## HINTS

### CÓDIGOS HTTP

Existen múltiples códigos HTTP, y dentro del presente Text Class hemos revisado solo aquellos que son más utilizados. Puedes buscar más información al respecto, y recuerda que es más importante entender el concepto, su implementación y los contextos para su utilización, que aprenderse de memoria todos los códigos que existen; pues siempre podrás consultarlos.

### DEBUGGING

Al iniciar, suele darse poca importancia a la depuración de programas, principalmente porque en tu estudio del mundo de la programación estás trabajando con programas pequeños. Pero una vez que trabajas en proyectos más grandes, entenderás la importancia de lo que significa el control de errores y la búsqueda de bugs. Debes tener en cuenta el concepto de escribir código limpio y modular desde un inicio, ya que más adelante en tu carrera, éstos te ayudarán a ahorrar tiempo. El desarrollo implica una gran dedicación a la búsqueda de errores, e incluso, de casos de borde que pueden afectar a tu programa, el cual puede o no estar preparado para ellos.

### TESTING

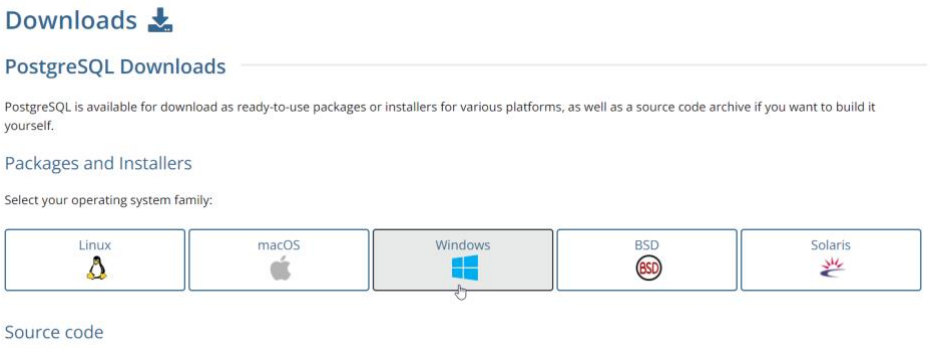
- La idea detrás de frameworks como mocha y chai, es poder hacer testing de calidad, con una sintaxis que sea fácil de leer. Existen múltiples métodos que te ayudarán a comprobar las respuestas de tu test, y la sintaxis de éstas hacen mucho más sentido cuanto entiendes inglés (ya que está escrita en este idioma). Puedes ayudarte de un traductor para conocer el significado semántico de los test, o leer la documentación en detalle para conocer la función de cada método. Todo esto permitirá que entiendas mejor, y en menos tiempo, un test, y por otra parte, que puedas crear unos que sean más fáciles de leer para el resto.
- Existen muchas posibilidades para crear tus tests, por lo cual es necesario encontrar el equilibrio entre uno demasiado especifico, o uno demasiado general. Recuerda que el tiempo es un factor importante a la hora de realizar un proyecto de desarrollo, y escribir tanto una función, como su test, toma el doble de tiempo.

- Recuerda siempre agregar mocha (o algún otro framework de testing) a tu archivo package.json, pues de lo contrario tu programa no será capaz de leer tus test.
- Recuerda que, si tu servidor utiliza llamadas asíncronas, debes emplear el método done () para enviar la señal de término del test a mocha.
- Recuerda que cada test ejecutado afectará los datos que estén involucrados en las peticiones realizadas a tu servidor.

