

Contenido

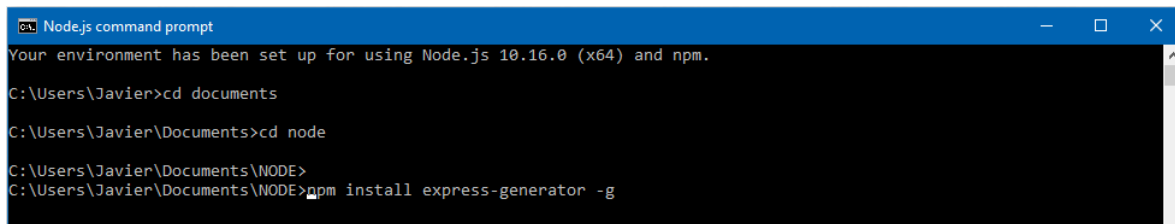
| | |
|--|----|
| 1. INSTALACION DE EXPRESS | 1 |
| 2. INSTALACION DE DEPENDENCIAS (NODEMON - MONGOOSE)..... | 2 |
| 3. CONFIGURACION DE CONEXIÓN CON MONGOOSE | 4 |
| 4. CONFIGURACION NODEMON | 4 |
| 5. DESARROLLO DE APLICACIÓN WEB | 5 |
| 5.1 CREACION DE MODELO (USUARIOS) | 5 |
| 5.2 CREACION DE RUTAS | 6 |
| 5.2.1 RUTA (INDEX) | 6 |
| 5.2.2 RUTA (ADD) | 6 |
| 5.2.3 RUTA (ADD/USER) | 7 |
| 5.2.4 RUTA (EDIT/:ID) | 7 |
| 5.2.5 RUTA (EDIT/:ID/USUARIO)..... | 8 |
| RUTA (DELETE/:ID) | 8 |
| 5.3 CREACION DE VISTAS..... | 9 |
| 5.3.1 VISTA (header.ejs) | 9 |
| 5.3.2 VISTA (footer.ejs)..... | 9 |
| 5.3.3 VISTA (INDEX) | 10 |
| 5.3.4 VISTA (ADD) | 11 |
| 5.3.5 VISTA (EDIT)..... | 13 |
| 5.3.6 VISTA FINAL DE CRUD (LOCAL) | 15 |
| 7. SUBIR CRUD A SERVIDOR (AZURE CON MONGO ATLAS) | 16 |
| 7.1 CREAR SERVICIO EN MICROSOFT AZURE | 16 |
| 7.2 CREAR BD EN MOGO ATLAS | 18 |
| 7.3 SUBIR CRUD A MICROSOFT AZURE | 24 |

MANUAL CRUD

1. INSTALACION DE EXPRESS

Para la elaboración de nuestro CRUD debemos instalar express de la siguiente manera.

Paso 1: Abrimos nuestra terminal de node y nos dirigimos a la carpeta en donde queremos que se instale y guarde nuestro nuevo proyecto. Seguido de ello ejecutamos la siguiente instrucción



```
Node.js command prompt
Your environment has been set up for using Node.js 10.16.0 (x64) and npm.

C:\Users\Javier>cd documents
C:\Users\Javier\Documents>cd node
C:\Users\Javier\Documents\NODE>
C:\Users\Javier\Documents\NODE>npm install express-generator -g
```

Paso 2: Al finalizar la instalación de express procedemos a ejecutar la siguiente instrucción la cual definirá el motor de arranque en nuestras vistas, seguido del nombre de nuestro proyecto

```
C:\Users\Javier\Documents\NODE>express --view=ejs Individual
```

Paso 3: Al finalizar la creación de nuestro proyecto nos mostrara la siguiente pantalla

```
create : Individual\views\error.pug
create : Individual\views\index.pug
create : Individual\views\layout.pug
create : Individual\app.js
create : Individual\package.json
create : Individual\bin\
create : Individual\bin\www

change directory:
> cd Individual

install dependencies:
> npm install

run the app:
> SET DEBUG=individual:* & npm start
```

Paso 4: Nos dirigimos a la carpeta de nuestro proyecto en este caso cd Individual y ejecutamos el comando npm install

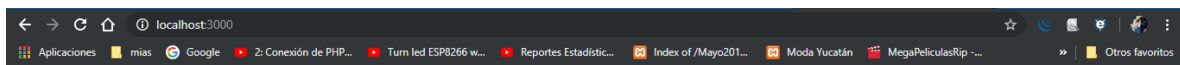
```
C:\Users\Javier\Documents\NODE>cd Individual
C:\Users\Javier\Documents\NODE\Individual>npm install
[.....] / loadIdealTree:loadAllDepsIntoIdealTree: sill install loadIdealTree
```

Este proceso comenzara a instalar los archivos necesarios de nuestro proyecto

Paso 5: Al finalizar Ejecutamos la siguiente instrucción y nuestro servidor express iniciara

```
C:\Users\Javier\Documents\NODE\Individual>SET DEBUG=individual:* & npm start
> individual@0.0.0 start C:\Users\Javier\Documents\NODE\Individual
> node ./bin/www
individual:server Listening on port 3000 +0ms
```

Para verificar que nuestro servicio se está ejecutando de forma correcta nos dirigimos a nuestro navegador y hacemos la siguiente búsqueda localhost:3000



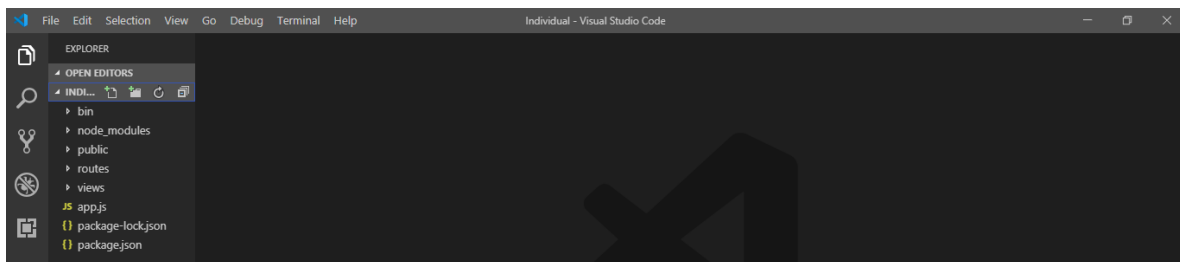
Express

Welcome to Express

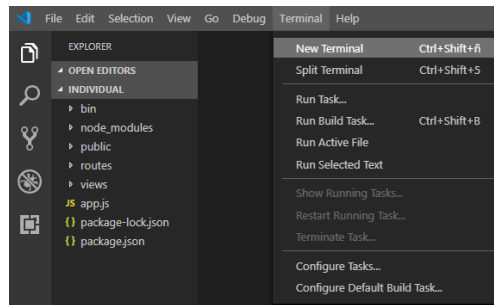
Como pueden observar el servidor se instaló de manera correcta

2. INSTALACION DE DEPENDENCIAS (NODEMON - MONGOOSE)

Paso 6: Para llevar a cabo la instalación de nuestras dependencias, abrimos nuestra carpeta o proyecto creado en visual code.



