

# Aplicación Telecontrol Robot.

Martínez García Erick Antonio.

Servicio Social.



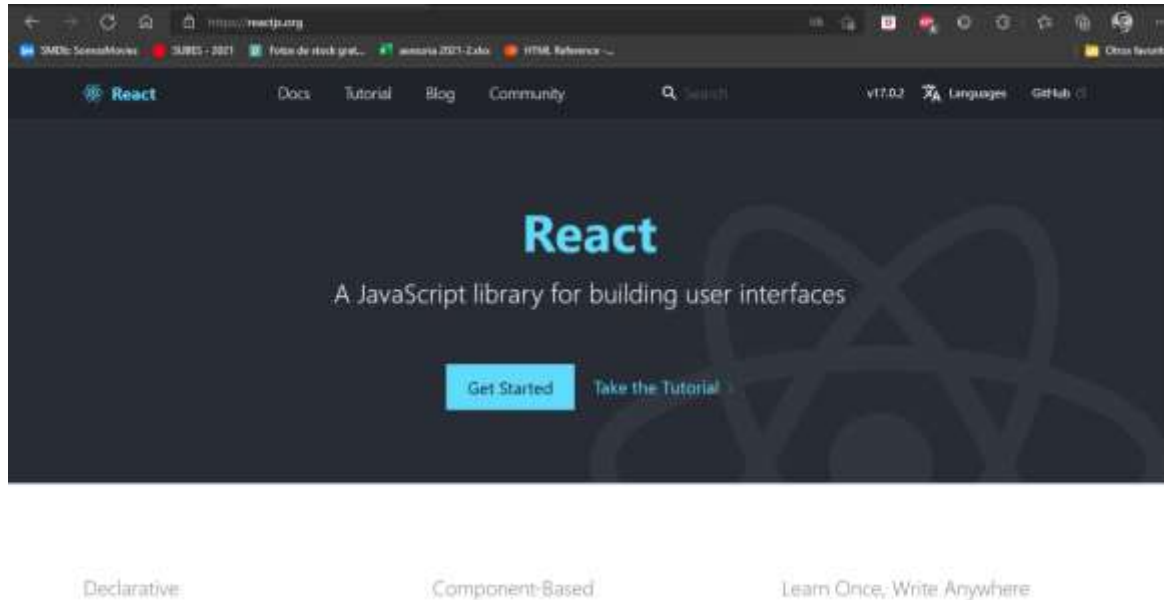
## Índice.

|  |    |
|--|----|
| React. ....  | 3  |
| Instalación de React. ....   | 3  |
| Instalación de Nodejs. ....  | 6  |
| Comprobación de versión npm. ....  | 10 |
| Creación de primer proyecto en React. ....                                 | 11 |
| Editor de código. ....   | 13 |
| Abrir el proyecto de React en Visual Studio Code. ....                     | 15 |
| Ejecución de nuestro proyecto. ....  | 15 |
| React Native. ....   | 17 |
| Requerimientos iniciales. ....   | 17 |
| Instalación de Git. ....   | 17 |
| Comprobación de herramientas instaladas. ....                              | 19 |
| Instalación de React Native. ....  | 20 |
| Instalación de Expo. ....  | 21 |
| Creación de primer proyecto con React Native y Expo. ....                  | 23 |
| Aplicación Web – Joystick. ....  | 29 |
| NodeMCU 8266. ....   | 29 |
| Especificaciones Técnicas. ....  | 30 |
| Diagrama. ....   | 31 |
| Código Fuente HTML/JavaScript. ....  | 32 |
| Código Fuente Arduino ID. ....   | 39 |
| Vista Pagina Web. ....   | 46 |
| Datos Obtenidos. ....  | 46 |
| Circuito 1. NodeMCU esp8266 y Arduino UNO. ....                            | 47 |
| Circuito 2. Node MCU esp8266 y Arduino MEGA. ....                          | 49 |
| Código fuente Arduino ID control de velocidad y dirección de motores. .... | 51 |
| Evidencias. ....   | 56 |
| Referencias. ....  | 57 |

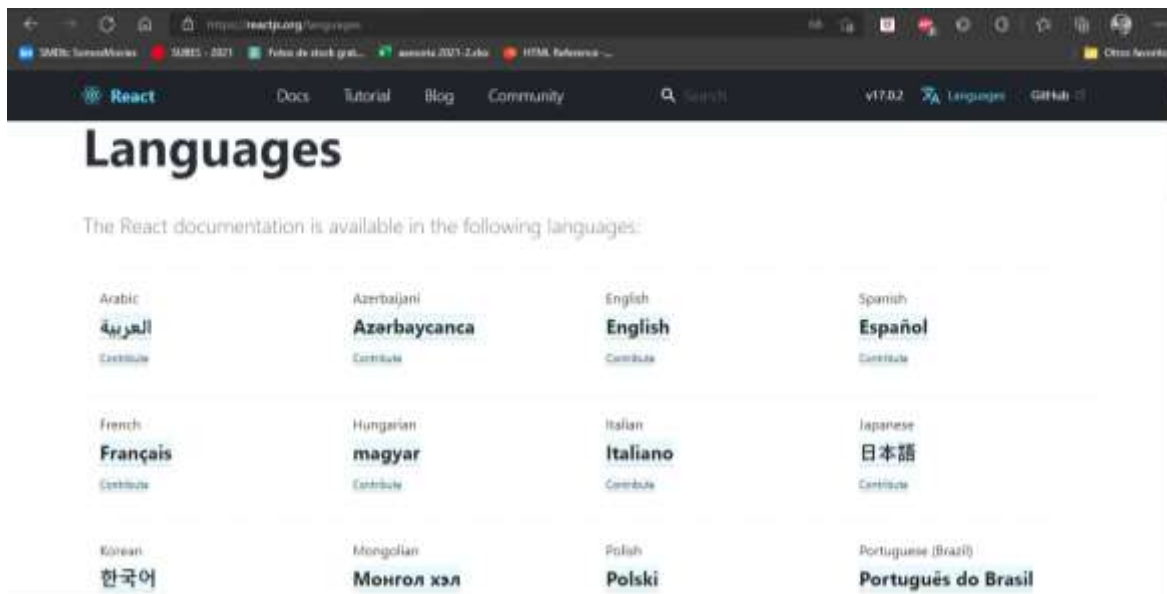
# React.

## Instalación de React.

Para poder instalar la biblioteca de React podremos en el Browser la siguiente dirección <https://reactjs.org>.



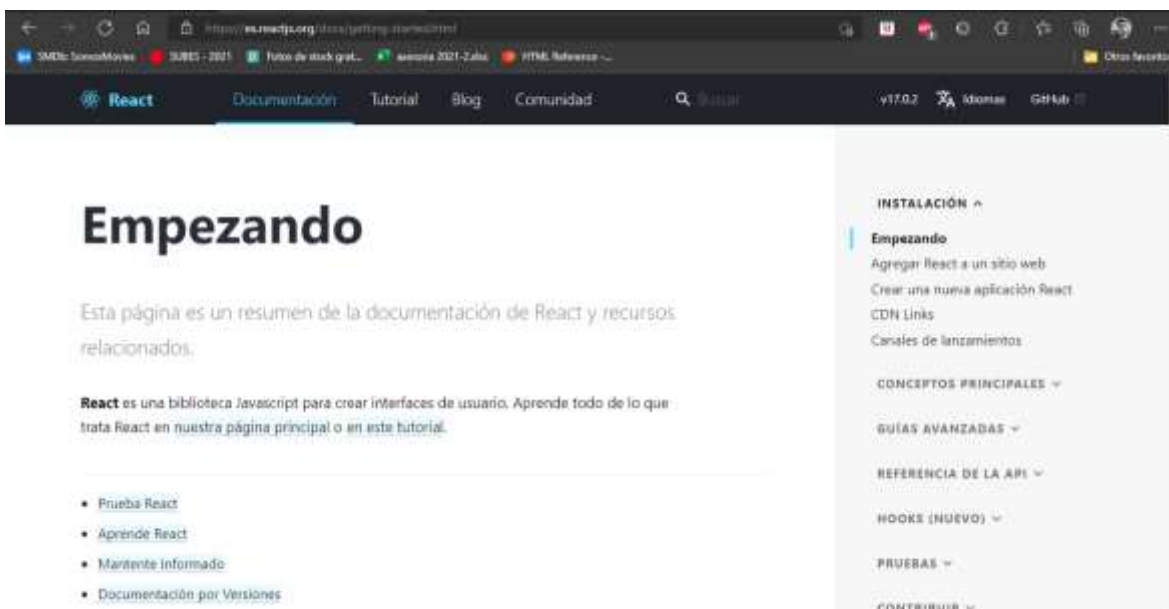
En la barra de principal de la página, daremos un clic en el botón de “Languages” esto con el fin de poder traducir al español la información de la pagina para su mayor entendimiento.



Una vez que hayamos cambiado el idioma al que mejor nos sintamos cómodos, en la pagina de inicio daremos clic en el botón “Comienza”.



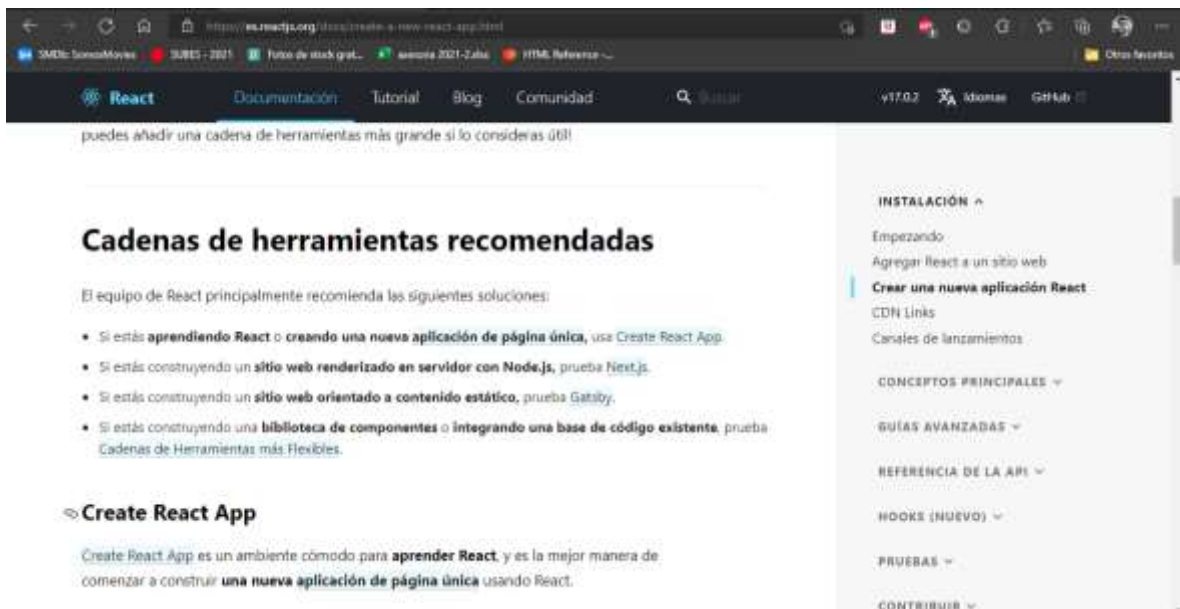
En este botón nos mostrara una guía (tutorial), para poder instalar paso a paso React.



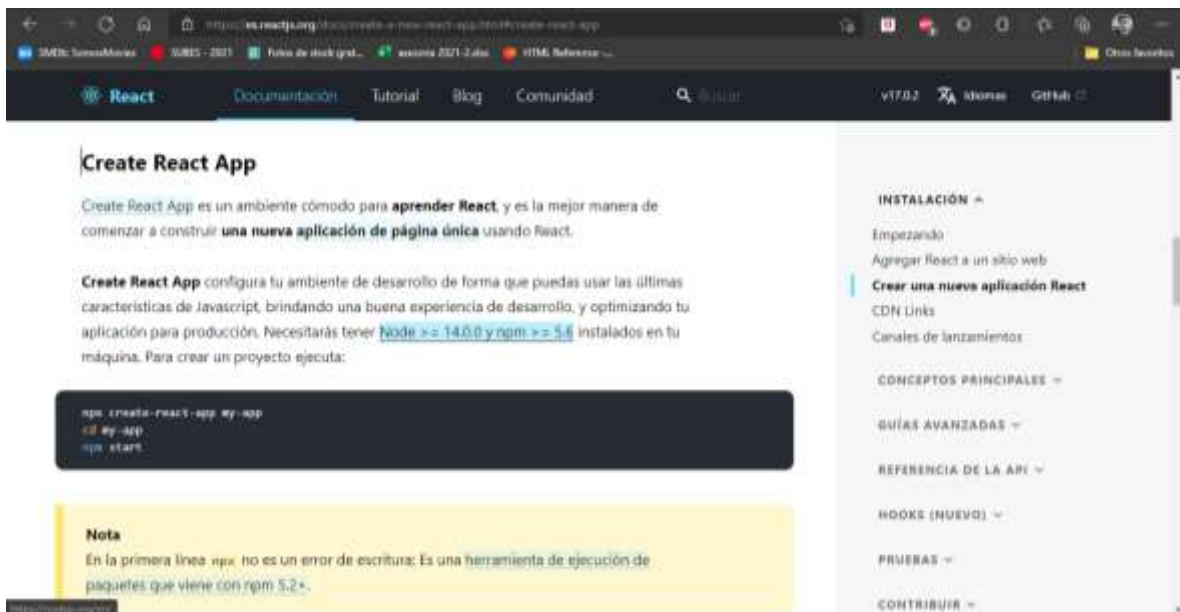
En la parte derecha se encontrará un menú, daremos clic en la opción “Crear una nueva aplicación React”.



Bajaremos a la sección “Cadena de herramientas recomendadas” en esta sección se nos recomendaran cuatro formas de comenzar con React, para este caso haremos uso de la primera opción, daremos clic en el botón “Create React App”.



Una vez que hayamos dado clic en “Create React App” nos dará una guía con los comandos necesarios para poder instalar React.



## Instalación de Nodejs.

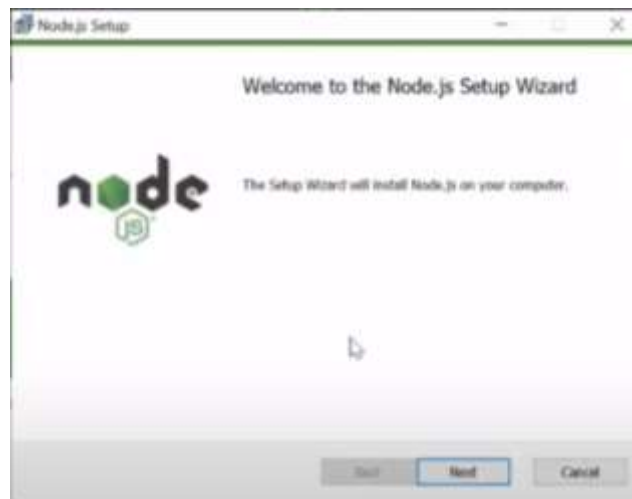
Para seguir con la instalación de React, debemos de instalar Nodejs para su instalación debemos de ir a la página <https://nodejs.org/en/>.



