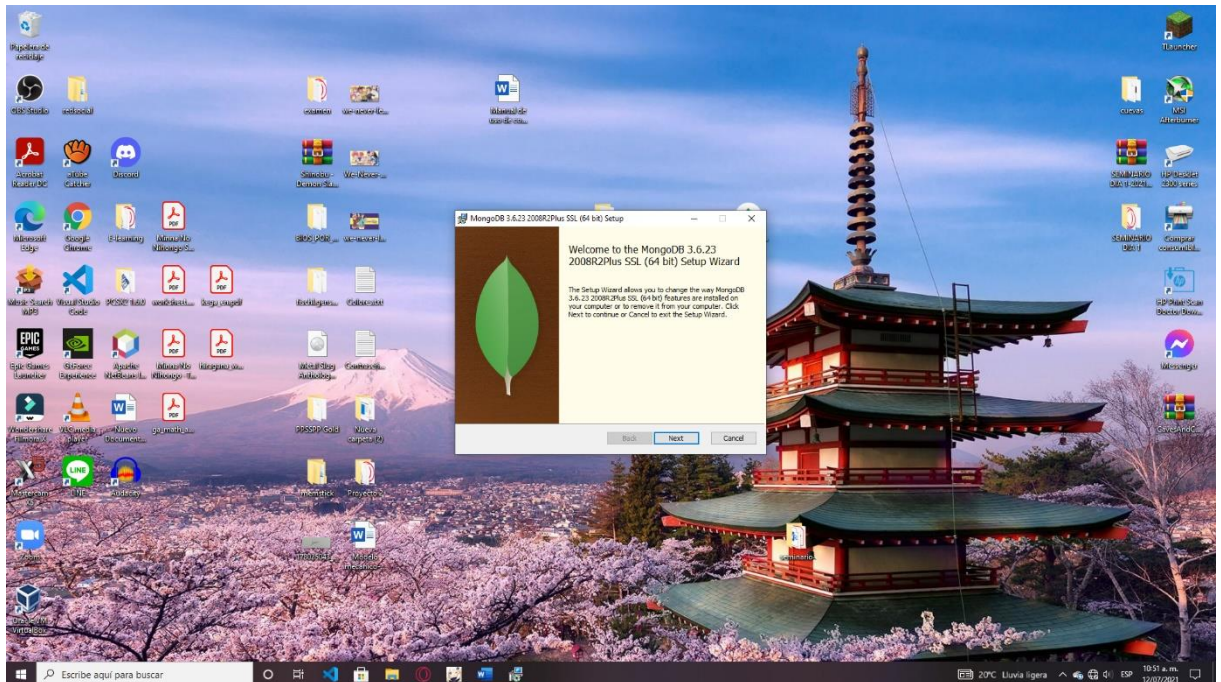


- Una vez estando dentro de la página nos dará la opción de descargar mongo para varias plataformas y diferentes versiones que existen de este mismo (descargaremos la versión que aguante nuestro equipo).



## 1.2 Instalación de Mongo db.

- Una vez descargado el mongodb procederemos a instalarlo como cualquier otro instalador.

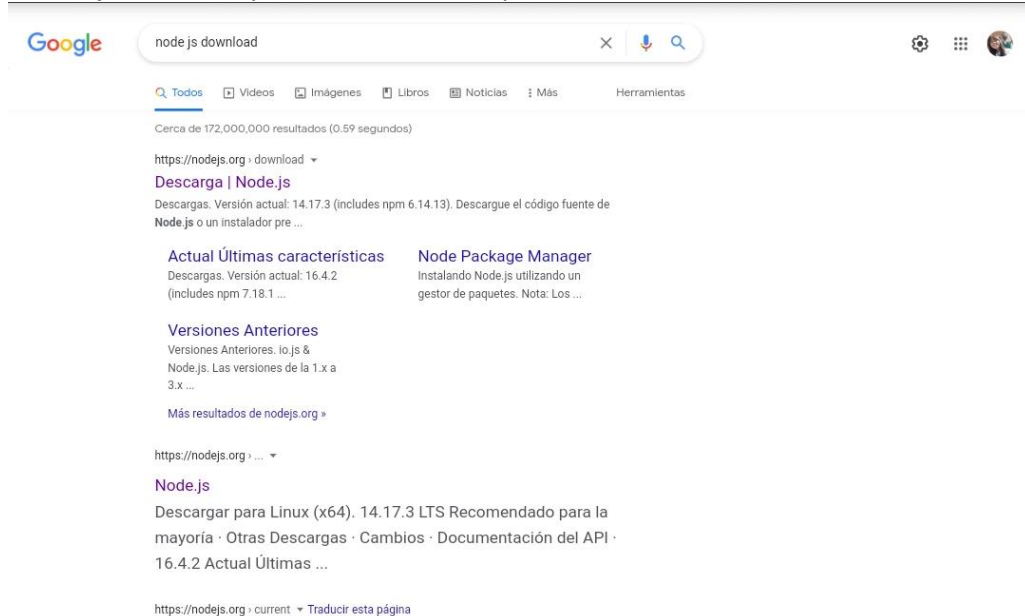


- Ahora un problema que nos puede surgir al instalar mongodb y que nos falle en algun futuro (cabe mencionar que hay excepciones y que podemos saltar este paso) para evitar esto debemos buscar la carpeta mongodb/server/la version de mongo que instalamos/bin dentro de nuestro disco duro (a continuacion mostrare un video sobre como encontrarlo).  
<https://youtu.be/GOydVurMvt0>
- Ya que tenemos ubicado la carpeta bin vamos a copiar esta dirección y vamos a pegarla en el siguiente directorio: en el icono de Windows y daremos clic derecho aquí vamos a seleccionar sistema vos va a abrir una nueva pestaña, dependiendo de la versión de Windows se ubicara una opción llamada opciones avanzadas del sistema aquí vamos a seleccionarla y nos mostrara otra ventana aquí buscaremos el botón llamado variables de entorno, nuevamente nos va a abrir una pestaña aquí en la sección de variables del sistema buscaremos la que lleve por nombre path la seleccionaremos y vamos a editarla, una vez dentro iremos hasta lo más bajo posible en uno de los espacios vacíos y ahí pegaremos nuestra dirección esto para Windows 10. En el caso de tener la versión 8 es casi lo mismo solo que al final nos va a mostrar una pequeña ventana con muchas direcciones juntas iremos al final y pondremos un ; seguido de nuestra dirección.  
<https://youtu.be/oirJCQm8VuU>
- Y con esto se terminaría con la instalación de Mongodb por el momento dejemos abierto estas ventanas.

## 2 INSTALACION DE NODE JS.

### 2.1 Descargar Node js.

- Otro elemento básico que si o si necesitaremos para comenzar a crear nuestro chat es Node js para poder descargarlo el proceso es similar a Mongo db solo bastara con buscar en el buscador Node js download y seleccionaremos el primer link.



- Dentro de esta página podremos seleccionar si queremos la versión de Node js para un sistema de 32 bits o una de 64 (descargaremos una u otra dependiendo de nuestro equipo).



#### Descargas

Versión actual: **14.17.3** (includes npm 6.14.13)

Descargue el código fuente de Node.js o un instalador pre-compilado para su plataforma, y comience a desarrollar hoy.

LTS	Actual	
Recomendado para la mayoría	Últimas características	
		
Instalador Windows	Instalador macOS	Código Fuente
node-v14.17.3-x86.msi	node-v14.17.3.pkg	node-v14.17.3.tar.gz
Instalador Windows (.msi)	32-bit	64-bit
Binario Windows (.zip)	32-bit	64-bit
Instalador macOS (.pkg)	64-bit	
Binario macOS (.tar.gz)	64-bit	
Binario Linux (x64)	64-bit	
Binario Linux (ARM)	ARMv7	ARMv8
Código Fuente	node-v14.17.3.tar.gz	

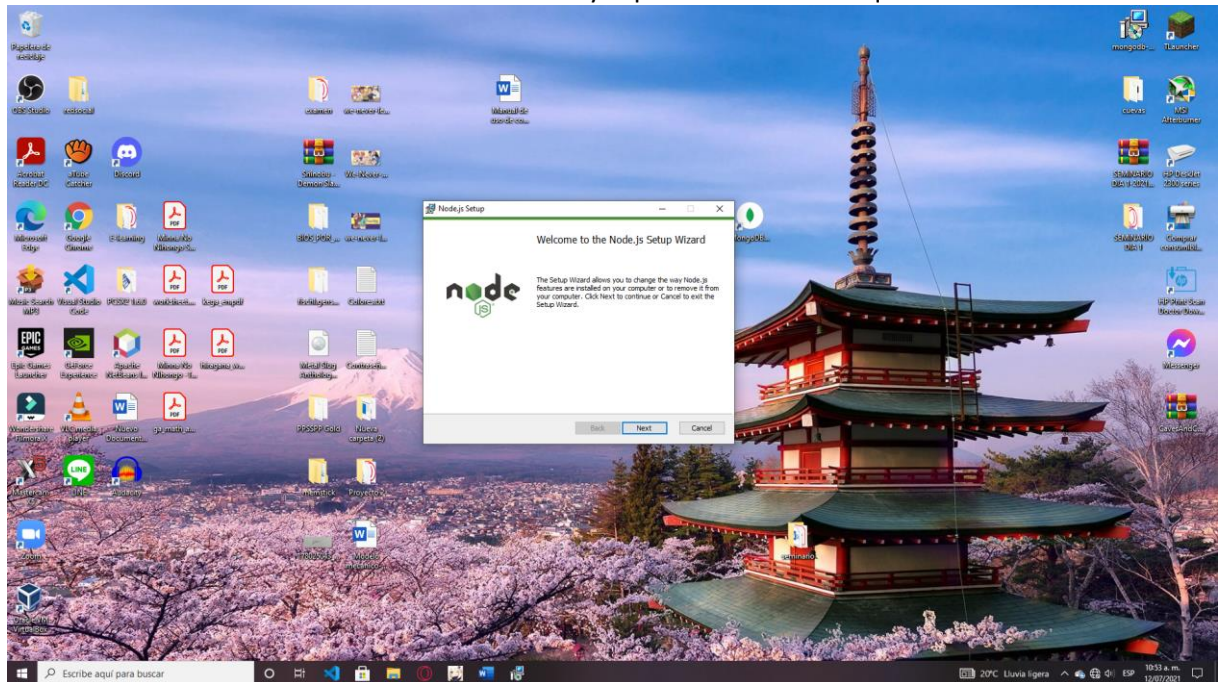
#### Plataformas adicionales

Imagen Docker	Imagen Docker Oficial Node.js
Linux en Power LE Systems	64-bit
Linux en System z	64-bit
AIX en Power Systems	64-bit



## 2.2 Instalación de Node js.

- Nuevamente la instalación es bastante sencilla ya que será como cualquier otro instalador.