Paso 7: Una vez abierto nuestro proyecto procedemos a abrir nuestra terminal



Paso 8: Una vez abierta nuestra terminal ejecutamos el siguiente comando para la instalación de mongoose, el cual nos servirá para establecer conexión con nuestra base de datos mongodb

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
PS C:\Users\Javier\Documents\NODE\Individual> npm install --save mongoose
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ mongoose@5.6.7
added 21 packages from 17 contributors and audited 2449 packages in 27.469s
found 0 vulnerabilities
```

Si la instalación se realizó de manera correcta se observará como la imagen previa.

Paso 9: Para instalara nodemon en nuestro proyecto ejecutamos el siguiente comando en nuestra terminal

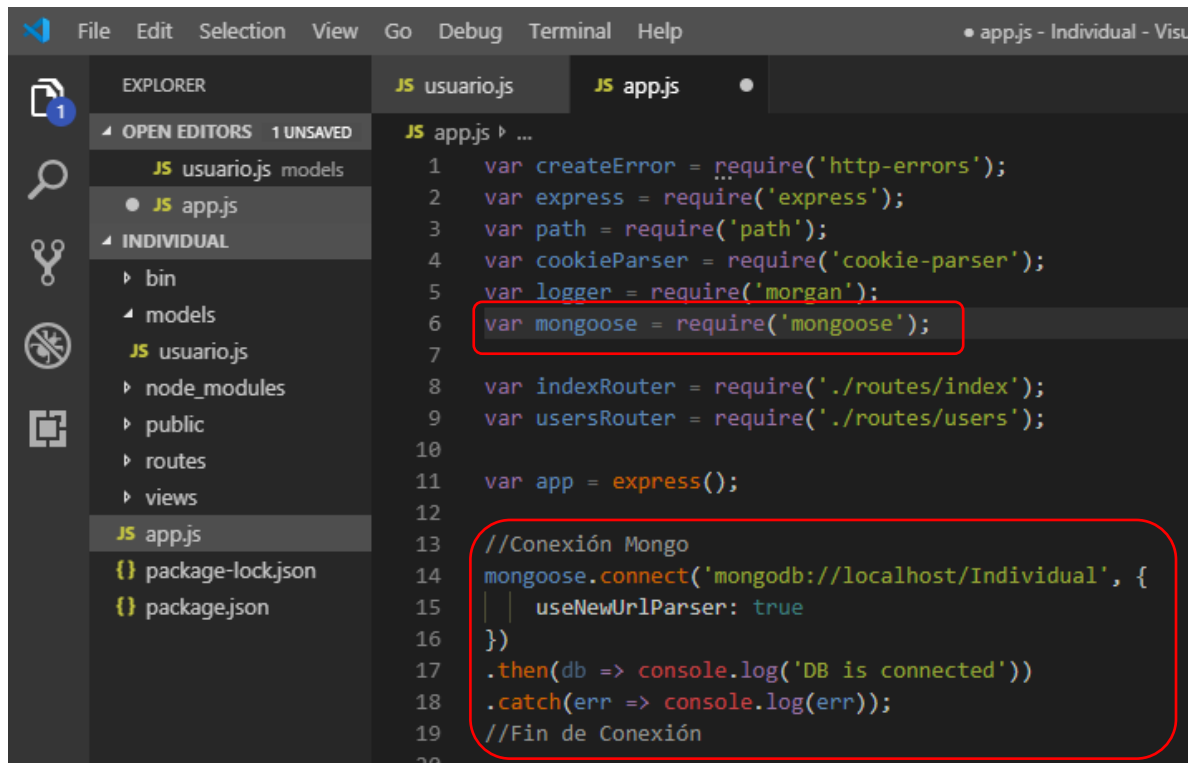
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\Javier\Documents\NODE\Individual> npm install --save nodemon
```

Una vez instaladas nuestras dependencias necesarias para el desarrollo de nuestro proyecto iniciaremos con la codificación.

3. CONFIGURACION DE CONEXIÓN CON MONGOOSE

Paso 10: Para establecer conexión entre nuestro proyecto y mongodb debemos llamar a nuestra dependencia previamente instalada y establecer la configuración de conexión de la siguiente manera en nuestro archivo app.js



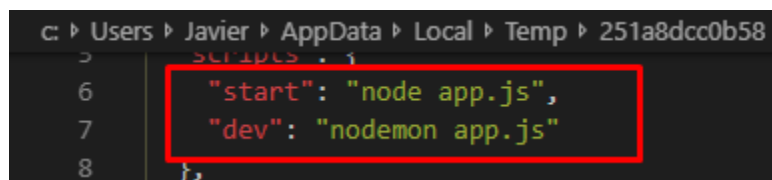
The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows the project structure: bin, models, usuario.js, node_modules, public, routes, views, app.js, package-lock.json, and package.json. The main editor window shows the content of app.js. The code includes requirements for http-errors, express, path, cookie-parser, morgan, and mongoose. The mongoose connection is configured to connect to 'mongodb://localhost/Individual' with useNewUrlParser set to true. The connection is then tested with a log message 'DB is connected' and an error catch. The code is as follows:

```
1 var createError = require('http-errors');
2 var express = require('express');
3 var path = require('path');
4 var cookieParser = require('cookie-parser');
5 var logger = require('morgan');
6 var mongoose = require('mongoose');
7
8 var indexRouter = require('./routes/index');
9 var usersRouter = require('./routes/users');
10
11 var app = express();
12
13 //Conexión Mongo
14 mongoose.connect('mongodb://localhost/Individual', {
15   | useNewUrlParser: true
16 })
17 .then(db => console.log('DB is connected'))
18 .catch(err => console.log(err));
19 //Fin de Conexión
20
```

En la conexión se defina que la base de datos será local y tendrá el nombre de (Individual), ya con ello al realizar cualquier movimiento nuestra base de datos se creara de forma automática.

4. CONFIGURACION NODEMON

Paso 11: Esta dependencia nos ayuda a no detener nuestro servicio para realizar un cambio en nuestro proyecto ya que, aunque este en ejecución se podrá llevar a cabo el desarrollo de nuestro proyecto sin interrupciones. Para configurar la dependencia nos dirigimos a package.json y definimos la siguiente sentencia.



