



# **Manual Técnico**

## **Contacto telefónico**



## Índice

Índice .....	2
Introducción .....	4
Desarrollo.....	4
Configuración del servidor local.....	5
Instalación de NodeJS Windows .....	5
Desplegando proyecto base express NodeJS y express .....	5
Proyecto Contacto telefónico .....	7
Módulo Contactos .....	7
Estructura del código fuente.....	7
app / config .....	7
app / controller.....	7
app / database .....	8
app / entities.....	8
app / helper.....	8
app / middleware.....	8
app / model .....	8
app / routes.....	8
app / views .....	9
Public.....	9
Descargar código del proyecto desde GIT .....	9
Estructura de datos .....	10
MySQL.....	10
Despliegue en local .....	11
Despliegue en Azure y el servicio de App Web y BD.....	11
Creación del App services.....	12
Carga de la BD de MySQL en Azure .....	15
Despliegue de la app en Azure .....	17
Consideraciones importantes.....	18
Deploy en el servicio Azure.....	20
Evidencia de servicios POSTMAN.....	22
Usuario .....	22



Login .....	22
Registro usuario .....	22
Contacto.....	23
Listar contactos .....	23
Nuevo Contacto .....	23
Actualizar contacto.....	24
Eliminar contacto .....	24
Conclusión .....	25
Referencias.....	25



## Introducción

La siguiente documentación tiene como finalidad mostrar la parte teórica y técnica del desarrollo del sistema de *Contacto telefónico* que es parte del entregable de la materia de Ingeniería y desarrollo en la Web de la Maestría en Dirección e ingeniería de sitios Web. Se tratarán la terminología, la arquitectura de diseño implementada, donde se encuentra alojado tanto el sistema como el código fuente para su revisión y análisis de quien guste utilizarlo.

## Desarrollo

En este manual se verán la parte de la integración en un equipo local, requerimientos de software, configuración técnica, arquitectura de desarrollo, el acceso y persistencia de los datos, alojamiento del código fuente la red y por último como es su despliegue en un sistema gratuito en la nube proporcionado por Microsoft de nombre Web App.

La integración del sistema está hecha en un servidor de aplicaciones web bajo la tecnología de NodeJS, la arquitectura de software del proyecto se encuentra bajo el esquema Modelo Vista Controlador. El alojamiento del código fuente se encuentra en los repositorios web proporcionados por GitHub.

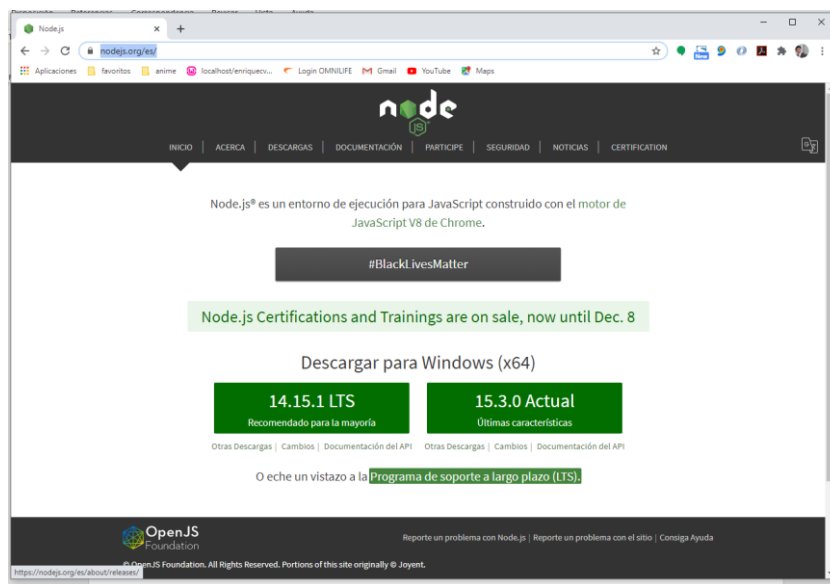
- El desarrollo del código fuente esta integrado en un solo proyecto, pero está completamente estructurado que podríamos considerar separado el front y el back
- El front esta hecho exclusivamente en HTML5, CSS3, JavaScript y jQuery y consume los servicios del Back por medio de peticiones AJAX
- El back esta realizado en Servicios Web, el cual cuenta con autenticación de Json Web Token (JWT), y utilizando los verbos de comunicación HTTP básicos entre un cliente y servidor: GET, POST, PUT y DELETE.
- Modelo del back, la persistencia de los datos del sistema se encuentra en base de datos MySQL.
- Controlador del back: Esta construido exclusivamente para delegar las peticiones que se reciben al consumir los servicios he ir hacia el modelo de la persistencia.
- Código fuente: Sera alojado en internet que es proporcionado por <https://github.com/>



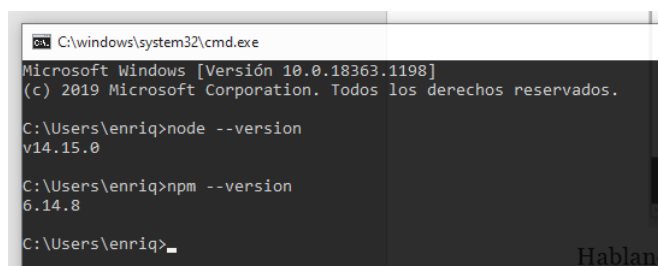
## Configuración del servidor local

### Instalación de NodeJS Windows

Primero debemos preparar el servidor en nuestro equipo local; debemos descargar el instalador de la página oficial <https://nodejs.org/es/> y seleccionamos la versión recomendada.



Hablando de Windows es como instalar cualquier programa, con el famoso “siguiente, siguiente, aceptar términos, siguiente, siguiente y finalizar”; en este caso para el instalador de NodeJS no debemos cambiar nada de lo predeterminado, al finalizar tendremos todo configurado incluido el path para node y el manejador de paquetes npm, podemos comprobarlo en la línea de comandos de Windows.



### Desplegando proyecto base express NodeJS y express

Para probar nuestro servidor y el framework de aplicaciones express, instalaremos la dependencia de express con el comando de manejador de paquetes npm:

```
npm install express --save
```



```
C:\windows\system32\cmd.exe

C:\servers\node\demo_js>npm install express --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\servers\node\demo_js\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\servers\node\demo_js\package.json'
npm WARN demo_js No description
npm WARN demo_js No repository field.
npm WARN demo_js No README data
npm WARN demo_js No license field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 2.901s
found 0 vulnerabilities

C:\servers\node\demo_js>
```

Seguido creamos hay que crear un archivo con extensión en javascript “.js” en algún directorio de Windows que desee agregar el proyecto, agregamos el código para levantar el proyecto con express con el siguiente código:

```
var express = require('express');
var app = express();

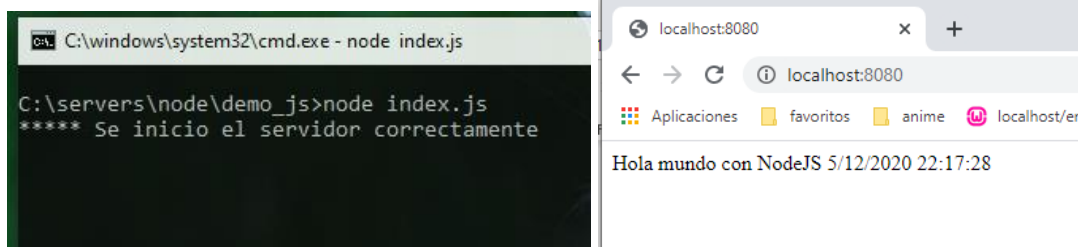
app.get('/', function(req, res) {
  res.send('Hola mundo con NodeJS ' + new Date().toLocaleString());
});

app.listen(8080, function() {
  console.log('***** Se inicio el servidor correctamente ');
});
```

En este caso salvamos el archivo con el nombre de **index.js**

Al final lo desplegamos con el comando de node:

*node index.js*



Es recomendable utilizar un IDE de desarrollo para facilitar nuestro trabajo codificando, puede ser libre como [Visual Studio Code](#), [NetBeans](#), [SublimeText](#), [Atom](#), [Eclipse](#); o de paga como [WebStorm](#), [PhpStorm](#) por mencionar algunos.



