

Guía para Despliegue de API REST NodeJs con Postgresql en Render

Guía paso a paso.

Prerrequisitos

Para el desarrollo de la presente guía se requiere contar con las siguientes aplicaciones y prerrequisitos:

- a) IDE para el Desarrollo (Visual Studio Code).
- b) Cuenta de Github (<https://github.com/>)
- c) Cuenta de Render (<https://render.com/>)
- d) PostgreSQL
- e) NodeJs
- f) Git
- g) Herramienta para pruebas servicios web (Ej. Postman)

Desarrollo

1. El primer paso es crear una BD en **PostgreSql** y seguidamente crear una tabla llamada “**usuarios**” con el siguiente script:

```
CREATE TABLE usuarios (  
  Id SERIAL PRIMARY KEY,  
  nombre VARCHAR(30),  
  edad VARCHAR(6),  
  tipo VARCHAR(30)  
);
```

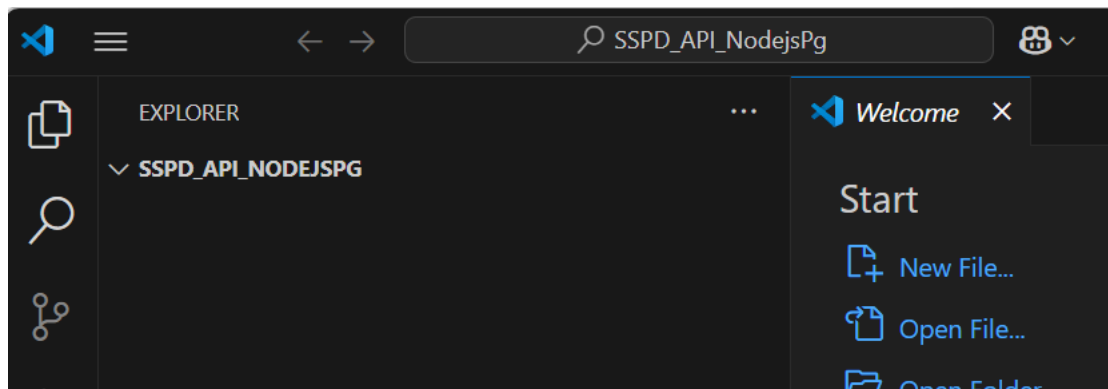
Insertar en la tabla el siguiente registro:

```
INSERT INTO usuarios (nombre,edad,tipo) VALUES ('Luis Diaz','27','Delantero');
```

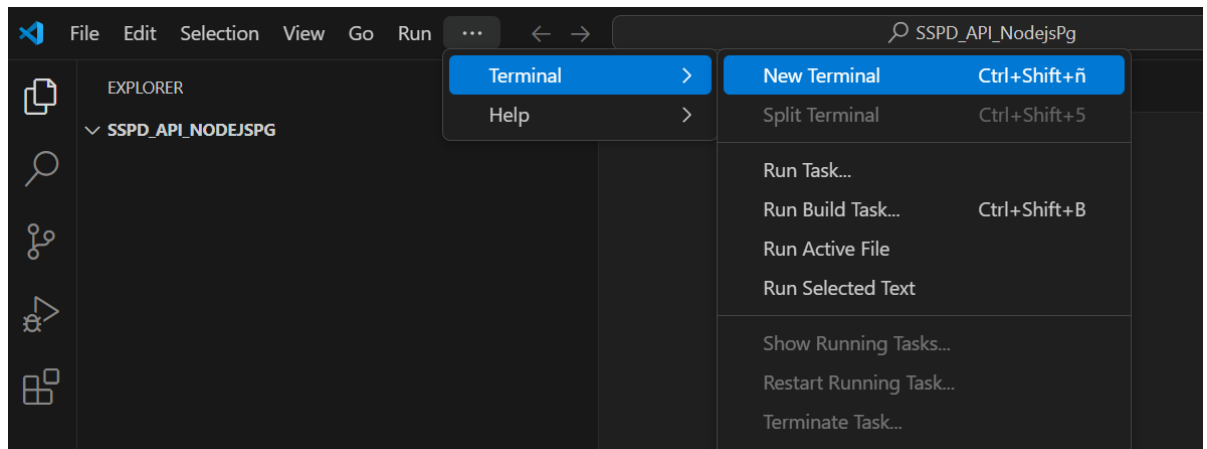
2. El segundo paso es crear una pequeña aplicación en NodeJs que permita desplegar un servidor web y crear un servicio web tipo REST para consultar y registrar datos en una tabla llamada “usuarios”.

Para crear un proyecto en NodeJs seguir los siguientes pasos:

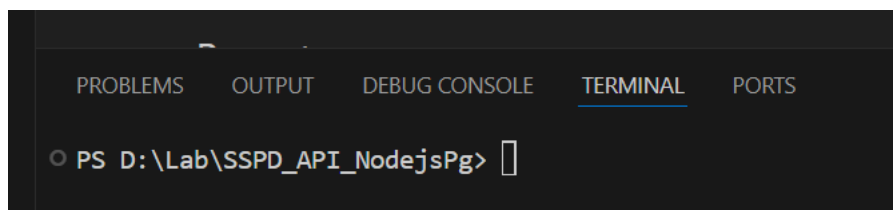
- a. Crear una carpeta llamada “SSPD_API_NodejsPg”.
- b. Abrir la herramienta Visual Studio Code
- c. Arrastrar la carpeta a la aplicación de Visual Studio Code.



- d. Una vez dentro de la aplicación seleccionar la opción **Terminal -> New Terminal**



- e. Desde la consola (Terminal) empezar a crear el proyecto de NodeJs siguiendo los siguientes pasos:



- **npm init** (Comando para iniciar la creación del proyecto)
Responder ENTER a cada pregunta.
Al terminar la operación, NodeJs crea un archivo llamada Package.Json que contiene las configuraciones de inicio, librerías, despliegues, etc.