Dentro de la página de nodejs daremos clic en el botón “17.2.0 Current” para iniciar la descarga”.



Una vez descargado daremos clic en el instalador para que se inicie el administrador de instalación de nodejs, damos clic en “Next”.



Aceptamos los términos y condiciones y clic en “Next”, esto lo repetimos hasta llegar al final.

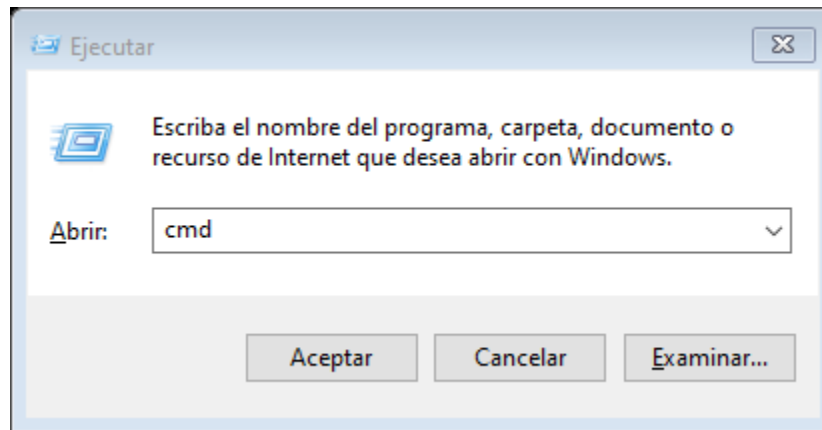


Ya que haya acabado la instalación solo daremos clic en “finish”.

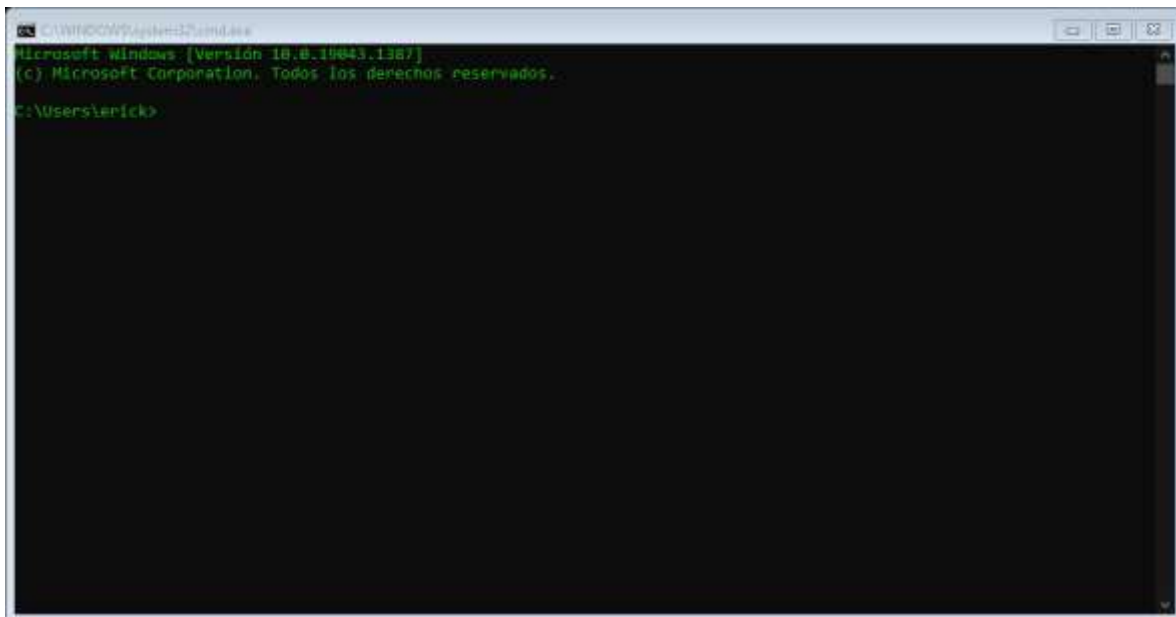




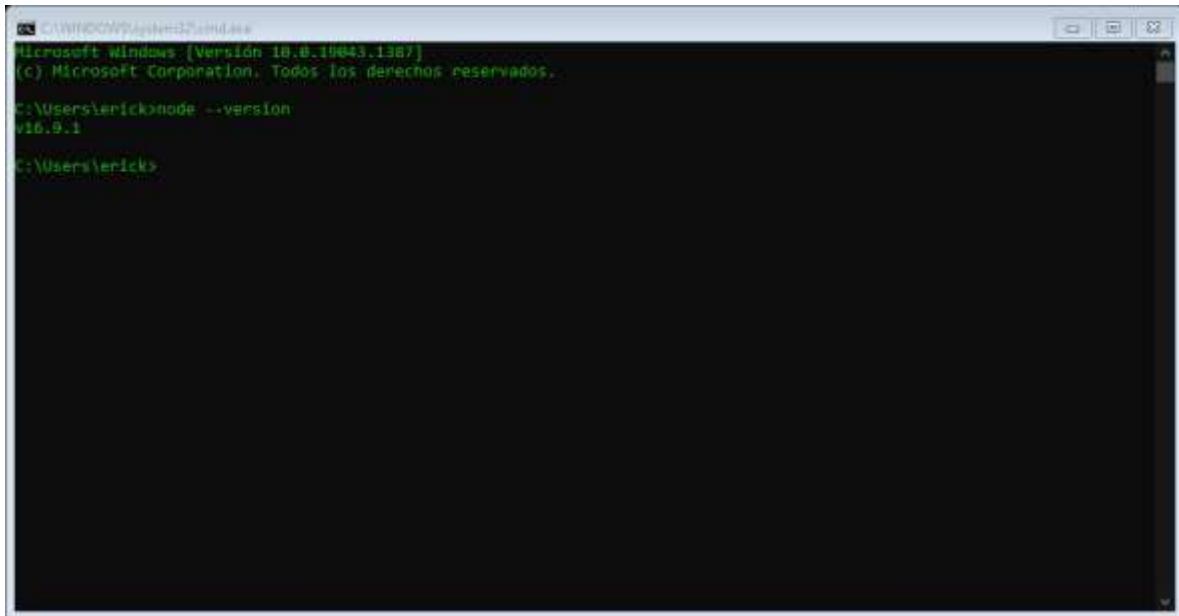
Para comprobar la instalación de nodejs, pulsaremos la tecla Windows mas la tecla R, se abrirá la ventana de Ejecución pondremos el comando "cmd".



Esto abrirá la consola de Windows.



Dentro de la consola tecleamos el comando “node - -version” y damos un enter, una vez que se ejecute este comando nos arrojará la versión de nodejs que hayamos instalado.



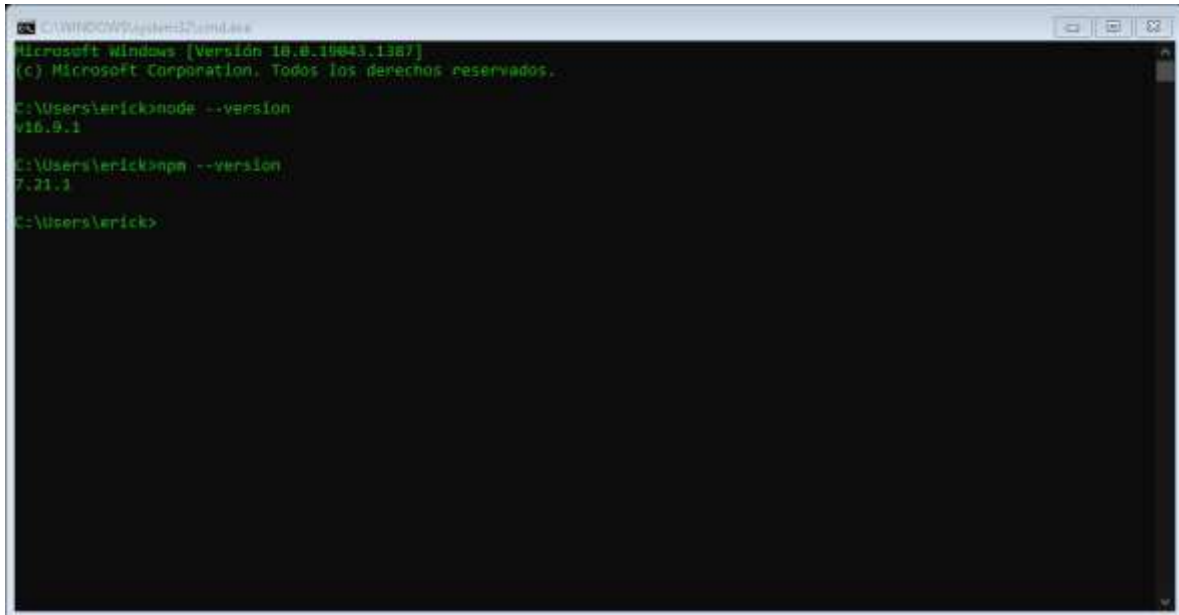
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1387]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\erick>node --version
v16.9.1

C:\Users\erick>
```

### Comprobación de versión npm.

Para comprobar la versión o la instalación de npm en la misma consola que tenemos abierta tecleamos el comando “npm - -version”.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1387]
(c) Microsoft Corporation. Todos los derechos reservados.

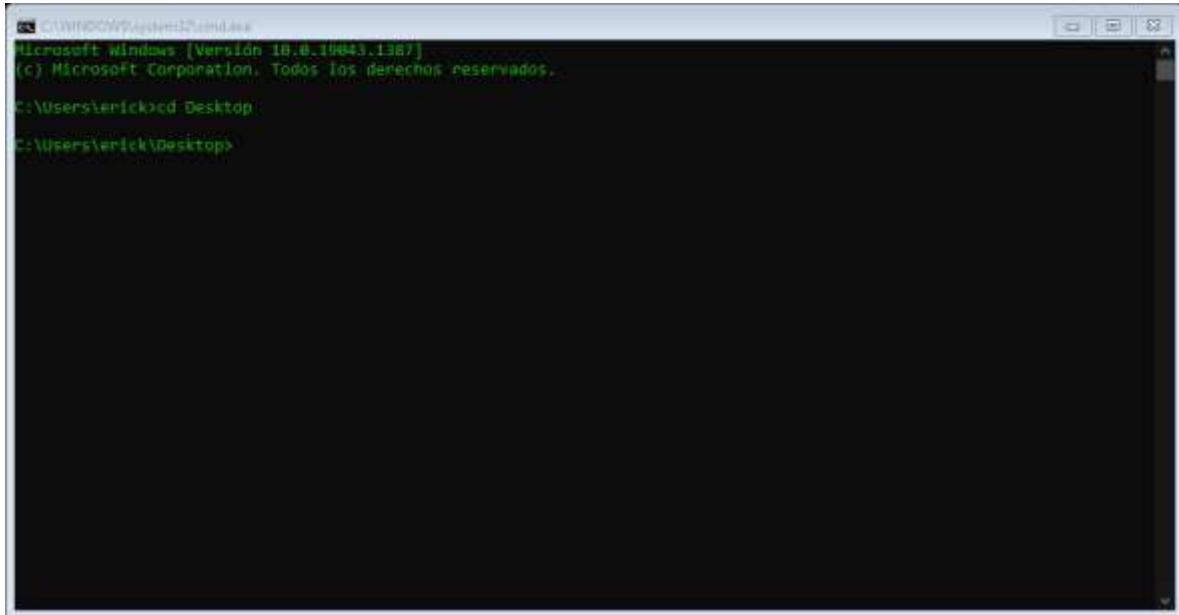
C:\Users\erick>node --version
v16.9.1

C:\Users\erick>npm --version
7.21.1

C:\Users\erick>
```

## Creación de primer proyecto en React.

Para esto debemos tener en cuenta en donde vamos a alojar la carpeta del proyecto, para este ejemplo pondremos la carpeta en el escritorio, para ello tecleamos el comando “cd Desktop”.

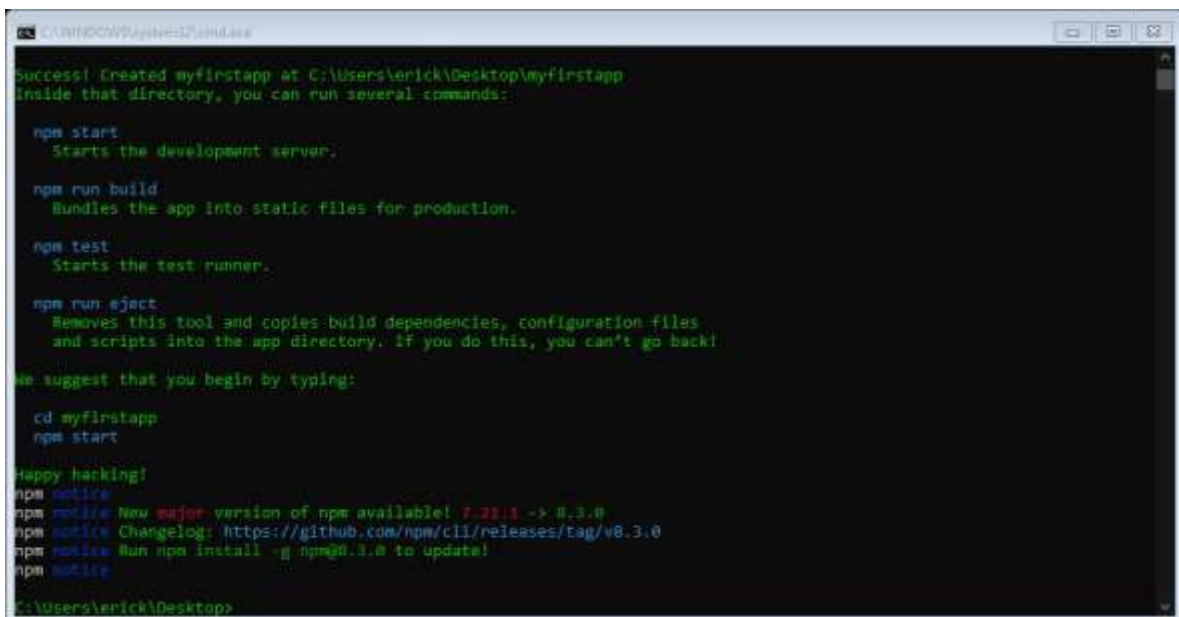


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1387]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\erick>cd Desktop
C:\Users\erick\Desktop>
```

⚠ Esto depende del lugar en donde se guardará la carpeta usaremos el comando “cd” con la diferencia de la ruta la cual debemos de especificar en la consola.

Ya que tengamos la ruta en la que se va a guardar la carpeta del proyecto de React, procedemos a teclear el comando “npx create-react-app” seguido del nombre que tendrá la carpeta, el comando quedaría de la siguiente manera “npx create-react-app myfirstapp” donde “myfirstapp” es el nombre que escogimos para nuestra carpeta, damos enter.