## Proyecto Contacto telefónico

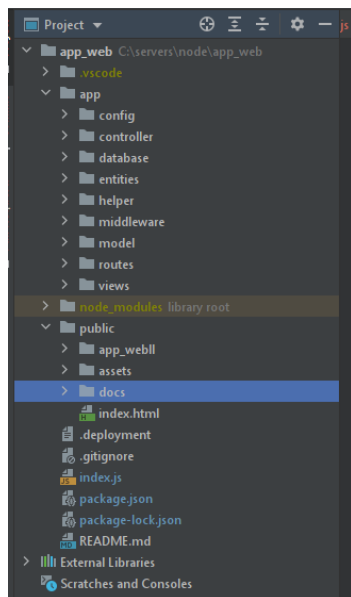
### Módulo Contactos

Sistema que se dedicará al CRUD de contactos y que se almacenará en la BD de MySQL que estará alojada en el propio Servidor de la aplicación.

Cabe mencionar que el código se integro en el mismo proyecto previo porque presente problemas en una implementación diferente para subirlo a la nube por los servicios alojados en la web que tengo activos.

### Estructura del código fuente

El código fuente del proyecto será:



- app\_web (proyecto raíz)
- | --- app (carpeta principal del proyecto)
- | | --- config
- | | --- controller
- | | --- database
- | | --- entities
- | | --- helper
- | | --- middleware
- | | --- model
- | | --- routes
- | | --- views
- | --- node\_modules (carpeta de librerías de node js)

- | --- public (carpeta destinada para el express static)

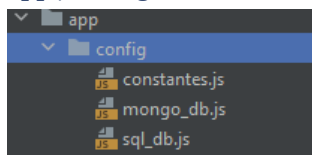
- | | --- assets

- | | --- docs

- | | --- index.js (archivo principal del proyecto)

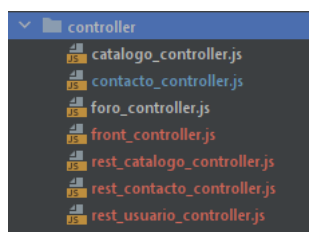
- | | --- package.json

app / config



Carpeta que contendrá la configuración del acceso de la base de datos y la configuración de las constantes

app / controller

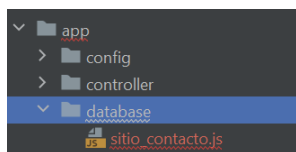


Carpeta que contendrá todos los controladores de la aplicación web, en estos estarán todas las funciones del CRUD y responderán con las vistas HTML necesarias o en su defecto un texto plano en formato JSON; validaran formularios que



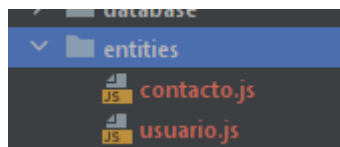
llegaron desde el front y se conectara con los modelos correspondiente de cada función en el controlador.

#### app / database



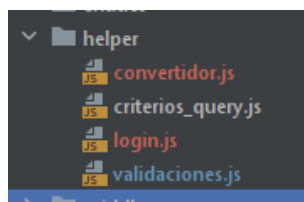
Carpeta que contendrá el script de BD que se crea en el motor de MySQL

#### app / entities



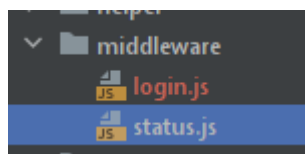
Carpeta que contendrá las entidades que se utilizaran en el modelo del proyecto del back

#### app / helper



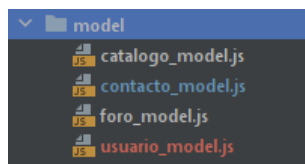
Carpeta que contendrá los ayudantes de código en el proyecto, por ejemplo: el constructor de la query adicional para el formulario de búsqueda y el validador del formulario de datos que se recibió de front

#### app / middleware



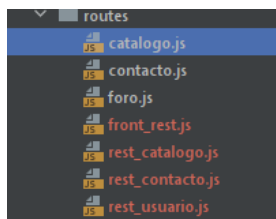
Carpeta que contendrá los middlewares que nos ayudaran a manejar los errores que se pueden presentar al momento de consumir los servicios de nuestra API rest o de cuando se trata de consumir servicios que necesitan de autenticación y esta no está presente

#### app / model



Carpeta para almacenar los modelos del proyecto, estos se dedicarán a cargar los datos de configuración de la BD y nos ayudarán a persistir los datos, por ejemplo: en el consultar, registrar, actualizar y eliminar datos. En este caso el catalogo\_model (obtendrá solo la lista del catálogo); el contacto\_model (configuración de BD con MySQL y que obtendrá el listado de contactos, el registro de contacto por id, actualizar/insertar un contacto y eliminación de un contacto; por último, la nueva entidad de usuario que contendrá el obtener los datos de inicio de sesión y el poder registrar un usuario nuevo al sistema para el login.

#### app / routes

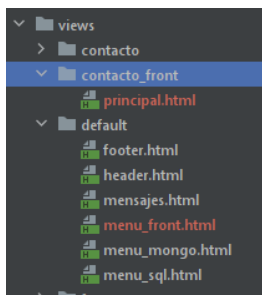


Carpeta que contendrá las rutas de todo el proyecto, en este caso las del catálogo, contacto y para las funciones del CRUD del proyecto.



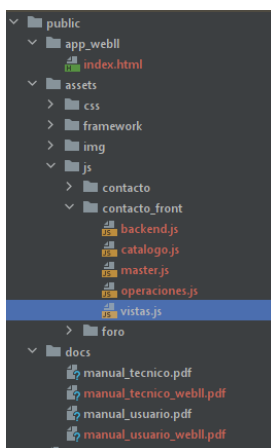


app / views



Carpeta que es exclusiva para las funciones del frontend, ya que contendrá las vistas que devolverán el controlador, se dividió en subdirectorios uno para las vistas por default, para las de `contact_front` con el único archivo de principal.

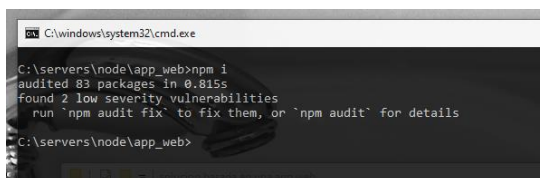
## Public



Carpeta public utilizada para el express para que tenga acceso el cliente a la única vista que carga el front, en ella se almacenaran los extras al sitio en sí, por ejemplo: los css, los framework de Bootstrap 4, jquery, las imágenes, los javascript utilizados y los documentos que tiene alojados la web de Contactos.

## Descargar código del proyecto desde GIT

Para desplegar el proyecto del repositorio es ubicarnos en el directorio donde lo vamos a descargar y descargar el código desde la fuente ([https://github.com/enriquecr1990/demo\\_nodejs](https://github.com/enriquecr1990/demo_nodejs)) y almacenarlo en dicho directorio. Una vez que lo tengamos vamos a actualizar las dependencias con el comando `npm i` y nos descargue de internet todo lo necesario para que funcione el proyecto.