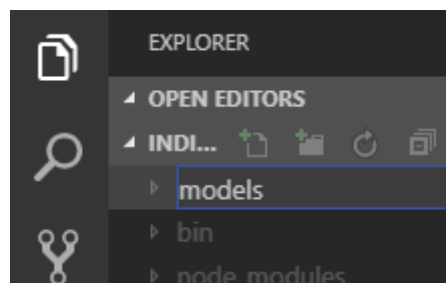
The screenshot shows a snippet of the package.json file. The 'start' script is set to 'node app.js' and the 'dev' script is set to 'nodemon app.js'. The code is as follows:

```
6 "start": "node app.js",
7 "dev": "nodemon app.js"
8
```

5. DESARROLLO DE APLICACIÓN WEB

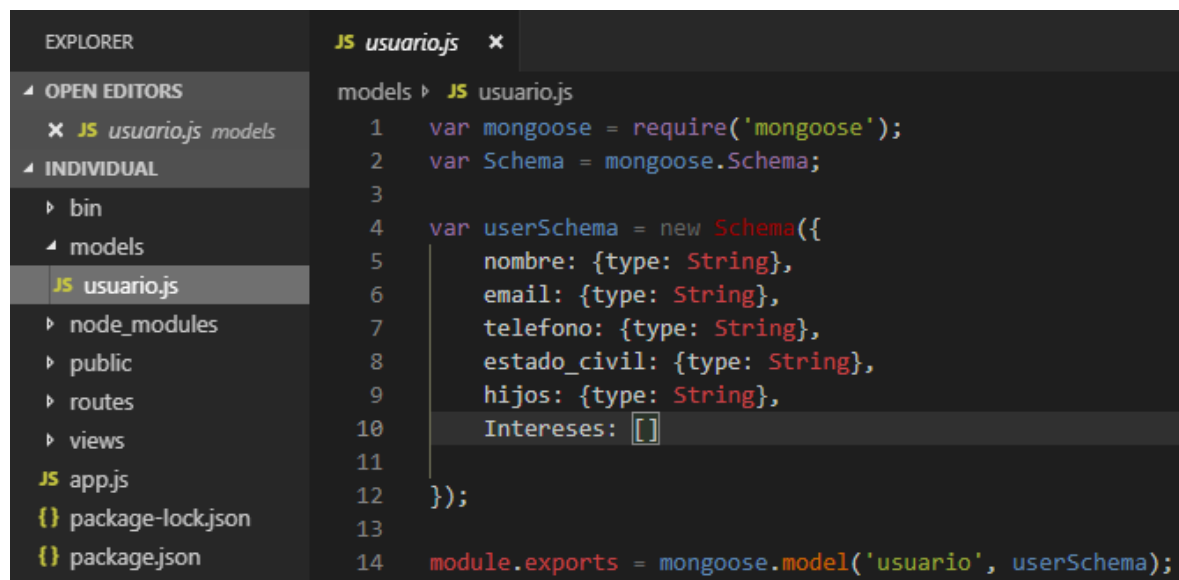
5.1 CREACION DE MODELO (USUARIOS)

Para llevar a cabo la elaboración de nuestro modelo usuario, primero debemos crear una carpeta en donde se guardará nuestro archivo usuario.js



Una vez creada nuestra carpeta creamos nuestro archivo usuario.js en el cual se definirán los campos que utilizaremos al momento del llenado de nuestro registro.

Primero se debe establecer la conexión con mongoose y luego definir los parámetros que ocuparemos de la siguiente manera



La tabla que se generará en nuestra base de datos tendrá el nombre usuario.

5.2 CREACION DE RUTAS

Para definir las rutas de nuestro proyecto debemos dirigirnos a la carpeta routes y editar el archivo index.js

En este se editan las rutas hacia las vistas de nuestro proyecto.

Lo primero es llamar a los archivos necesarios para nuestro proyecto, Usuario hace referencia a la conexión con nuestra base de datos en nuestro modelo usuario.

```
routes ▸ JS index.js ▸ ...
1  var express = require('express');
2  var router = express.Router();
3
4  var Usuario = require('../models/usuario');
```

5.2.1 RUTA (INDEX)

Esta ruta nos mostrara la tabla principal con todos los datos recuperados de mongodb.

Se define la ruta que tendrá en nuestro navegador y se hace una instancia de búsqueda en nuestro modelo usuario el cual devolverá los valores encontrados en la vista index por medio de la variable usuario.

```
6  router.get('/', async (req, res) =>{
7    var usuario = await Usuario.find();
8    res.render('index',{ usuario });
9  });
10
```

5.2.2 RUTA (ADD)

Esta ruta nos dirigirá a nuestro formulario para el llenado de los campos solicitados.

```
12 router.get('/add', function(req, res, next) {
13   res.render('add');
14 });
```

5.2.3 RUTA (ADD/USER)

Esta ruta nos permitirá llevar a cabo el guardado de datos mediante el envío por medio del método post de los campos llenados en nuestro formulario de la vista add. Al finalizar el guardado de datos nos enviara a nuestra página principal.

```
16 router.post('/add/user', async (req, res, next) => {
17   console.log(new Usuario(req.body));
18   var usuario = new Usuario(req.body);
19   await usuario.save();
20
21   res.redirect('/');
22 });
```

5.2.4 RUTA (EDIT/:ID)

Esta ruta nos permite recuperar los datos de un elemento seleccionado en la tabla principal por medio de su id. Después de recuperar los datos los envía de nuevo al formulario en la vista edit para su modificación.

Se llevó a cabo la realización de un ciclo for para la recuperación de datos en el array[] intereses para su edición en el formulario.

```
26 router.get('/edit/:id', async (req, res, next) => {
27   const {id} = req.params;
28   const usuario = await Usuario.findById(id);
29   var L = "";
30   var M = "";
31   var D = "";
32   var O = "";
33   for(i=0; i<usuario.Intereses.length; i++){
34     if(usuario.Intereses[i] === 'Libros'){
35       L = 'Libros';
36     }if(usuario.Intereses[i] === 'Musica'){
37       M = 'Musica';
38     }if(usuario.Intereses[i] === 'Deportes'){
39       D = 'Deportes';
40     }if(usuario.Intereses[i] === 'Otros'){
41       O = 'Otros';
42     }
43   }
44   console.log(L, M, D, O);
45   res.render('edit', {usuario, L, M, D, O});
46 });
```


5.2.5 RUTA (EDIT/:ID/USUARIO)

Esta ruta recibirá los datos modificados del formulario de la vista edit por medio del método post y llevará a cabo una actualización de ellos en la base de datos por medio de los datos recibidos de los campos editados.

```
49 router.post('/edit/:id/usuario', async (req, res, next) => {  
50   const { id } = req.params;  
51   await Usuario.update({_id: id}, req.body);  
52   res.redirect('/');  
53 });  
54
```

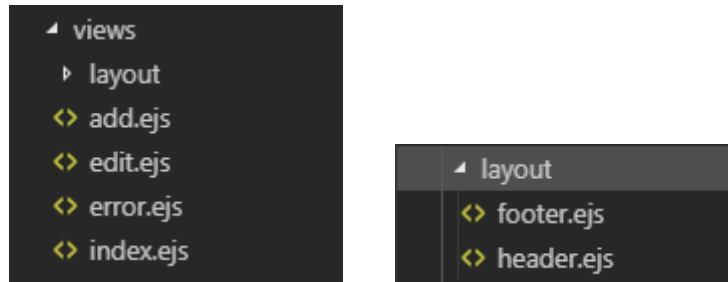
RUTA (DELETE/:ID)