```
C:\WINDOWS\system32\cmd.exe
Success! Created myfirstapp at C:\Users\erick\Desktop\myfirstapp
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd myfirstapp
  npm start

Happy hacking!
npm notice
npm notice New major version of npm available! 7.21.1 -> 8.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.3.0
npm notice Run npm install -g npm@8.3.0 to update!
npm notice
C:\Users\erick\Desktop>
```

Una vez que hayamos esperado unos minutos a que se instalaran los complementos que vamos a tener que usar, en esta parte hay que mencionar cuatro comandos importantes para los cuales son:

- `npm start`: Ejecuta un servidor el cual va a escuchar los cambios que hagamos al proyecto.
- `npm run build`: Nos permite comenzar a construir nuestra aplicación.
- `npm test`: Inicia una compilación al código.
- `npm run eject`: Elimina esta herramienta y copia las dependencias de compilación, los archivos de configuración y scripts en el directorio de la aplicación.

```
npm start
  Starts the development server.

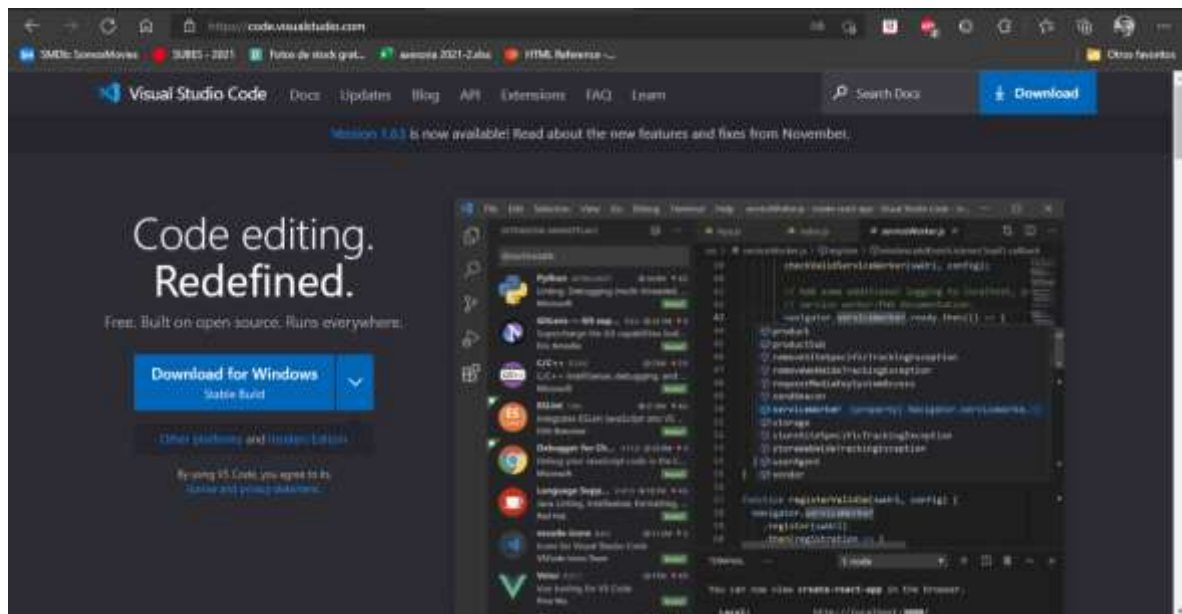
npm run build
  Bundles the app into static files for production.

npm test
  Starts the test runner.

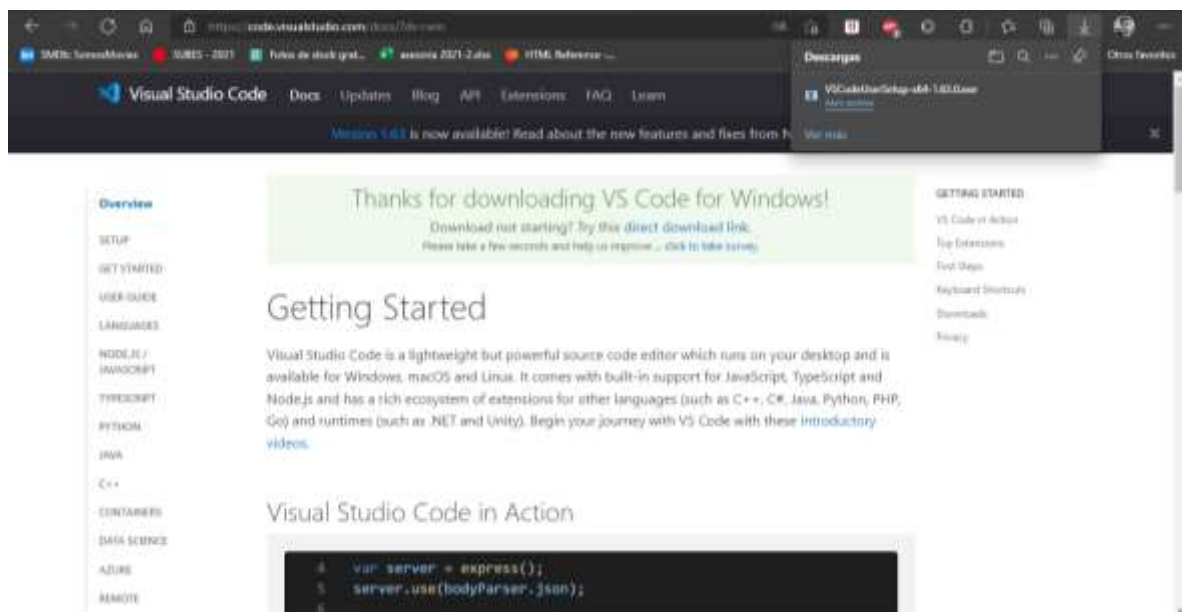
npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!
```

## Editor de código.

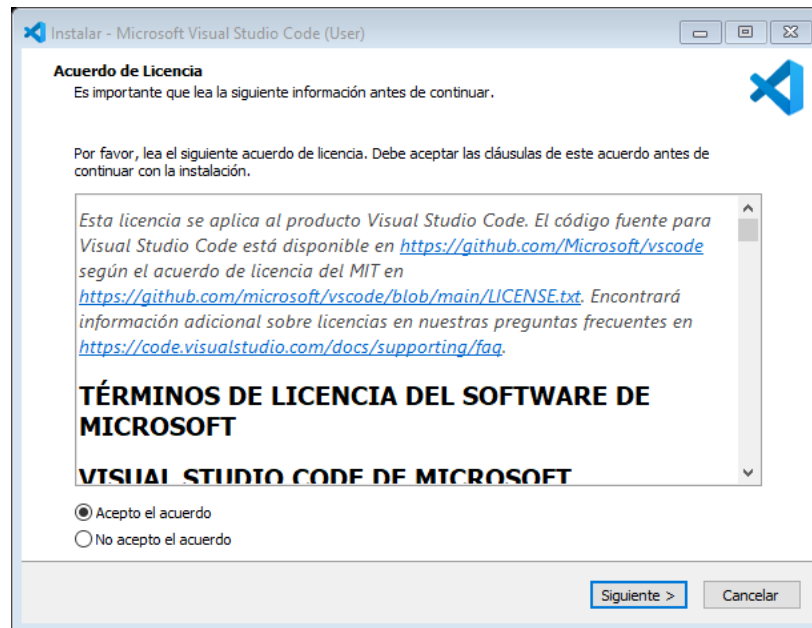
Para este caso usaremos el editor de código Visual Studio Code, para su descarga nos dirigiremos a su página <https://code.visualstudio.com>.



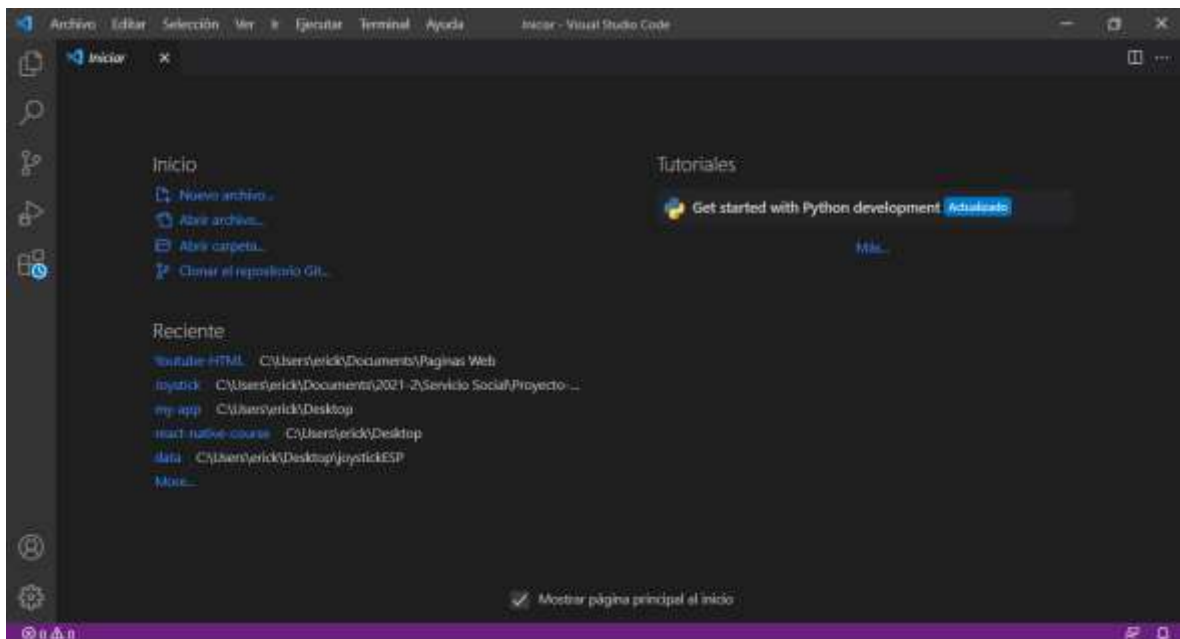
Damos clic en “Download for Windows” y se iniciara la descarga”.



Para su instalación solo aceptamos los términos y condiciones, seguimos los pasos del administrador de instalación.

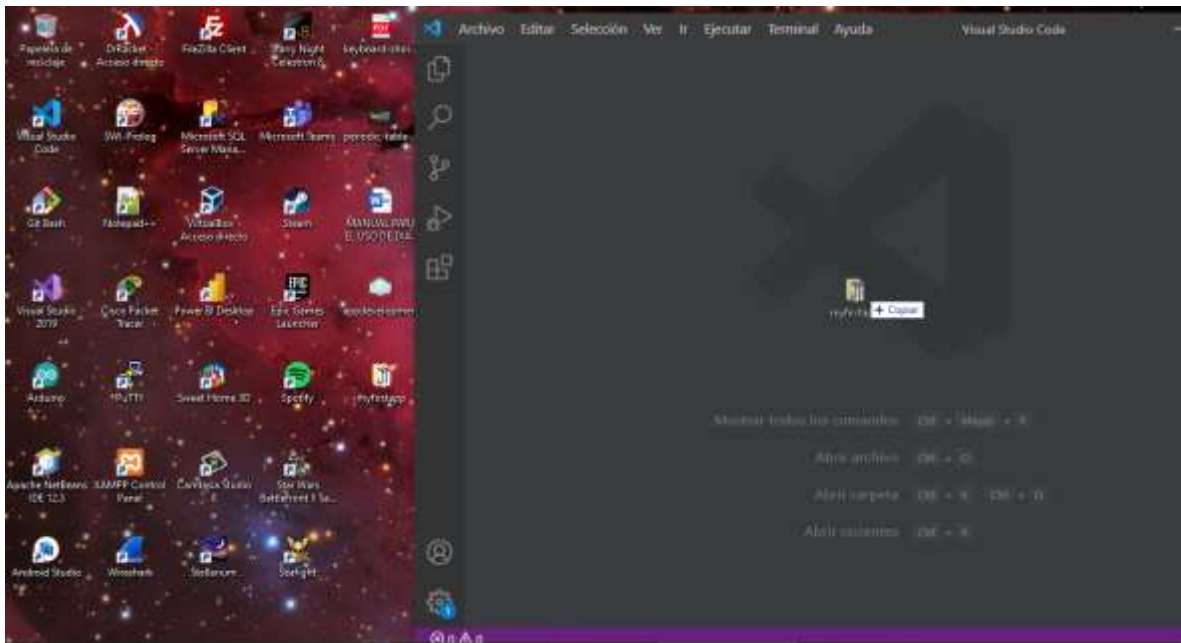


Una vez que termine la instalación, al iniciar Visual Studio Code se vera la pantalla de la siguiente manera.



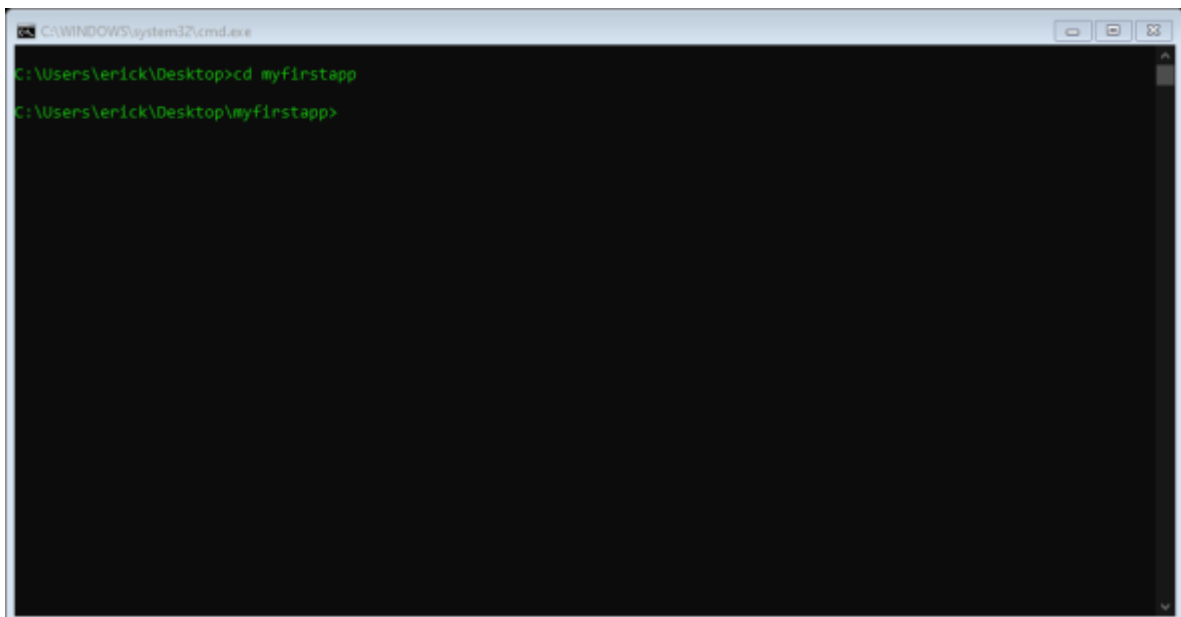
Abrir el proyecto de React en Visual Studio Code.

Para este proceso es algo muy fácil ya que solo implica sostener con el mouse la carpeta de nuestro proyecto y arrastrarla hacia la zona vacía de Visual.

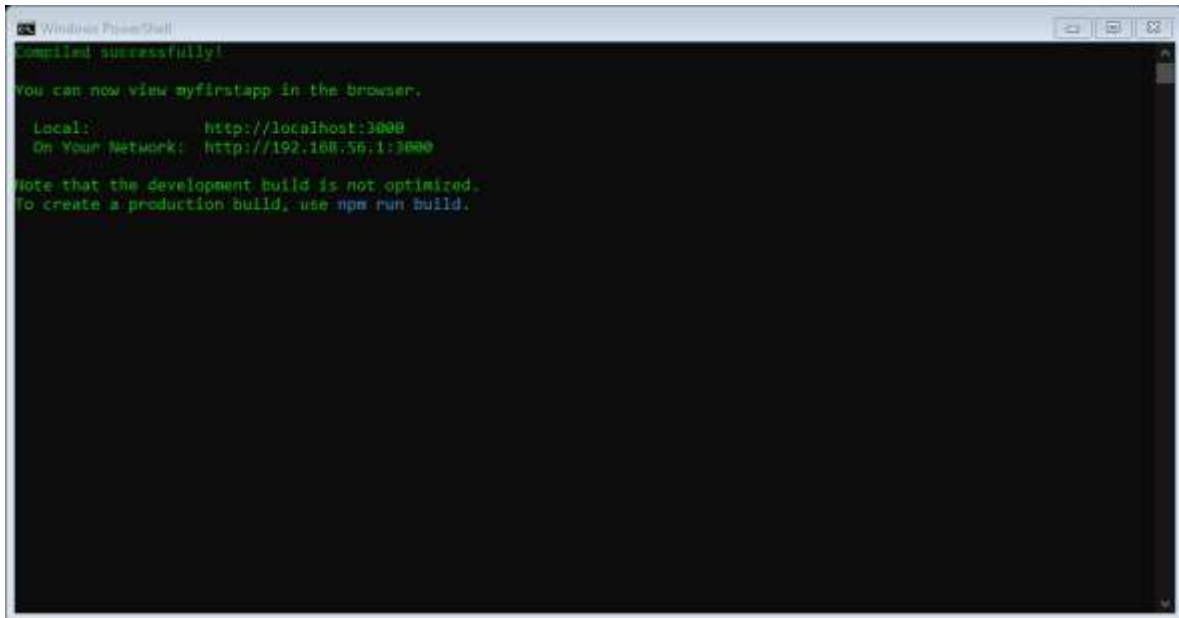


Ejecución de nuestro proyecto.

Para ejecutar el proyecto, en la consola de comandos tecleamos el comando “cd” junto al nombre de nuestra carpeta “cd myfirstapp”.



Una vez que estemos dentro de nuestra carpeta de proyecto tecleamos el comando “npm start”.