- Para que nosotros podamos ocupar los comandos de Node js en el cmd debemos de hacer lo mismo que con mongodb solo que con encontrar la carpeta y meternos dentro con esto bastaría.

- Nuevamente vamos a copiar esta dirección y como ya tenemos abierta nuestra ventana de path bastara con poner la dirección antes seleccionada como se hizo con Mongodb.

[https://youtu.be/k\\_A1\\_gayP1M](https://youtu.be/k_A1_gayP1M)

- Ya con esto tenemos los dos programas para comenzar con la programación de nuestro chat.

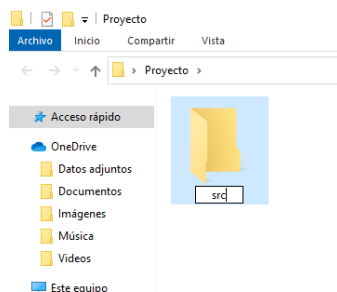
### 3 CREACION DE CARPETAS DE NUESTRO PROYECTO.

Comenzando con el proyecto primero vamos a crear todas las carpetas necesarias.

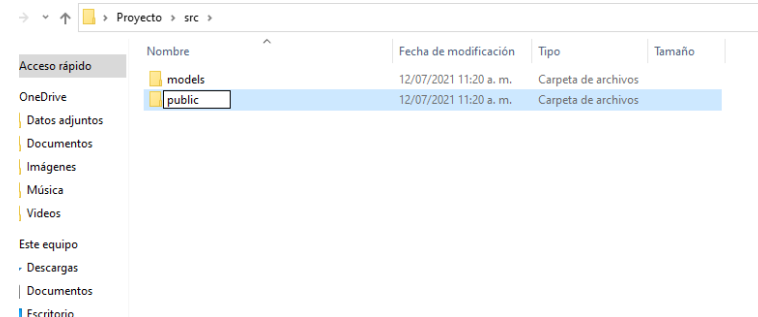
- Primero vamos a crear la carpeta principal podemos poner en nombre que nosotros queramos.



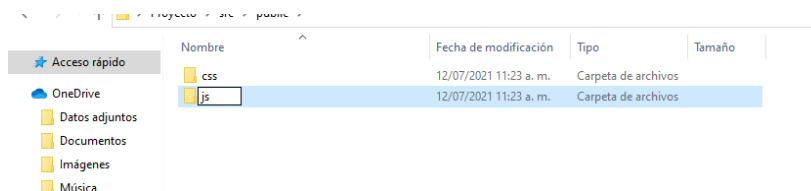
- Dentro de esta carpeta vamos a crear una nueva llamada src.



- Nuevamente nos metemos a la carpeta de src y dentro de aquí vamos a crear otras dos carpetas una llamada **public** y otra con el nombre de **models**.



- Y para finalizar dentro de la carpeta public vamos a crear otras dos carpetas mas una con el nombre js y otra con el nombre css.

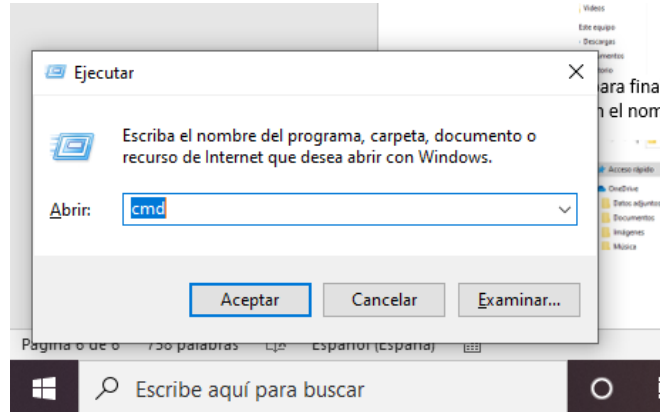




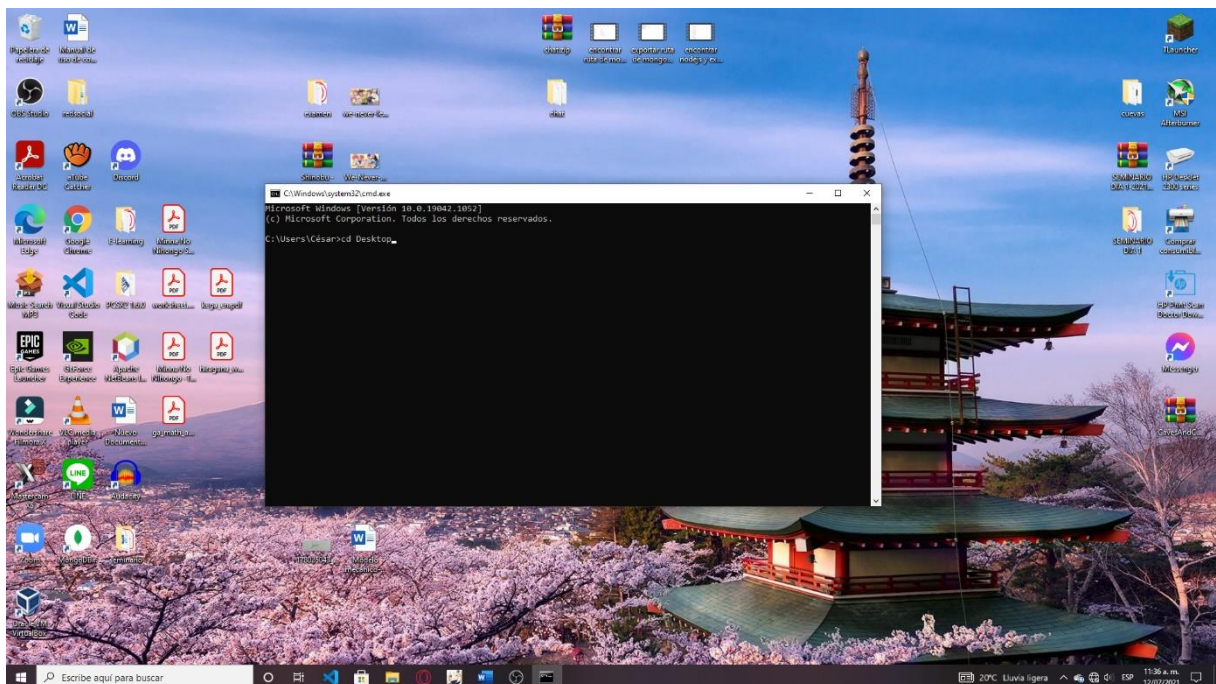
## 4 INSTALAR JSON Y DEMAS DEPENDENCIAS.

### 4.1 Navegar hasta nuestro proyecto.

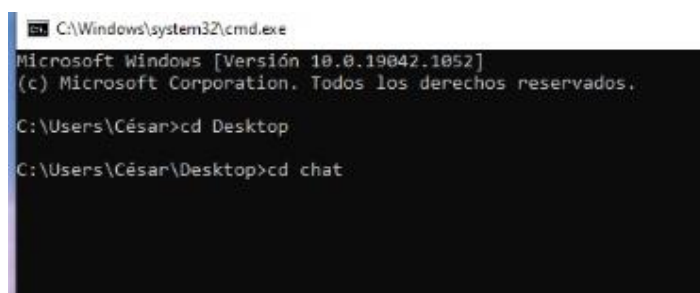
- Primero para poder instalar las dependencias en nuestro proyecto vamos a tener que abrir nuestro proyecto en un cmd, para abrir un cmd en el buscador que nos da Windows podemos escribirlo y dar enter o en otros casos presionar la combinación de teclas de Windows + R y aquí escribir cmd.



- Nos abrirá una nueva ventana en donde debemos navegar hasta la carpeta de nuestro proyecto con el comando **cd** seguido de la ruta por ejemplo mi proyecto se encuentra en el escritorio así que pondré **cd desktop** y daremos enter.



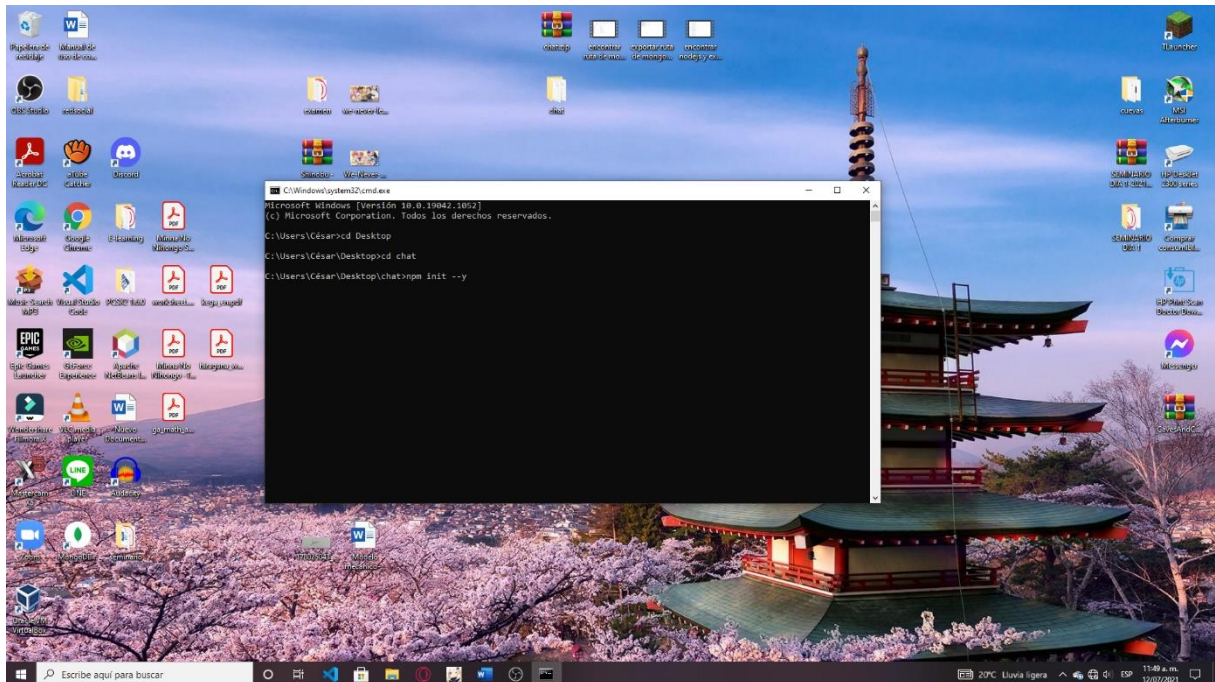
- Nuevamente ocuparemos en comando **cd** seguido del nombre de nuestro proyecto en mi caso **cd chat**.