Esta ruta recibe el id de la tabla principal y elimina el conjunto de datos correspondiente a esa búsqueda. Al finalizar redirección a la página principal.

```
55 /* GET delete page. */  
56 router.get('/delete/:id', async (req, res, next) => {  
57   let { id } = req.params;  
58   await Usuario.remove({_id: id});  
59   res.redirect('/');  
60 });
```

5.3 CREACION DE VISTAS

En la carpeta views se llevará a cabo la elaboración de nuestras vistas a las cuales serán la interfaz gráfica que vera el usuario. Se crearán las siguientes vistas con la terminación ejs



5.3.1 VISTA (header.ejs)

Esta vista tendrá el encabezado, además de que se hace referencia al llamado de bootstrap para el diseño de nuestro CRUD.

```
views > layout > header.ejs > html > body
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
8     integrity="sha384-ggOyR0iXCbMQV3Xipma34MD+dH/1fQ784/j6cY/1JTQUO0hcWr7x9JvoRxt2M2w1T" crossorigin="anonymous">
9   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
10     integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaEfF/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
11   <title>CRUD</title>
12 </head>
13 <body>
```

5.3.2 VISTA (footer.ejs)

Esta vista tendrá el pie de página de nuestro CRUD

```
views > layout > footer.ejs
1 </body>
2 </html>
```

Estas vistas se hicieron con la finalidad de no tener que estar poniendo el mismo condigno en las vistas principales, si se tuviera que hacer un cambio de encabezado únicamente se modificarían estos archivos y no todas las vistas.

5.3.3 VISTA (INDEX)

En esta vista se mostrará la tabla de los datos guardados el código HTML se muestra a continuación.

Para llamar a la vista header en esta vista index hay que llamarla por medio del include como se muestra a continuación.

```
index.ejs
views > index.ejs > div.container > div.row > div.col-sm-12 > div.card > div.card-body > table.table.table-hover > tbody > tr > td > a
1  <% include ./layout/header %>
2  <div class="container" style="margin-top: 40px">
3      <div class="row">
4          <div class="col-sm-12">
5              <div class="card">
6                  <h5 class="card-header">
7                      <a href="/add">
8                          <button type="button" class="btn btn-outline-success">Agregar Nuevo Registro</button>
9                      </a>
10                 </h5>
11                 <div class="card-body">
12                     <p class="card-text">En esta tabla puede observar la informacion de los registros guardados.</p>
13                     <table class="table table-hover">
14                         <thead>
15                             <tr>
16                                 <th scope="col">Nombre</th>
17                                 <th scope="col">Email</th>
18                                 <th scope="col">Telefono</th>
19                                 <th scope="col">Estado Civil</th>
20                                 <th scope="col">¿Hijos?</th>
21                                 <th scope="col">Pasatiempos</th>
22                                 <th scope="col">Opciones</th>
23                             </tr>
24                         </thead>
25                     </table>
26                 </div>
27             </div>
28         </div>
29     </div>
30 </div>
```

Llama a la vista header

Para recibir los datos enviados desde la consulta de la ruta anteriormente establecida se utiliza el siguiente código.

Se establece un foreach para la búsqueda de la variable usuario y se asigna de la siguiente manera para recopilar los datos buscados en nuestra bd.

```
27  <% usuario.forEach(function(usuarios) { %>
28      <tr>
29          <th><%= usuarios.nombre %></th>
30          <td><%= usuarios.email %></td>
31          <td><%= usuarios.telefono %></td>
32          <td><%= usuarios.estado_civil %></td>
33          <td><%= usuarios.hijos %></td>
34          <td><%= usuarios.Intereses %></td>
35          <td colspan="2">
36              <a href="/edit/<%= usuarios._id %>">
37                  <button type="button" class="btn btn-outline-primary">Editar</button>
38              </a>
39              <a href="/delete/<%= usuarios.id %>">
40                  <button type="button" class="btn btn-outline-danger">Eliminar</button>
41              </a>
42          </td>
43      </tr>
44  <% }) %>
45  </tbody>
46  </table>
47  </div>
48  </div>
49  </div>
50  </div>
51  </div>
52  <% include ./layout/footer %>
```

Recupera datos

Envia el id para la edicion en la vista edit

Envia el id para la eliminacion del registro en la bd

5.3.4 VISTA (ADD)

En esta vista se elaborará el formulario correspondiente para el llenado de datos.

add.js x

views > add.js > ? > div.container > div.row > div.col-sm-10 > div.card > div.card-body > form

```

1  <? include ../layout/header ?>
2  <div class="container" style="margin-top: 40px">
3    <div class="row">
4      <div class="col-sm-1"></div>
5      <div class="col-sm-10">
6        <div class="card">
7          <div class="card-body">
8            <h3 class="card-text" style="text-align: center">NUEVO REGISTRO</h3>
9            <form action="/add/producto" method="post">
10              <div class="form-group row">
11                <label class="col-sm-2 col-form-label h3">Nombre</label>
12                <div class="col-sm-10">
13                  <input type="text" class="form-control" name="nombre" placeholder="Nombre" required>
14                </div>
15              </div>
16              <div class="form-group row">
17                <label class="col-sm-2 col-form-label h3">Email</label>
18                <div class="col-sm-10">
19                  <input type="email" class="form-control" name="email" placeholder="Email" required>
20                </div>
21              </div>
22              <div class="form-group row">
23                <label class="col-sm-2 col-form-label h3">Telefono</label>
24                <div class="col-sm-10">
25                  <input type="text" class="form-control" name="telefono" placeholder="Telefono" required>
26                </div>
27              </div>

```

Enviara los datos a la ruta especificada para su ejecucion

Los datos se envian por el metodo post

Se asignan el nombre de las variables declaradas en la bd en los campos correspondientes

[illegible]

5.3.5 VISTA (EDIT)

En esta vista se recuperan los datos enviados desde la vista index por medio del botón editar.