```
Windows PowerShell
Compiled successfully!

You can now view myfirstapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Esto lo que hará es crear un localhost en el puerto 3000 con el fin de desarrollar y ver los cambios que hagamos en nuestro proyecto.





## React Native.

### Requerimientos iniciales.

Para la instalación, necesitamos de tres herramientas nodejs, git y el editor de código Visual Studio Code.

Los pasos para seguir la instalación de nodejs y Visual Studio Code están en la parte de arriba.

### Instalación de Git.

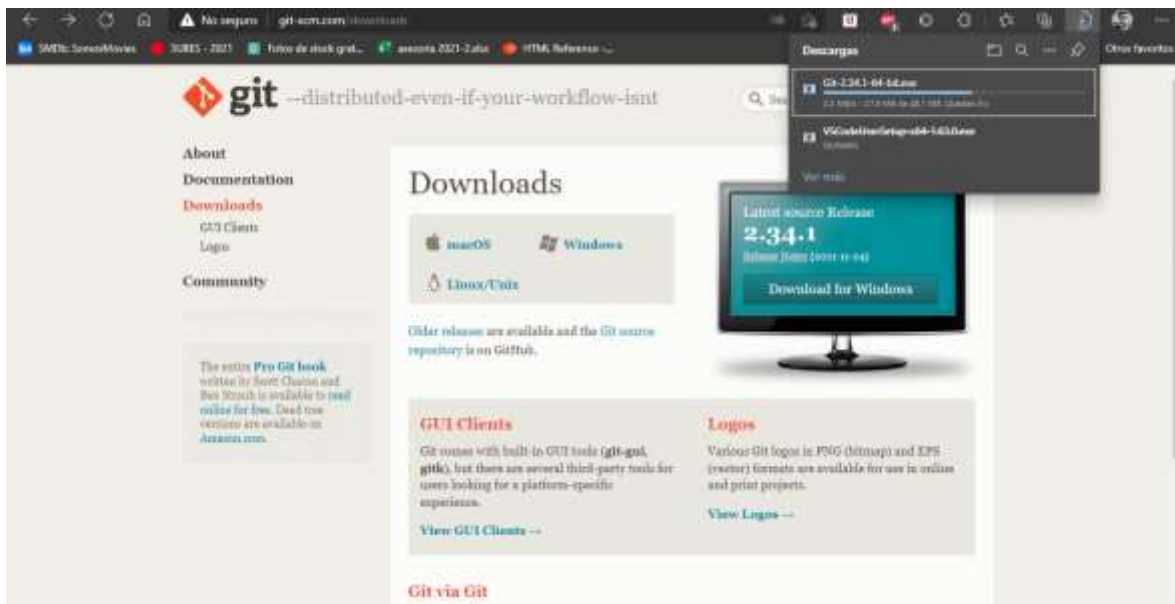
Para descargar el instalador de git tendremos que ir a su pagina la cual se encuentra en [git.scm.com](http://git.scm.com).



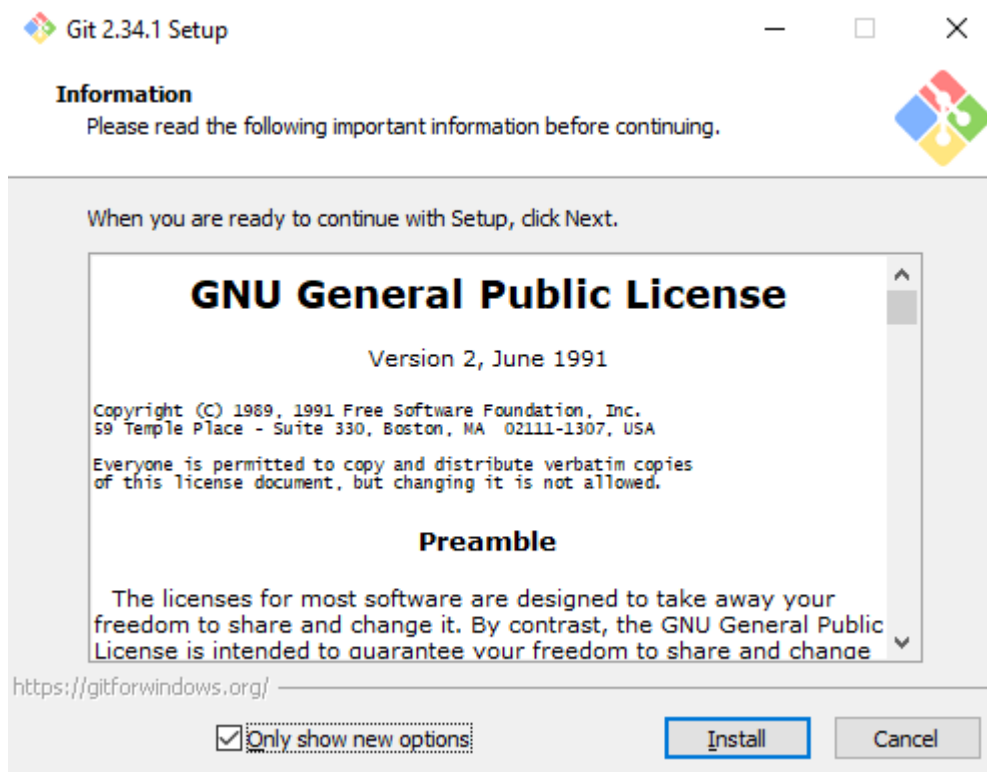
Damos clic en la opción de Downloads.



Clic en “Windows” y se iniciara la descarga.



Para su instalación solo basta con aceptar los términos y condiciones, dar clic en “install” y esperar unos minutos.

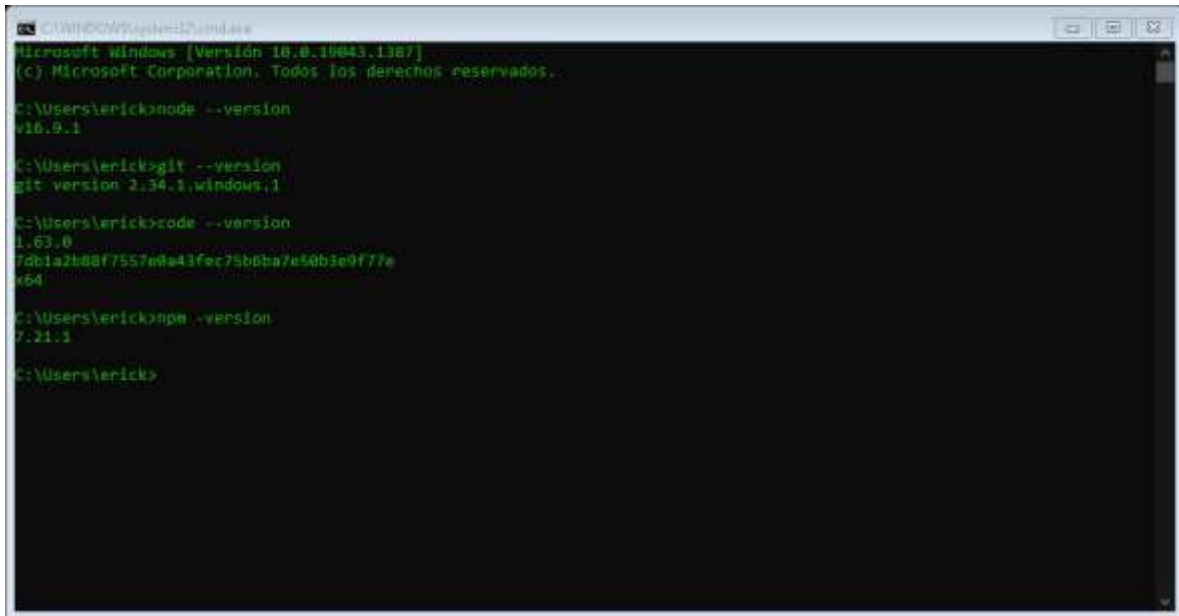


## Comprobación de herramientas instaladas.

En este paso pulsaremos las teclas "Windows" mas "R" con el fin de abrir una consola, dentro de la consola tecleamos los siguientes comandos.

- node - -version.
- git - -version.
- code - -version.
- npm - -version.

Lo que harán estos comandos es ver si tenemos esas herramientas instaladas en nuestro equipo si es así nos darán la versión que tenemos instalada de estas cuatro herramientas.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1387]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\erick>node --version
v16.9.1

C:\Users\erick>git --version
git version 2.34.1.windows.1

C:\Users\erick>code --version
1.63.0
7db1a2b88f7557a9a41fec75b6ba/e50b3e9f77e
x64