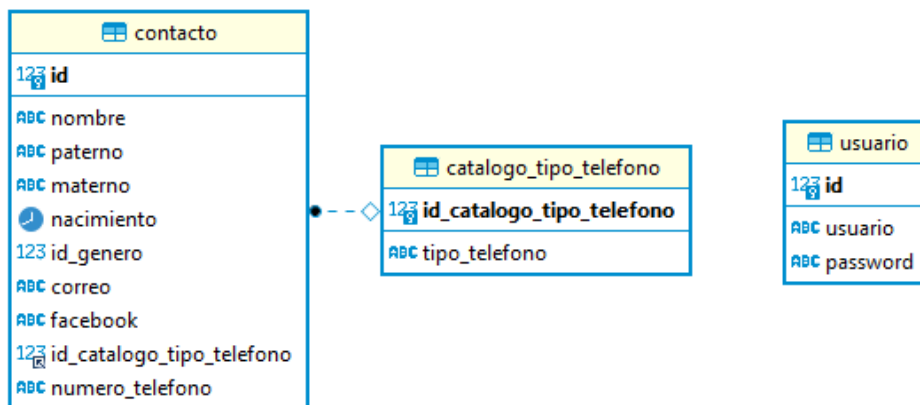




## Estructura de datos

### MySQL

La estructura de datos para el sistema en el módulo de contactos constara de dos tablas sencillas y solo tendrá una relación de uno a uno.



```
CREATE DATABASE IF NOT EXISTS `sitio_contacto` DEFAULT CHARACTER SET utf8 ;
USE `sitio_contacto` ;

CREATE TABLE IF NOT EXISTS `sitio_contacto`.`catalogo_tipo_telefono` (
  `id_catalogo_tipo_telefono` INT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `tipo_telefono` VARCHAR(150) NOT NULL,
  PRIMARY KEY (`id_catalogo_tipo_telefono`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `sitio_contacto`.`contacto` (
  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(75) NOT NULL,
  `paterno` VARCHAR(45) NOT NULL,
  `materno` VARCHAR(45) NOT NULL,
  `nacimiento` DATE NOT NULL,
  `id_genero` INT(1) UNSIGNED NOT NULL,
  `correo` VARCHAR(100) NOT NULL,
  `facebook` VARCHAR(150) NOT NULL,
  `id_catalogo_tipo_telefono` INT(3) UNSIGNED NOT NULL,
  `numero_telefono` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_contacto_catalogo_tipo_telefono_idx` (`id_catalogo_tipo_telefono` ASC),
  CONSTRAINT `fk_contacto_catalogo_tipo_telefono`
FOREIGN KEY (`id_catalogo_tipo_telefono`)
REFERENCES `sitio_contacto`.`catalogo_tipo_telefono` (`id_catalogo_tipo_telefono`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `sitio_contacto`.`usuario` (
  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `usuario` VARCHAR(100) NOT NULL,
  `password` VARCHAR(500) NOT NULL,
  PRIMARY KEY (`id`)
)
ENGINE = InnoDB;

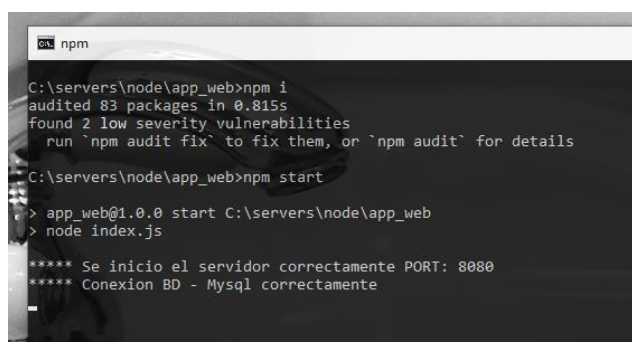
START TRANSACTION;
USE `sitio_contacto`;
INSERT INTO `sitio_contacto`.`catalogo_tipo_telefono` (`id_catalogo_tipo_telefono`,
`tipo_telefono`) VALUES (1, 'Celular');
INSERT INTO `sitio_contacto`.`catalogo_tipo_telefono` (`id_catalogo_tipo_telefono`,
`tipo_telefono`) VALUES (2, 'Casa');
INSERT INTO `sitio_contacto`.`catalogo_tipo_telefono` (`id_catalogo_tipo_telefono`,
`tipo_telefono`) VALUES (3, 'Oficina');
```



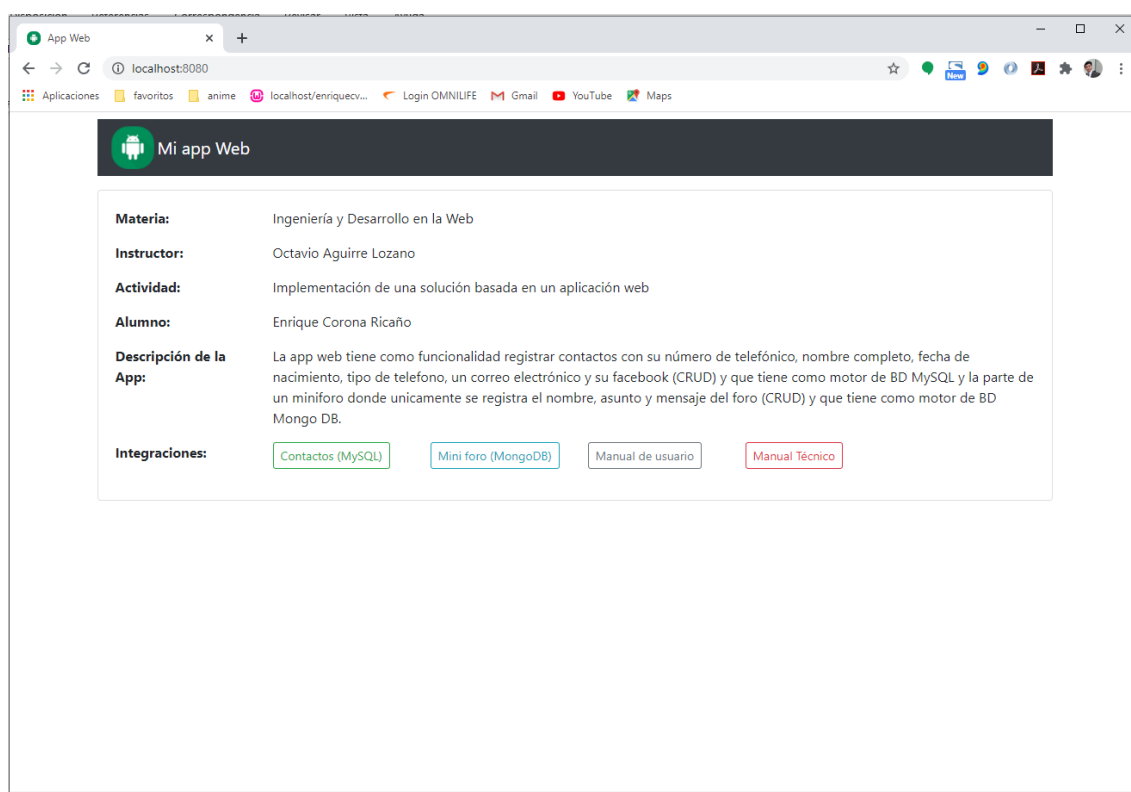
```
INSERT INTO `sitio_contacto`.`catalogo_tipo_telefono` (`id_catalogo_tipo_telefono`,  
`tipo_telefono`) VALUES (4, 'Fax');  
  
COMMIT;
```

### Despliegue en local

Una vez descargado el proyecto e instalado las dependencias y configurado la BD local en MySQL podemos proceder para su implementación en local. Para ello vamos a ubicarnos en la carpeta del proyecto y abrir una terminal de comandos y ejecutar el *npm start*. Si todo sale bien podremos ver el mensaje del despliegue correctamente y podremos abrir el servidor desde un navegador web. Con la ruta <http://localhost:8080>



```
npm  
C:\servers\node\app_web>npm i  
audited 83 packages in 0.815s  
found 2 low severity vulnerabilities  
run 'npm audit fix' to fix them, or 'npm audit' for details  
  
C:\servers\node\app_web>npm start  
  
> app_web@1.0.0 start C:\servers\node\app_web  
> node index.js  
  
***** Se inicio el servidor correctamente PORT: 8080  
***** Conexion BD - Mysql correctamente
```