El id seleccionado lo recibe la ruta edit/:id y realiza una búsqueda, esta se almacena en una variable la cual se recibe en esta vista.

```
1 <% include ../layout/header %>
2 <div class="container" style="margin-top: 40px">
3   <div class="row">
4     <div class="col-sm-1"></div>
5     <div class="col-sm-10">
6       <div class="card">
7         <div class="card-body">
8           <h3 class="card-text" style="text-align: center">EDITAR REGISTRO</h3>
9           <form action="/edit/<%= usuario.id %>/usuario" method="post">
10             <div class="form-group row">
11               <label for="inputEmail3" class="col-sm-2 col-form-label h3">Nombre</label>
12               <div class="col-sm-10">
13                 <input type="text" class="form-control" name="nombre" value="<%= usuario.nombre %>" placeholder="Nombre" required>
14               </div>
15             </div>
16             <div class="form-group row">
17               <label for="inputEmail3" class="col-sm-2 col-form-label h3">Email</label>
18               <div class="col-sm-10">
19                 <input type="email" class="form-control" name="email" value="<%= usuario.email %>" placeholder="Email" required>
20               </div>
21             </div>
22             <div class="form-group row">
23               <label for="inputPassword3" class="col-sm-2 col-form-label h3">Telefono</label>
24               <div class="col-sm-10">
25                 <input type="text" class="form-control" name="telefono" value="<%= usuario.telefono %>" placeholder="Telefono" required>
26               </div>
27             </div>
28             <div class="form-group row">
29               <label class="col-sm-2 col-form-label h3">Estado Civil</label>
30               <div class="col-sm-10">
31                 <select class="form-control" name="estado_civil">
32                   <% if(usuario.estado_civil=='SOLTERO') { %>
33                     <option value="SOLTERO">SOLTERO</option>
34                     <option value="CASADO">CASADO</option>
35                     <option value="DIVORCIADO">DIVORCIADO</option>
36                   <% } else if(usuario.estado_civil=='CASADO') { %>
37                     <option value="CASADO">CASADO</option>
38                     <option value="SOLTERO">SOLTERO</option>
39                     <option value="DIVORCIADO">DIVORCIADO</option>
40                   <% } else if(usuario.estado_civil=='DIVORCIADO') { %>
41                     <option value="DIVORCIADO">DIVORCIADO</option>
42                     <option value="CASADO">CASADO</option>
43                     <option value="SOLTERO">SOLTERO</option>
44                   <% } %>
45                 </select>
46               </div>
47             </div>
48           </form>
49         </div>
50         <div class="form-group">
51           <div class="row">
52             <div class="col-sm-10">
53               <div class="form-check">
54                 <input class="form-check-input" type="radio" name="hijos" value="SI" <% if(usuario.hijos=='SI') { %> checked="true" <% } %>>
55                 <label class="form-check-label" for="gridRadios1">SI</label>
56               </div>
57               <div class="form-check">
58                 <input class="form-check-input" type="radio" name="hijos" value="NO" <% if(usuario.hijos=='NO') { %> checked="true" <% } %>>
59                 <label class="form-check-label" for="gridRadios1">NO</label>
60               </div>
61             </div>
62           </div>
63         </div>
64       </div>
65     </div>
66   </div>
67 </div>
```

Al finalizar la edicion se envian los datos a la ruta especificada

se envian por medio del metodo post

Se asignan los nombres de la bd para su edicion

Recupera los datos y los muestra en el formulario

Realiza una validacion dependiendo del esta civil recuperado de la bd, Este sera elemento que se mostrara en el formulario

Realiza una validacion para marcar el checkinput correspondiente dependiendo del dato recuperado de la bd.

[illegible]

5.3.6 VISTA FINAL DE CRUD (LOCAL)

[Agregar Nuevo Registro](#)

En esta tabla puede observar la informacion de los registros guardados.

| Nombre | Email | Telefono | Estado Civil | ¿Hijos? | Pasatiempos | Opciones |
|--------|------------------|----------|--------------|---------|-----------------|---|
| Javier | fcof07@gmail.com | 1881266 | SOLTERO | NO | Musica,Deportes | Editar Eliminar |

NUEVO REGISTRO

Nombre

Email

Telefono

Estado Civil

SOLTERO

¿Tiene Hijos?

☒ SI ☐ NO

INTERESES

☐ Libros ☐ Musica ☐ Deportes ☐ Otros

Regresar

Guardar

7. SUBIR CRUD A SERVIDOR (AZURE CON MONGO ATLAS)

7.1 CREAR SERVICIO EN MICROSOFT AZURE

Para llevar a cabo esto es necesario crear una cuenta en azure, después de ello iniciamos sesión e ingresamos a app services para crear nuestro servicio web.

Para llevar a cabo la configuración se siguen los siguientes pasos.

Microsoft Azure

Home > App Services

App Services

Seleccionamos para agregar un servicio

+ Add Edit columns Refresh Assign tags Start Restart Stop Delete

Subscription: Universidad Tecnológica Metropolitana

Filter by name... (Disabled) Evaluación gra... All resource groups All locations All tags No grouping

0 items

NAME STATUS APP TYPE APP SERVICE PLAN LOCATION SUBSCRIPTION

No results.

Subscription: Azure para estudiantes

Resource Group: Select existing Create new

Instance Details

Name: practicaordinario

Publish: Code Docker Image

Runtime stack: Node LTS

Operating System: Linux Windows

Region: Central US

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. Learn more

Linux Plan (Central US) ASP-Zapateria-9c15 (F1) Create new

Definimos el nombre que queremos

Seleccionamos nuestra cuenta

Creamos un nuevo grupo



Nombre de nuestro proyecto

Motor de desarrollo

Region preferible

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app.
[Learn more](#)

* Linux Plan (Central US)  ASP-Zapateria-9c15 (F1) 
[Create new](#)

* Sku and size Free F1


[Review and create](#) [Next: Tags >](#)

Finalizado seleccionamos esta opcion para crear nuestro servicio

Después de esto nos mostrara una ventana de confirmación

Web App
Create

Summary

 **Web App**
by Microsoft

Details

| | |
|----------------|--------------------------------------|
| Subscription | c1ef46fc-05b9-4817-81b7-372207e657b7 |
| Resource Group | ordinario |
| Name | practicaordinario |
| Publish | Code |
| Runtime stack | Node LTS |

App Service Plan

| | |
|------------------|-----------------------|
| Name | ASP-Zapateria-9c15 |
| Operating System | Linux |
| Region | Central US |
| SKU | Free |
| Size | |
| ACU | Shared infrastructure |

[Create](#) [Previous](#)


Seleccionamos para crear el servicio
[Download a template for automation](#)

Finalmente, nuestro servicio se comenzará a ejecutar

Delete Cancel Redeploy Refresh

■ ■ ■

Your deployment is underway



Deployment name: Microsoft.Web-WebApp-Portal-4aa31ddd-ae9b
Subscription: [Azure para estudiantes](#)
Resource group: ordinario

Start time: 8/1/2019, 9:19:13 PM
Correlation ID: 021e24c4-7c34-4086-8061-c14607c0c57


^

Deployment details [\(Download\)](#)

Delete Cancel Redeploy Refresh

✓

Your deployment is complete



Deployment name: Microsoft.Web-WebApp-Portal-4aa31ddd-ae9b
Subscription: [Azure para estudiantes](#)
Resource group: ordinario

Start time: 8/1/2019, 9:19:13 PM
Correlation ID: 021e24c4-7c34-4086-8061-c14607c0c57

∨

Deployment details [\(Download\)](#)

^


Next steps

7.2 CREAR BD EN MOGO ATLAS

Para poder llevar a cabo este proceso es necesario crear una cuenta en mongo atlas una vez realizado esto podremos crear nuestra base de datos en la nube.