C:\Users\erick>npm --version
7.21.1

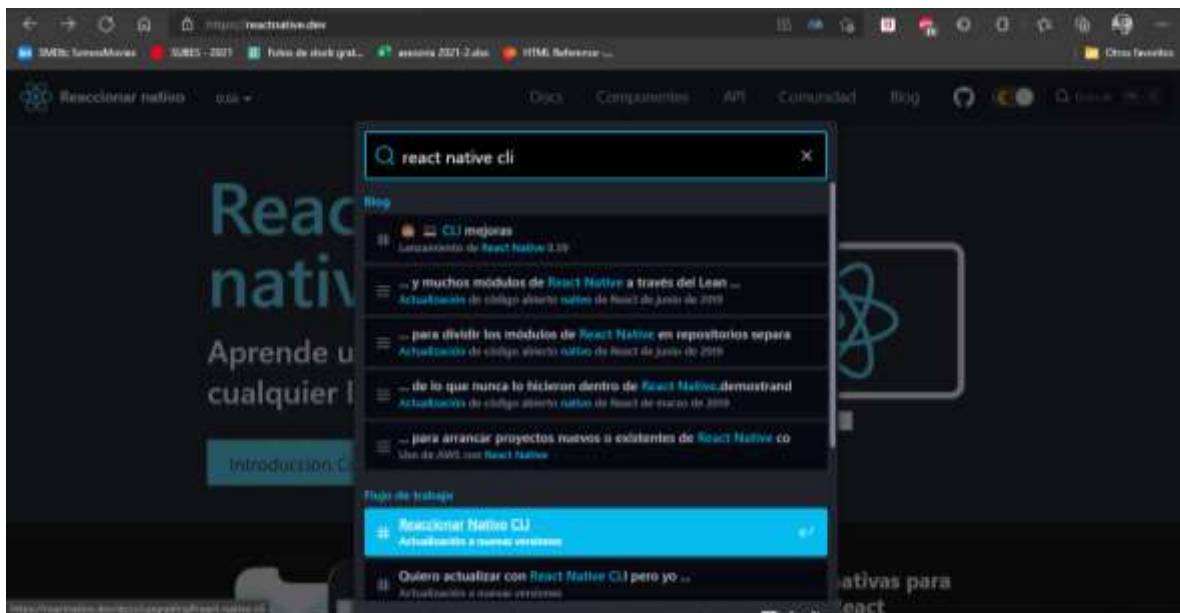
C:\Users\erick>
```

## Instalación de React Native.

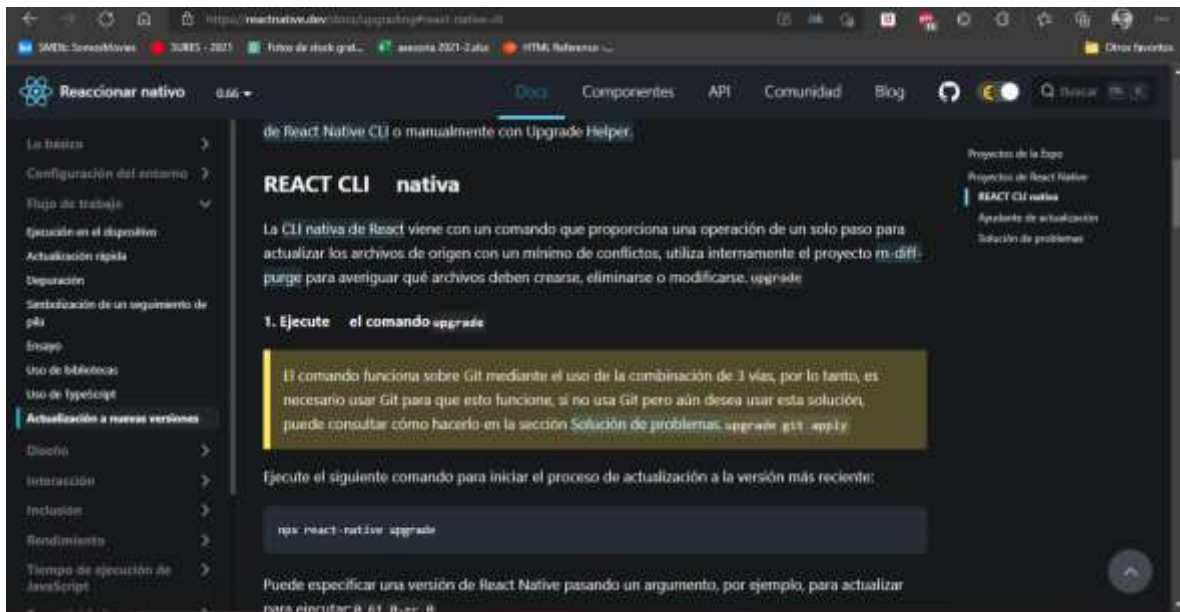
Para esto debemos de ir a la página <https://reactnative.dev>.



Iremos al buscador que está dentro de la página y tecleamos “React Native CLI”.

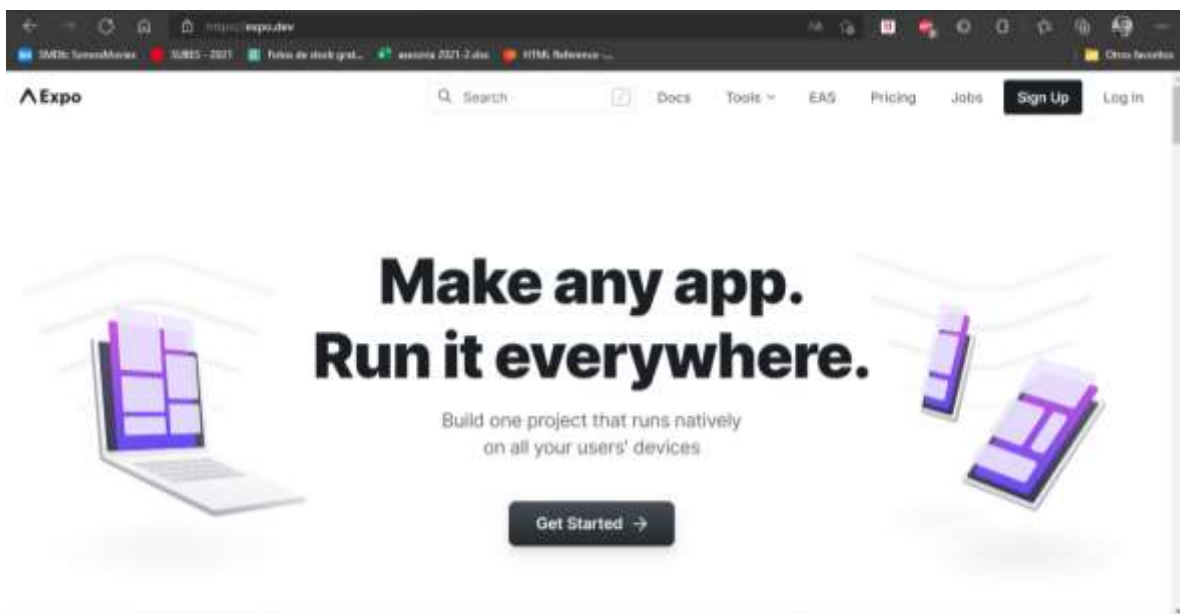


Dentro de esta documentación encontraremos una guía de como instalar y usar react native.

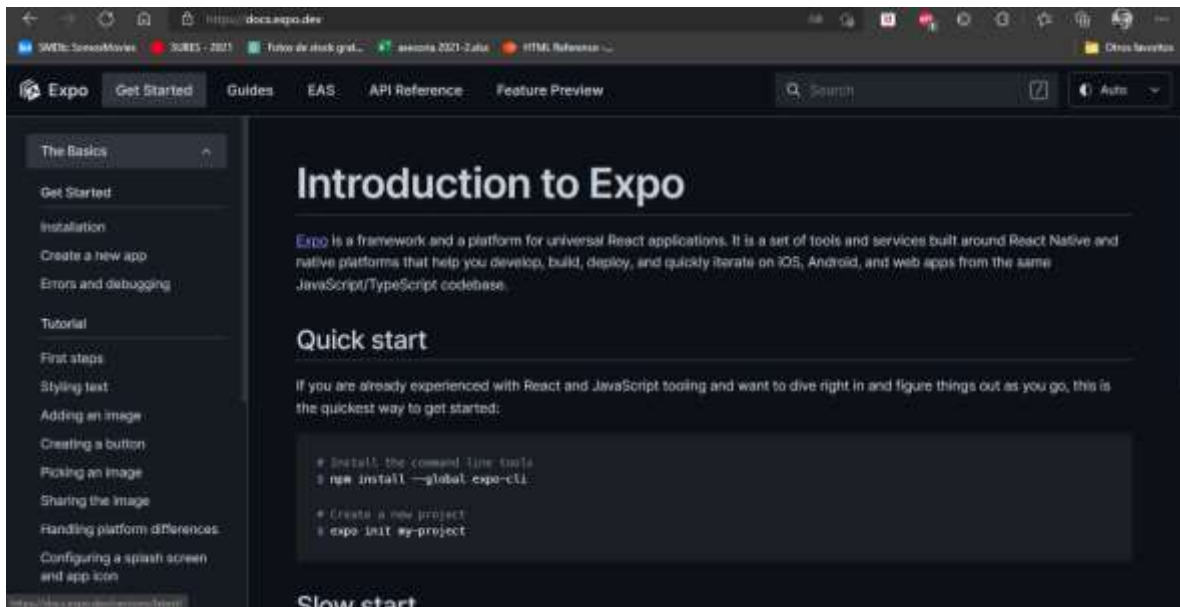


## Instalación de Expo.

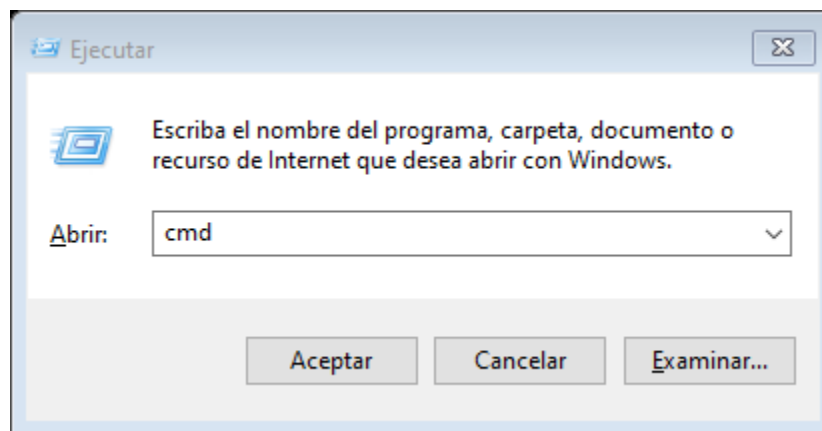
Para poder comenzar con la instalación de Expo no dirigiremos a su página web la cual es <https://expo.dev>, dentro de la página daremos clic en el botón “Get Started”.



El botón nos llevará a la documentación de instalación de Expo, donde nos dará un comando “npm install --global expo-cli”.



Para este proceso tendremos que abrir una consola en Windows como ya sabemos para abrir la consola pulsaremos la tecla Windows + R y escribiremos el comando “cmd”, posteriormente daremos enter.



Para empezar, pondremos el comando “npm install -g expo-cli”, esto instalará la cli de expo la cual nos permitirá crear un proyecto de React Native.

```

C:\Users\Facth> npm install -g expo-cli
npm WARN deprecated @shapi/joi@17.1.1: joi is leaving the @shapi organization and moving back to 'joi' (https://github.com/sideway/joi/issues/2411)
[.....] | loadDap:babel-plugin-dynamic-import-node: 15.1 /babel-core@7.20.6 @babel/helper-esplode-assign

```

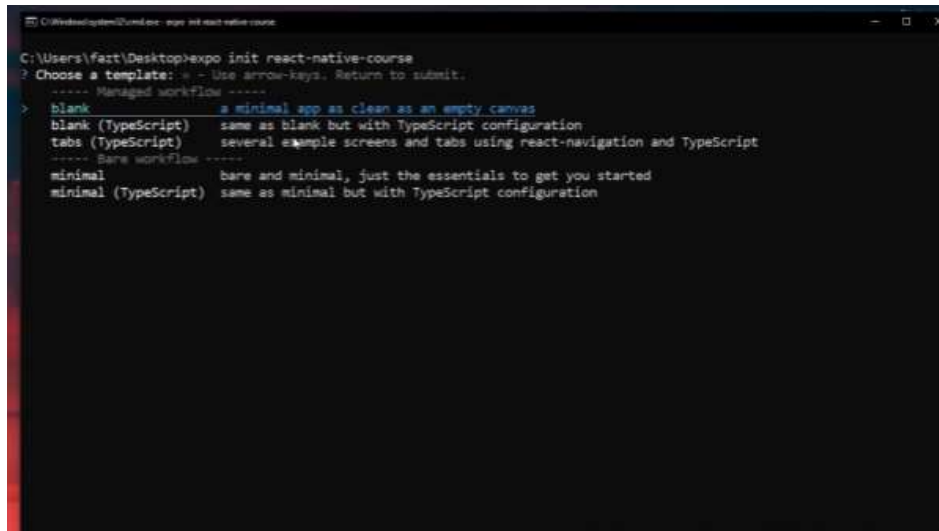
## Creación de primer proyecto con React Native y Expo.

Una vez que hayamos esperado unos minutos a que haya terminado la instalación de expo tendremos que situarnos en la ruta en la que nuestro proyecto será guardado, para este caso nos situaremos en el Escritorio, para ello pondremos el comando “cd Desktop”.

[illegible]



Para crear nuestro primer proyecto pondremos el siguiente comando “expo init react-native-course” donde “react-native-course” es el nombre que nosotros le pondremos al proyecto.

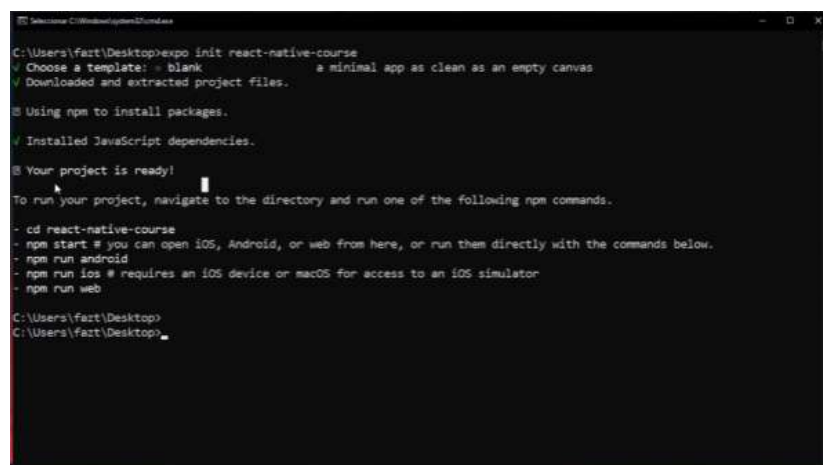


```
C:\Users\fast\Desktop>expo init react-native-course
? Choose a template: - Use arrow-keys. Return to submit.
----- Managed workflow -----
> blank                a minimal app as clean as an empty canvas
  blank (TypeScript)   same as blank but with TypeScript configuration
  tabs (TypeScript)    several example screens and tabs using react-navigation and TypeScript
----- Bare workflow -----
  minimal              bare and minimal, just the essentials to get you started
  minimal (TypeScript) same as minimal but with TypeScript configuration
```

Nos dará unas opciones las cuales son:

- blank: Crear un proyecto en blanco, utilizando react native.
- blank (TypeScript): Crear un proyecto en blanco utilizando TypeScript.
- tabs (TypeScript): Crear un proyecto con tabs.

Para nuestro caso utilizaremos la primera opción, la seleccionamos y damos enter, esperamos a que se cree el proyecto (esto tardara el tiempo de acuerdo con la velocidad de nuestro internet).



```
C:\Users\fast\Desktop>expo init react-native-course
✓ Choose a template: - blank                a minimal app as clean as an empty canvas
✓ Downloaded and extracted project files.

@ Using npm to install packages.
✓ Installed JavaScript dependencies.

@ Your project is ready!
To run your project, navigate to the directory and run one of the following npm commands.
- cd react-native-course
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web

C:\Users\fast\Desktop>
C:\Users\fast\Desktop>
```

Como podemos ver nuestro proyecto ya esta creado, ya que en la ruta donde lo creamos estará una carpeta con el nombre que le asignamos y dentro archivos ejemplo del proyecto.



En la consola habrá cinco comandos para arrancar el proyecto de React Native.

