2020

# Sistema de Registro de Usuarios

MANUAL TECNICO

ESTEBAN MISAEL GONZALEZ FLORIDO

# Introducción

Con el fin de proporcionar mayor ayuda en el uso del software se realiza este manual destinado a quienes tengan que modificar, en un futuro, y mejorar el funcionamiento del mismo.

Se pondrá de la forma más clara y concisa las características con las cuales se desarrolló este proyecto de software.

# Herramientas necesarias

Para la creación de este software se utilizó:

Visual Studio Code v. 1.43.

PostgreSQL v. 11

Git y Github v. 2.26

Ademas de los frameworks:

NodeJS v. 12.16

NPM  v. 6.13

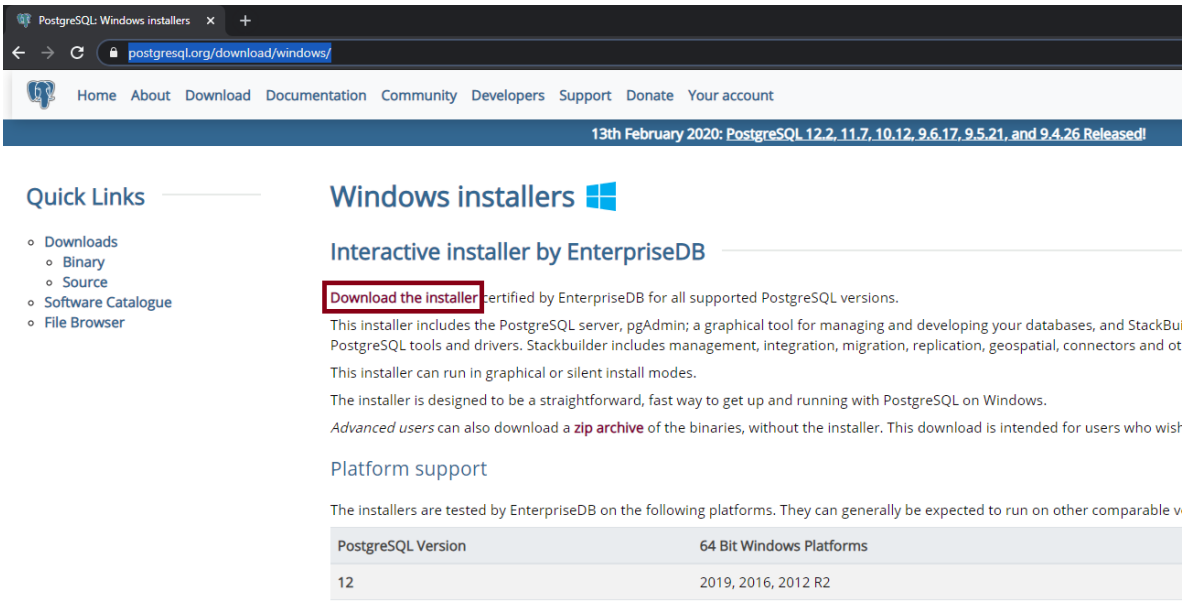
PugJS v. 2.6

Todo esto con el lenguaje de desarrollo de SQL (para la creación de base de Datos) y JavaScript (para la creación del servidor local de nodejs)

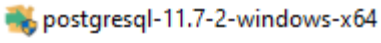
# Instalación de las herramientas

Postgresql.

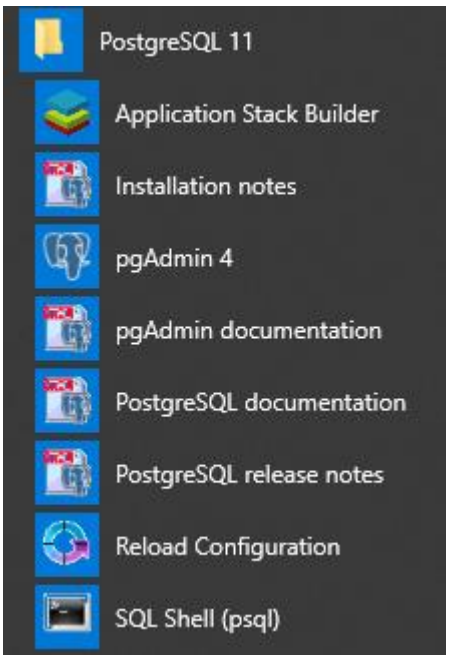
Ir a: https://www.postgresql.org/download/windows/ y dar clic en el recuadro rojo:



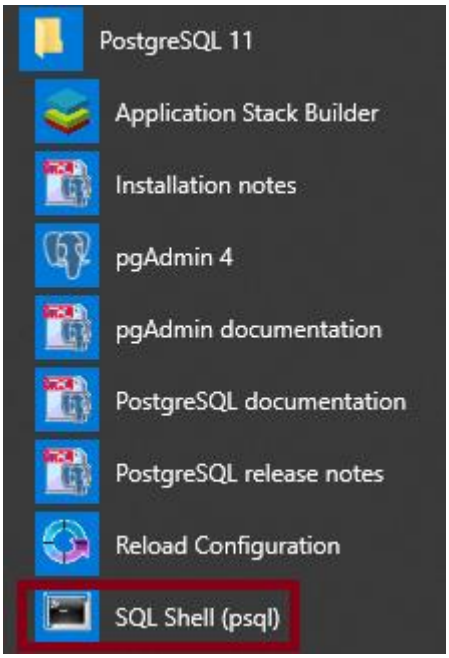

Se descargará el instalador:



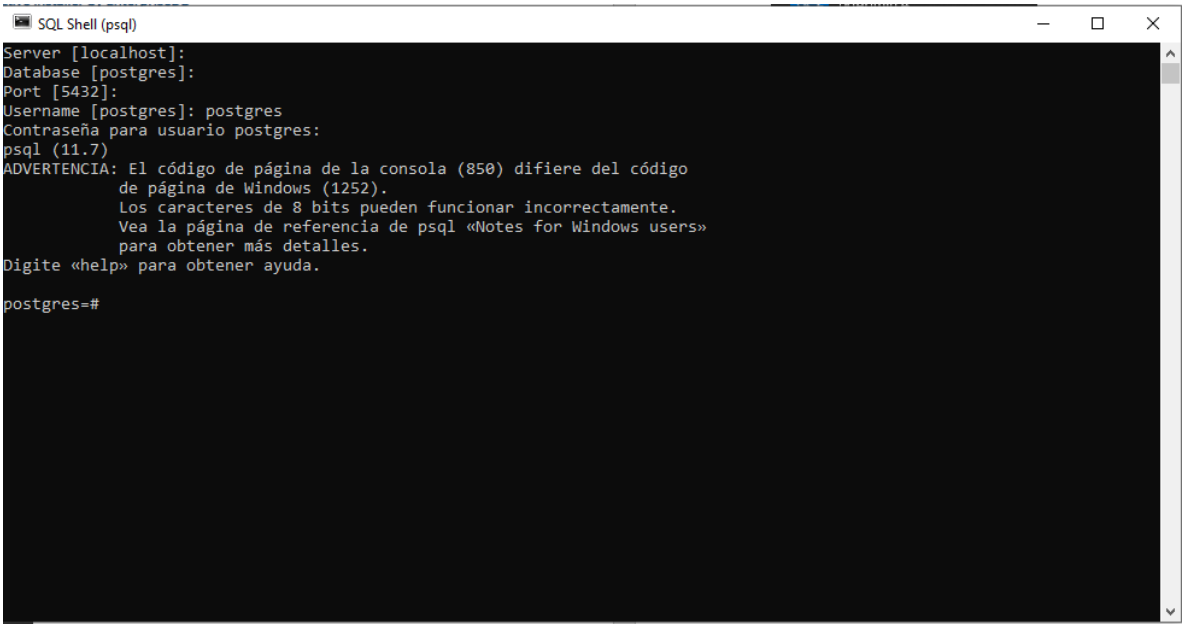

Ejecutamos y simplemente damos siguiente y finalizamos la instalación:




Ahora creamos la base datos necesaria, damos clic en SQL Shell:




Y nos abre la ventana de comandos:




Si observamos únicamente pondremos nuestro usuario, por lo general es postgres y nuestra contraseña, depende si en la instalación pusimos o no contraseña, haciendo esto ya nos deja entrar.