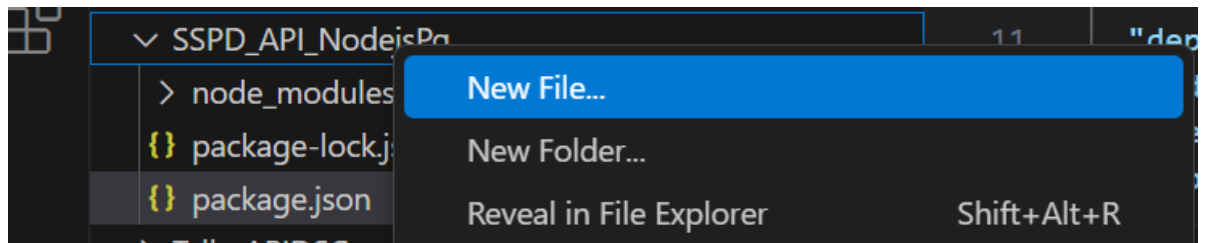
Instalar Librerías

- **npm install express** (Comando para instalar la librería para gestionar un servidor web)
- **npm install pg** (Comando para instalar la librería para acceder al motor de BD PostgreSQL)
- **npm install dotenv** (Comando para instalar la librería para la gestión de variables de entorno)

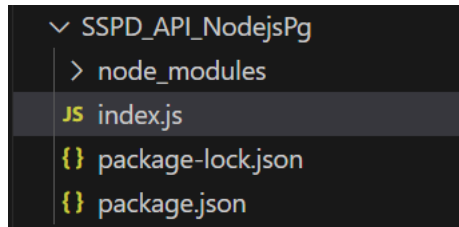
Al finalizar las instalaciones el archivo “**package.json**” deberá tener un contenido como el siguiente:

```
{
  "name": "sspd_api_nodejspg",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Fabrizio",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "pg": "^8.13.1"
  }
}
```

- f. Seguidamente se deberá crear un archivo llamado “**Index.js**” en la ruta raíz.



Para crear el archivo, presionar click derecho en el panel izquierdo (Explorer) justo debajo de la carpeta del proyecto (Ruta Raiz). Crear el archivo “Index.js”



El archivo “index.js” deberá contener el siguiente código:

```
const express = require("express");
const app = express();
const bodyParser = require('body-parser');
const { connectionString } = require("pg/lib/defaults");
require('dotenv').config()
const Pool = require('pg').Pool
//const connectionString = process.env.DB

app.use(bodyParser.json())
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
)

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
})

const getUsuario = (request, response) => {
  pool.query('SELECT * FROM usuarios ORDER BY id ASC', (error,
results) => {
    if (error) {
      throw error
    }
    response.status(200).json(results.rows)
  })
}
```

```

    }

    const crearUsuario = (request, response) => {
      const { nombre, edad, tipo } = request.body

      console.log("Nombre:", nombre, "edad:", edad, "tipo:", tipo)
      pool.query('insert into usuarios (nombre, edad, tipo) values ($1, $2, $3)', [nombre, edad, tipo], (error, results) => {
        if (error) {
          throw error
        }
        response.status(201).json({ UsuarioAgregado: 'Ok' })
      })
    }

    app.get('/', function (req, res) {
      res.json({ Resultado: 'Bienvenido al Taller Despliegue NodeJs y Postgres - Render' })
    });

    app.get('/usuarios', getUsuario)
    app.post('/usuarios', crearUsuario)

    const port = process.env.PORT || 1337;

    app.listen(port, () => {
      console.log("El servidor está inicializado en http://localhost:%d", port);
    });
  });
}

```

- g. De la misma forma y en la ruta raíz se deberá crear un archivo llamado “.env” en el cual se definirán las variables de entorno para la conexión al motor de BD PostgreSQL.

El contenido del archivo será el siguiente:

```

DATABASE_URL = postgresql://postgres:123456@localhost:5432/SSPD

```

postgresql://**usuario:password@host:puerto/database**

Nota: El contenido de los datos y credenciales de acceso al motor PostgreSQL deberán cambiarse por los de su ambiente local

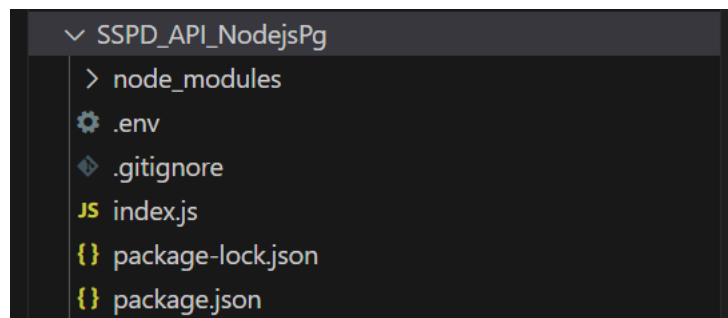
- h. De la misma forma, se deberá crear el archivo “**.gitignore**” que será utilizado por NodeJs para evitar subir al repositorio de GitHub algunos archivos o librerías que no son necesarias.

El contenido de este archivo es el siguiente:

```
node_modules
```

Cuando se realice la operación PUSH al repositorio, la aplicación lee este archivo e ignora las carpetas o archivos configurados en él y así evitará subir la carpeta de los archivos de los módulos instalados localmente al repositorio de GitHub.

- i. Una vez creados los archivos anteriormente descritos, el proyecto se verá con la siguiente configuración:



- j. Una vez terminada la configuración de los archivos, el siguiente paso es ejecutar la aplicación y desplegar de manera local.

Para ejecutar deberá escribir el siguiente comando desde la terminal de comandos:

Ejecución

npm run start (Comando para ejecutar la aplicación)

La etiqueta “start” esta definida en el archivo “package.json”

```
SSPD_API_NodejsPg > {} package.json > ...
1  {
2    "name": "sspd_api_nodejspg",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "start": "node index.js"
7    },
8    "author": "Fabrizio",
9    "license": "ISC",
10   "description": "",
11   "dependencies": {
12     "dotenv": "^16.4.7",
13     "express": "^4.21.2",
14     "pg": "^8.13.1"
15   }
16 }
17
```

Al ejecutar la aplicación, se mostrará lo siguiente:

```
(nenv2015) PS D:\Entornos2024\LabNodeJsAngular\SSPD_API_NodejsPg> npm run start
> sspd_api_nodejspg@1.0.0 start
> node index.js

El servidor está inicializado en http://localhost:1337
█
```

Se muestra el mensaje que el servidor está inicializado en el puerto indicado.