```
@ Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd react-native-course
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web
```

El primero es para situarnos en la ruta de la carpeta donde está nuestro proyecto, lo tecleamos y damos un enter.

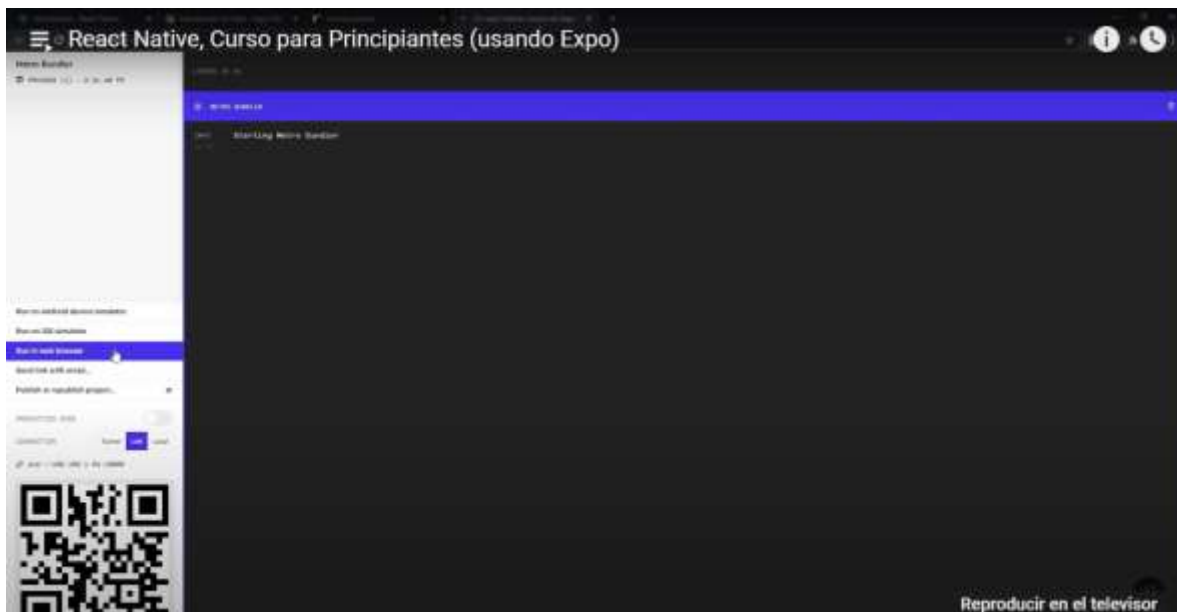
“cd react-native.course”

```
C:\Users\Fazt\Desktop>
C:\Users\Fazt\Desktop>cd react-native-course
C:\Users\Fazt\Desktop\react-native-course>
```

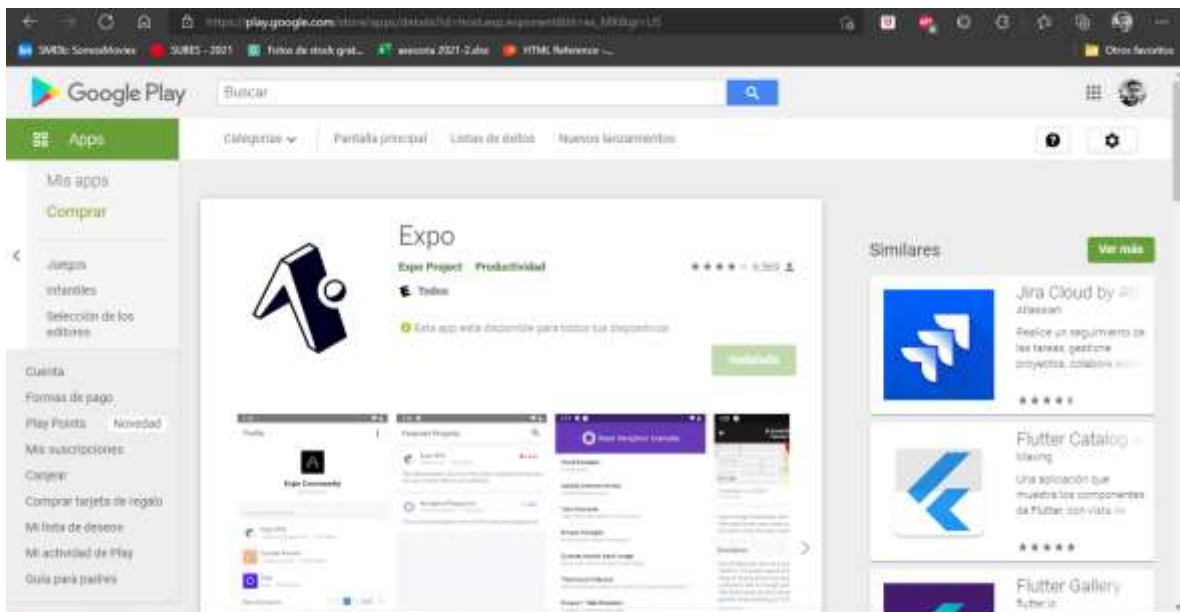
Para iniciar el proyecto tecleamos el segundo comando.

“npm start”

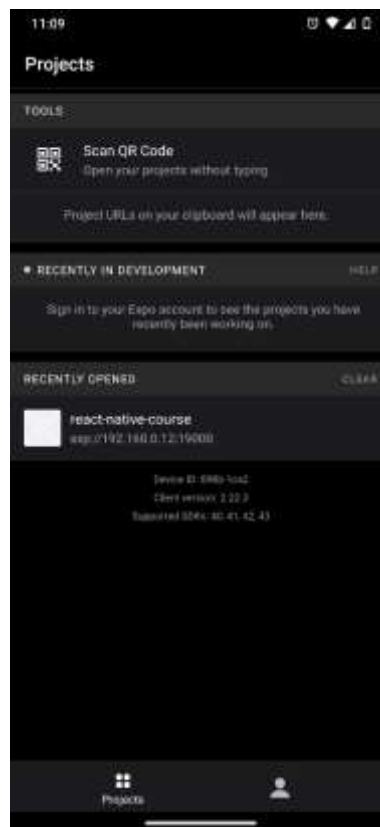
Este comando lo que hará es abrir una pestaña en el browser, donde habrá una cli, lo que significa que ya hemos arrancado el proyecto.



En la pantalla habrá una ventana de lado izquierdo con distintas opciones para poder correr la aplicación, abajo estará un código QR, este nos servirá siempre y cuando descarguemos la app de la Play Store (Android) o Apps Store (IOS).



Esta aplicación nos dará la oportunidad de conectar nuestro dispositivo físico (smartphone) con el proyecto.



Para poder conectar la aplicación con nuestro proyecto solo basta dar clic en la opción "Scan QR code" el cual esta en la pestaña del browser y en automático se conectará.



Una vez escaneado el código se iniciará la aplicación en nuestro dispositivo físico.



## Aplicación Web – Joystick.

### NodeMCU 8266.

NodeMCU ESP8266 es una plataforma de desarrollo similar a Arduino especialmente orientada al Internet de las cosas (IoT). La placa NodeMcu v2 ESP8266 tiene como núcleo al SoM ESP-12E que a su vez está basado en el SoC Wi-Fi ESP8266, integra además el conversor USB-Serial TTL CP2102 y conector micro-USB necesario para la programación y comunicación a PC. NodeMcu v2 ESP8266 está diseñado especialmente para trabajar montado en protoboard o soldado sobre una placa. Posee un regulador de voltaje de 3.3V en placa, esto permite alimentar la placa directamente del puerto micro-USB o por los pines 5V y GND. Los pines de entradas/salidas (GPIO) trabajan a 3.3V por lo que para conexión a sistemas de 5V es necesario utilizar convertidores de nivel como: Conversor de nivel 3.3-5V 4CH o Conversor de nivel bidireccional 8CH - TXS0108E.

NodeMCU viene con un firmware pre-instalado el cual nos permite trabajar con el lenguaje interpretado LUA, enviándole comandos mediante el puerto serial (CP2102). Las tarjetas NodeMCU y Wemos D1 mini son las plataformas más usadas en proyectos de Internet de las cosas (IoT). No compite con Arduino, pues cubren objetivos distintos.

El SoC(System On a Chip) ESP8266 de Espressif Systems es un chip especialmente diseñado para las necesidades de un mundo conectado, integra un potente microcontrolador con arquitectura de 32 bits (más potente que el Arduino Due) y conectividad Wi-Fi. El SoM(System on Module) ESP-12E fabricado por Ai-Thinker integra en un módulo el SoC ESP8266, memoria FLASH, cristal oscilador y antena WiFi en PCB.

La plataforma ESP8266 permite el desarrollo de aplicaciones en diferentes lenguajes como: Arduino, Lua, MicroPython, C/C++, Scratch. Al trabajar dentro del entorno Arduino podremos utilizar un lenguaje de programación conocido y hacer uso de un IDE sencillo de utilizar, además de hacer uso de toda la información sobre proyectos y librerías disponibles en internet. La comunidad de usuarios de Arduino es muy activa y da soporte a plataformas como el ESP8266. Dentro de las principales placas de desarrollo o módulos basados en el ESP8266 tenemos: ESP-01, ESP-12E, Wemos D1 mini y NodeMCU v2.

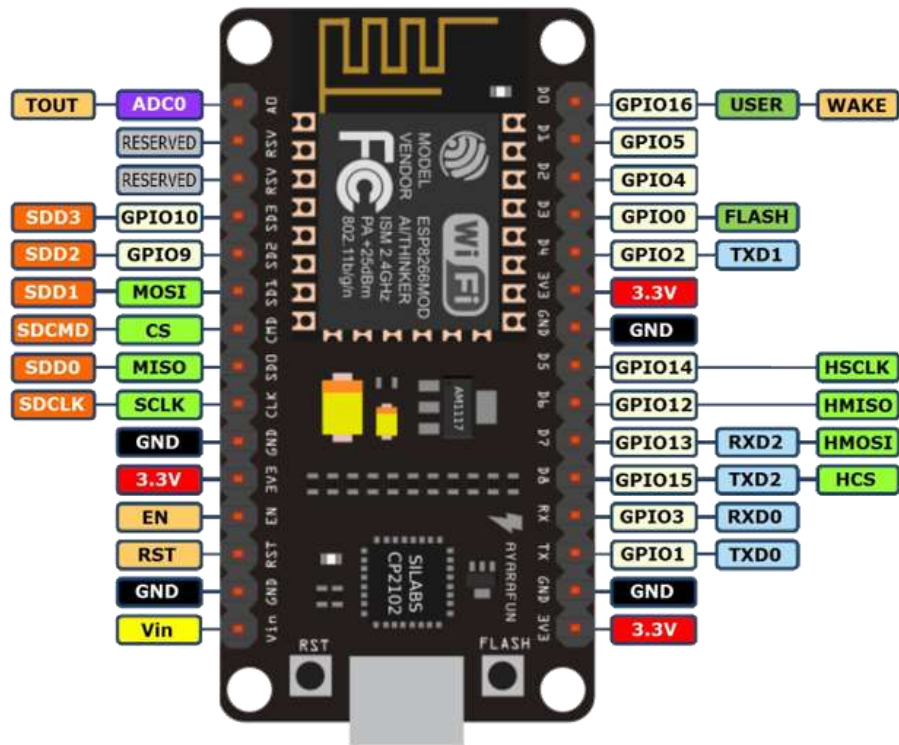


### Especificaciones Técnicas.

- Voltaje de Alimentación: 5V DC
- Voltaje de Entradas/Salidas: 3.3V DC (No usar 5V)
- Placa: NodeMCU v2 (Amica)
- Chip conversor USB-serial: CP2102
- SoM: ESP-12E (Ai-Thinker)
- SoC: ESP8266 (Espressif)
- CPU: Tensilica Xtensa LX3 (32 bit)
- Frecuencia de Reloj: 80MHz/160MHz
- Instruction RAM: 32KB
- Data RAM: 96KB
- Memoria Flash Externa: 4MB
- Pines Digitales GPIO: 17 (4 pueden configurarse como PWM a 3.3V)
- Pin Analógico ADC: 1 (0-1V)
- Puerto Serial UART: 2
- Certificación FCC
- Antena en PCB
- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Stack de Protocolo TCP/IP integrado
- PLLs, reguladores, DCXO y manejo de poder integrados
- Potencia de salida de +19.5dBm en modo 802.11b
- Corriente de fuga menor a 10uA
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Consumo de potencia Standby < 1.0mW (DTIM3)
- Pulsador RESET y FLASH
- Leds indicadores: 2

- Dimensiones: 49\*26\*12 mm
- Peso: 9 gramos

Diagrama.



## Código Fuente HTML/JavaScript.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Joystick WIFI NODEMCU8266</title>
  <style>
    body {
      text-align: center;
      font-size: width/2pt;
    }

    h1 {
      font-weight: bold;
      font-size: width/2pt;
    }

    h2 {
      font-weight: bold;
      font-size: width/2pt;
    }

    button {
      font-weight: bold;
      font-size: width/2pt;
    }
  </style>
  <script>
    var canvas_width = 400, canvas_height = 400;
    var radio_base = 150;
    var radio_handle = 72;
    var radio_shaft = 120;
    var rango = canvas_width / 2 - 10;
    var step = 18;
    var ws;
    var joystick = { x: 0, y: 0 };
    var click_state = 0;

    var ratio = 1;

    function inicio() {
```



```

var width = window.innerWidth;
var height = window.innerHeight

if (width < height)
    ratio = (width - 50) / canvas_width;
else
    ratio = (height - 50) / canvas_width;

canvas_width = Math.round(canvas_width * ratio);
canvas_height = Math.round(canvas_height * ratio);
radio_base = Math.round(radio_base * ratio);
radio_handle = Math.round(radio_handle * ratio);
radio_shaft = Math.round(radio_shaft * ratio);
rango = Math.round(rango * ratio);
step = Math.round(step * ratio);

var canvas = document.getElementById("remote");
canvas.width = canvas_width;
canvas.height = canvas_height;

canvas.addEventListener("touchstart", mouse_down);
canvas.addEventListener("touchend", mouse_up);
canvas.addEventListener("touchmove", mouse_move);
canvas.addEventListener("mousedown", mouse_down);
canvas.addEventListener("mouseup", mouse_up);
canvas.addEventListener("mousemove", mouse_move);

var ctx = canvas.getContext("2d");
ctx.translate(canvas_width / 2, canvas_height / 2);
ctx.shadowBlur = 20;
ctx.shadowColor = "LightGray";
ctx.lineCap = "round";
ctx.lineJoin = "round";

actualizarVista();
}
function conectar() {
    if (ws == null) {
        ws = new WebSocket('ws://' + window.location.hostname + ':81');
        document.getElementById("ws_state").innerHTML = "CONECTADO";
        ws.onopen = ws_onopen;
        ws.onclose = ws_onclose;
        ws.onmessage = ws_onmessage;
    }
}