### Despliegue en Azure y el servicio de App Web y BD

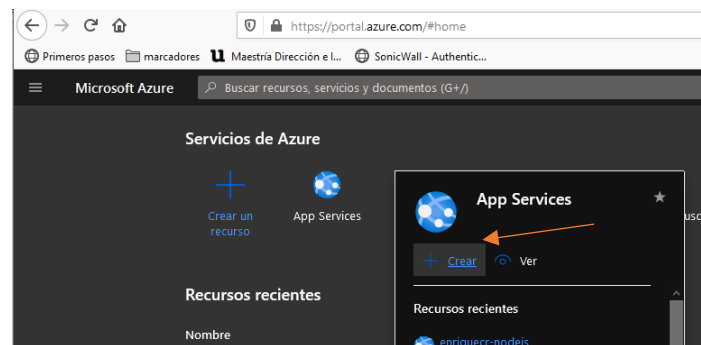
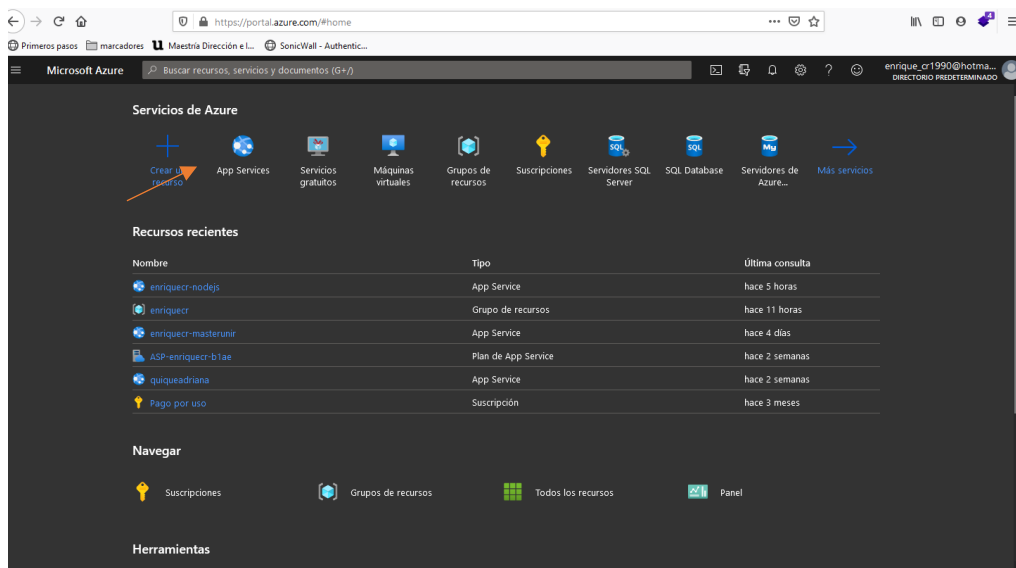
Para la parte de la implementación en el servicio de Azure, primero hay que considerar en realizar una cuenta en su portal; como requisito indispensable es tener una Tarjeta



bancaria para que se pueda ligar a cuenta, aunque el servicio que usemos es gratuito es parte de sus requerimientos que se de alta la tarjeta (se hará un cargo de por lo regular 1 dólar, que será reembolsable).

### ***Creación del App services***

Ingresamos al portal de Azure y buscamos la opción de crear servicio, para lograrlo vamos a la opción de *App Services* y de la ventana emergente que nos aparezca clickear en la opción de *crear*



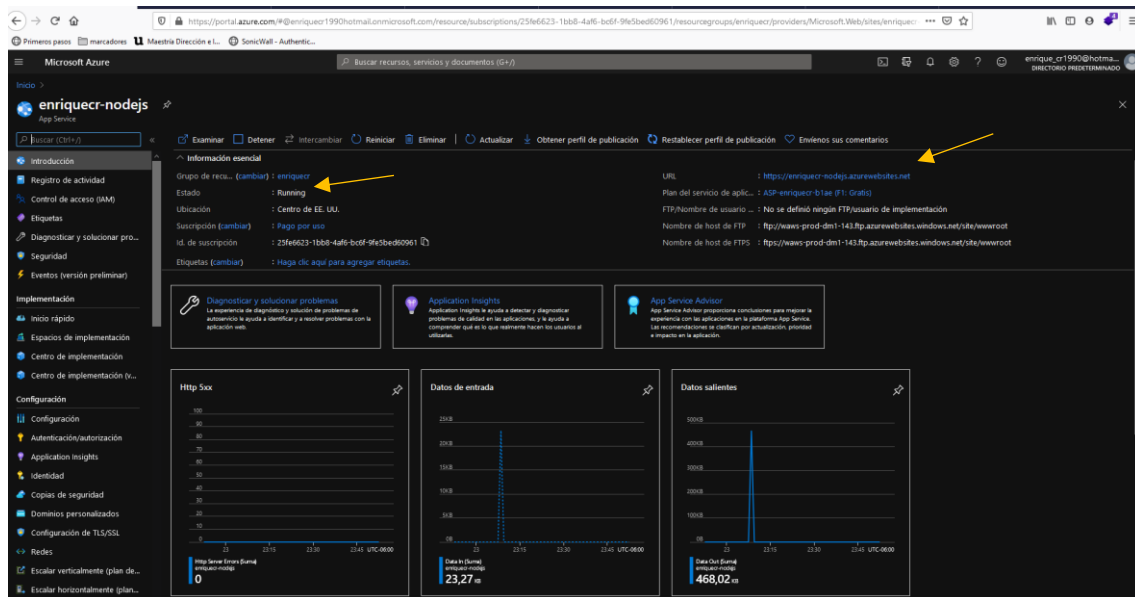
La siguiente pantalla es para configurar nuestra aplicación como lo es la suscripción, el grupo de recursos (en caso de que no exista darle la opción de crear uno y poner un nombre), en el nombre de la instancia nos sirve para darle la dirección de internet URL para que podamos acceder a ella, en publicar lo dejamos en la opción predeterminada y la pila de ejecución lo dejamos en la versión mas estable para el servicio (Node 12 LTS), para el sistema operativo pueden seleccionar Linux o Windows (en mi caso deje Windows que es el que me permite alojar una BD pequeña en MySQL), la región dejamos la predeterminada. Para el apartado de plan de servicio hay que poner atención si solamente queremos la parte gratuita, para ello nos iremos a cambiar el plan de tamaño,