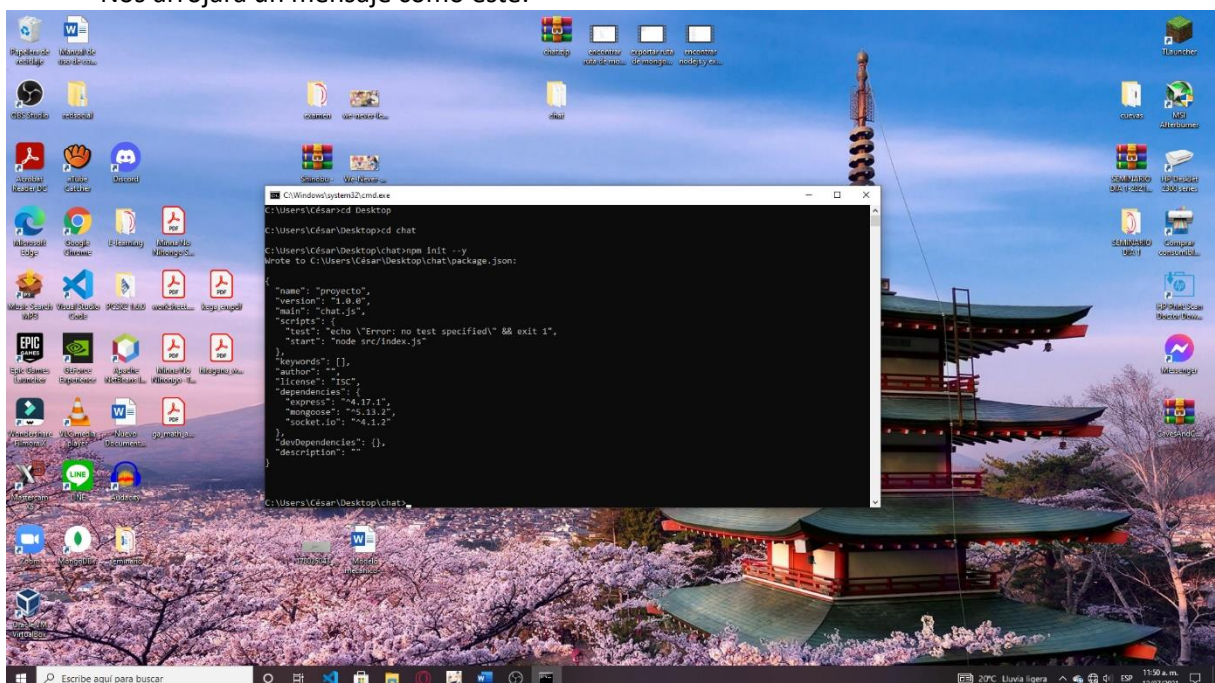
- Con esto ya estaremos dentro de nuestro proyecto y ahora si podemos instalar nuestras dependencias.

## 4.2 Instalar JSON.

- Para instalar la primera dependencia, en nuestro cmd vamos a poner **npm init --y** (Debo mencionar que para hacer esto debemos tener conexión a internet ya que estaremos descargando archivos).



- Nos arrojará un mensaje como este:

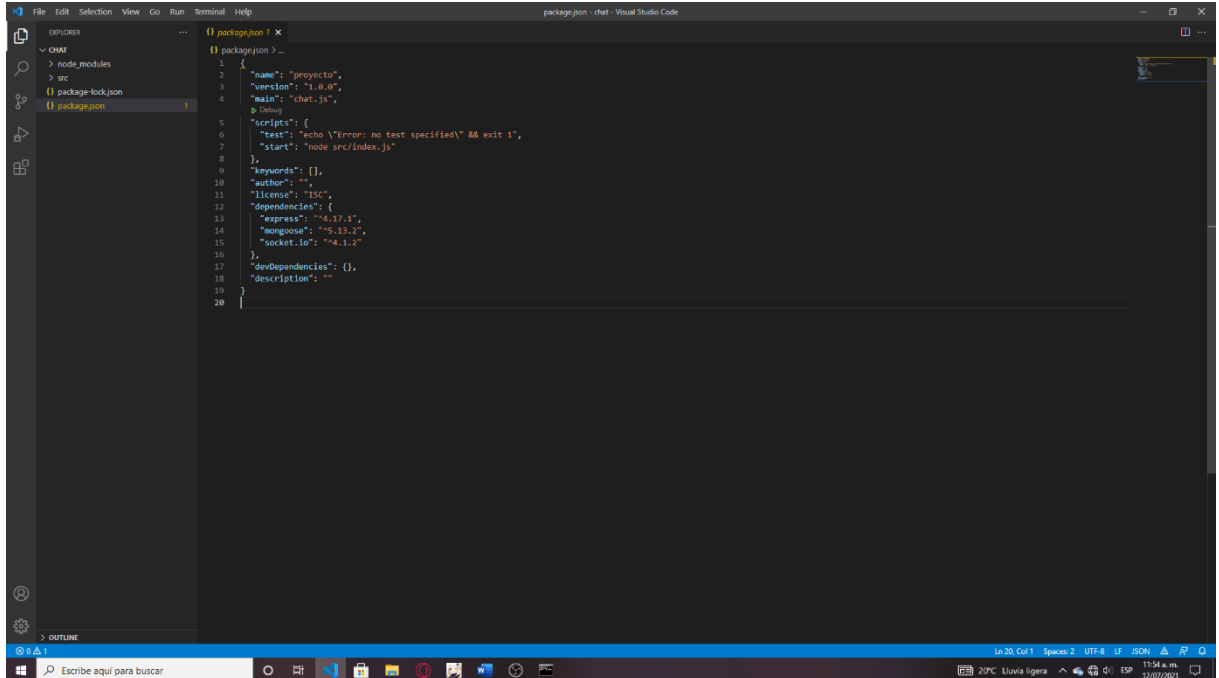


- Una vez instalado veremos que en la carpeta principal de nuestro proyecto se agregaron dos nuevos archivos con el nombre de **package.json** y de **package-lock.json**.



### 4.3 Solucionar error de JSON.

- En nuestro editor de texto favorito vamos a abrir el archivo con el nombre package.json en este tendremos que fijarnos que exista una línea de texto que inicie con “start” en el caso de no tenerla vamos a tener que escribirla manualmente como se muestra en nuestra captura (es posible que nos marque error, pero eso ya lo solucionaremos más adelante).

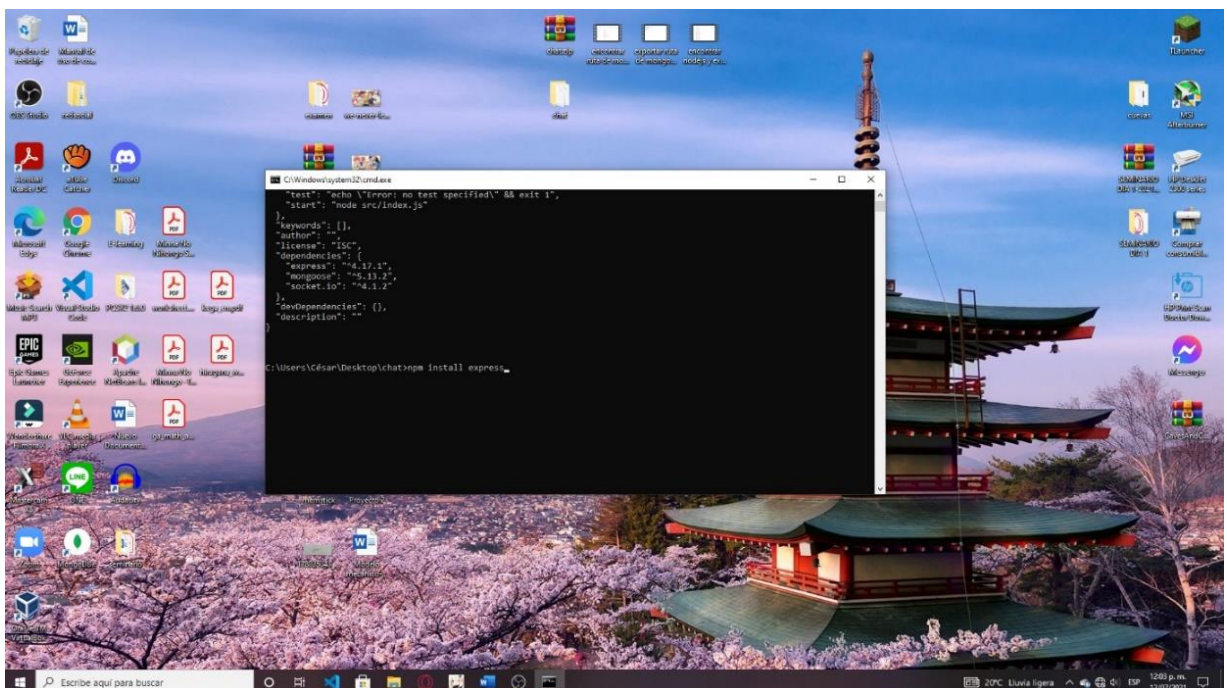


```
1 {
2   "name": "proyecto",
3   "version": "1.0.0",
4   "main": "chat.js",
5   "scripts": {
6     "test": "echo \\\"error: no test specified\\\" && exit 1",
7     "start": "node src/index.js"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "express": "^4.17.1",
14    "mongoose": "^5.13.2",
15    "socket.io": "^4.1.2"
16  },
17  "devDependencies": {},
18  "description": ""
19 }
```

- Este archivo se va a encargar de la configuración de nuestro proyecto

### 4.4 Instalar express.

- Una vez instalado y checado que no falte nada nuevamente vamos a nuestro cmd y vamos a escribir otra línea de código: **npm install express** este es un framework que se va a encargar de ayudar a escribir el código de nuestro server mucho más rápido.