- k. El siguiente paso es abrir un navegador y comprobar el acceso al servidor y a las rutas de las API definidas:

Por ejemplo, al abrir el navegador Firefox y escribir la ruta:

<http://localhost:1337/>

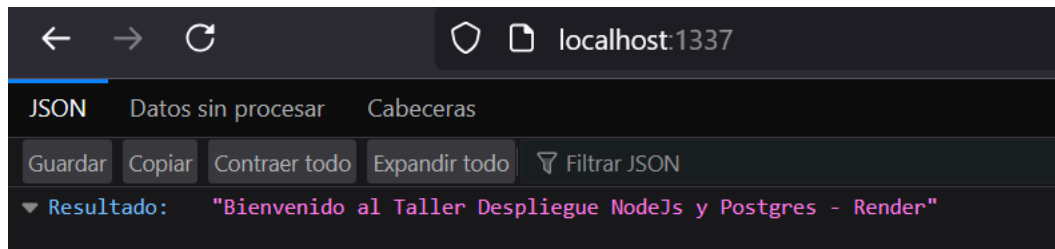
Veremos lo siguiente:

```

  <  >  ↻  ⓘ  localhost:1337

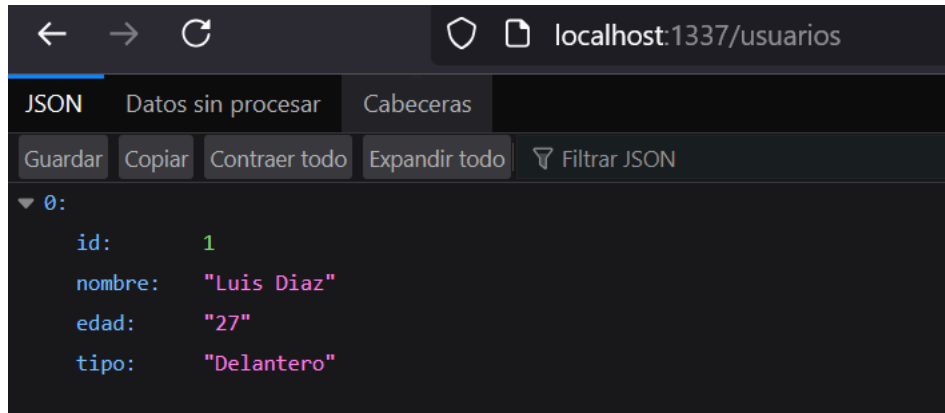
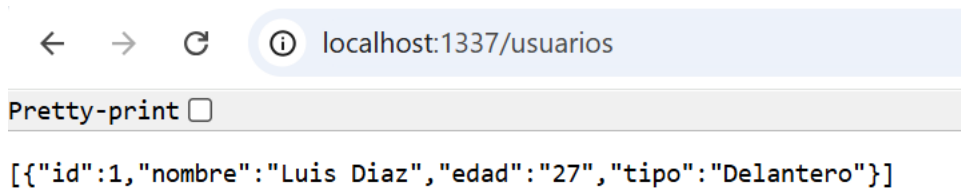
Pretty-print ☐

{"Resultado":"Bienvenido al Taller Despliegue NodeJs y Postgres - Render"}
```



Al solicitar la ruta /usuarios/ veremos lo siguiente:

<http://localhost:1337/usuarios/>



Recordemos que la ruta “/usuarios”, según el código llama a la función `getUsuario` que a su vez consulta todos los registros de la tabla “usuarios” de la BD PostgrSql y los devuelve en una variable “results”, la cual se le da el formato tipo JSON.

```
app.get('/usuarios', getUsuario)
```

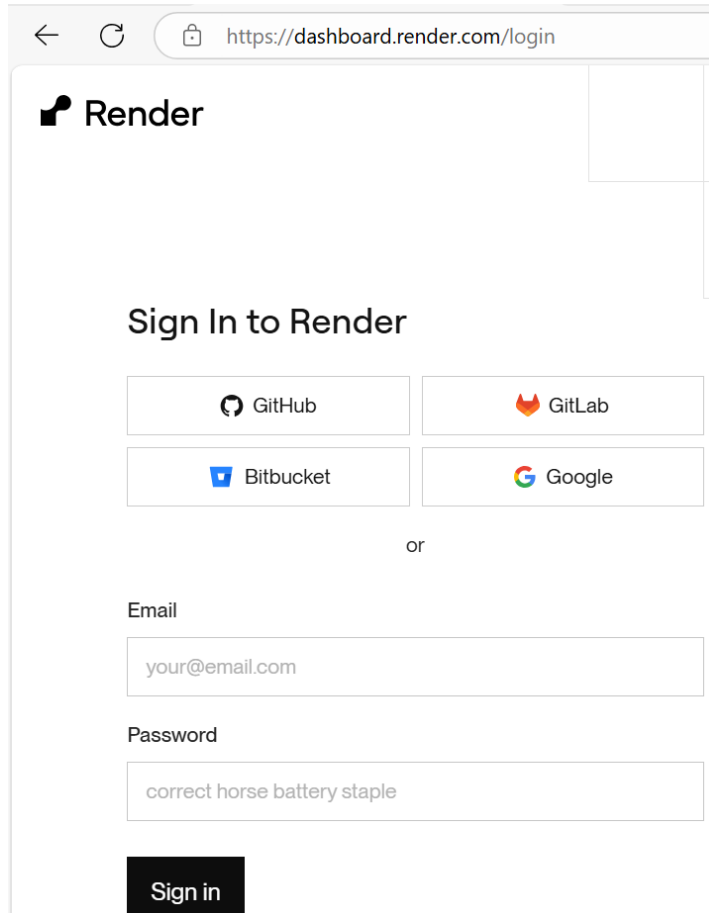
```
const getUsuario = (request, response) => {
  pool.query('SELECT * FROM usuarios ORDER BY id ASC', (error,
results) => {
    if (error) {
      throw error
    }
    response.status(200).json(results.rows)
  })
}
```



```
} )  
}
```

3. Crear la cuenta en **Render** (<https://render.com/>).









Para crear la cuenta en la plataforma Render puedes utilizar tu cuenta de Github y autorizar el acceso.



The screenshot shows the login page of the Render platform. The browser's address bar displays 'https://dashboard.render.com/login'. The page features the Render logo at the top left. Below the logo, the heading 'Sign In to Render' is centered. Underneath this heading, there are four buttons for social login: GitHub, GitLab, Bitbucket, and Google. Below these buttons, the word 'or' is centered. Further down, there are two input fields: one for 'Email' with the placeholder text 'your@email.com' and another for 'Password' with the placeholder text 'correct horse battery staple'. At the bottom of the form is a black button with the text 'Sign in' in white.

Una vez dentro de la plataforma Render, se podrá visualizar lo siguiente:

Get started in minutes

| | | | |
|---|--|---|---|
|  Static Sites Static Sites are automatically served over a global CDN. Add a custom domain and get free, fully-managed SSL. New Static Site |  Web Services Web Services include zero-downtime deploys, persistent storage and PR previews. Scale up and down with ease. New Web Service |  Private Services Private Services are only accessible within your Render network and can speak any protocol. New Private Service |  Background Workers Background Workers are suitable for long running processes like consumers for queues and streaming. New Worker |
|  Cron Jobs With Cron Jobs, you can schedule any command or script to run on a regular interval. New Cron Job |  PostgreSQL Fully-managed hosted PostgreSQL with internal and external connectivity, and automated daily backups. New PostgreSQL |  Redis A cloud based in-memory key value datastore. Render offers fully managed hosted Redis instances. New Redis |  Blueprints A Blueprint specifies your Infrastructure as Code in a single file. Use it to set up all your services at once. New Blueprint |

Se pueden desplegar, sitios estáticos, web services, código en segundo plano, código para ejecutar después de un tiempo (serverless), Jobs, bases de datos postgres y redis.