```

```

        else
            ws.close();
    }
    function ws_onopen() {
        document.getElementById("ws_state").innerHTML = "<font color =
'blue'>CONECTADO</font>";
        document.getElementById("bt_connect").innerHTML = "DESCONECTADO";
        actualizarVista();
    }
    function ws_onclose() {
        document.getElementById("ws_state").innerHTML = "<font color =
'gray'>CERRADO</font>";
        document.getElementById("bt_connect").innerHTML = "CONECTADO";
        ws.onopen = null;
        ws.onclose = null;
        ws.onmessage = null;
        ws = null;
        actualizarVista();
    }
    function ws_onmessage(e_msg) {
        e_msg = e_msg || window.event; //MessageEvent
    }
    function enviarDatos() {
        var x = joystick.x, y = joystick.y;
        var joystick_rango = rango - radio_handle;
        x = Math.round(x * 100 / joystick_rango);
        y = Math.round(-(y * 100 / joystick_rango));

        if (ws != null)
            ws.send(x + ":" + y + "\r\n");
    }
    function actualizarVista() {
        var x = joystick.x, y = joystick.y;

        var canvas = document.getElementById("remote");
        var ctx = canvas.getContext("2d");

        ctx.clearRect(- canvas_width / 2, - canvas_height / 2, canvas_width, canvas_height);

        ctx.lineWidth = 3;

        ctx.strokeStyle = "black";
        ctx.fillStyle = "hsl(0, 0%, 20%)";
        ctx.beginPath();

```

```

ctx.fillRect(-250, -250, 500, 500);
ctx.stroke();
ctx.fill();

ctx.strokeStyle = "black";
ctx.fillStyle = "hsl(0, 0%, 35%)";
ctx.beginPath();
ctx.arc(0, 0, radio_base - 50, 0, 2 * Math.PI);
ctx.stroke();
ctx.fill();

ctx.strokeStyle = "black";

var lineWidth = radio_shaft;
var pre_x = pre_y = 0;
var x_end = x / 5;
var y_end = y / 5;
var max_count = (radio_shaft - 10) / step;
var count = 1;

while (lineWidth >= 10) {
    var cur_x = Math.round(count * x_end / max_count);
    var cur_y = Math.round(count * y_end / max_count);
    console.log(cur_x);
    ctx.lineWidth = lineWidth;
    ctx.beginPath();
    ctx.lineTo(pre_x, pre_y);
    ctx.lineTo(cur_x, cur_y);
    ctx.stroke();

    lineWidth -= step;
    pre_x = cur_x;
    pre_y = cur_y;
    count++;
}

var x_start = Math.round(x / 3);
var y_start = Math.round(y / 3);
lineWidth += step;

ctx.beginPath();
ctx.lineTo(pre_x, pre_y);
ctx.lineTo(x_start, y_start);
ctx.stroke();

```

```

count = 1;
pre_x = x_start;
pre_y = y_start;

while (lineWidth < radio_shaft) {
    var cur_x = Math.round(x_start + count * (x - x_start) / max_count);
    var cur_y = Math.round(y_start + count * (y - y_start) / max_count);
    ctx.lineWidth = lineWidth;
    ctx.beginPath();
    ctx.lineTo(pre_x, pre_y);
    ctx.lineTo(cur_x, cur_y);
    ctx.stroke();

    lineWidth += step;
    pre_x = cur_x;
    pre_y = cur_y;
    count++;
}

var grd = ctx.createRadialGradient(x, y, 0, x, y, radio_handle);
for (var i = 85; i >= 50; i -= 5)
    grd.addColorStop((85 - i) / 35, "hsl(0, 100%, " + i + "%)");

ctx.fillStyle = grd;
ctx.beginPath();
ctx.arc(x, y, radio_handle, 0, 2 * Math.PI);
ctx.fill();
}

function procesarEvento(event) {
    var pos_x, pos_y;
    if (event.offsetX) {
        pos_x = event.offsetX - canvas_width / 2;
        pos_y = event.offsetY - canvas_height / 2;
    }
    else if (event.layerX) {
        pos_x = event.layerX - canvas_width / 2;
        pos_y = event.layerY - canvas_height / 2;
    }
    else {
        pos_x = (Math.round(event.touches[0].pageX -
event.touches[0].target.offsetLeft)).canvas_width / 2;
        pos_y = (Math.round(event.touches[0].pageY -
event.touches[0].target.offsetTop)).canvas_height / 2;
    }
}

```

```

    }

    return { x: pos_x, y: pos_y }
}
function mouse_down() {
    if (ws == null)
        return;

    event.preventDefault();
    var pos = procesarEvento(event);

    var delta_x = pos.x - joystick.x;
    var delta_y = pos.y - joystick.y;

    var dist = Math.sqrt(delta_x * delta_x + delta_y * delta_y);

    if (dist > radio_handle)
        return;

    click_state = 1;

    var radio = Math.sqrt(pos.x * pos.x + pos.y * pos.y);

    if (radio < (rango - radio_handle)) {
        joystick = pos;
        enviarDatos();
        actualizarVista();
    }
}
function mouse_up() {
    event.preventDefault();
    click_state = 0;
}
function mouse_move() {
    if (ws == null)
        return;

    event.preventDefault();

    if (!click_state)
        return;

    var pos = procesarEvento(event);

```

```

        var radio = Math.sqrt(pos.x * pos.x + pos.y * pos.y);

        if (radio < (rango - radio_handle)) {
            joystick = pos;
            enviarDatos();
            actualizarVista();
        }
    }
    window.onload = inicio;
</script>
</head>

<body>
    <h1>Aplicacion joystick NODE MCU</h1>
    <canvas id="remote"></canvas>
    <h2>
        <p>
            WebSocket : <span id="ws_state">null</span>
        </p>
        <button id="bt_connect" type="button" onclick="conectar();">CONECTAR</button>
    </h2>
</body>

</html>

```

## Código Fuente Arduino ID.

```
#include <ESP8266WiFi.h>
```

```
#include <WiFiClient.h>
```

```
#include <ESP8266WebServer.h>
```

```
#include <WebSocketsServer.h>
```

```
#include <Hash.h>
```

```
#include <FS.h>
```

```
//"ROBOTECISC" Nombre de la red del router vinculado al robot **ssid**.
```

```
//"sanitizador" Contraseña de la red del router vinculado al robot **password**.
```

```
const char* ssid = "IZZI-2023"; //Nombre de la red a usar.
```

```
const char* password = "3C0461FC2023"; //Contraseña de la red a usar.
```

```
//Motor Derecha
```

```
int OUTPUT4 = 16; // D0 => 16 o //D0 => '16'
```

```
int OUTPUT3 = 5; //D3 => '0' o //D1 o 5
```

//El marcador " hace referencia a los pines si se usa el Arduino MEGA, si no estan marcados son pines del Arduino uno, pero para ambas tarjetas sirven cualquiera.

```
//Motor Izquierda
```

```
int OUTPUT2 = 4; //D2 => 4 o //D4 => '2'
```

```
int OUTPUT1 = 0; //D3 => 0 0 //D5 => '14'
```

```
long duracion = 0;
```

```
//Se define el puerto 81 para el WebSocket y puerto 80 para pagina web
```

```
WebSocketsServer websocket = WebSocketsServer(81);
```

```
ESP8266WebServer server(80);
```

```
void setup(void) {
```

```
delay(1000);  
//Velocidad  
Serial.begin(115200);  
//Configuracion de Salidas  
pinMode (OUTPUT1, OUTPUT);  
pinMode (OUTPUT2, OUTPUT);  
pinMode (OUTPUT3, OUTPUT);  
pinMode (OUTPUT4, OUTPUT);  
  
//Establecer la conexion con el AP  
WiFi.begin(ssid, password);  
while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
}  
  
//Se establece la conexion y presenta la IP del cliente  
IPAddress myIP = WiFi.localIP();  
Serial.print("IP: ");  
Serial.println(myIP);  
  
//Inicia lectura del archivo  
SPIFFS.begin();  
//Inicia Websocket  
WebSocket.begin();  
WebSocket.onEvent(WebSocketEvent);  
//En caso de error 404 no se encuentra  
server.onNotFound([] () {
```



```

    if (!handleFileRead(server.uri()))
        server.send(404, "text/plain", "Archivo no encontrado");
    });

//Servidor Web iniciado
server.begin();
Serial.println("Servidor HTTP Iniciado");
}

void loop(void) {
    //Websocket a la espera de conexiones
    websocket.loop();
    //Servidor Web a la espera de conexiones
    server.handleClient();
    delay(10);
}

//Funcion predefinida de un WebSocket
void websocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t lenght) {
    switch (type) {
        //En caso de que un cliente se desconecte del websocket
        case WStype_DISCONNECTED: {
            Serial.printf("Usuario #%u - Desconectado\n", num);
            break;
        }
        //Cuando un cliente se conecta al websocket presenta la informacion del cliente conectado, IP y
        ID
        case WStype_CONNECTED: {
            IPAddress ip = websocket.remoteIP(num);

```

```

    Serial.printf("Nueva conexión: %d.%d.%d.%d Nombre: %s ID: %u\n", ip[0], ip[1], ip[2], ip[3],
payload, num);

    break;
}

case WType_TEXT: {
    String entrada = "";
    //Se lee la entrada de datos y se concatena via el servidor Websocket
    for (int i = 0; i < lenght; i++) {
        entrada.concat((char)payload[i]);
    }
    //Se separan los datos de la posicion X y Y del Joystick
    String data = entrada;
    if (data) {
        int pos = data.indexOf(':');
        long x = data.substring(0, pos).toInt();
        long y = data.substring(pos + 1).toInt();
        //Imprime en monitor serial
        Serial.print("x:");
        Serial.print(x);
        Serial.print(", y:");
        Serial.println(y);
        //De acuerdo al valor de X y Y del Joystick, se ejecuta la funcion para habilitar los motores
        if (((x <= 50 && x > - 50) && (y <= 50 && y > - 50))) { //Parar
            parar();
        } else if (x > 50 && (y < 50 || y > - 50)) { //Derecha
            derecha();
        } else if (x < - 50 && (y < 50 || y > - 50)) { //Izquierda
            izquierda();
        } else if (y > 50 && (x < 50 || x > - 50)) { //Atras

```

```

        atras();
    } else if (y < - 50 && (x < 50 || x > - 50)) { //Adelnte
        adelante();
    }
}
break;
}
}
}

```

//Funcion para indentificar el tipo de contenido de los archivos del servidor web

```

String getContentType(String filename) {
    if (server.hasArg("download")) return "application/octet-stream";
    else if (filename.endsWith(".htm")) return "text/html";
    else if (filename.endsWith(".html")) return "text/html";
    else if (filename.endsWith(".css")) return "text/css";
    else if (filename.endsWith(".js")) return "application/javascript";
    else if (filename.endsWith(".png")) return "image/png";
    else if (filename.endsWith(".gif")) return "image/gif";
    else if (filename.endsWith(".jpg")) return "image/jpeg";
    else if (filename.endsWith(".xml")) return "text/xml";
    else if (filename.endsWith(".pdf")) return "application/x-pdf";
    else if (filename.endsWith(".zip")) return "application/x-zip";
    else if (filename.endsWith(".gz")) return "application/x-gzip";
    return "text/plain";
}

```

//Funcion para cargar el archivo del servidor web index.html

```

bool handleFileRead(String path) {
#ifdef DEBUG
    Serial.println("handleFileRead: " + path);
#endif
    if (path.endsWith("/")) path += "index.html";
    if (SPIFFS.exists(path)) {
        File file = SPIFFS.open(path, "r");
        size_t sent = server.streamFile(file, getContentType(path));
        file.close();
        return true;
    }
    return false;
}

```

//Funciones para accionar los motores de acuerdo a los valores que se envian por medio del joystick

```

void adelante() {
    Serial.println("adelante");
    digitalWrite(OUTPUT1, 0);
    digitalWrite(OUTPUT2, 0);
    digitalWrite(OUTPUT3, 1);
    digitalWrite(OUTPUT4, 0);
}

```

```

void atras() {
    Serial.println("atras");
    digitalWrite(OUTPUT1, 1);
    digitalWrite(OUTPUT2, 0);
    digitalWrite(OUTPUT3, 0);
}

```

```
digitalWrite(OUTPUT4, 0);  
}
```

```
void derecha() {  
    Serial.println("derecha");  
    digitalWrite(OUTPUT1, 1);  
    digitalWrite(OUTPUT2, 0);  
    digitalWrite(OUTPUT3, 1);  
    digitalWrite(OUTPUT4, 0);  
}
```

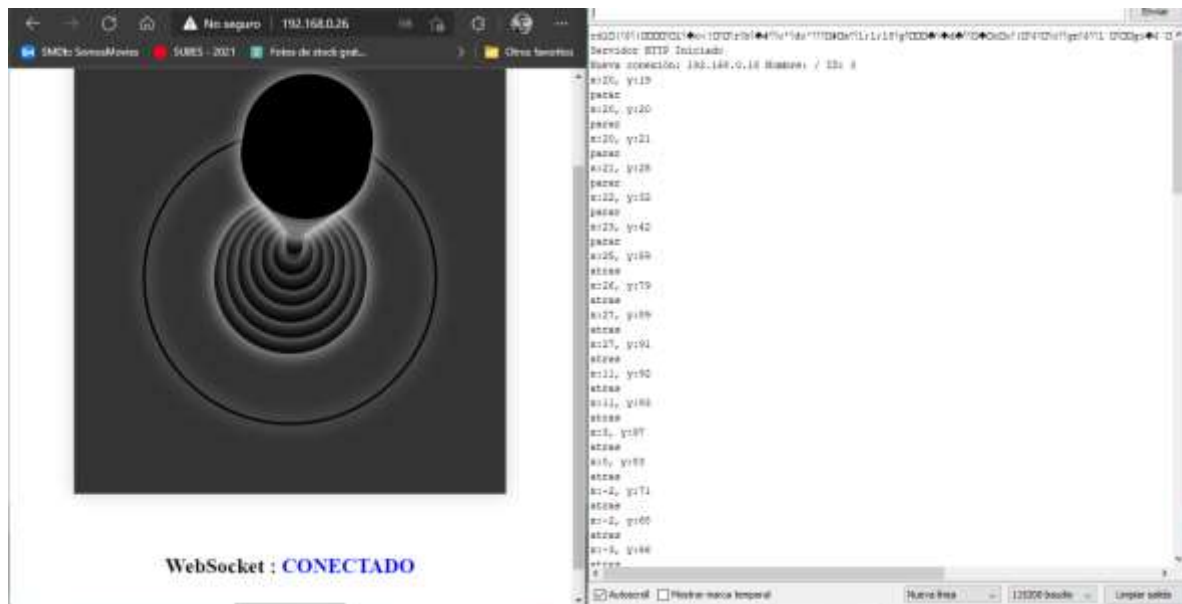
```
void izquierda() {  
    Serial.println("izquierda");  
    digitalWrite(OUTPUT1, 0);  
    digitalWrite(OUTPUT2, 1);  
    digitalWrite(OUTPUT3, 1);  
    digitalWrite(OUTPUT4, 0);  
}
```

```
void parar() {  
    Serial.println("parar");  
    digitalWrite(OUTPUT1, 0);  
    digitalWrite(OUTPUT2, 0);  
    digitalWrite(OUTPUT3, 0);  
    digitalWrite(OUTPUT4, 0);  
}
```

## Vista Pagina Web.



## Datos Obtenidos.



## Circuito 1. NodeMCU esp8266 y Arduino UNO.

En este punto, el siguiente circuito esta esquematizado con el Arduino UNO esto con el fin de que el autor no cuenta con la placa Arduino MEGA en la cual esta hecho el circuito original por lo cual las conexiones cambian, esto con el fin de hacer pruebas.