## 5 CREACION DE NUESTRO SERVIDOR.

### 5.1 Importar librerías.

- En nuestro editor de texto vamos a crear un archivo nuevo dentro de la carpeta src con el nombre de index.js este se encargará de crear nuestro servidor.

 index.js 12/07/2021 12:10 p. m. Archivo JavaScript 2 KB

- Lo primero que vamos a hacer es exportar las librerías necesarias, exportaremos un total de 3.
- Para esto escribiremos la línea de código const “nombre de la librería” = require(“nombre de la librería”); (esto será para nuestras 3 librerías).

```
index.js  X
c > JS index.js > ...
1 //importacion de librerias
2 const http = require('http');
3 const path = require('path');
4 const express = require('express');
5
```

### 5.2 Creación de variables.

- Ahora mandaremos a llamar a una función y crear un servidor.

```
6 //creacion de variables
7 const app = express();
8 const server = http.createServer(app);
9
10
11 const socketio = require('socket.io')(server);
12
```

### 5.3 Llamar a mongoose.

- Como nosotros ya tenemos instalado mongodb debemos mandar a llamarlo y asignarle un puerto en este caso le asignaremos el puerto 8000.

```
18
19 app.set('port', process.env.PORT || 8000);
20
```

- Ahora solo le diremos que escuche al servidor con el siguiente código.

```
21 const io = socketio.listen(server);
22
```

- Bien ahora nosotros podemos ejecutar nuestro servidor en un cmd y en nuestro navegador abrirlo, pero por el momento no nos mostrará nada

### 5.4 Conexión a base de datos.

- Vamos a conectar nuestra aplicación a una base de datos de mongodb de manera local más adelante si nosotros queremos subir nuestra app a un servidor deberemos de modificar esta línea de código
- Para conectar nuestro proyecto a una base de datos debemos escribir mongoose.connect(“mongodb://127.0.0.1/chat-database”) la dirección 127.0.0.1 podemos también ponerla como localhost y el chat-database es el nombre de nuestra base de datos así que dependerá del nombre que le queramos poner

```
25 mongoose.connect('mongodb://127.0.0.1/chat-database')
```

- Bien por si la conexión a nuestra base de datos llegase a fallar vamos a usar un try y catch, pero en js el try cambia por un then y lo vamos a escribir debajo de la conexión a nuestra base de datos, en este caso los códigos quedando así:
- `.then(db => console.log('conectado a la base de datos'))`
- `.catch(err => console.log(err));`
- Console.log sirve para mandar un mensaje y en este caso then servirá para mandar un mensaje confirmando que la conexión a la base de datos fue exitosa y catch por si fallo la conexión

```
26 .then(db => console.log('conectado a la base de datos'))
27 .catch(err => console.log(err));
28
```

### ***5.5 Llamada al funcionamiento de la base de datos.***

- Una vez creada nuestra conexión ahora vamos a llamar a otro archivo .js que por el momento no tenemos creado así que es posible que nos mande algún error.
- Para esto usaremos la función require, pero también podemos ocupar la función socket.
- Y vamos a escribir `require('./sockets')(io);` el `./sockets` dependiera de donde tengamos ubicado este archivo y del nombre que le pusimos como yo lo tengo al mismo nivel que `index.js` entonces se ocupa un punto seguido de una diagonal si estuviera dentro de una carpeta entonces seria : `('nombre de la carpeta/nombre del archivo');`

```
31 require('./sockets')(io);|
32
```

### ***5.6 Creando archivos estadísticos.***

- Ahora lo que vamos a hacer es que vamos a mandar archivos estadísticos ocupando la siguiente linea de código:

```
34 app.use(express.static(path.join(__dirname, 'public')));
35
```

## 5.7 Iniciando el servidor.

- Bien anteriormente nosotros ya creamos un servidor, pero ahora lo que vamos a hacer es que nos mande un mensaje confirmando que el servidor se pudo iniciar correctamente.
- Para esto ocuparemos que al momento de la llamada de nuestro server nosotros agarremos el puerto y podamos verlo junto con un mensaje:

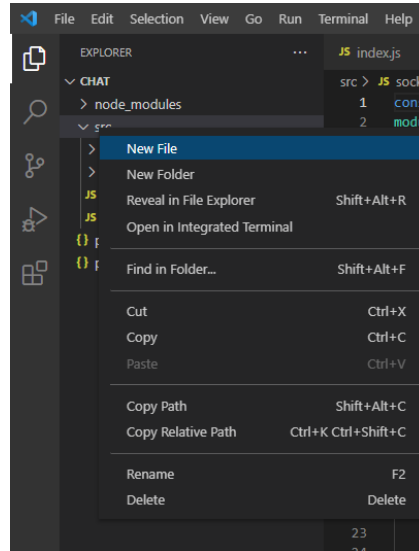
```
37  server.listen(app.get('port'),()=> {  
38    console.log("escuchando en el puerto", app.get('port'))  
39  });  
40
```

Y con esto finalizamos con la creación de nuestro server y conexión a mongo db ahora pasaremos a crear nuestro archivo que tendrá el funcionamiento de conectar usuarios.

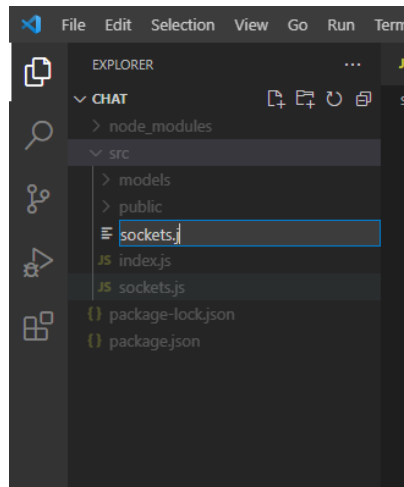
## 6 CREAR ARCHIVO SOCKETS.

### 6.1 Crear archivo.

- Nosotros vamos a crear el archivo sockets al mismo nivel donde tenemos el index.js para eso nosotros en nuestro editor de texto vamos a seleccionar la carpeta src y dar clic derecho ahí nosotros vamos a crear un nuevo archivo.



- A este le pondremos por nombre sockets.js



- Una vez creado nuestro archivo vamos a comenzar haciendo que agarre las variables de otro archivo y que le pondremos por nombre Chat.

```
1  const Chat = require('./models/Chat')
2  module.exports = function(io){
3    let users = {}
```

- En este caso nosotros estamos tomando modulos de que están en un archivo llamado Chat aquí es posible que nos marque error ya que no tenemos creado este archivo.



## 6.2 Registrar usuarios.

- Ahora vamos a hacer que nos registre y cuando esto pase en el cmd nos mandara un mensaje de nuevo usuario conectado también hará que una lista de usuarios se actualice al momento de que uno nuevo se registre

```
io.on('connection', async socket=>{
  console.log('Nuevo usuario conectado');
  let mensajes = await Chat.find({});
  socket.emit('mandaoldmsg', mensajes);

  socket.on('new user',(data,cb)=>{
    console.log(data);
    //if(users.indexOf(data) != -1){}
    if(data in users){
      cb(false);
    }else{cb(true);
      socket.usnam = data;
      users[socket.usnam] = socket;
      //users.push(socket.usnam);
      actualizaUsuarios();
    }
  });
});
```

- Los comandos que están con dos diagonales no se van a ejecutar solo es otra manera en que podría funcionar.

## 6.3 Mandar mensajes y mensajes privados.

- aquí vamos a hacer que nuestro chat nos permita mandar mensajes solo que también nos deja enviar mensajes vacíos si ningún carácter y mensajes privados, para los mensajes privados para eso es la función /w seguido de un espacio con el nombre del usuario y nuestro mensaje en el caso de mandar un mensaje a un usuario que no existe nos mandara una alerta de que el usuario no existe y si no pusimos ningún mensaje nos mandara otra alerta.

```
socket.on('send message', async (data, cb) => {
  var msg = data.trim();
  if (msg.substr(0,3) === '/w ') {
    msg = msg.substr(3);
    const index = msg.indexOf(' ');
    if(index !== -1){
      var name = msg.substring(0, index);
      var message = msg.substring(index + 1);
      if(name in users){
        users[name].emit('whisper',{
          message,
          nick: socket.usnam
        });
      }else{
        cb('Error, por favor ingresa tu nombre de usuario');
      }
    }else{
      cb('Por favor ingresa tu mensaje');
    }
  }else{
    var nuevoMsg = new Chat({
      usuario : socket.usnam,
      msg: msg
    })
    await nuevoMsg.save();

    io.sockets.emit('new message',{
      us: socket.usnam,
      msg
    });
  }
});
```

## 6.4 Desconexión de un usuario.

- Ahora solo nos falta que al momento de que nosotros salgamos de del chat el usuario se borre de la base de datos y este actualice nuestros usuarios conectados.

```
socket.on('disconnect', data=>{
  if(!socket.usnam) return;
  delete users[socket.usnam];
  //users.splice(users.indexOf(socket.usnam), 1);
  actualizaUsuarios();
});

function actualizaUsuarios(){
  io.sockets.emit('userss', Object.keys(users));
}

});
```

## 7 CREAR Y CONFIGURAR EL ARCHIVO CHAT.

### 7.1 Crear archivo chat.

- Para crear el archivo chat basta con hacer lo mismo que con el archivo sockets solo que esta vez vamos a crear nuestro archivo dentro de la carpeta models el archivo también tendrá la extensión .js

### 7.2 Configurar el archivo.

- Una vez creado el archivo vamos a crear dentro de este lo que llevará nuestra tabla que llevará por nombre tablaChat en este caso solo tendrá que guardar usuarios y mensajes del tipo string.


```
src > models > JS Chat.js > ...
1  const mongoose = require('mongoose');
2  const {Schema} = mongoose;
3
4  const tablaChat = new Schema({
5    usuario: String,
6    msg: String,
7    creacion: {
8      type: Date,
9      default: Date.now
10   }
11 });
12
13 module.exports = mongoose.model('Chat', tablaChat);
14
15 //end mongoose
```