Creamos la base de datos con el siguiente comando:

CREATE DATABASE nombredelabase;

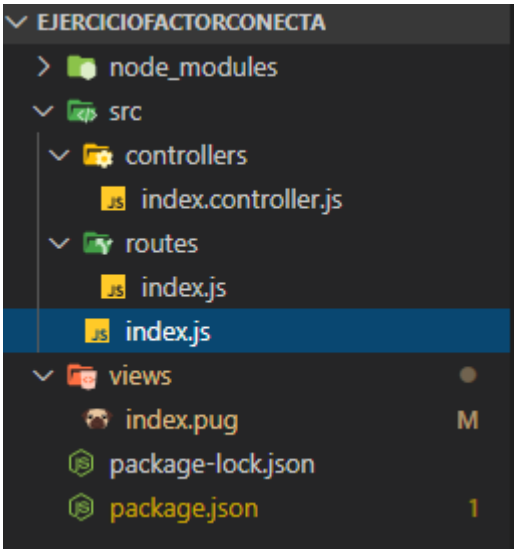
Una vez creada accedemos a la base datos para crearla con el comando:

\c nombrebasedatos

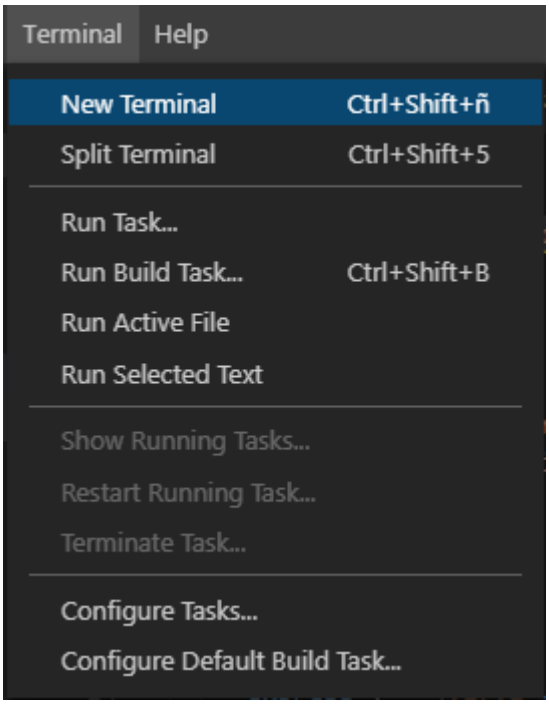
Suponiendo que tenemos instalado visual studio code, git, nodejs, npm y además tenemos una cuenta en github pasamos a desarrollar la aplicación.

Creamos una carpeta la cual contendrá nuestro desarrollo y la abrimos en visual studio code iniciamos con poniendo el siguiente comando en una terminal dentro de visual estudio code, paso a paso:

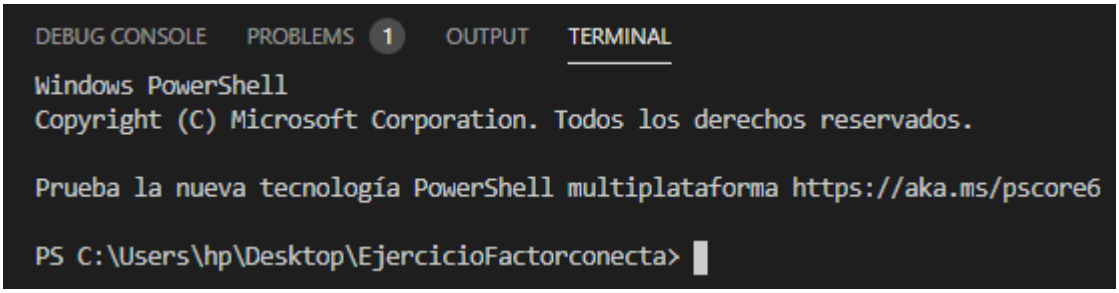
Antes, creamos las siguientes carpetas y archivos:

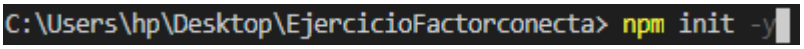



Escogemos New Terminal



Como podemos ver nos da por default la ruta donde se encuentra nuestro proyecto:

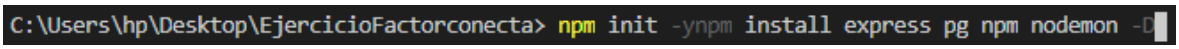



Escribimos: 

para iniciar nuestro proyecto.