← → ↻ 🏠 🔒 https://cloud.mongodb.com/user?signedOut=true#/atlas/login

Aplicaciones mias Google 2: Conexión de PHP... Turn led ESP8266 w... Reportes Estadistic... Index of /Mayo201... Moda Yucatán MegaPelículasRip -... Otros favoritos



We've launched a unified login experience that gives you access to MongoDB Cloud, Support, and Jira with a single identity.
[LEARN MORE](#)

👤 Login

Usually, this is the email you used to register.

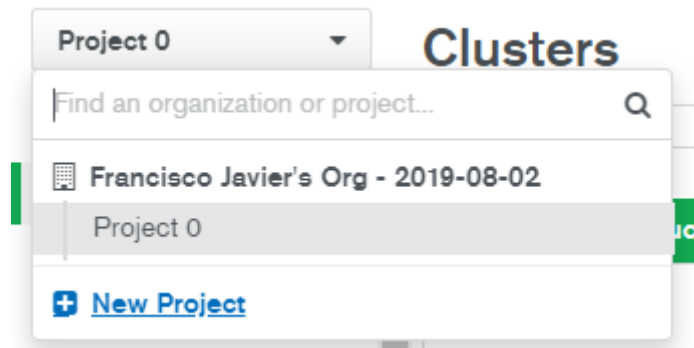
Get up and running
in minutes
with a free managed
MongoDB replica set

[Log in to launch your cluster](#)

Iniciamos sesión para continuar con el proceso.

18

A continuación, crearemos un nuevo cluster o base de datos en nuestra nube.



Name Your Project

Project names have to be unique within the organization (and other restrictions).

ordinario|

Asignamos el nombre de nuestro proyecto

Cancel

Next

Add Members and Set Permissions

Invite new or existing users via email address...

Give your members access permissions below.

iscodominguez07@gmail.com
(you)

Project Owner

Si queremos compartir nuestros permisos agregamos los correos correspondientes
de lo contrario seleccionamos crear proyecto

Cancel

← Go Back

Create Project

Una vez hecho lo anterior nos mostrara la siguiente pantalla



Create a cluster




Choose your cloud provider, region, and specs.

Build a Cluster

Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#).













Seleccionamos construir cluster y nos mostrara los servidores disponibles para guardar nuestra bd, es importante mencionar que hay servidores de paga y de prueba, en este caso seleccionamos un servicio de prueba por lo cual está limitado en capacidad y velocidad y tomara un tiempo en desplegarse para su utilización. Seleccionamos el cluster y lo creamos.

Cloud Provider & Region AWS, N. Virginia (us-east-1) ▾



Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the **M0** cluster tier below.

★ Recommended region ⓘ

| NORTH AMERICA | EUROPE | ASIA |
|--|---|--|
|  N. Virginia (us-east-1) ★ FREE TIER AVAILABLE |  Stockholm (eu-north-1) ★ |  Hong Kong (ap-east-1) ★ |
|  Ohio (us-east-2) ★ |  Ireland (eu-west-1) ★ FREE TIER AVAILABLE |  Tokyo (ap-northeast-1) ★ |
|  N. California (us-west-1) |  London (eu-west-2) ★ |  Seoul (ap-northeast-2) |
|  Oregon (us-west-2) ★ |  Paris (eu-west-3) ★ |  Singapore (ap-southeast-1) ★ |

FREE Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Cancel **Create Cluster**

A continuación, te pedirá algunos parámetros de configuración los cuales se muestran en las siguientes imágenes.

FRANCISCO JAVIER'S ORG - 2019-08-02 > ORDINARIO

Database Access

MongoDB Users

MongoDB Roles

+ ADD NEW USER

User Name ↕

Authentication Method ↕

MongoDB Roles

Actions



Create a database user

➔ Agregamos un nuevo usuario

Add New User

SCRAM Authentication

SCRAM is MongoDB's default authentication method.

Nombre del usuario

isco

e.g. new-user_31

Contraseña del usuario

....

SHOW

Autogenerate Secure Password

User Privileges

Atlas admin

Read and write to
any database

Only read any
database

Select Custom
Role

Add Default Privileges

☐ Save as temporary user

Cancel

Add User

Agregamos al usuario

Después de ello nos dirigimos a la sección Network Access la cual es necesario configurar para poder tener acceso desde cualquier máquina que trate de acceder desde la red.

FRANCISCO JAVIER'S ORG - 2019-08-02 > ORDINARIO

Network Access

IP Whitelist

Peering

Agregamos la ip

+ ADD IP ADDRESS

| IP Address | Comment | Status | Actions |
|------------|---------|--------|---------|
|------------|---------|--------|---------|



Whitelist an IP address

Add Whitelist Entry

Add a whitelist entry using either CIDR notation or a single IP address. [Learn more.](#)

ADD CURRENT IP ADDRESS

ALLOW ACCESS FROM ANYWHERE

Seleccionamos esta opcion
ya que permite el acceso desde cualquier
lugar

Whitelist Entry:

0.0.0.0/0

Comment:

Optional comment describing this entry

IP por defecto

☐ Save as temporary whitelist

Confirmamos la
configuracion

Cancel

Confirm

Una vez realizado lo anterior nos mostrara el estatus del servicio activo.

| IP Address | Comment | Status | Actions |
|--|---------|----------|---|
| 0.0.0.0/0 (includes your current IP address) | | ● Active | EDIT DELETE |

Esto nos mostrara nuestro cluster, ahora hay que configurar la conexión entre nuestra base de datos y nuestro proyecto

Clusters [Build a New Cluster](#)

Find a cluster...

SANDBOX

Cluster0
Version 4.0.11

CONNECT METRICS COLLECTIONS ...

CLUSTER TIER
M0 Sandbox (General)

REGION
AWS / N. Virginia (us-east-1)

TYPE
Replica Set - 3 nodes

LINKED STITCH APP
None Linked

Operations R: 0 W: 0 100.0%
Last 6 Hours

Logical Size 0.0 B 512.0 MB max
Last 6 Hours

Connections 0 100 max
Last 6 Hours

Enhance Your Experience
For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!
[Upgrade](#)

Connect to Cluster0

✓ Setup connection security > Choose a connection method > Connect

Choose a connection method [View documentation](#)

See methods to add data and diagnostics in the [Command Line Tools](#) shortcut from within your cluster.