## 8 FUNCIONAMIENTO DE LA APP.

- Vamos a comenzar a crear lo que serán los eventos que van a ocurrir dentro de nuestra página para ello vamos a crear otro archivo .js dentro de la carpeta js antes creada, el archivo llevara por nombre codigo.js

### 8.1 Configurar el archivo.

- Una vez creado el archivo vamos a comenzar editándolo.
- Primero vamos a crear una función:

```
src > public > js > JS codigo.js >  $() callback
1  $(function(){
2
3      const socket = io();
4
```

- Dentro de esta función vamos a obtener datos del dom, en este caso vamos a obtener el formulario de los mensajes, los mensajes y el cuerpo del chat

```
6
7      const formulario = $('#formulario-mensajes');
8      const mensaje = $('#mensajes');
9      const cuerpoChat = $('#cuerpoChat');
10
```

- Nuevamente vamos a obtener más datos solo que esta vez serán de nuestro login (login form, usuario, error de login y usuarios)

```
12
13      const loginform = $('#loginform');
14      const usuario = $('#usuario');
15      const loginError = $('#loginError');
16      const usuarios = $('#usuarios');
17
```

### 8.2 Creación de eventos.

- Ahora vamos a crear los eventos el primero se encargará del login:

```
19  loginform.submit(e =>{
20      e.preventDefault();
21      socket.emit('new user', usuario.val(), data=>{
22          if(data){
23              $('#contentlogin').hide();
24              $('#contenedor').show();
25              usuario.append(usuario);
26          }else{
27              loginError.html('<div class="alert-danger">este usuario ya existe</div>');
28          }
29      })
30      usuario.val('');
31  });
32
```

- Y por último terminaremos de cerrar nuestra función con ;

- Nuestro segundo evento será para que nuestros mensajes sean guardados y estos sean acomodados en forma de lista, también será para que usuarios nuevos puedan ver los mensajes que fueron enviados con anterioridad por otros usuarios, al igual que los mensajes que sea mandados por privado solo podrán ser vistos por la persona a quien se lo mandamos.

```

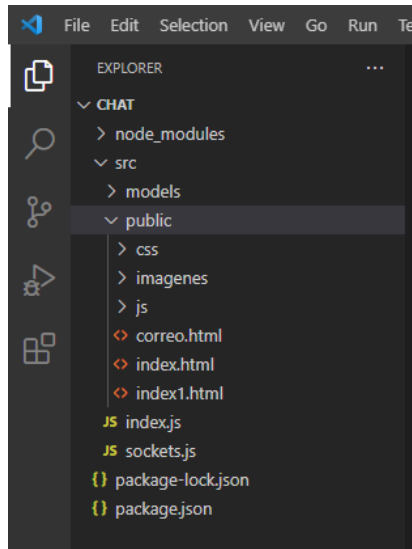
37 formulario.submit(e =>{
38     e.preventDefault();
39     socket.emit('send message', mensaje.val(), data =>{
40         cuerpoChat.append('<p class="error">'+ data+'</p>');
41     });
42     mensaje.val('');
43 });
44
45 socket.on('new message', function(data) {
46     cuerpoChat.append('<b>'+data.us+'</b>'+': '+data.msg+'<br>');
47 });
48
49
50 socket.on('userss', data => {
51     let html = '';
52     for (let i = 0; i < data.length; i++) {
53         html += '<i class="fas fa-user"></i>'+ data[i]+'<br>';
54     }
55     usuarios.html(html);
56 });
57
58 socket.on('whisper', data =>{
59     cuerpoChat.append('<b>'+data.nick+'</b>'+': '+<p class="whisper">'+ data.message + '</p><br><br>')
60 });
61
62 socket.on('mandaoldmsg', data =>{
63     for (var i = 0; i < data.length; i++) {
64         //cuerpoChat.append('<b>'+data.usuario+'</b>'+': '+<p class="whisper">'+ data.msg + '</p><br>')
65         muestraMsg(data[i]);
66     }
67 });
68
69
70 function muestraMsg(data){
71     cuerpoChat.append('<b>'+data.usuario+'</b>'+': '+<p class="whisper">'+ data.msg + '</p><br>');
72 }
73
74 }
75

```

## 9 CREAR EL DISEÑO DE NUESTRO CHAT.

### 9.1 Crear archivo.

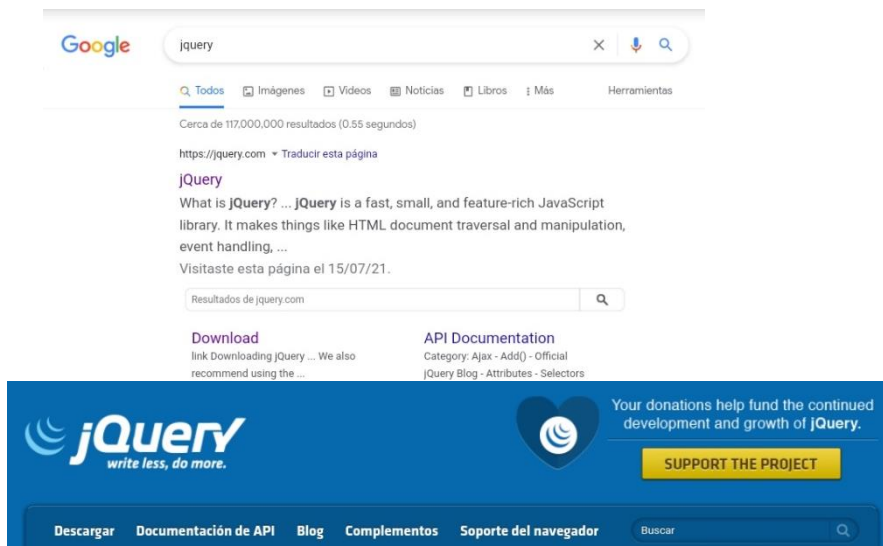
- Para comenzar vamos a crear un archivo con la extensión HTML en este caso se llamará index.html este archivo estará dentro de la carpeta public.



- Primero vamos a escribir html seguido de tabulador para que nos genere una estructura de html.
- En este caso dentro de la etiqueta title pondremos el nombre que mas no guste para el chat.

### 9.2 Poner etiqueta JQuery.

- Ahora debajo de title, pero dentro del head vamos a copiar unas etiquetas de internet en este caso de JQUERY y de una página llamada FONTAWESOME.
- En Google buscaremos JQUERY y en la primera pagina vamos a dar clic dentro de esta vamos a abrir la sección de descargas.



- Aquí vamos a donde dice usando JQuery con una CDN.



- Dentro de este vamos a buscar una liga: <https://code.jquery.com> y nos mandara a otra página.



- Una vez estando aquí en jQuery 3.x vamos a dar clic en minified y nos mandara el link que ocuparemos.



- Este lo vamos a poner dentro de nuestro head (para ocupar esto debemos de tener si o si una conexión a internet, aunque puede que después de la primera vez usándolo si nos desconectamos y refrescamos la página aun podremos ver los elementos que ocupamos por que se quedan almacenados en el cache).

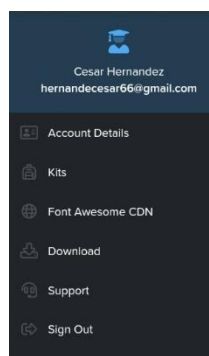
```

6
7
8 <!-- JavaScript Bundle with Popper -->
9 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFY1zcL8BNI+ntUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/twtIaxVM" cross
10

```

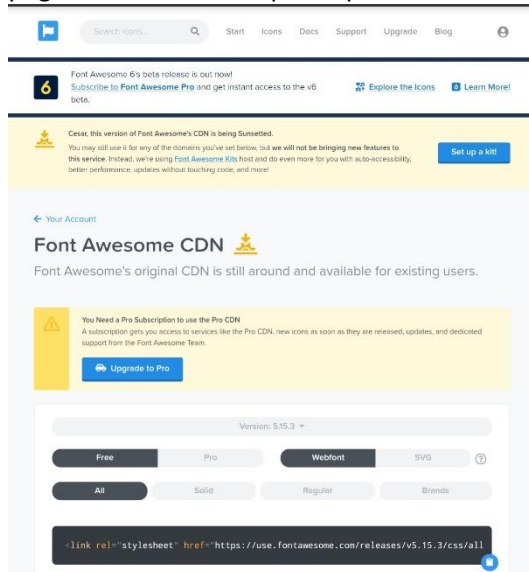
### 9.3 Poner etiqueta de fontawesome.

- Nuevamente en una pestaña de Google vamos a buscar Fontawesome y vamos a seleccionar la primera opción que nos salga.
- A diferencia de JQuery aquí vamos a tener que crearnos una cuenta.
- Una vez creada nuestra cuenta aquí vamos a dar clic en nuestro perfil vamos a fontawesome CDN.





- En esta página nos dará la etiqueta que necesitamos.



- Esta la vamos a pegar nuevamente dentro de nuestro head.

```
3 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.3/css/all.css" integrity="sha384-SZxX4wh179/gErwOYf+zWLeJdY/qpuqC4cAa9rOGUstPomtqpuNWT9wdPEn2Fk" crossorigin="anonymous">
4
5
```

## 9.4 Etiqueta Bootstrap.

- Ahora vamos a buscar Bootstrap documentación y daremos clic en la primera página.

Introducción · Bootstrap  
¡Hay una versión más nueva de Bootstrap 4! Inicio ·  
Documentación · Ejemplos · Temas · Expo · Blog. v4.1. Más reciente (4.1.x) ...  
Descargar · Tematización · JavaScript  
Visitaste esta página el 15/07/21.

- Una vez estando dentro en la sección JS vamos a copiar el primer script.

### JS

Muchos de nuestros componentes requieren el uso de JavaScript para funcionar. Específicamente, requieren jQuery, Popper.js y nuestros propios complementos de JavaScript. Coloque los siguientes <script>s cerca del final de sus páginas, justo antes de la </body> etiqueta de cierre, para habilitarlos. Primero debe aparecer jQuery, luego Popper.js y luego nuestros complementos de JavaScript.