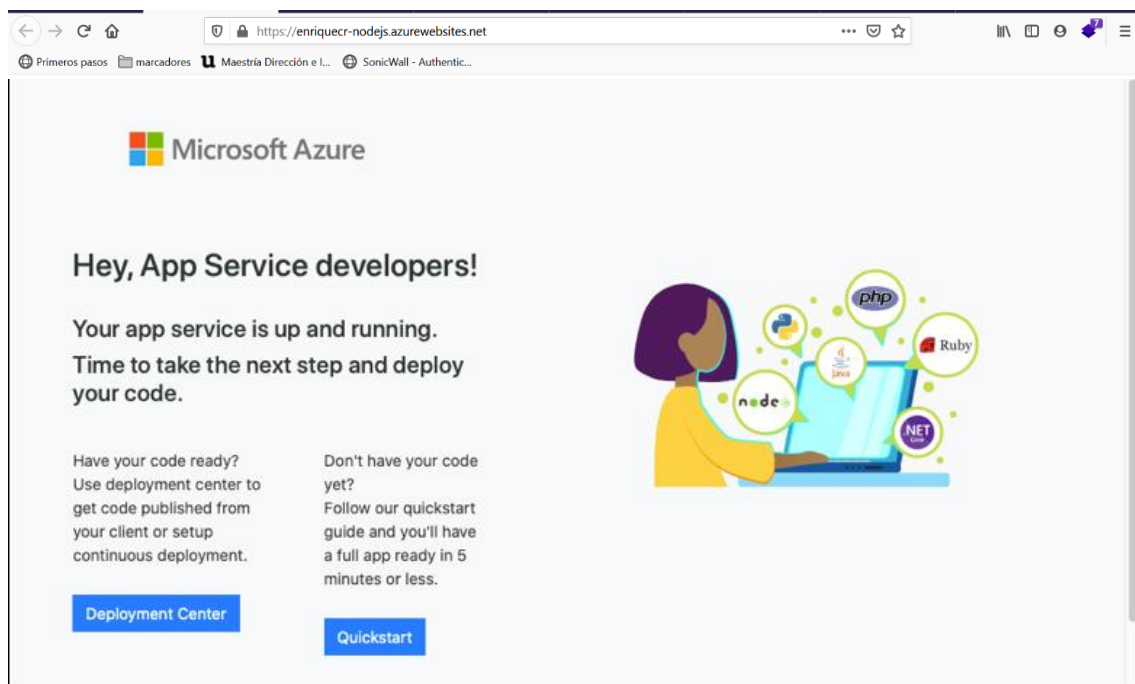
buscamos la opción de desarrollo y pruebas y buscamos la opción de 1 giga de RAM con 60 minutos diarios de uso y en aplicar.

Al finalizar la configuración solo basta con darle click en revisar y crear

Si todo sale bien como esperamos, tendremos una vista como esta y en estado de corriendo



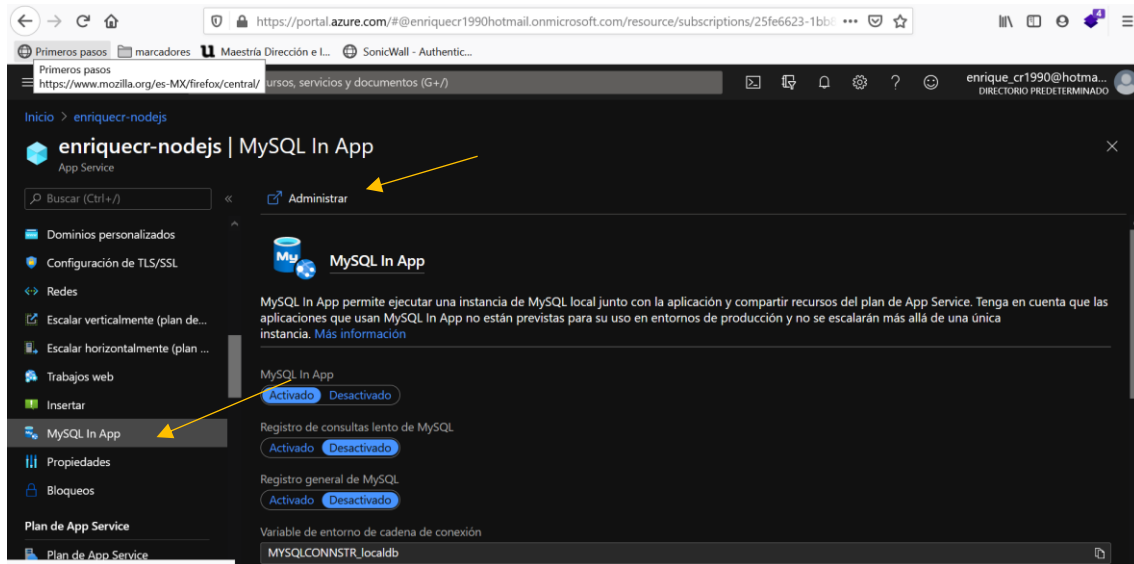
Cabe mencionar que al ser un servicio gratuito, este se activara prácticamente al momento de ingresar a la url del sitio; por ello es importante antes de intentar hacer despliegues (y en mi caso crear la BD) ingrese por lo menos una vez a la dirección del servicio. Nos daremos cuenta qué todo estará bien si vemos una pantalla como esta.





## Carga de la BD de MySQL en Azure

Con el paso anterior, hicimos que se crearan el servicio; ya podremos ingresar al admin de MySQL, para ello buscaremos la opción de MySQL In App del menú izquierdo de nuestro servicio e ir a la opción de Administrar; al hacerlo nos mostrara el sistema de phpMyAdmin y podremos crear nuestro esquema de BD y nuestras tablas correspondientes.



En caso de que nuestro servicio de MySQL estuviera desactivado, basta solo con activarlo para ponerlo en funcionamiento (hay que esperar unos segundos después de su activación para poder entrar, si no lo hicieramos nos mostraria una pantalla de Access a socket forbidden)