Se crear un archivo JSON: 

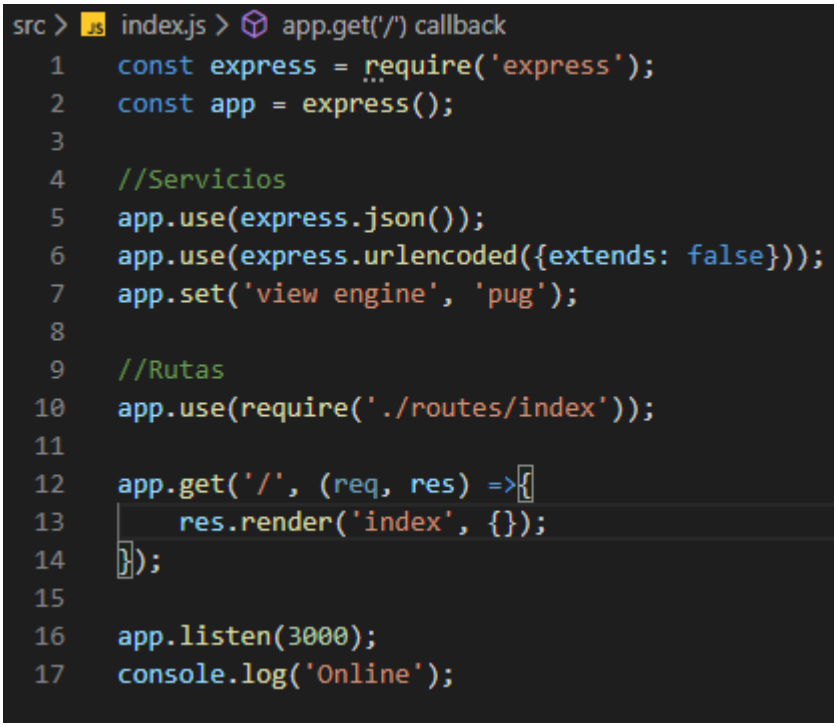

Instalaremos dependencias para poder desarrollar la aplicación: npm install express pg npm pug nodemon –D :




Express es para nuestro servidor local, pg es para la conexión con postgreSQL y nodemon para evitar actualizar manualmente el servidor, si vemos que tiene una –D esto significa que es solo para desarrollo, que en la aplicación final no es necesario.

Ahora creamos nuestro servidor local de la siguiente manera:

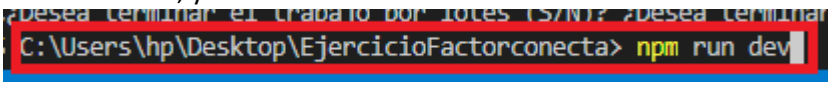
En nuestro index.js :



Las primeras dos líneas corresponden a la biblioteca requerida a usar, en este caso express, como se dan cuenta se guardan en constantes para que sea mas fácil su utilización, del 5 al 7 son sevicios a utilizar para que funcione de forma correcta, tenemos nuestra ruta principal, línea 10 y nuestra ruta donde se cuentra  el archivo principal de pug, línea 12 a 14, la línea 16 nos dice el puerto de nuestro servidor y la línea 17 nos da un mensaje por consola que esta en línea y de forma correcta.

Tenemos configurar nodemon para que funcione, así que nos vamos al JSON y hacemos modificaciones:


Modificamos el archivo agregando la línea de código en el recuadro rojo, es un script para poder usar nodemon, y tecleamos en la terminal:




Resultado:

```
> nodemon src/index.js

[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
body-parser deprecated undefined extended: provide extended option src\index.js:6:17
Online
```

Ahora en nuestro index.controller.js:

```
const { Pool } = require('pg'); // Biblioteca requerida para conectar a
postgreSQL


const pool = new Pool ({
    host: 'localhost',
    user: 'postgres',              //nos indican la configuracion
    password: 'toor',              // para ingresar a postgres
    database: 'ejercicio',
    port: '5432'
})

//Apartir de aquí es donde empieza la api Rest para poder manipular la base
de datos, ingresando, modificando, mostrando y eliminando usuarios.

const obtenerUsuarios = async (req, res) => {
    const respuesta = await pool.query('SELECT * FROM usuario');
    res.status(200).json(respuesta.rows);
};

const crearUsuario = async (req, res) => {
    const {nick, nombre, apellidoPa, apellidoMa, constra, rol, correo } = re
q.body;
    const respuesta = await pool.query('INSERT INTO usuario (nick, nombre, a
pellidoPa, apellidoMa, constra, rol, correo) VALUES ($1, $2, $3, $4, $5, $6,
 $7)', [nick, nombre, apellidoPa, apellidoMa, constra, rol, correo]);
    res.json({
        message: 'Creado',
        body: {
            usuario: {nick, nombre, apellidoPa, apellidoMa, constra, rol, co
rreo}
        }
    })
};
```