Usamos la versión delgada de jQuery, pero la versión completa también es compatible.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-qJiW17zQYUA25HxIyUl6tKPQF7SGHfdQ1dN+PgLCH0" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZT7ju0IZdYXqYzVlPaYQ4Yq3G16U3YT96D187LPt1ZPc11ZUggfJd474wt" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-3Bp7Bq7huz7bOs1q5uBn61UQduY0AZhYW3MnOKV65OwXoFA3FsIv8ObXrPwUMAX" crossorigin="anonymous"></script>
```

- Este lo pondremos hasta abajo y dentro de nuestro body.

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3O3JyUwaoQ7nJy7v6wD6u4OjW6d3B8Qz2Lj4u0=" crossorigin="anonymous"></script>
<script type="text/javascript" src="/socket.io/socket.io.js"></script>
<script type="text/javascript" src="/js/codigo.js"></script>
```

- Aprovechando debajo de este vamos a escribir otros dos script donde agarremos nuestro archivo sockets.js y el archivo codigo.js que creamos con anterioridad.
- Sin salirnos de la página de Bootstrap en la sección de css vamos a copiar el script.

### CSS

Copie y pegue la hoja de estilo <link> en su <head> antes de todas las demás hojas de estilo para cargar nuestro CSS.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-TCSf1pe5HaaKsfRpPit1b4hAFeeh2XX7qhUupWPDYTM6V7P8Y8V838YRv0wX0yVT6o" crossorigin="anonymous">
```

- Y lo pegaremos dentro del head.

```
<!-- CSS only -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOML
```

- Vamos a agarrar nuevamente un archivo en este caso aun no lo tenemos creado mas adelante lo vamos a crear en este caso el archivo termina con .css

```
<link rel="stylesheet" type="text/css" href="css/estilo.css">
```

- Con todos los scripts puestos nuestro head quedaría así:

```
<head>
  <title>Intercambio Cultural</title>

  <!-- JavaScript Bundle with Popper -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integ

  <!-- CSS only -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="styles

  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.3/css/all.css" integrity

  <link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
```

- No vamos a cerrar la pagina vamos a dejarla abierta.

## 10 DAR DISEÑO.

### 10.1 Configura body.

- Lo primero que haremos es crear un simple header a este no le vamos a poner nada.

```
19 <body>
20   <header>
21 </header>
22 <section>
```

- Ahora vamos a crear un section dentro de este vamos a trabajar con navbar.

```
22 <section>
23   <nav class="navbar navbar-light bg-warning">
24     <a class="navbar-brand mx-auto" href="/index1.html">Chat de Intercambio Cultural
25   </a>
26 </nav>
```

- Este será un simple texto que nos mandará a la misma pagina en este caso index.html o cualquier otra página que queramos.
- Ahora vamos a crear un div este tendrá un titulo (H3) y también un botón y una barra donde podremos escribir nuestro nombre.

```
27 <div class="card col md-4 mt-5 mx-auto" id="contentlogin">
28   <div class="card-header">
29     <h3>Ingresa tu nombre de usuario:</h3>
30   </div>
31   <p id="loginError"></p>
32   <div class="card-body">
33     <form id="loginform">
34       <input type="text" id="usuario" class="form-control">
35       <button type="submit" class="btn btn-warning mt-2">Registrar</button>
36     </form>
37   </div>
38 </div>
```

- Ahora nosotros vamos a crear el arte visual del chat y lo primero que vamos a hacer es la parte de los mensajes nuevamente vamos a poner un titulo una card donde podamos ver los mensajes y un botón de enviar.

```
39 <div class="container" id="contenedor">
40   <div class="row">
41     <div class="col md-6 offset-md-1 mt-4">
42       <div class="card">
43         <div class="card-header">
44           <h4>Vamos a chatear</h4>
45         </div>
46         <div id="cuerpoChat" class="card-body"></div>
47         <form id="formulario-mensajes" class="card-footer">
48           <div class="input-group">
49             <input type="text" id="mensajes" class="form-control">
50             <div class="input-group-append">
51               <button type="submit" class="btn btn-warning">Enviar</button>
52             </div>
53           </div>
54         </form>
55       </div>
56     </div>
```

- Y para finalizar vamos a crear la parte donde se podrá visualizar los usuarios que se conecten.

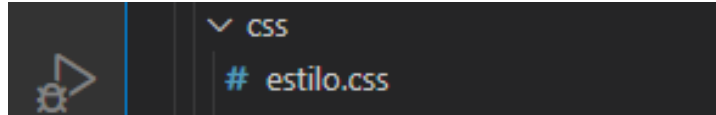
```
57     <div class="card col -md-3">
58         <div class="card-header">
59             <h3>Usuarios Conectados</h3>
60         </div>
61         <div class="card-body">
62             <div id="usuarios">
63
64             </div>
65         </div>
66     </div>
67 </div>
68 </div>
69 </section>
```

- Y con esto cerrara nuestro section.
- Si nosotros ejecutamos nuestra pagina veremos que todo esta junto y no se ve nada bien para ello vamos a crear nuestro archivo de estilo.

## 11 CREAR ARCHIVO DE ESTILO.

### 11.1 Crear css.

- Para finalizar vamos a crear nuestro archivo css en la carpeta de este nombre pondremos como nombre estilo.css



### 11.2 Editar archivo

- para finalizar vamos a escribir este código que hará que nuestro fondo de la página cambie de color y que nuestro primer div que creamos sirva como una pestaña que se oculte al nosotros registrar nuestro usuario.

```
1 body{
2   background: #1162ac; /*fallback for old browsers*/
3   background: -webkit-linear-gradient(left, red, cyan, lightgreen);
4   background: -webkit-linear-gradient(200px top, ellipse, white, #1162ac, black);
5 }
6
7 #contenedor{
8   display: none;
9 }
10
11 .error{
12   color: red;
13 }
14
15 .whisper{
16   color: grey;
17   font-size: italic;
18   display: inline;
19 }
20
```

## 12 CREAR MAS PAGINAS PARA UN MEJOR DISEÑO.

- Aquí ya dependerá del gusto de cada quien si nosotros queremos vamos a crear un nuevo index.html y modificando el nombre de nuestro primer index.html
- En caso de querer agregar una nueva página haremos lo siguiente.

### 12.1 Crear página usando Bootstrap

- Como anterior mente no cerramos la pagina de Bootstrap solo bastara con volver a abrirla y en la parte izquierda buscar **Componentes** aquí podremos ver un sinfín de cosas que podremos agregar a nuestra pagina como lo son: alertas, navbar, dropdowns entre otras cosas, cabe mencionar que también debemos pasar las librerías que en nuestro anterior archivo pegamos dentro del head.
- En mi caso hice una pagina inicial usando estos códigos:

```
index.html - chat - Visual Studio Code

index.html x
src > public > index.html > html > body > nav.navbar.navbar-expand-lg.navbar-light.bg-light > div#navbarSupportedContent.collapse.navbar-collapse > ul.navbar-nav.mr-auto > li.nav-item > a.nav-link

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Intercambio Cultural</title>
5
6 <!-- JavaScript Bundle with Popper -->
7 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-Mrcw6ZMFY1zcLA8NML+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
8
9
10 <!-- CSS only -->
11 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztTCQTWfSpd3yD65VohhpuuCOmLAS" crossorigin="anonymous">
12
13 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.3/css/all.css" integrity="sha384-SZxX4whJ79/gErwc0Yf+zWLeJdY/qpuqC4cAa9r0GUstPomtqpuNWT9wdPen2fk" crossorigin="anonymous">
14
15 <link rel="stylesheet" type="text/css" href="css/estilo.css">
16
17 </head>
18
19
20 <body>
21   <div class="col-md-3"></div>
22   <nav class="navbar navbar-expand-lg navbar-light bg-light">
23     <a class="navbar-brand" href="index1.html">Chat</a>
24     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
25       <span class="navbar-toggler-icon"></span>
26     </button>
27   <div class="col-md-3"></div>
28   <div class="collapse navbar-collapse" id="navbarSupportedContent">
29     <ul class="navbar-nav mr-auto">
30       <li class="nav-item active">
31         <div class="col-md-3"></div>
32         <a class="nav-link" href="index1.html">Chat <span class="sr-only"></span></a>
33       </li>
34       <div class="col-md-3"></div>
35       <li class="nav-item">
36         <a class="nav-link" href="https://m.youtube.com/channel/UC2Fkx0paZSe2ewIyNSITW-g">Youtube</a>
37       </li>
38       <div class="col-md-3"></div>
39       <li class="nav-item">
40         <a class="nav-link" href="https://www.facebook.com/elshidoris/">Contactanos</a>
41       </li>
42     </ul>
43     <div class="col-md-3"></div>
44     <li class="nav-item">
45       <a class="nav-link" href="mailto:mandanosuncorreo@gmail.com">Mandanos un correo</a>
46     </li>
47   </div>
48   <div class="col-md-5"></div>
49   <form class="form-inline my-2 my-lg-0">
50     <input class="form-control mr-sm-2" type="search" placeholder="Buscar" aria-label="Search">
51   </form>
52 </body>
53 </html>
```

```

src > public > index.html > html > body > nav.navbar.navbar-expand-lg.navbar-light.bg-light > div#navbarSupportedContent.collapse.navbar-collapse > ul.nav
48 |         <form class="form-inline my-2 my-lg-0">
49 |             <input class="form-control mr-sm-2" type="search" placeholder="Buscar" aria-label="Search">
50 |             <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Buscar</button>
51 |         </form>
52 |     </div>
53 | </nav>
54 | <!-- tarjetas -->
55 | <p></p>
56 | <div class="card" style="width: 30rem;">
57 |     
58 |     <div class="card-body">
59 |         <h5 class="card-title">Conocenos</h5>
60 |         <p class="card-text">Descubre más sobre Nosotros en nuestra sección Conócenos. Tenemos las respuestas a tus preguntas.</p>
61 |         <a href="https://m.twitch.tv/frontcesard12/profile" class="btn btn-primary">Ir al sitio web</a>
62 |     </div>
63 | </div>
64 | <p></p>
65 | <div></div>
66 |
67 | </body>
68 | </html>

```

- Toda la personalización quedara a nuestro gusto, como por ejemplo yo solo puse un navbar y una card todos estos con links a mis redes sociales.



### 13 PROBAR NUESTRA PAGINA

- Para comenzar con a pruebas de nuestra pagina lo que haremos es abrir 3 ventanas de cmd.
- En nuestra primera ventana escribiremos mongod.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\César>mongod_
```