En esta guía, se va crear un Web Service y una base de datos en Postgres.

4. Para crear la base de datos en Postgresql seleccionamos la opción y se presenta la siguiente ventana.

New PostgreSQL

[View docs](#)

Name

A unique name for your PostgreSQL instance.

Project Optional

Add this database to a [project](#) once it's created.



Create a new project to add this to?

You don't have any projects in this workspace. [Projects](#) allow you to group resources into environments so you can better manage related resources.

[+ Create a project](#)

En la opción Name ingresaremos el nombre que tendrá nuestra instancia de BD.

Database Optional
 The PostgreSQL `dbname`

randomly generated unless specified

User Optional

randomly generated unless specified

Region
 Your services in the same [region](#) can communicate over a [private network](#). You currently have services running in **Oregon**.

☒ Oregon (US West)

1 existing service

Deploy in a new region +

PostgreSQL Version

16

Datadog API Key Optional
 The API key to use for sending metrics to Datadog. Setting this will enable Datadog monitoring.

La opción Database y User son opcionales para definir y lo recomendable es dejarlas en blanco para que la aplicación de Render las genere automáticamente.

La región de despliegue también viene definida por defecto.

La versión de Postgres también se puede seleccionar. Es recomendable seleccionar la versión 15.

La opción Datadog API Key es para generar logs y es recomendable dejarla en blanco.

Seguidamente se debe seleccionar la opción **Free**.

Plan Options

Instance Type
 Set your database's RAM and CPU. You can change your instance type later.

New
 You can now set your database's storage separately from its instance type [Learn more](#).

☒ **Free**
 For testing out PostgreSQL on Render

Free

256 MB (RAM)

\$0 / month

0.1 CPU

1 GB (Storage)

☐ **Basic**
 Reliability and performance for hobby projects. Starting at \$6 / month plus storage.

☐ **Pro**
 Perfect for production use cases at any scale. Starting at \$55 / month plus storage.

Seguidamente hay otras opciones que se recomienda dejar igual a lo sugerido.

Storage

Your database's capacity, in GB. You can increase storage at any time, but you can't decrease it. Specify 1 GB or any multiple of 5 GB.

1

GB

\$0 / month

Enable High Availability

Run a standby instance of your database and automatically fail over to it if the primary encounters an issue. [Learn more](#)

☐ Disabled

ⓘ

 Only available for **Pro** instances and higher.

Monthly Total

Database instances are billed and prorated by the second.

Instance: **Free**

\$0 / month

Total

\$0 / month

Create Database

Finalmente dar click en el botón Create Database

Create Database

A partir de este momento la plataforma Render generará la BD (*Esto tarda unos minutos*)

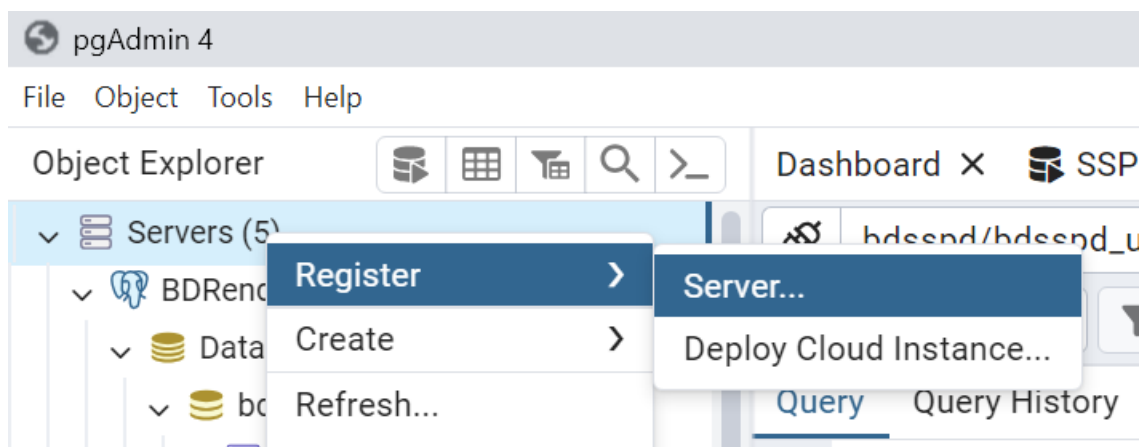
5. Revisar la configuración de la Base de Datos Postgresql

Una vez creada la base de datos podemos visualizar las opciones de conexión y propiedades.

Connections

| | |
|---|----------------------------|
| Hostname An internal hostname used by your Render services. | dpg-cuba9hi3esus73ersav0-a |
| Port | 5432 |
| Database | bdsspd |
| Username | bdsspd_user |
| Password | |
| Internal Database URL | |
| External Database URL | |
| PSQL Command | |

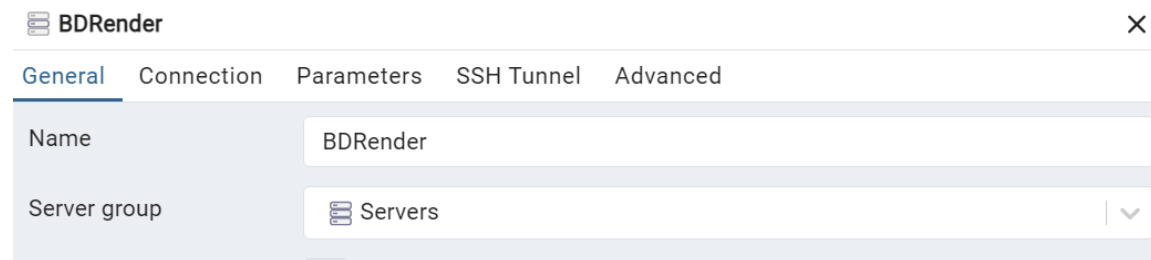
Para realizar las pruebas, debemos crear la tabla **usuarios** y para ello procedemos a crear la conexión desde la herramienta PgAdmin y creamos una nueva conexión de servidor utilizando los valores definidos en la variable **External Database URL**



External Database URL

postgresql://bdsspd_user:iB75BioCB7vivaBfuWXjLquhNPi8nO7b@dpg-cuba9hi3esus73ersav0-a.oregon-postgres.render.com/bdsspd

En la pestaña General asignamos un nombre a la conexión.