Pines Digitales Arduino Uno.

- PIN 8.
- PIN 9.
- PIN 10.
- PIN 11.

Pines de alimentación Arduino UNO.

- PIN 3v3.
- PIN GND.

Pines Node MCU esp8266.

- PIN D0
- PIN D1
- PIN D2
- PIN D3

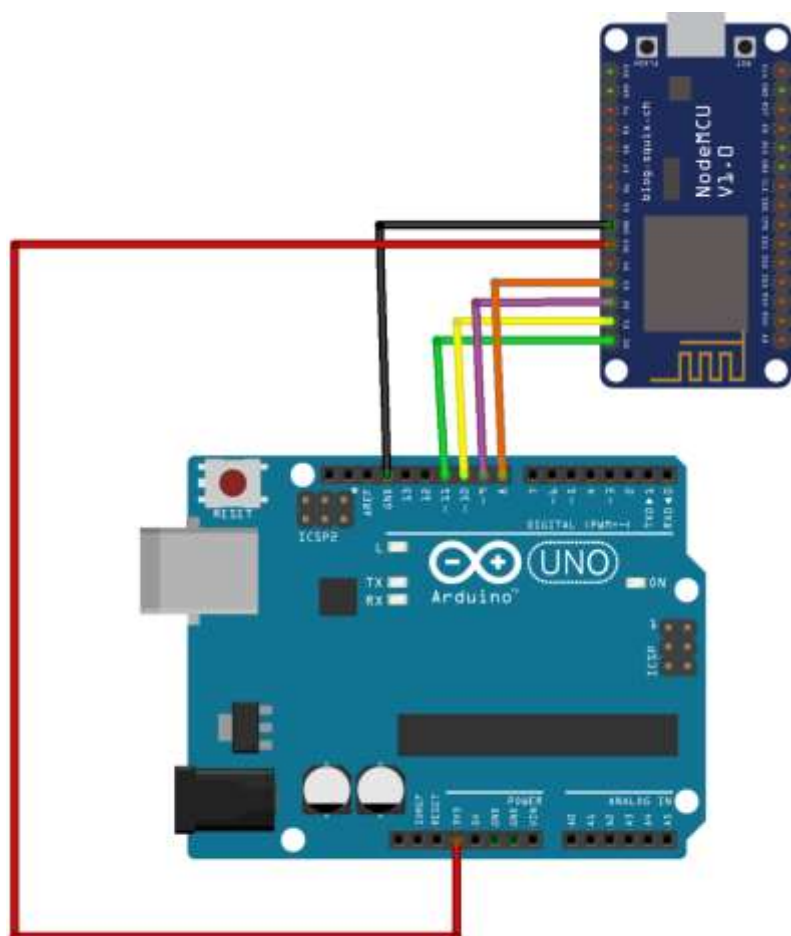
Pines de alimentación Node MCU esp8266.

- PIN 3v.
- PIN G.

Tabla de conexiones.

| Node MCU esp8266 | Arduino Uno. |
|------------------|--------------|
| PIN D0           | PIN 11       |
| PIN D1           | PIN 10       |
| PIN D2           | PIN 9        |
| PIN D3           | PIN 8        |
| PIN 3v           | PIN 3V3      |
| PIN G            | PIN GND      |

Diagrama de conexiones.



fritzing



## Circuito 2. Node MCU esp8266 y Arduino MEGA.

Este es el circuito original el cual esta montado en el robot, en este apartado se pondrán los pines con las que se realizaron las conexiones, así como su respectivo diagrama.

Pines Digitales Arduino MEGA.

- PIN 8.
- PIN 9.
- PIN 10.
- PIN 11.

Pines de alimentación Arduino MEGA.

- PIN 3v3.
- PIN GND.

Pines Node MCU esp8266.

- PIN D0.
- PIN D3.
- PIN D4.
- PIN D5.

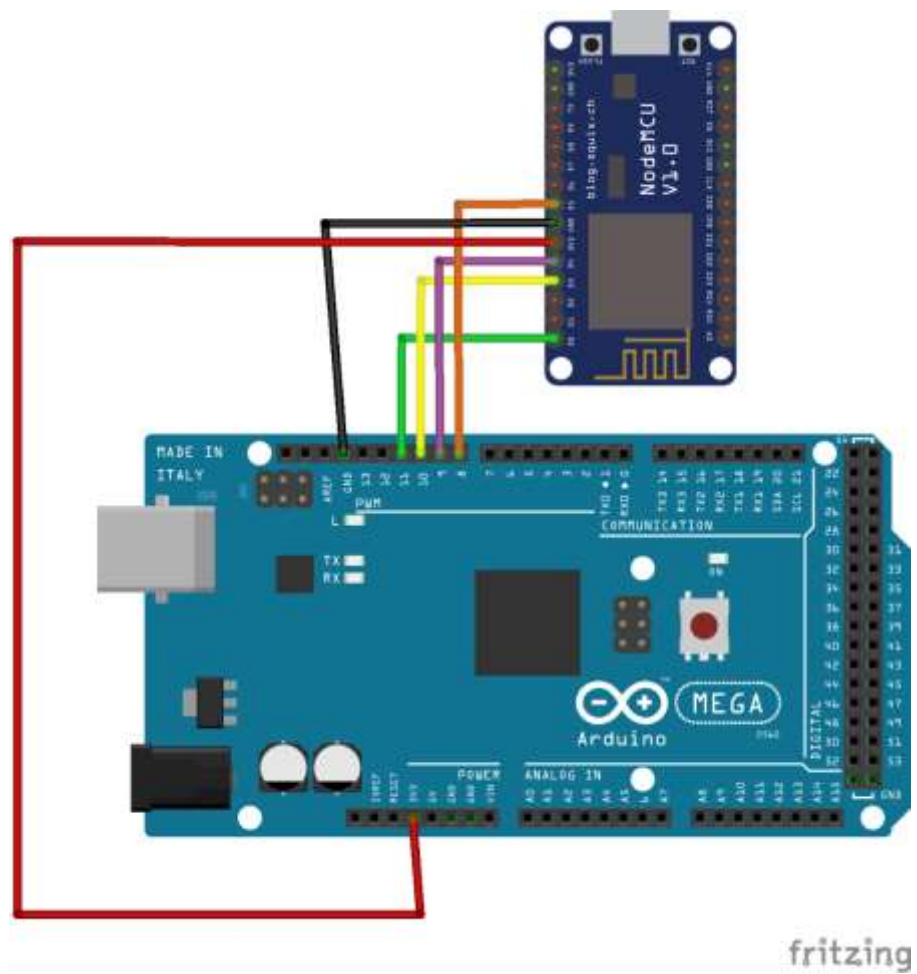
Pines de alimentación Node MCU esp8266.

- PIN 3v.
- PIN G.

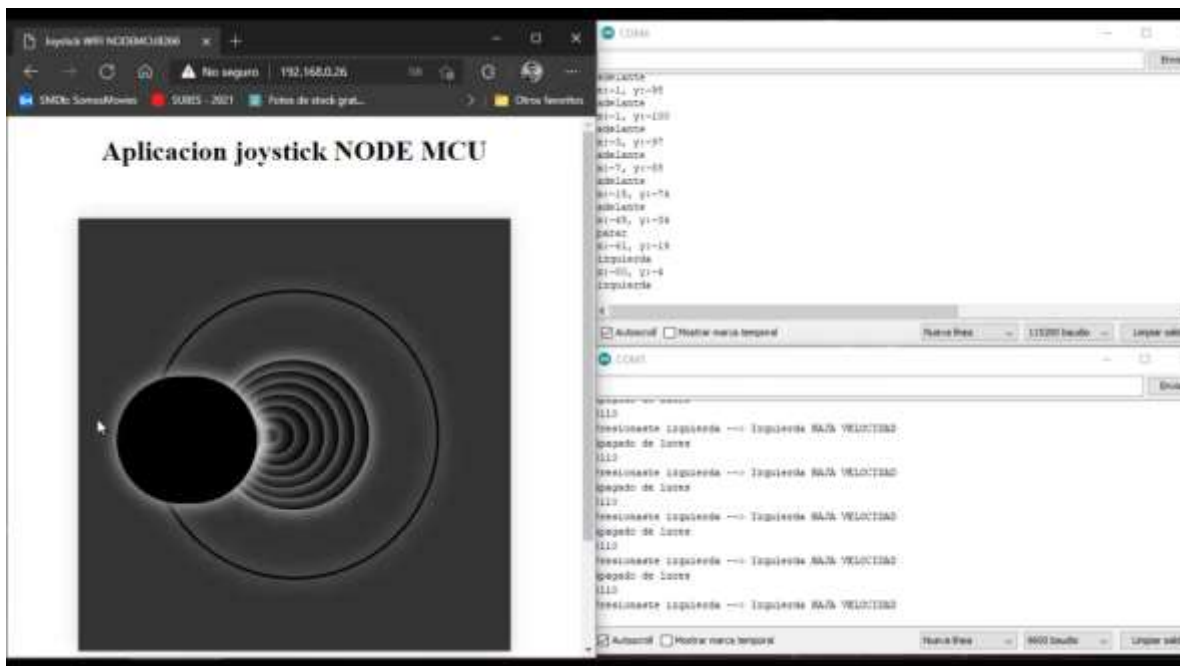
Tabla de conexiones.

| Node MCU esp8266 | Arduino MEGA. |
|------------------|---------------|
| PIN D0           | PIN 11        |
| PIN D3           | PIN 10        |
| PIN D4           | PIN 9         |
| PIN D5           | PIN 8         |
| PIN 3v           | PIN 3V3       |
| PIN G            | PIN GND       |

Diagrama de conexiones.




Funcionamiento.



## Código fuente Arduino ID control de velocidad y dirección de motores.

En esta sección se expone el código el cual este alojado en la tarjeta de Arduino ya sea "Uno" o "MEGA", este código es para el control de velocidad y dirección de los motores del robot.

 El código expuesto es solo una porción del código original, ya que solo se huso con fines de prueba y con la particularidad de que para este proyecto solo se requirió de esta porción de código.

```
//RECIBEN LA CADENA DE DATOS
```

```
int Recibe1=8;
```

```
int Recibe2=9;
```

```
int Recibe3=10;
```

```
int Recibe4=11;
```

```
//LENTURA DE DATOS
```

```
int P1=0;
```

```
int P2=0;
```

```
int P3=0;
```

```
int P4=0;

//PUENTE H
int A=3; //A  AZUL
int B=4; //B  AMARILLO
int C=5; //C  VERDE
int D=6; //D  NARANJA

//LUCES
int Luces=13;

void setup() {
  Serial.begin(9600);
  Serial.println("-----LISTO PARA TRABAJO-----");
  pinMode(A,OUTPUT);
  pinMode(B,OUTPUT);
  pinMode(C,OUTPUT);
  pinMode(D,OUTPUT);
  analogWrite(A,127);
  analogWrite(B,127);
  analogWrite(C,127);
  analogWrite(D,127);
  delay(3000);

  // PINES RECIBE DATOS
  pinMode(Recibe1,INPUT);
  pinMode(Recibe2,INPUT);
  pinMode(Recibe3,INPUT);
  pinMode(Recibe4,INPUT);
```

```
//ENCENDIDO DE LUCES
```

```
pinMode(Luces,OUTPUT);
```

```
}
```

```
void loop() {
```

```
P1=digitalRead(Recibe1);
```

```
P2=digitalRead(Recibe2);
```

```
P3=digitalRead(Recibe3);
```

```
P4=digitalRead(Recibe4);
```

```
Serial.print(P1);
```

```
Serial.print(P2);
```

```
Serial.print(P3);
```

```
Serial.print(P4);
```

```
Serial.print('\n');
```

```
if (P3==0 && P2==0 && P1==1)
```

```
{
```

```
Serial.println("Presionaste arriba -->Adelante VELOCIDAD 1");
```

```
analogWrite(A,155);
```

```
analogWrite(B,135);
```

```
analogWrite(C,158);
```

```
analogWrite(D,130);
```

```
}
```

```
if (P3==0 && P2==1 && P1==0)
```

```
{
```

```
Serial.println("Presionaste arriba -->Adelante VELOCIDAD 2");
```

```
analogWrite(A,180);
```

```
analogWrite(B,130);
```

```
analogWrite(C,185);
```

```
analogWrite(D,130); }
```

```

if (P3==0 && P2==1 && P1==1)
{
    Serial.println("Presionaste arriba -->Adelante VELOCIDAD 3");
    analogWrite(A,255);
    analogWrite(B,127);
    analogWrite(C,255);
    analogWrite(D,127);
}
if (P3==1 && P2==0 && P1==0)
{
    Serial.println("Presionaste abajo -->Reversa BAJA VELOCIDAD");
    analogWrite(A,0);
    analogWrite(B,127);
    analogWrite(C,0);
    analogWrite(D,127);
}
if (P3==0 && P2==0 && P1==0)
{
    Serial.println("Presionaste STOP --> ALTO TOTAL");
    analogWrite(A,127);
    analogWrite(C,127);
    analogWrite(D,127);
    analogWrite(B,127);
}
if (P3==1 && P2==0 && P1==1)
{
    Serial.println("Presionaste derecha --> Derecha BAJA VELOCIDAD");
    analogWrite(A,127);
    analogWrite(B,255);
    analogWrite(C,127);

```

```
analogWrite(D,255);
}
if (P3==1 && P2==1 && P1==0)
{
    Serial.println("Presionaste izquierda --> Izquierda BAJA VELOCIDAD");
    analogWrite(A,127);
    analogWrite(B,0);
    analogWrite(C,127);
    analogWrite(D,0);
}
if (P4==1)
{
    Serial.println("Encendido de luces");
    digitalWrite(Luces,HIGH);
}
else
{
    Serial.println("Apagado de luces");
    digitalWrite(Luces,LOW);
}

//delay(10);
}
```

## Evidencias.





## Referencias.

Fazt. (24 de Junio de 2019). Reactjs, Curso Práctico para Principiantes (React 16) [video]. YouTube. Obtenido de <https://youtu.be/zlY87vU33aA>

Fazt. (18 de Diciembre de 2020). React Native, Curso para Principiantes (usando Expo) [video]. YouTube. Obtenido de <https://youtu.be/hXDMWeD0ERM>

Ing. René Domínguez. (08 de Febrero de 2018). IOT NodeMCU ESP8266 - WiFi Robot Car WebSocket WebServer [video]. YouTube. Obtenido de [https://youtu.be/s\\_7wOnzSMLo](https://youtu.be/s_7wOnzSMLo)