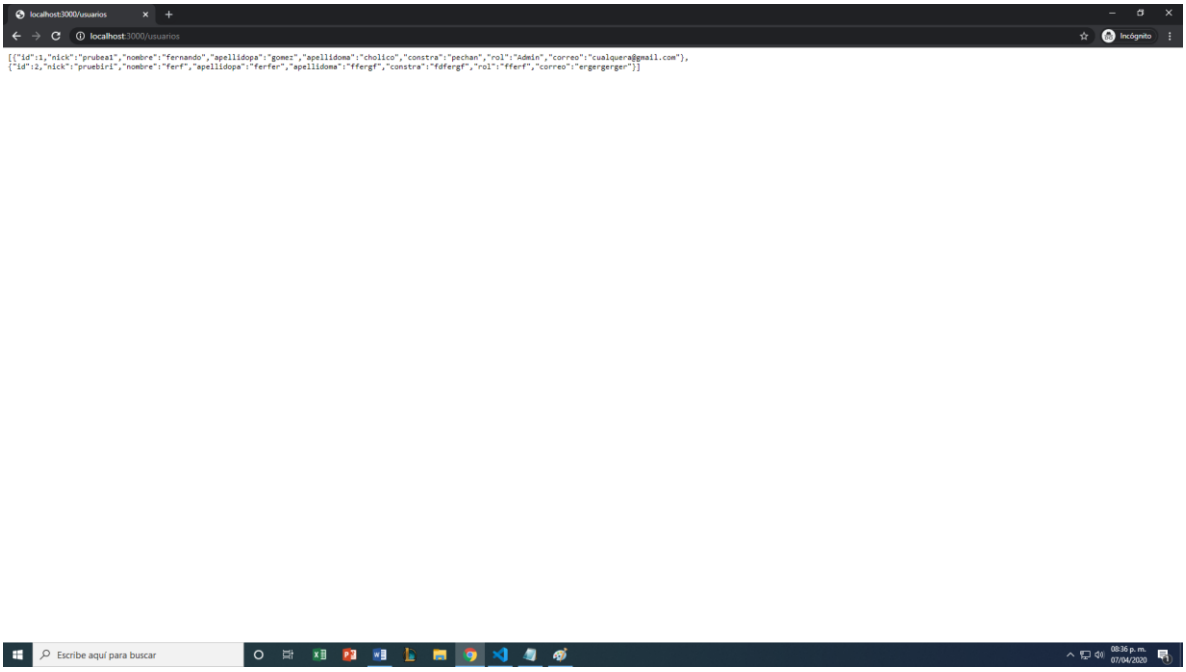
```
const obtenerUsuario = async (req, res) => {
    const id = parseInt(req.params.id);
    const response = await pool.query('SELECT * FROM usuario WHERE id = $1',
 [id]);
    res.json(response.rows);
};

const actualizarUsuario = async (req, res) => {
    const id = parseInt(req.params.id);
    const {nick, nombre, apellidoPa, apellidoMa, constra, rol, correo} = req
.body;

    const respuesta =await pool.query('UPDATE usuario SET nick = $1, nombre
= $2, apellidoPa = $3, apellidoMa = $4, constra = $5, rol = $6, correo = $7
WHERE id = $8', [
        nick,
        nombre,
        apellidoPa,
        apellidoMa,
        constra,
        rol,
        correo,
        id
    ]);
    res.json('Actualizado');
};

const eliminarUsuario = async (req, res) => {
    const id = parseInt(req.params.id);
    await pool.query('DELETE FROM usuario where id = $1', [
        id
    ]);
    res.json(`Usuario ${id} eliminado`);
};

//Modulos necesarios para exportar y puedan ser usados.

module.exports = {
    obtenerUsuarios,
    crearUsuario,
    obtenerUsuario,
    actualizarUsuario,
    eliminarUsuario

}
```

En nuestro routes/index.js.

```
src > routes > index.js > ...
1   const {Router} = require('express');
2   const router = Router();
3
4   const { obtenerUsuarios, crearUsuario, obtenerUsuario, actualizarUsuario, eliminarUsuario } = require('../controllers/index.controller')
5
6   router.get('/usuarios', obtenerUsuarios);
7   router.post('/usuarionuevo', crearUsuario);
8   router.put('/usuario/:id', actualizarUsuario);
9   router.delete('/usuario/:id', eliminarUsuario);
10  router.get('/usuario/:id', obtenerUsuario);
11
12  module.exports = router;
```

Tenemos las rutas para poder acceder a cada función del api creada.

Pruebas de funcionamiento:




Pd. Se uso postman para probar la api.