Connect with the Mongo Shell
Mongo Shell with TLS/SSL support is required >

Connect Your Application
Get a connection string and view driver connection examples >

Connect with MongoDB Compass
Download Compass to explore, visualize, and manipulate your data >

[Go Back](#) [Close](#)

Connect to Cluster0

✓ Setup connection security > ✓ Choose a connection method > Connect

1 Choose your driver version

| DRIVER | VERSION |
|---------|-----------------|
| Node.js | 2.2.12 or later |

Seleccionamos nuestra version de node

2 Add your connection string into your application code

Copiamos la cadena de conexion

Connection String Only Full Driver Example

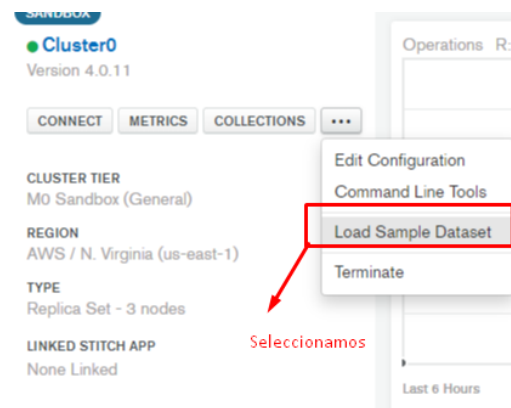
```
mongodb://isco:<password>@cluster0-shard-00-00-uary2.mongodb.net:2701
```

Copy

Replace **<password>** with the password for the **isco** user. Nos dice que remplacemos esto con nuestro password

When entering your password, make sure that any special characters are [URL encoded](#).

Una vez realizado lo anterior regresamos a nuestro cluster y seleccionamos lo siguiente



Esto nos permitirá cargar nuestra base de datos

Load Sample Dataset

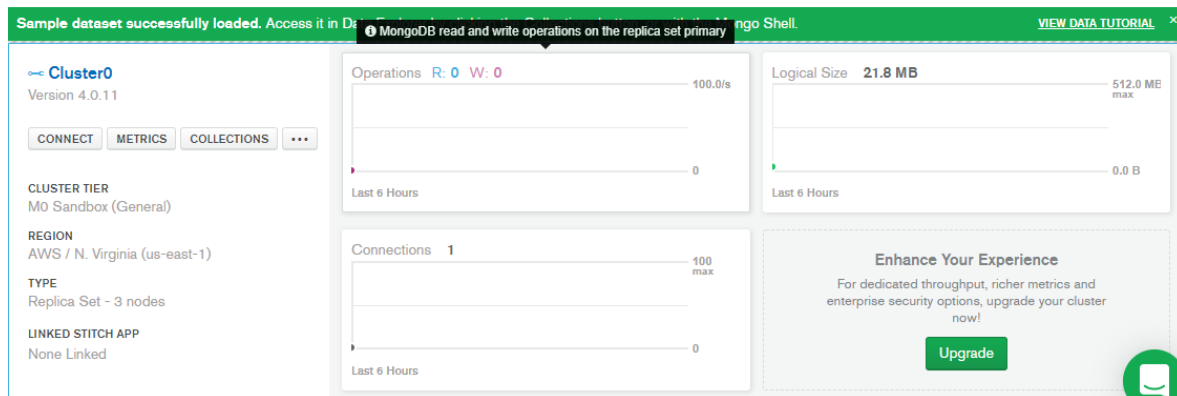
We've created a sample dataset to help you test features on Cluster0.

[Sample Dataset](#)
Size: ~350 MB

Please confirm that you want to load this sample dataset.

Cancel Load Sample Dataset

Confirmamos y nos debe mostrar la siguiente pantalla.



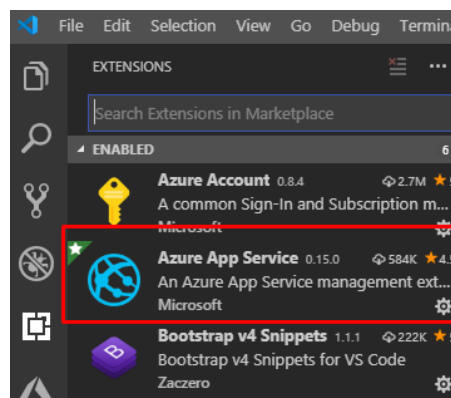
Finalmente.

```
//Conexión Mongo
mongoose.connect('mongodb://isco:[password]@cluster0-shard-00-00-uary2.mongodb.net:27017,cluster0-shard-00-01-uary2.mongodb.net?ssl=true&replicaSet=rs0&serverSelectionTimeoutMS=10000&socketTimeoutMS=10000&familyName=IPv6', {
  useNewUrlParser: true
})
.then(db => console.log('DB is connected'))
.catch(err => console.log(err));
//Fin de Conexión
```

Cambiamos la cadena de conexion en nuestro proyecto y de igual manera asignamos el passwor establecido al momento de crear nuestro cluster

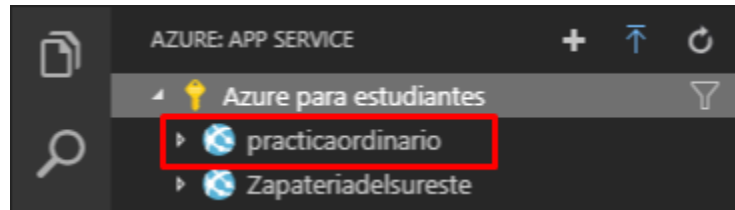
7.3 SUBIR CRUD A MICROSOFT AZURE

Para llevar a cabo esto, debemos descargar la siguiente app en nuestro visual code

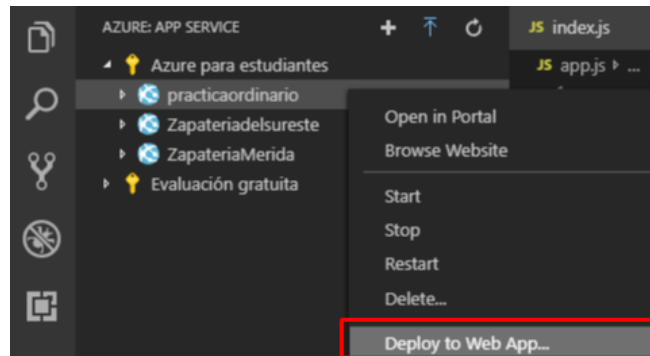


Una vez instalado nos solicitara iniciar sesión de nuestra cuenta de azure, al ingresar nuestros datos nos mostrara los servicios disponibles en nuestra cuenta.

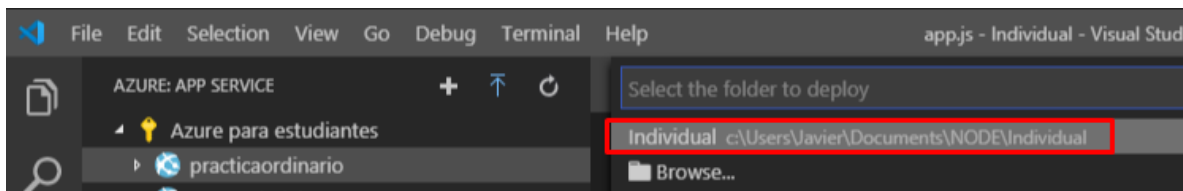
En este caso podremos observar que nuestro servicio previamente creado se encuentra entre la cuenta correspondiente a nuestro azure.