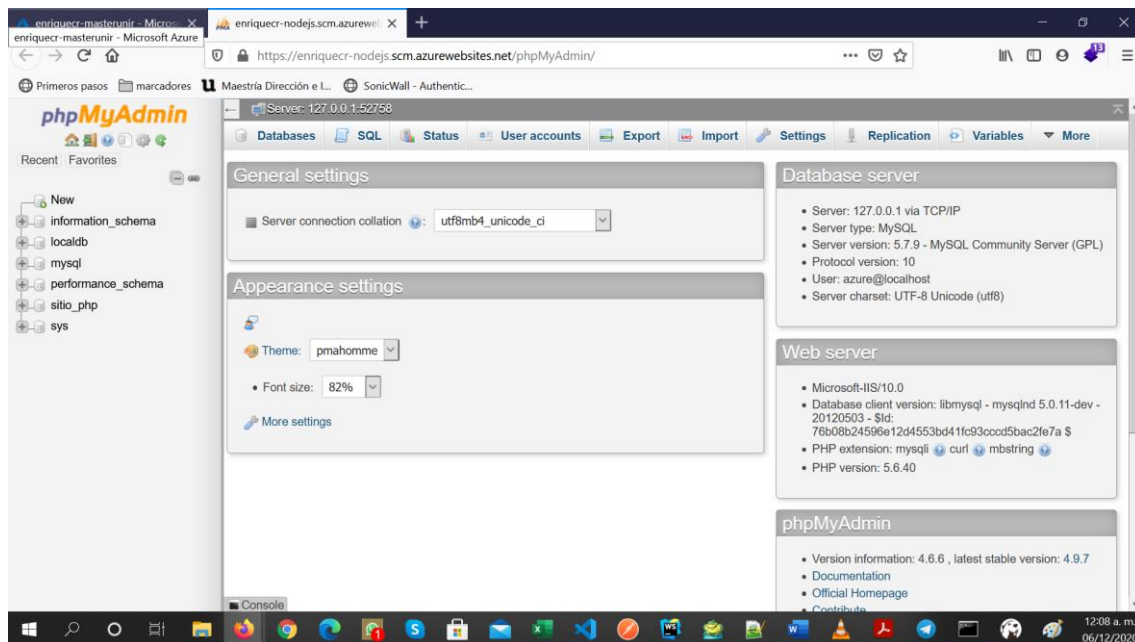


Welcome to phpMyAdmin

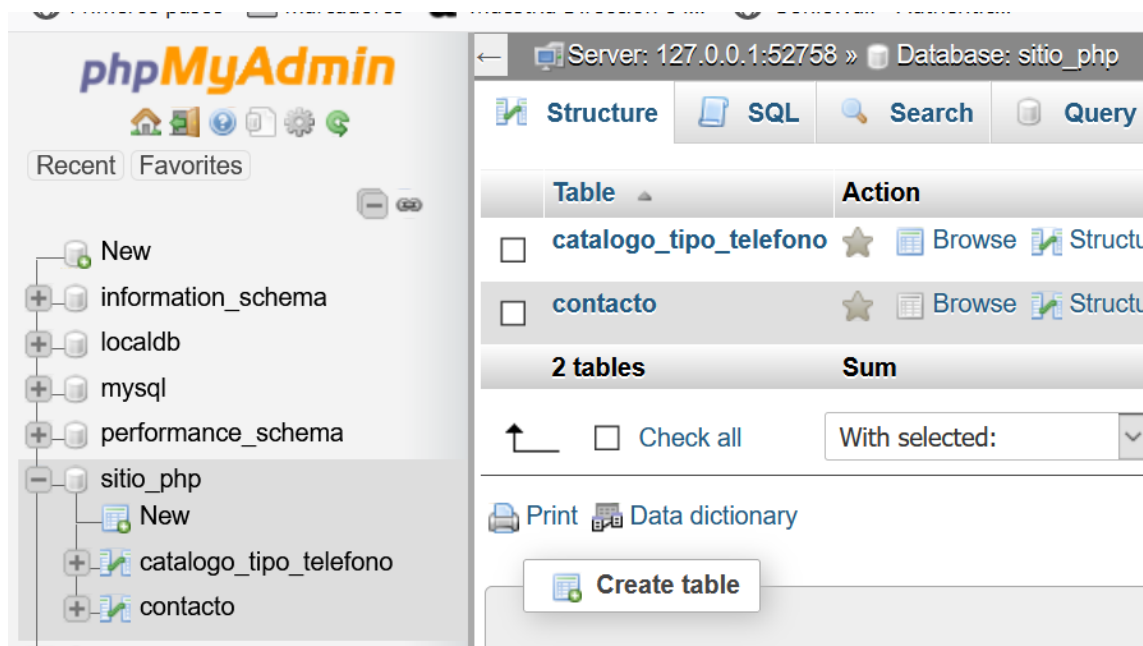




Si todo sale bien podremos ver el portal de phpMyAdmin y podremos agregar nuestra BD como nuestro script SQL para crear las tablas



En el phpMyAdmin nos servirá para saber que puerto esta usando el servidor para la BD, en este caso es el 52758 y que usaremos en el archivo de configuración sql de nuestra app



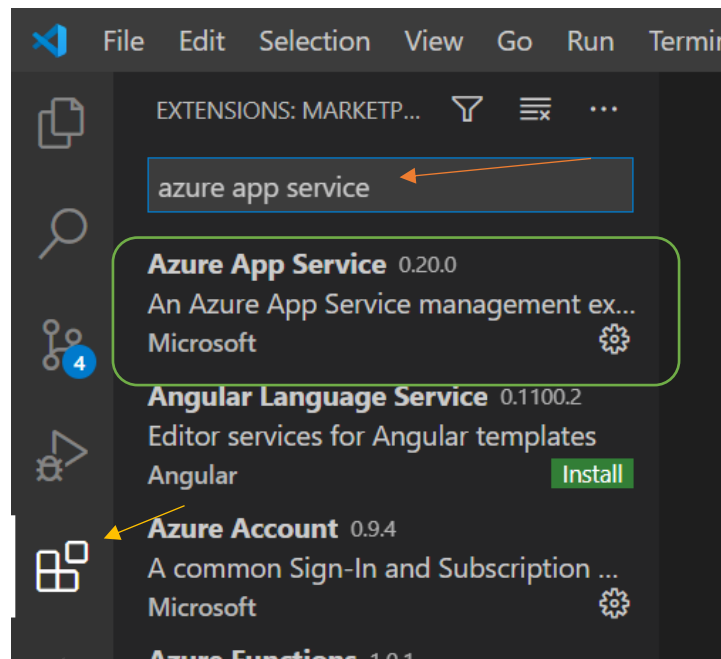
Con esto ya tenemos todo listo para hacer nuestro despliegue de la app hecha con nodeJS



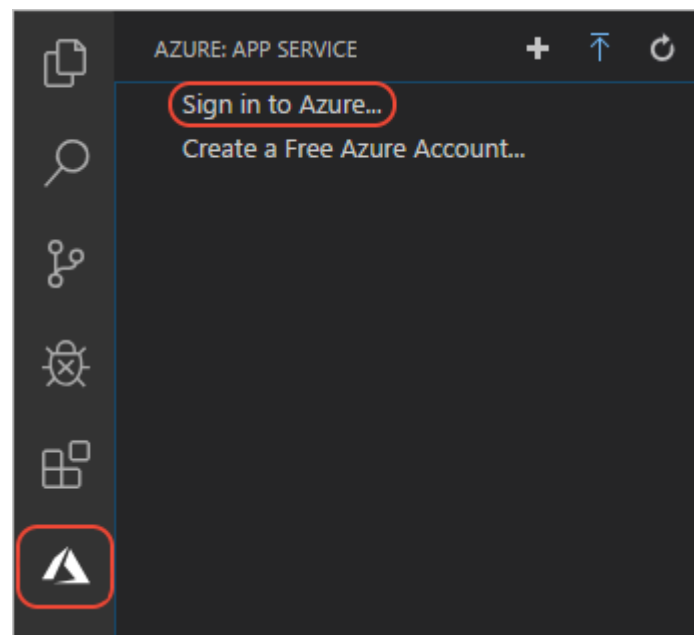


### ***Despliegue de la app en Azure***

Para poder realizar nuestro despliegue, haremos uso del IDE de desarrollo Visual Studio Code pero primero debemos instalar la extensión de nombre AZURE APP SERVICE, lo que nos ayudara a conectarnos a nuestra cuenta y poder realizar el despliegue de nuestro proyecto.



Ingresamos a la nueva opción de la extensión e iniciamos sesión con nuestras credenciales correspondientes



Con lo anterior listo, hay que abrir el folder donde se encuentra alojado nuestro proyecto localmente y poderlo ver en la parte de archivos



### Consideraciones importantes

Para que no exista ningún problema en el despliegue, nuestra aplicación hay que configurar el puerto por el cual se levanta nuestro server; en el servicio de Azure existen variables y puertos configurados que nosotros no tenemos acceso (en las cuentas gratuitas), es por eso que debemos dejar lista nuestra app para esa eventualidad

Lo anterior se solventa con el código y hacemos uso de las variables de configuración del server, y si no existieran dejamos un puerto por default (para nuestro ambiente local); creamos la constante y lo dejamos listo para que escuche ese puerto

```
const PORT = process.env.PORT || 8080;

.
.
.

//se agrega el listen para arrancar el servidor en el puerto de la constante
app.listen(PORT,function(){
  console.log('***** Se inicio el servidor correctamente PORT: ' + PORT);
});
```

El despliegue de la aplicación en la nube también considera que exista el archivo package.json y que contenga el comando “start” en el arreglo de scripts y apuntarlo a nuestro JS principal (en mi caso index.js) que es lo que hará que nuestra app con node JS pueda inicializarse.

```
JS index.js  {} package.json X
{} package.json > ...
1  {}
2  "name": "app_web",
3  "version": "1.0.0",
4  "description": "Implementación de una solución basada en una
5  "main": "index.js",
6  > Debug
7  "scripts": {
8    "start": "node index.js",
9    "test": "echo \"Error: no test specified\" && exit 1"
10 },
    "repository": {
```