The screenshot shows the 'General' tab of the BDRender configuration window. The 'Name' field is set to 'BDRender'. The 'Server group' dropdown menu is open, showing 'Servers' as the selected option.

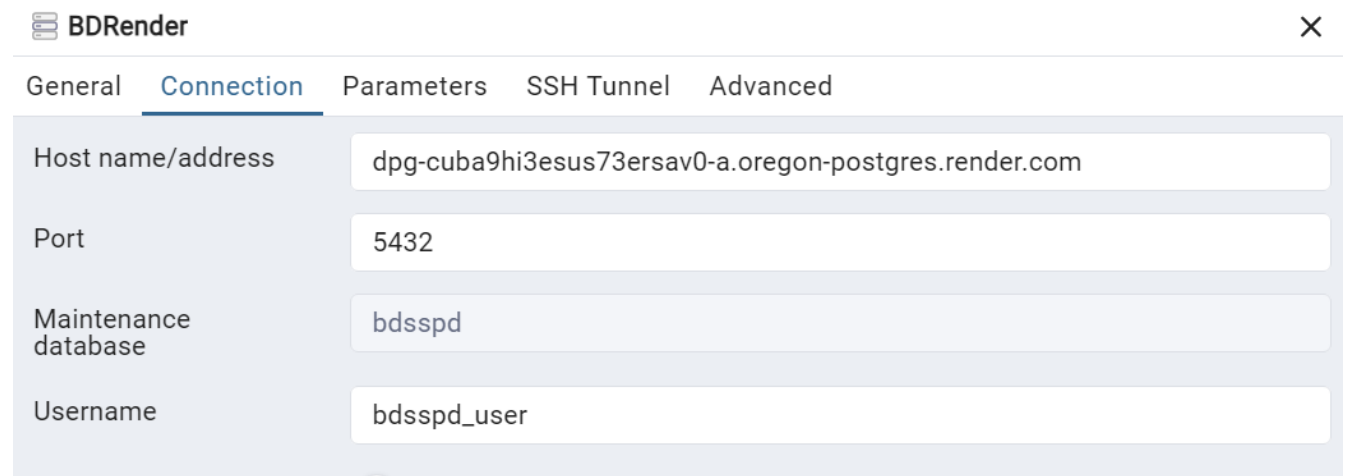
En la pestaña Conexión configuramos cada parámetro definido en la cadena **External Database URL**

Username: bdsspd_user

Contraseña: iB75BioCB7vivaBfuWXjLquhNPi8nO7b

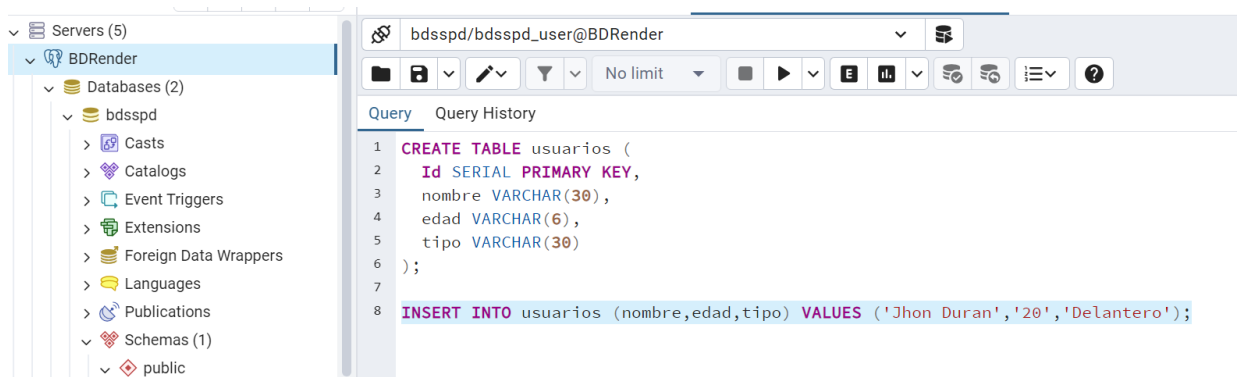
Database: bdsspd

Host: dpg-cuba9hi3esus73ersav0-a.oregon-postgres.render.com



The screenshot shows the 'Connection' tab of the BDRender configuration window. The 'Host name/address' field is set to 'dpg-cuba9hi3esus73ersav0-a.oregon-postgres.render.com'. The 'Port' field is set to '5432'. The 'Maintenance database' field is set to 'bdsspd'. The 'Username' field is set to 'bdsspd_user'.

Una vez configurada la conexión, la herramienta PgAdmin permitirá el acceso y seguidamente podremos crear la tabla e ingresar unos registros.



```
CREATE TABLE usuarios (
  Id SERIAL PRIMARY KEY,
  nombre VARCHAR(30),
  edad VARCHAR(6),
  tipo VARCHAR(30)
);
```

```
INSERT INTO usuarios (nombre,edad,tipo) VALUES ('Jhon Duran','20','Delantero');
```

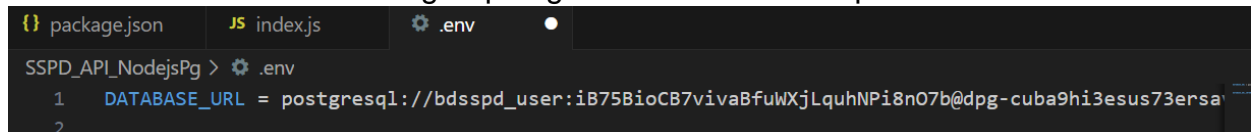
NOTA: Es importante precisar que esta tabla y este registro se insertaron en la base de datos de Postgresql creada en la plataforma Render.

6. Pruebas del API local de Node Js conectado a la BD Postgres en Render.

Para realizar las pruebas desde nuestra aplicación NodeJs actualizamos la variable del archivo **.env** **DATABASE_URL** por los valores definidos en la plataforma en la opción External Database URL lo cual nos permitirá conectarnos desde una aplicación externa..