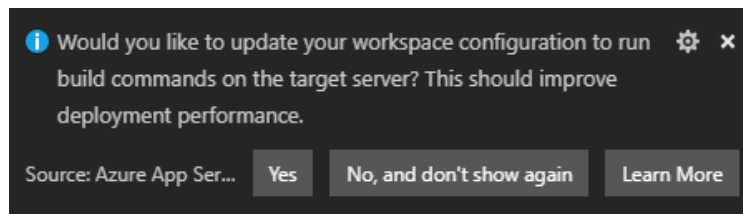
Ya en nuestro proyecto CRUD seleccionamos la siguiente opción



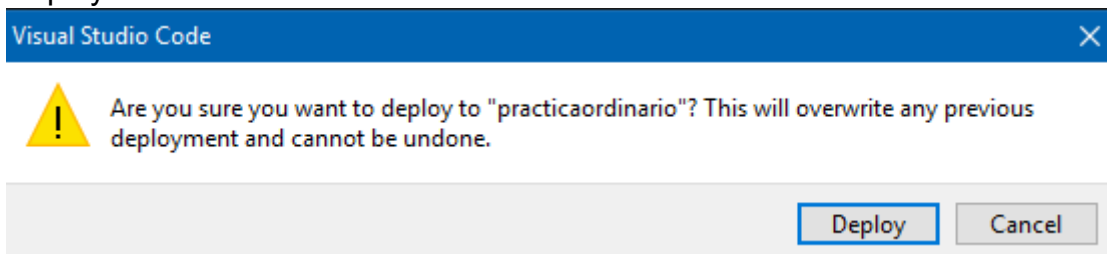
Nos preguntara que proyecto queremos subir a nuestro servidor, lo seleccionamos



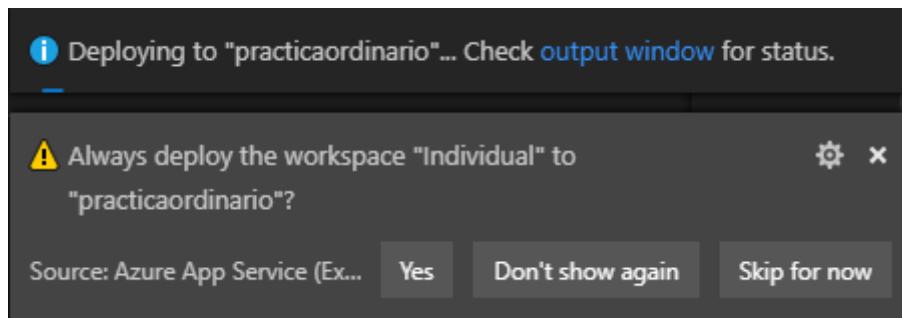
Aceptamos la siguiente cuestión.



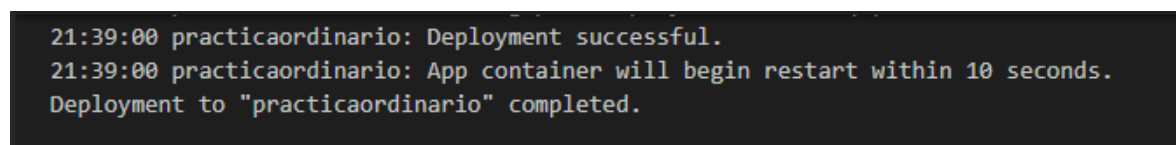
Nos mostrara una advertencia de si queremos cargar los datos al servidor ocasionara que se borren datos previamente guardados, pero como es primera vez que subimos nuestro proyecto no habrá ningún problema. Seleccionamos Deploy



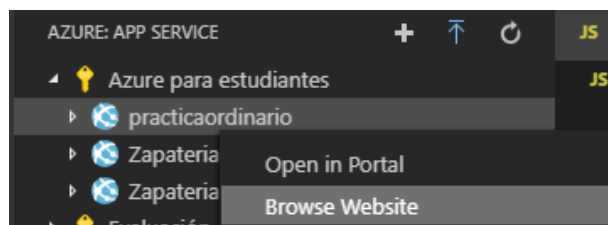
Nos preguntara si nuestro proyecto trabajara en nuestro servidor de manera constante al guardar los cambios, seleccionamos que sí, mientras nuestro proyecto se carga.



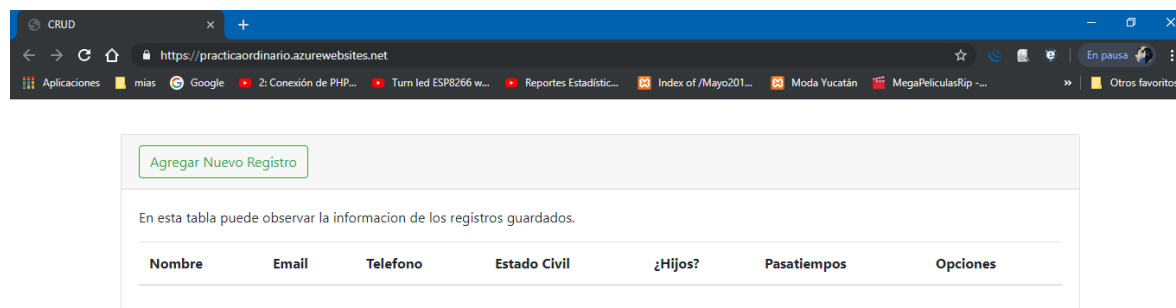
Al finalizar nos mostrara que nuestro proyecto se ha cargado con éxito.



Para poder confirmarlo seleccionamos la siguiente opción



Ahora podemos ver que nuestro servicio se está ejecutando de manera correcta en nuestro servidor de azure.



NUEVO REGISTRO

| | |
|--|--|
| Nombre | <input type="text" value="Nombre"/> |
| Email | <input type="text" value="Email"/> |
| Telefono | <input type="text" value="Telefono"/> |
| Estado Civil | <input type="text" value="SOLTERO"/> |
| ¿Tiene Hijos? | <input checked="" type="radio"/> SI <input type="radio"/> NO |
| INTERESES | <input type="checkbox"/> Libros <input type="checkbox"/> Musica <input type="checkbox"/> Deportes <input type="checkbox"/> Otros |
| <input type="button" value="Regresar"/> <input type="button" value="Guardar"/> | |

[Agregar Nuevo Registro](#)

En esta tabla puede observar la informacion de los registros guardados.

| Nombre | Email | Telefono | Estado Civil | ¿Hijos? | Pasatiempos | Opciones |
|--------|------------------|----------|--------------|---------|-----------------|---|
| Javier | fcof07@gmail.com | 1881266 | SOLTERO | NO | Musica,Deportes | <input type="button" value="Editar"/> <input type="button" value="Eliminar"/> |