Para tener el package.json de la forma correcta es correr el comando en la raíz del proyecto `npm init` y llenar los datos solicitados de su ejecución



```
C:\windows\system32\cmd.exe

C:\servers\node\demo_js>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

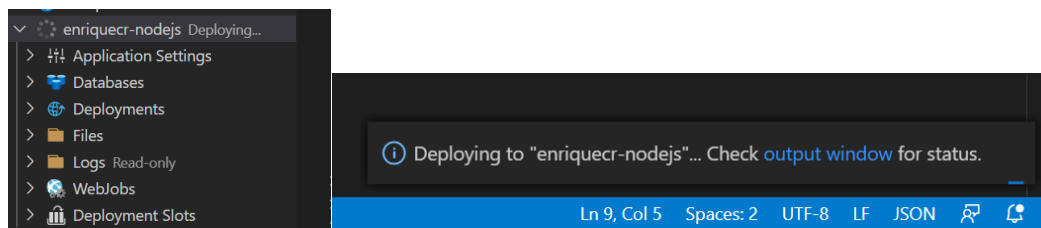
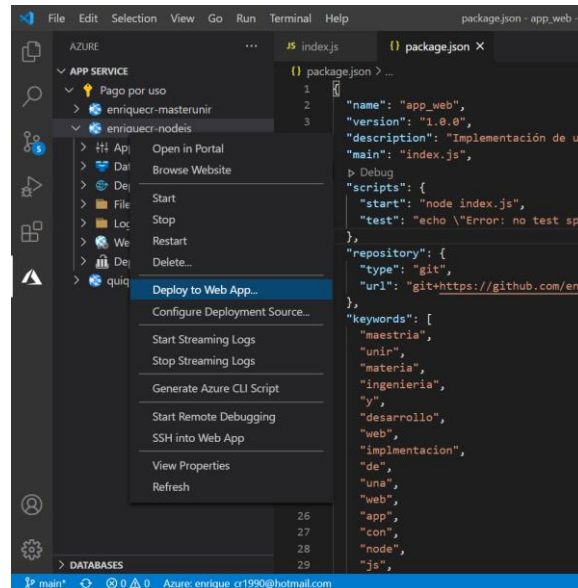
Press ^C at any time to quit.
package name: (demo_js)
version: (1.0.0)
description: la descripcion del proyecto node js
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\servers\node\demo_js\package.json:
{
  "name": "demo_js",
  "version": "1.0.0",
  "description": "la descripcion del proyecto node js",
  "main": "index.js",
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
C:\servers\node\demo_js>_
```

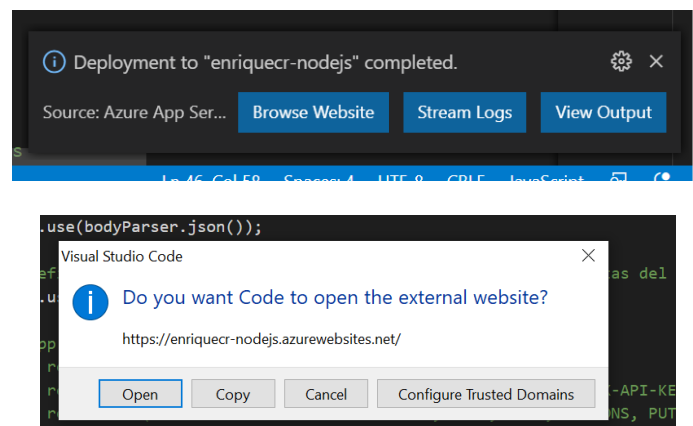


## Deploy en el servicio Azure

Para subir el código ya con todas las configuraciones previas, es ir a la extensión de azure y encontrar nuestro servicio creado y seleccionar la opción de “Deploy to Web App”, nos pedirá la configuración de la implementación y esperamos ah que termine de subir el código.



Una vez que termine el despliegue solo basta con visitar el sitio web desde la URL del servicio o dar click en el botón “Browse Website” del mensaje de la implementación finalizada.





## Manual técnico – Contacto telefónico



Mi app Web

**Materia:** Ingeniería y Desarrollo en la Web

**Instructor:** Octavio Aguirre Lozano

**Actividad:** Implementación de una solución basada en un aplicación web versión II

**Alumno:** Enrique Corona Ricaño

**Descripción de la App:** La app web tiene como funcionalidad registrar contactos con su número de telefónico, nombre completo, fecha de nacimiento, tipo de telefono, un correo electrónico y su facebook (CRUD) y que tiene como motor de BD MySQL y la parte de un miniforo donde unicamente se registra el nombre, asunto y mensaje del foro (CRUD) y que tiene como motor de BD Mongo DB. En la versión II se realiza la construcción de servicios Web REST el cual se consume como un api y cada funcion devuelve un texto en Formato JSON teniendo la siguiente estructura

1. /api/rest/catalogo: Para obtener el catalogo de tipo de telefono
2. /api/rest/contacto: Conjunto de peticiones para el contacto y que necesita login para poder consumirlos (integración realizada por Json Web Token JWT):
  - o GET: Obtiene el listado de contacto
  - o /id\_contacto GET: Obtiene el registro de contacto
  - o POST: Guardar un contacto nuevo
  - o /id\_contacto PUT: actualiza los datos de contacto
  - o /id\_contacto DELETE: actualiza los datos de contacto

**Integraciones:** [Contactos \(MySQL\)](#) [Manual de usuario](#) [Manual Técnico](#)

Mi app Web Enrique Corona App con MySQL UNIR México Cerrar Sesión

Resultados

[Nuevo](#)

#	Nombre completo	Genero	Cumpleaños	Número(s)	Contacto	
1	Enrique Coron Ricaño	Masculino	4 de junio	<b>Celular:</b> 2467575099	<b>Correo:</b> enrique_cr1990@hotmail.com <b>Facebook:</b> https://www.facebook.com/anoroc.euqirne/	<a href="#">Modificar</a> <a href="#">Eliminar</a>
2	Octavio Aguirre Lozano	Masculino	1 de octubre	<b>Celular:</b> 5555555555	<b>Correo:</b> octavio.aguirre@octavioaguirre.net <b>Facebook:</b>	<a href="#">Modificar</a> <a href="#">Eliminar</a>

Contacto

Nombre: \*

Nombre

Paterno: \*

Apellido paterno

Materno: \*

Apellido materno

Genero: \*

☐ Masculino  
☐ Femenino

Nacimiento: \*

dd / mm / aa

Numero de telefono: \*

-- Selecc --  
Número de telefoni

Por favor, llena este campo.

Correo:

Correo electrónico

Facebook:

Facebook

Los campos con \* son obligatorios

Cancelar

Guardar

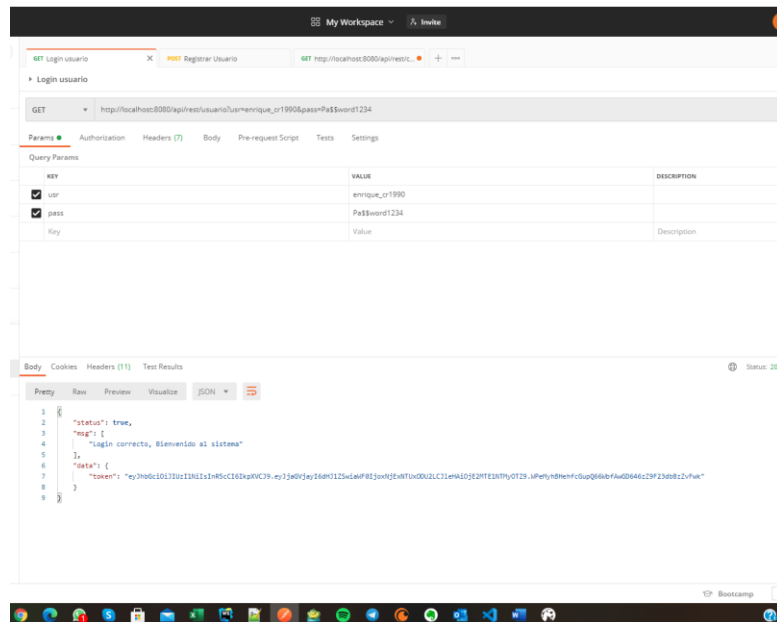


## Evidencia de servicios POSTMAN

### Usuario

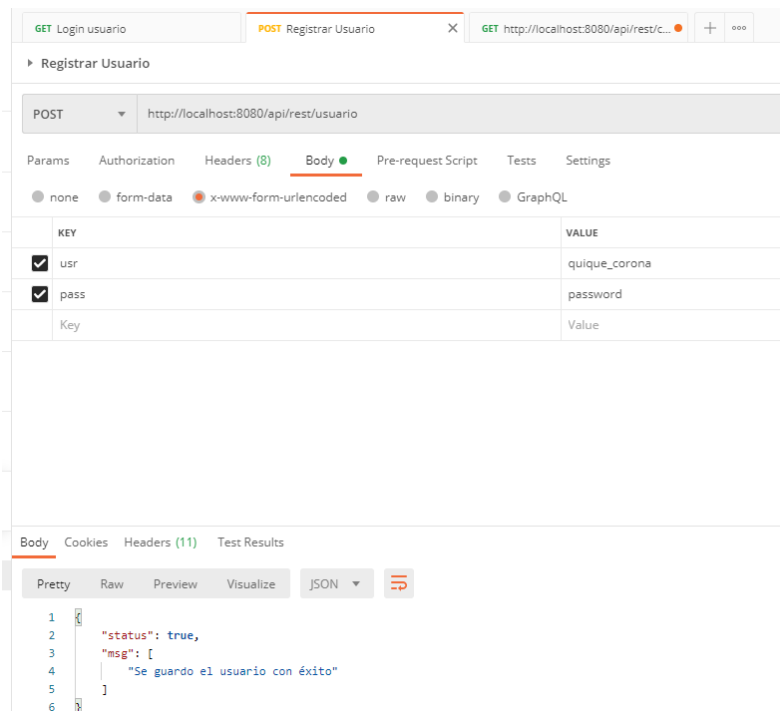
#### Login

Petición GET para el login en el sistema y obtener el token de autenticación para el uso de servicios



#### Registro usuario

Petición POST para poder registrar un usuario nuevo al sistema





## Contacto

### Listar contactos

#### Petición GET para listar los contactos registrados

GET Login usuario POST Registrar Usuario GET Listar contactos

GET http://localhost:8080/api/rest/contacto

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers 6 hidden

KEY	VALUE	DESCRIPTION
access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjaGVhZGciOiJ1ZjZ5aGVhZGciOiJ2ZW5iaW90aXNj	
Key	Value	Description

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": true,
3   "msg": "",
4   "data": [
5     {
6       "id": 1,
7       "nombre": "Quique Modificado",
8       "paterno": "Corona",
9       "materno": "Ricalto",
10      "nacimiento": "1990-04-06T06:00:00.000Z",
11      "id_genero": 1,
12      "correo": "enrique_cr1990@hotmail.com",
13      "facebook": "https://www.facebook.com/nifacel23456",
14      "id_catalogo_tipo_telefono": 1,
15      "numero_telefono": "246 757 5099",
16      "genero": "Masculino",
17      "tipo_telefono": "Celular"
18    },
19    {
20      "id": 5,
21      "nombre": "uno modificado",
22      "paterno": "nuevo",
23      "materno": "registro",
24      "nacimiento": "1990-04-06T06:00:00.000Z",
25      "id_genero": 1,
26      "correo": "kike@correo.com",
27    }
28  ]
29 }
```

### Nuevo Contacto

#### Petición POST para registrar un contacto nuevo

GET Login usuario POST Registrar Usuario GET Listar contactos POST Nuevo contacto

POST http://localhost:8080/api/rest/contacto

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers 7 hidden

KEY	VALUE
access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjaGVhZGciOiJ1ZjZ5aGVhZGciOiJ2ZW5iaW90aXNj
Key	Value

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": false,
3   "msg": [
4     "El campo nombre es requerido o no es valido",
5     "El campo apellido paterno es requerido o no es valido",
6     "El campo apellido materno es requerido o no es valido",
7     "El campo genero es requerido o no es valido",
8     "El campo tipo de teléfono es requerido o no es valido",
9     "El campo fecha de nacimiento es requerido o no es valido",
10    "El campo número de telefono es requerido o no es valido"
11  ]
12 }
```



## Actualizar contacto

## Petición PUT para actualizar un contacto existente

**My Workspace** ⌵ + Invite

---

GET Login usuario
POST Registrar Usuario
GET Listar contactos
POST Nuevo contacto
PUT Actualizar contacto

---

▶ Actualizar contacto

---

PUT
http://localhost:8080/api/rest/contacto/1

---

Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings

---

Headers
👁️ 7 hidden

KEY	VALUE
<input checked="" type="checkbox"/> access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjaGVjayl6dHJ1ZSwiaWF0IjoxNj
Key	Value

---

Body
Cookies
Headers (11)
Test Results

---

Pretty
Raw
Preview
Visualize
JSON ⌵
≡

```

1 {
2   "status": false,
3   "msg": [
4     "El campo nombre es requerido o no es valido",
5     "El campo apellido paterno es requerido o no es valido",
6     "El campo apellido materno es requerido o no es valido",
7     "El campo genero es requerido o no es valido",
8     "El campo tipo de teléfono es requerido o no es valido",
9     "El campo fecha de nacimiento es requerido o no es valido",
10    "El campo número de telefono es requerido o no es valido"
11  ]
12 }
```

## Eliminar contacto

## Petición DELETE para eliminar un contacto existente

The screenshot shows the REST Client interface. At the top, there's a dark bar with 'My Workspace' and an 'Invite' button. Below it, a row of buttons includes 'GET Login usuario', 'POST Registrar Usuario', 'GET Listar contactos', 'POST Nuevo contacto', 'PUT Actualizar contacto', and 'DEL Eliminar contacto'. The 'DEL Eliminar contacto' button is highlighted with a red border. Below this, a dropdown menu shows 'DELETE' and the URL 'http://localhost:8080/api/rest/contacto/1'. Underneath, tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible. The 'Headers (7)' tab is active, showing a table with one header row: 'KEY', 'VALUE', 'DESCRIPTION'. The first row has 'access-token' in the key column, a long alphanumeric string in the value column, and 'Description' in the description column. Below the headers tab, there are tabs for 'Body', 'Cookies', 'Headers (11)', and 'Test Results'. The 'Body' tab is active, showing a JSON response: 

```
{
  "status": true,
  "msg": [
    "Se elimino el contacto con éxito"
  ]
}
```





## Conclusión

Es la segunda vez que desarrollo un CRUD basado en NodeJS y utilizando servicios web, había trabajado más por mi experiencia laboral con tecnologías como Java y PHP en su gran mayoría, es por ello que llegar a desplegar un pequeño sistema con estas tecnologías nuevas para mi me agrado bastante y más llevarlo con un patrón de diseño como el MVC. Trate de llevar la misma arquitectura con los nuevos temas que utilizamos.

Se me hizo muy interesante el ver como se hacen las pruebas de servicio con el programa de postman, además de la integración de login para devolver un token de autenticación.

## Referencias

1. Varios autores By BBVA Api\_Market – API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos - <https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/>
2. Varios autores, by Refsnes Data. - W3Schools is Powered by W3.CSS - <https://www.w3schools.com/nodejs/default.asp>
3. Olatunde Michael Garuba 2017 – Build Node.js RESTful APIs in 10 minutes - <https://www.codementor.io/@olatundegaruba/nodejs-restful-apis-in-10-minutes-qosgsfhbd>
4. ASFO Senior Developoer - Autenticando un API Rest con NodeJS y JWT (JSON Web Tokens) - <https://asfo.medium.com/autenticando-un-api-rest-con-nodejs-y-jwt-json-web-tokens-5f3674aba50e>
5. Adnan Rahic 2017 - Securing Node.js RESTful APIs with JSON Web Tokens - <https://www.freecodecamp.org/news/securing-node-js-restful-apis-with-json-web-tokens-9f811a92bb52/>
6. Varios autores By NPMJS – Password hash - NPM - <https://www.npmjs.com/package/password-hash>
7. Varios autores By Microsoft – Creación de una aplicación web de Node.js en Azure - <https://docs.microsoft.com/es-mx/azure/app-service/quickstart-nodejs?pivots=platform-windows>