- en el segundo cmd vamos a monitorear nuestra base de datos para ello solo pondremos mongo.

```
C:\Windows\system32\cmd.exe - mongo
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\César>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c6172ada-4ea0-40e9-88f2-d9cbf2a9eaa") }
MongoDB server version: 4.4.6
```

- Y en nuestra tercera ventana de cmd vamos a navegar hasta nuestro proyecto con el comando cd seguido del nombre de donde se encuentre por ejempló si esta en el escritorio vamos a poner desktop y está en documentos pondremos documents.
- Ahora solo vamos a poner cd seguido del nombre del proyecto en mi caso sería cd chat

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\César>cd Desktop

C:\Users\César\Desktop>cd Proyecto

C:\Users\César\Desktop\Proyecto>cd chat

C:\Users\César\Desktop\Proyecto\chat>_
```

- Y ya para finalizar vamos a poner npm start indicando que inicie el servidor.
- Nos tendrá que mandar estos mensajes

```
C:\> npm
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\César>cd Desktop

C:\Users\César\Desktop>cd Proyecto

C:\Users\César\Desktop\Proyecto>cd chat

C:\Users\César\Desktop\Proyecto\chat>npm start

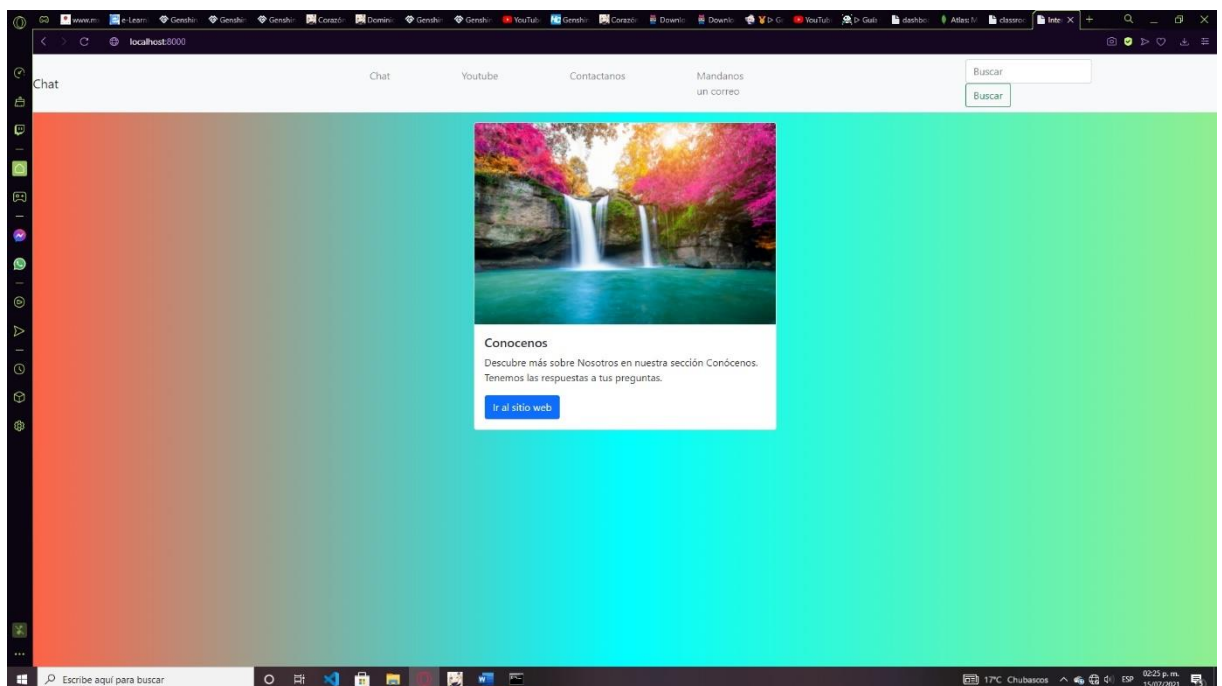
> proyecto@1.0.0 start C:\Users\César\Desktop\Proyecto\chat
> node src/index.js

(node:1060) DeprecationWarning: current URL string parser is
(node:1060) [MONGODB DRIVER] Warning: Current Server Discover
o the MongoClient constructor.
escuchando en el puerto 8000
conectado a la base de datos
```

- Ahora en nuestro navegador vamos a escribir localhost: el puerto que le pusimos en mi caso quedaría como localhost:8000.



- Y veremos que exitosamente nuestra pagina cargara ahora solo quedara de hacer la pruebas para ver que todo realmente funcione.



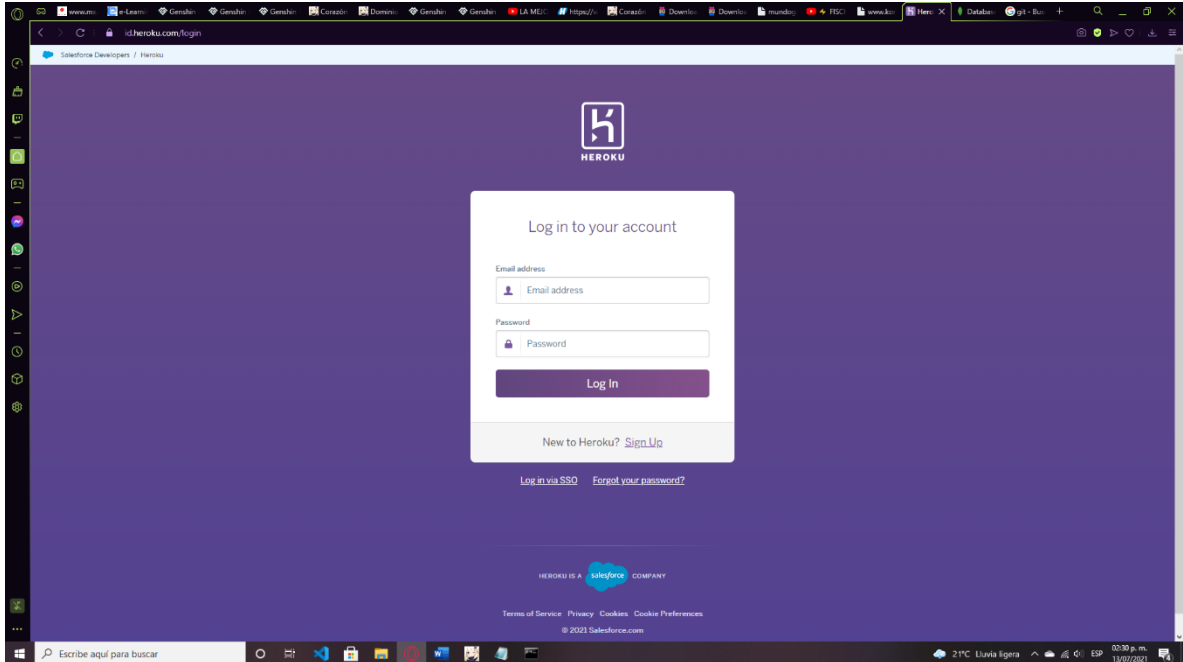
- La primera vez que nos registremos en mongodb se creara nuestra base de datos solo basta que en el cmd donde pusimos mongo pongamos **show dbs y veremos** que nuestra base de datos fue creada.