DATABASE_URL

postgresql://bdsspd_user:iB75BioCB7vivaBfuWXjLquhNPi8nO7b@dpg-cuba9hi3esus73ersav0-a.oregon-postgres.render.com/bdsspd

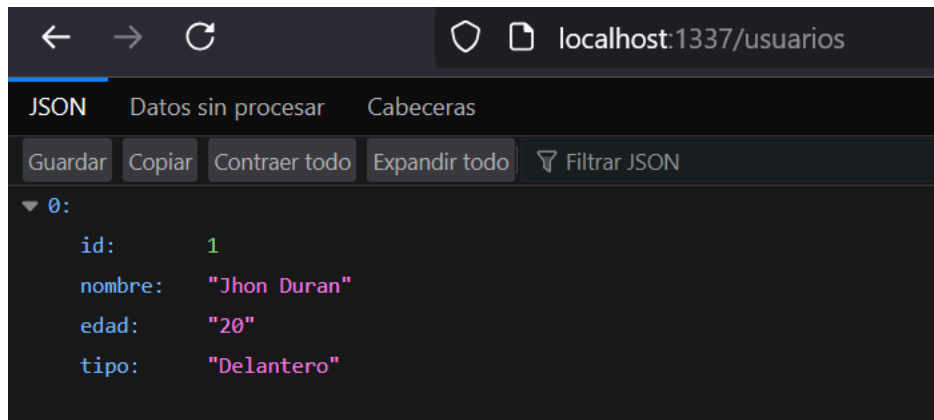


Adicionalmente, se debe realizar un ajuste en la función donde se define la cadena de conexión agregando una propiedad de SSL en True.

```
const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl : true
})
```

NOTA: Para pruebas desde el entorno local es importante agregar el parámetro SSL sin embargo cuando se vaya a realizar el despliegue, esta propiedad no es necesaria dado que el acceso estará en un entorno integrado.

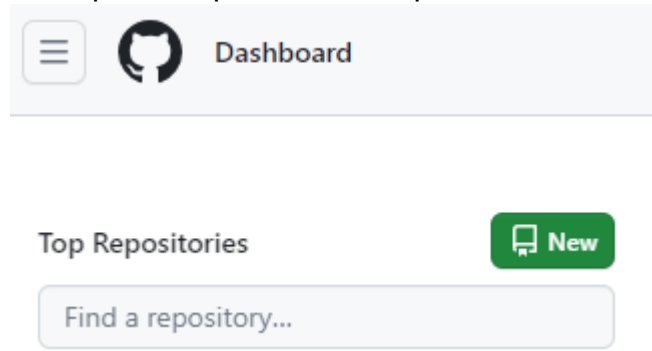
Luego de hacer la modificación, se realizan nuevamente las pruebas y se puede visualizar el registro agregado en la base de datos de Render.



7. El siguiente paso es crear un nuevo repositorio en GitHub con el fin de subir el proyecto realizado en NodeJs

Para este paso se deberá crear una cuenta en GitHub y seguidamente crear un nuevo repositorio como se detalla a continuación:

- a) En el panel izquierdo de la aplicación seleccionar el botón “New”.



- b) Seguidamente registrar el nombre del repositorio y verificar que este disponible.


Para este taller se podrá crear de tipo “Público” o “Privado”

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).



Owner * Repository name *

 fabriziobolano / BackendSSPD



✔ BackendSSPD is available.

Great repository names are short and memorable. Need inspiration? How about [literate-bassoon](#) ?

Description (optional)

- ☐  Public
Anyone on the internet can see this repository. You choose who can commit.
- ☒  Private
You choose who can see and commit to this repository.

c) Al terminar de configurar las opciones, presionar click en el botón “**Create repository**”

- ☐  Public
Anyone on the internet can see this repository. You choose who can commit.
- ☒  Private
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

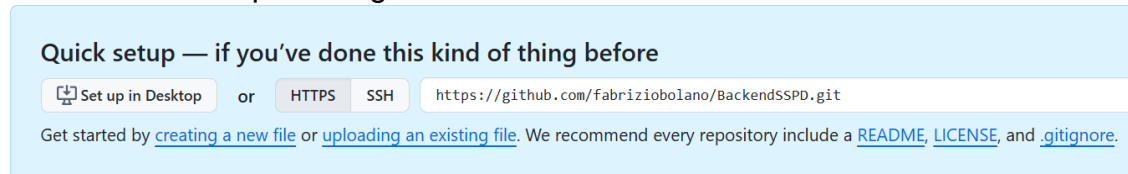
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a private repository in your personal account.

Create repository

d) Al crear el repositorio, Github le mostrará las diferentes formas de conexión y/o acceso al nuevo repositorio creado.

Desde HTTPS o para cargar dese archivos:



Desde la línea de comandos:

```
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/fabriziobolano/BackendSSPD.git
git push -u origin main
```

Instrucciones para hacer PUSH desde la línea de comandos:

```
git remote add origin https://github.com/fabriziobolano/BackendSSPD.git
git branch -M main
git push -u origin main
```

Al terminar estos pasos ya tendrá un repositorio disponible para cargar el nuevo proyecto desarrollado en NodeJs.

8. El siguiente paso es iniciar un repositorio con la herramienta Git y configurarlo para subirlo al repositorio vacío creado en el punto anterior en la plataforma GitHub.

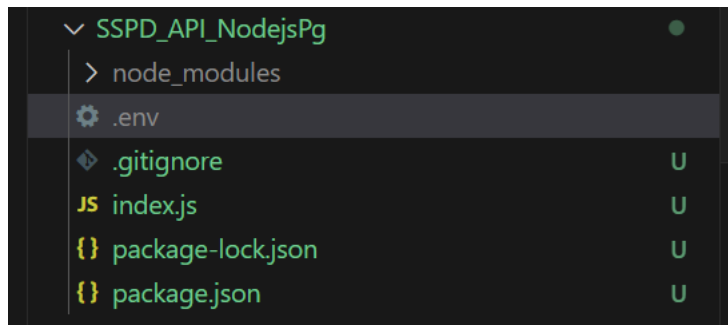
Para realizar este paso, se deberá haber instalado previamente la herramienta Git en su sistema operativo (Windows, Linux o Mac).

Al usar Git por primera vez, deberás establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque los "commits" de Git usan esta información, y es introducida de manera inmutable en los commits que envías:

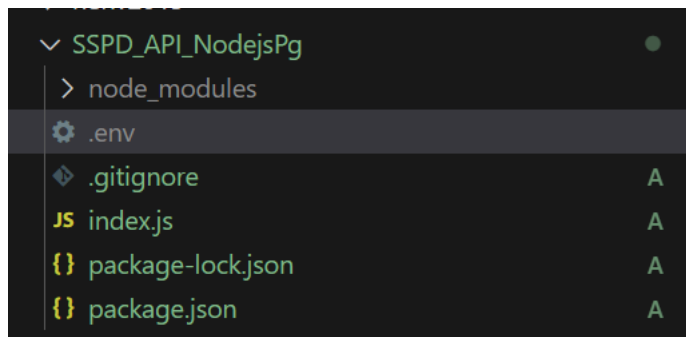
```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Luego se deberán seguir los siguientes pasos:

- git init



- `git add .`



- `git commit -m "primera version"`