```
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
> show dbs
admin              0.000GB
chat-database     0.000GB
config             0.000GB
local              0.000GB
> _
```

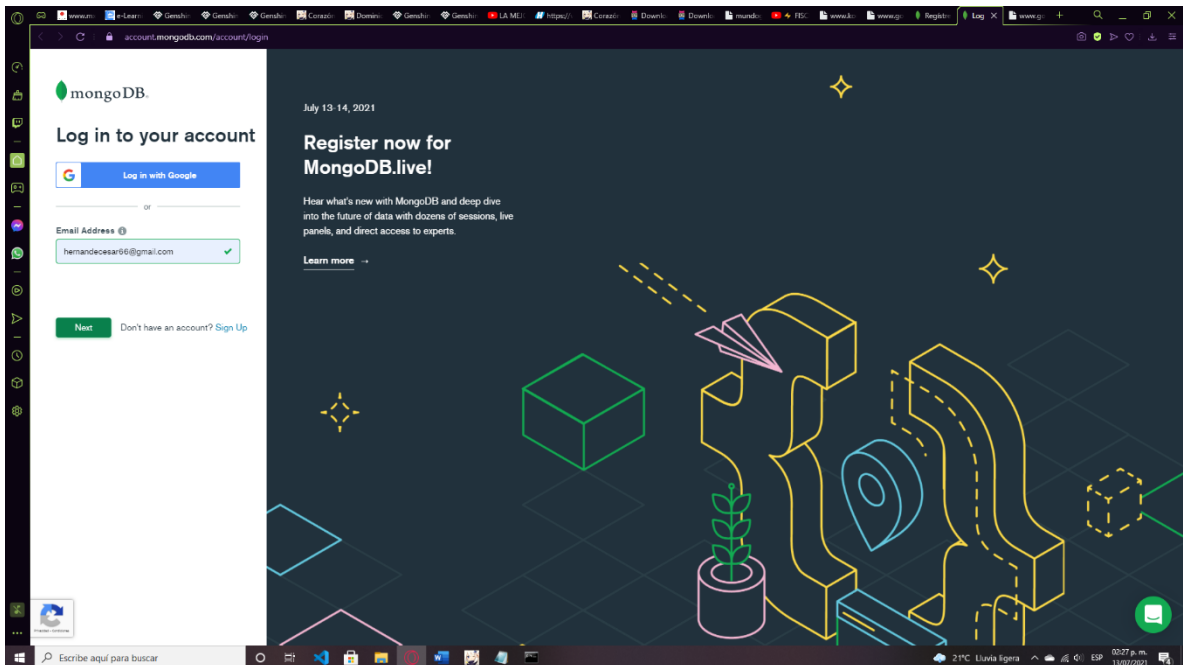
## 14 SUBIR NUESTRA PAGINA A INTERNET

### 14.1 Registrarse en páginas.

- Para subir nuestra página lo que debemos de hacer es registrarnos en 2 páginas que son:
- Heroku:

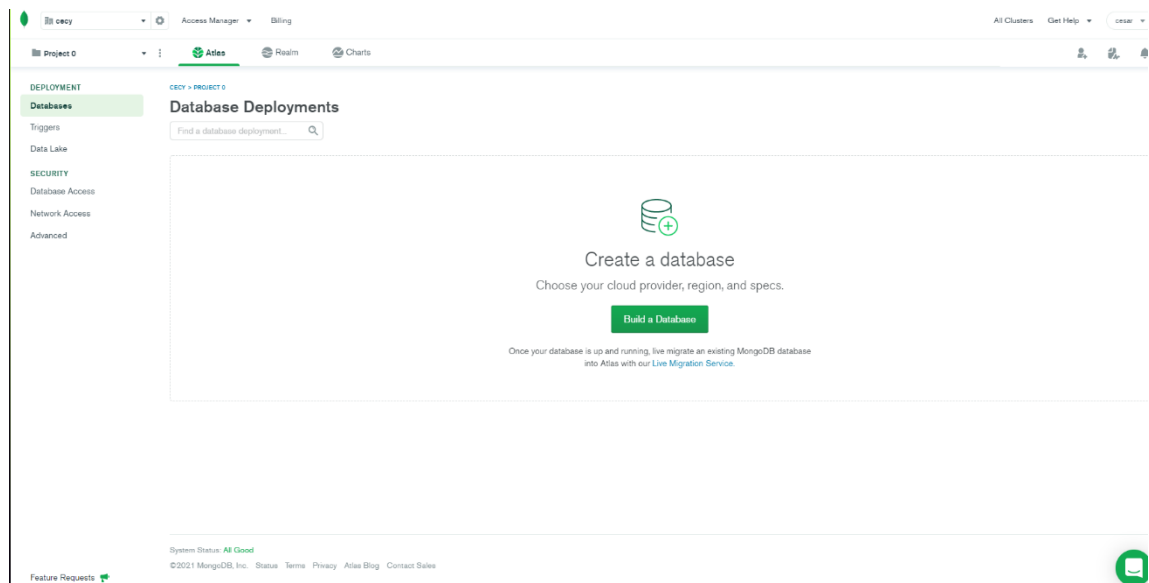


- MongoDB atlas:

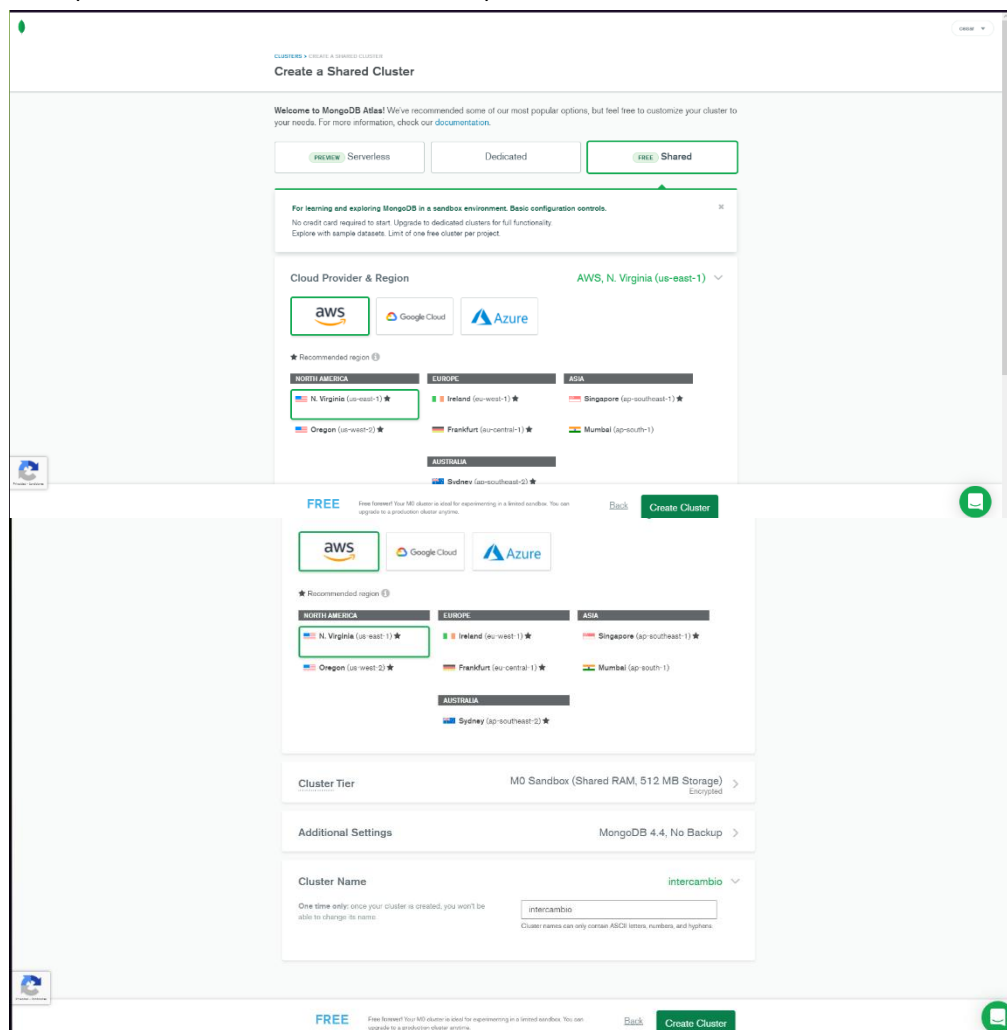


- Aunque no es necesario registrarse debemos de buscar una tercera página llamada GIT

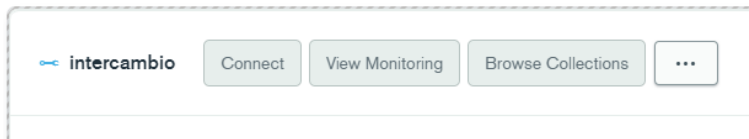
- En esta vamos a descargar la aplicación que nos da la página y procederemos a instalarla sin mover nada del instalador.
- Ahora en mongo db atlas vamos a crear un nuevo clúster



- aquí dejaremos todo por defecto, pero si queremos podemos cambiarle el nombre al clúster (esto no afectara nada en el futuro).



- Tardara esto unos minutos en lo que se configura solo quedara esperar.
- Una vez termine vamos dar en el botón conectar.



- Como es la primera vez que nos registramos vamos a añadir una nueva dirección ip que quedara en puros 0 y posteriormente vamos a crear un usuario aquí lo importante es la contraseña vamos a copiarla la que pusimos.

- Una vez configuramos vamos a dar en siguiente y nos mostrara esta otra ventana:

- Aquí vamos a seleccionar la opción de Connect your application.

- Aquí nos va a mostrar un código que vamos a cambiar en este caso la <password> quitaremos eso y pondremos la contraseña que pusimos en nuestro usuario y en myFirstDatabase borraremos eso y pondremos el nombre de nuestra base de datos

```
mongodb+srv://<username>:<password>@intercambio.h6fvy.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

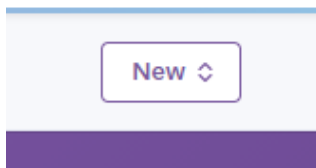
Replace **<password>** with the password for the **<username>** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

- Ahora ese código lo vamos a pegar en nuestro index.js y borraremos el código de mongoose.connect dejando el .then y el .catch quedando de la siguiente manera:
- Con esto ya tendremos nuestra conexión a base de datos en la nube ahora solo falta faltaría

```
25 /*mongoose.connect('mongodb://127.0.0.1/chat-database')
26 .then(db => console.log('conectado a la base de datos'))
27 .catch(err => console.log(err));*/
28 mongoose.connect('mongodb+srv://cesar:cesar@cluster0.h6fvy.mongodb.net/chat-database?retryWrites=true&w=majority', {
29   useNewUrlParser: true,
30   useUnifiedTopology: true
31 })
32 .then(db => console.log('conectado a la base de datos'))
33 .catch(err => console.log(err));
34
```

configurar heroku.

- Para esto una vez registrado vamos a darle en new app



- Aquí vamos a poner cualquier nombre y todo lo demás lo dejaremos tal cual esta.
- Nos mandara una nueva pagina en esta vamos a buscar la sección de heroku cli y lo vamos a descargar y lo instalaremos
- Ahora en un cmd escribiremos heroku login y nos va a abrir una pestaña de Google la cual

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19042.1052]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\César>h login
```

deberemos iniciar sesión.

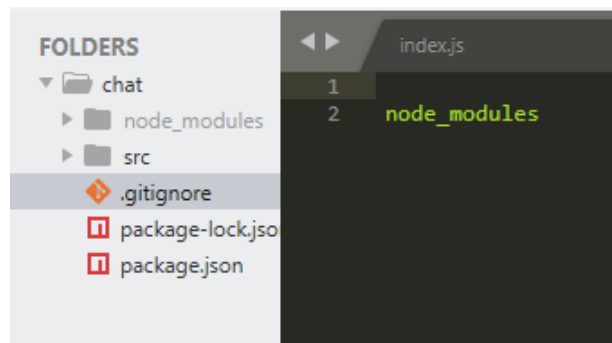
- En otro cmd vamos a navegar hasta nuestro proyecto.
- Vamos a crear un nuevo repositorio para esto vamos a ejecutar un nuevo comando en este caso es git init.

```
C:\Users\César>cd Desktop
C:\Users\César\Desktop>cd chat
C:\Users\César\Desktop\chat>git init
```

- Bien ahora en nuestro proyecto vamos a crear un archivo mas que llevara por nombre



.gitignore y dentro de este vamos a escribir el nombre de la carpeta node\_modules



- Regresando a nuestro ahora vamos a ejecutar el comando **git add .**

```
C:\Users\César\Desktop\chat>git add .
```

- Y para finalizar escribiremos 3 comandos más el primero será: heroku git: **remote -a “aquí va el nombre de a app creada en heroku”**

```
C:\Users\César\Desktop\chat>heroku git: remote -a chat
```

- El penúltimo código será **git commit -m “el mensaje que queramos”**

```
C:\Users\César\Desktop\chat>git commit -m "primera subida al server"
```

- Y nuestro último código será: **git push heroku master**

```
C:\Users\César\Desktop\chat>git push heroku master
```

- En este último cuando termine de subir nos dará un link y será cuestión de copiarlo y pasarlo a otros dispositivos y con eso tendrás tu página subida a internet