```
(nenv2015) PS D:\Entornos2024\LabNodeJsAngular\SSPD_API_NodejsPg> git commit -m "Version 1 API SSPD"
[master (root-commit) a6633b9] Version 1 API SSPD
4 files changed, 983 insertions(+)
create mode 100644 .gitignore
create mode 100644 index.js
create mode 100644 package-lock.json
create mode 100644 package.json
(nenv2015) PS D:\Entornos2024\LabNodeJsAngular\SSPD_API_NodejsPg> 
```

- `git branch -M main`

El siguiente comando es conectarse de manera remota al repositorio creado en el punto anterior:

- `git remote add origin https://github.com/fabriziobolano/BackendSSPD.git`

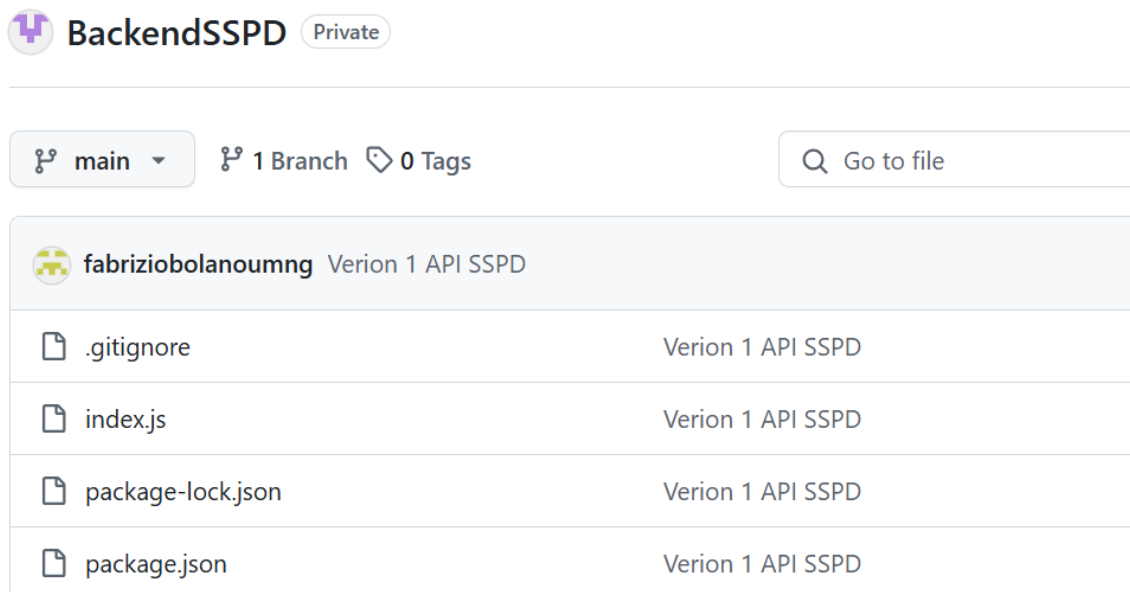
Nota: En este paso deberá permitir a Visual Studio Code conectarse a su cuenta de GitHub.

- `git push -u origin main`

Después de aplicar este comando, el proyecto estará cargado en el repositorio creado de GitHub. Para verificar deberá actualizar o refrescar el repositorio en Gitub.

```
(nenv2015) PS D:\Entornos2024\LabNodeJsAngular\SSPD_API_NodejsPg> git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 10.33 KiB | 10.33 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/fabriziobolano/BackendSSPD.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(nenv2015) PS D:\Entornos2024\LabNodeJsAngular\SSPD_API_NodejsPg>
```

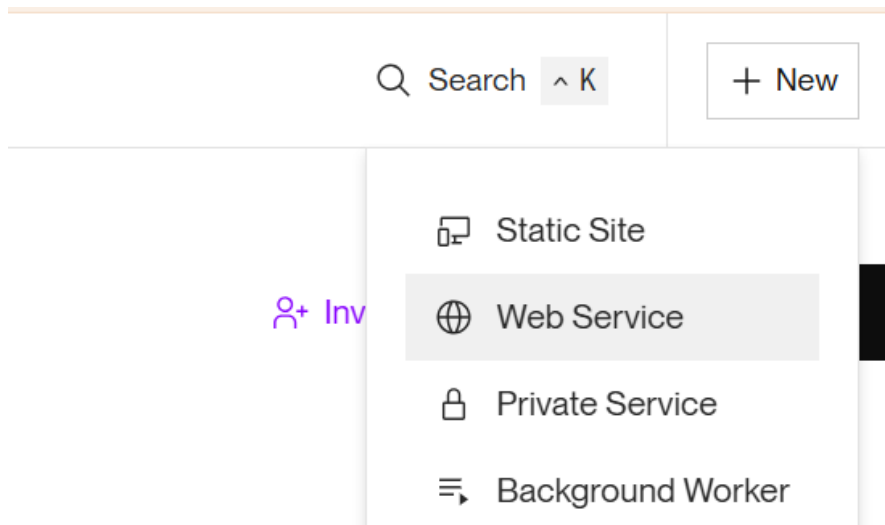
Luego de aplicar el comando anterior, se deben dirigir al repositorio de github y presionar la opción de actualizar página:



Seguidamente se podrá visualizar el código cargado al repositorio.

9. El siguiente paso es crear una aplicación de Web Services en la plataforma Render.

Para crear el web services vamos al botón New y seguidamente Web Services:



Seguidamente Render presenta una pantalla que permite conectarse a diferentes repositorios:




You are deploying a Web Service

Source Code

Git Provider Public Git Repository Existing Image

Connect Git provider

Connect your Git provider to deploy from your existing repositories.

 GitHub  GitLab  Bitbucket

Al seleccionar Github, la plataforma permite seleccionar algunos repositorios:



Install **Render**

Install on your personal account fabriziobolano



for these repositories:

☐ **All repositories**

This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

☒ **Only select repositories**

Select at least one repository.
Also includes public repositories (read-only).

Select repositories ▾

Para este taller, seleccionaremos el repositorio creado BacknedSSPD

☒ **Only select repositories**

Select at least one repository.
Also includes public repositories (read-only).

Select repositories ▾

Selected 1 repository.

fabriziobolano/BackendSSPD




with these permissions:

- ✓ **Read** access to Dependabot alerts, administration, code, and metadata
- ✓ **Read and write** access to actions, checks, commit statuses, deployments, environments, issues, pull requests, repository hooks, and workflows

Normalmente la plataforma pedirá confirmación de acceso al repositorio de Github



Confirm access

 Signed in as @fabriziobolano

Password

[Forgot password?](#)

Confirm

Una vez confirmado el acceso, se carga el repositorio.


You are deploying a Web Service




Source Code

Git Provider

Public Git Repository

Existing Image

 Credentials (1) ▾


 fabriziobolano /  BackendSSPD  15m ago




Seguidamente se selecciona el repositorio y se habilita el botón de **Conectar**

Git Provider

Public Git Repository

Existing Image

 Credentials (1) ▾

 fabriziobolano /  BackendSSPD  16m ago

✓

Connect →

Al presionar Conectar la plataforma de Render empezara a configurar las opciones para desplegar el Web Services.

Name

A unique name for your web service.

backendapisspd

Project Optional

Add this web service to a **project** once it's created.



Create a new project to add this to?

You don't have any projects in this workspace. **Projects** allow you to group resources into environments so you can better manage related resources.

+ Create a project

Language

Node

Branch

The Git branch to build and deploy.

main

Region

Your services in the same **region** can communicate over a **private network**. You currently have services running in **Oregon**.



Oregon (US West)

1 existing service

Deploy in a new region +

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a **monorepo**.

e.g. src

Build Command

Render runs this command to build your app before each deploy.

\$ npm install

Start Command

Render runs this command to start your app with each deploy.

\$ npm start

En la opción Start command es importante recordar que nuestra opción **npm start** que corresponde a la etiqueta definida en el archivo package.json.

Seguidamente seleccionar el tipo de instancia **Free**

Instance Type

For hobby projects

| | |
|-------------|--------------|
| Free | 512 MB (RAM) |
| \$0 / month | 0.1 CPU |

For professional use

For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

| | | | |
|----------------|--------------|-----------------|------------|
| Starter | 512 MB (RAM) | Standard | 2 GB (RAM) |
| \$7 / month | 0.5 CPU | \$25 / month | 1 CPU |

Upgrade to enable more features

Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

En la opción Variables de entorno agregamos la definida en nuestro archivo (.env) **DATABASE_URL** con el valor definido en la opción **Internal_URL** de la configuración de la base de datos PostgreSQL.

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more.](#)


| | | |
|--------------|-------|---|
| DATABASE_URL | |  |
|--------------|-------|---|


Finalmente presionamos click en el botón **Deploy Web Services**.

▼ **Advanced**

Deploy Web Service

Seguidamente la plataforma procederá a desplegar el Web Service (*puede tardar unos minutos*)

 Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. [Upgrade now](#)

January 26, 2025 at 5:57 PM  Building

[a6633b9](#) Verion 1 API SSPD **Cancel deploy**



Después de unos minutos, la plataforma muestra el detalle del despliegue y un mensaje al final que dice **"Your Service is Live"**:


```
All logs  Search  Live tail  GMT-5  ↑  ↵


Jan 26 05:58:04 PM 15 packages are looking for funding
Jan 26 05:58:04 PM run 'npm fund' for details
Jan 26 05:58:04 PM found 0 vulnerabilities
Jan 26 05:58:04 PM ==> Uploading build...
Jan 26 05:58:13 PM ==> Build uploaded in 8s
Jan 26 05:58:13 PM ==> Build successful 🎉
Jan 26 05:58:29 PM ==> Deploying...
Jan 26 05:58:49 PM ==> Running 'npm start'
Jan 26 05:58:50 PM > sspd_api_nodejspg@1.0.0 start
Jan 26 05:58:50 PM > node index.js
Jan 26 05:58:50 PM El servidor está inicializado en http://localhost:10000
Jan 26 05:58:59 PM ==> Your service is live 🎉
```

Luego la plataforma habilita una URL

Node Free [Upgrade your instance →](#)

 fabriziobolano / BackendSSPD 

<https://backendapisspd.onrender.com> 

 Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. [Upgrade now](#)

January 26, 2025 at 5:57 PM

✓ Live





[a6633b9](#) Verion 1 API SSPD

```
All logs  Search  Live tail  GMT-5  ↑  ↵

Jan 26 05:58:04 PM 15 packages are looking for funding
Jan 26 05:58:04 PM run 'npm fund' for details
```

10. Pruebas del Sitio Desplegado

Si todo esta correcto al presionar click sobre la nueva URL se podrá visualizar el Web Service desplegado:

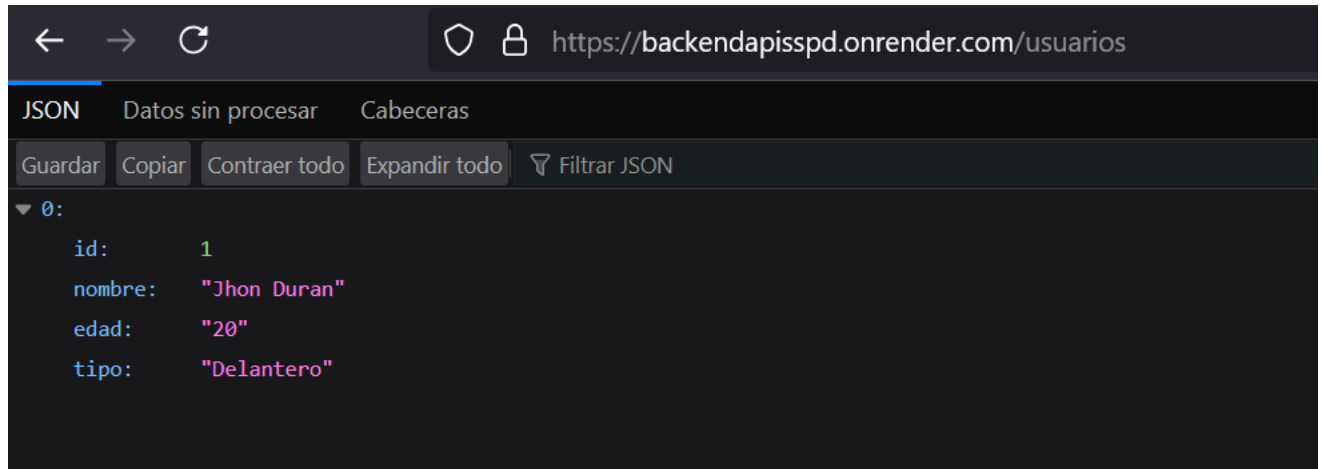
    backendapisspd.onrender.com

Pretty-print ☐

```
{"Resultado":"Bienvenido al Taller Despliegue NodeJs y Postgres - Render"}
```

Luego al realizar las pruebas del End Point “usuarios” podemos ver que las pruebas son exitosas:

<https://backendapisspd.onrender.com/usuarios>



11. Modificaciones y Despliegue Continuo.

Si se requiere realizar alguna modificación, se podrá realizar en el entorno local y luego realizar las opción de Git:

- Git add .
- git commit -m "ajuste"
- git push -u origin main

Luego del “git push” la plataforma de Render que está conectada al repositorio notará los cambios y realizará el despliegue de forma automática.