**CONEXIÓN BÁSICA A BASE DE DATOS POSTGRESQL CON PROYECTO NODE.JS**

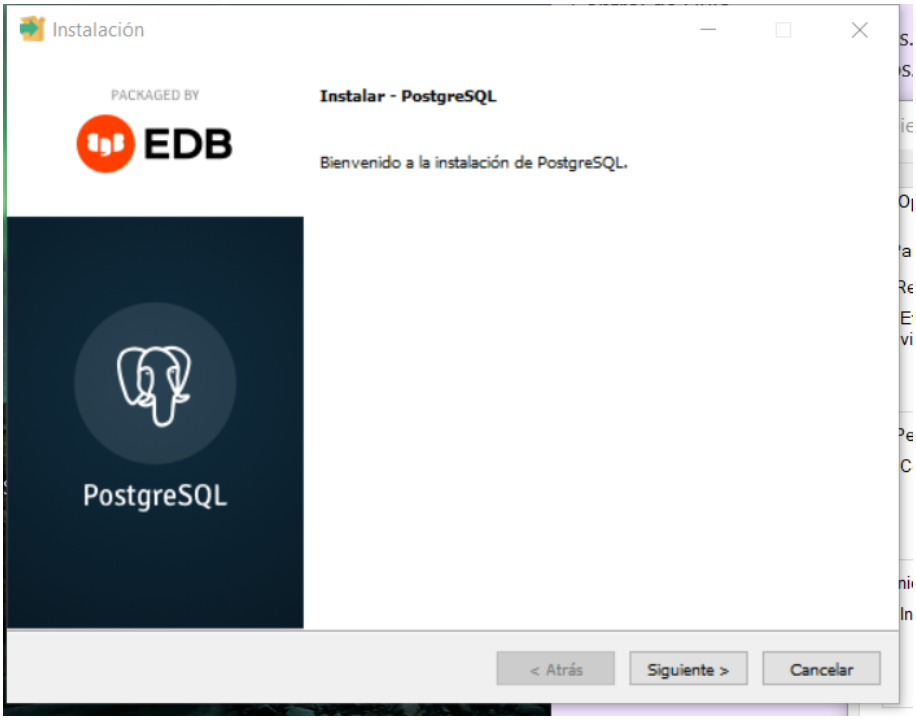
Explicaremos cómo podemos integrar la base de datos PostgreSQL, a nuestros proyectos creados con node.js.

En primer lugar, iremos a la página oficial: https://www.postgresql.org/download/, y seleccionamos el sistema operativo que utilizamos, en este caso Windows.




Hacemos clic en “Download the installer”.


Y seleccionamos la versión adecuada a nuestro sistema operativo. Esperamos que se termine la descarga, y ejecutamos como administrador el instalador.




Se nos preguntará el directorio de instalación. Lo recomendable es dejar el que viene por defecto.

Ahora se deberá elegir los componentes que queramos instalar:

- PostgreSQL Server: el servidor de PostgreSQL.
- pgAdmin: el equivalente de phpmyadmin, pero para Postgres.
- Stack Builder: es como un gestor de paquetes para Postgres; algo como lo que es NPM a Node.
- Command Line Tools: herramientas de la línea de comandos, que sirven para exportar, importar, entre otros.

Luego, nos pedirá elegir una ruta para almacenar los datos, y dejaremos la que viene por defecto. También solicitará crear una contraseña de superusuario para Postgres, la cual se debe recordar o anotar para usar más adelante.

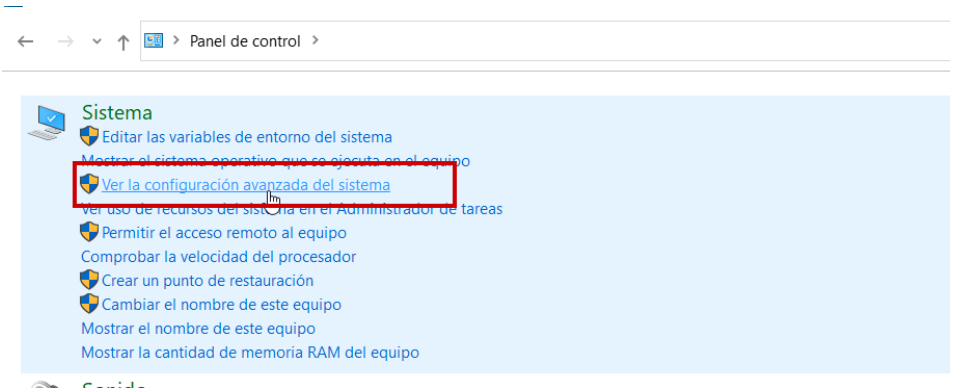
Lo siguiente será seleccionar el puerto. Es recomendable usar el **5432,** y si se encuentra ocupado, se puede dejar el **5433**.

A la configuración regional la dejaremos por defecto, y finalmente comenzamos la instalación. Una vez terminada, nos solicitará lanzar **stack builder** que es un gestor de paquetes, lo **seleccionaremos**, y le daremos clic a "Terminar".

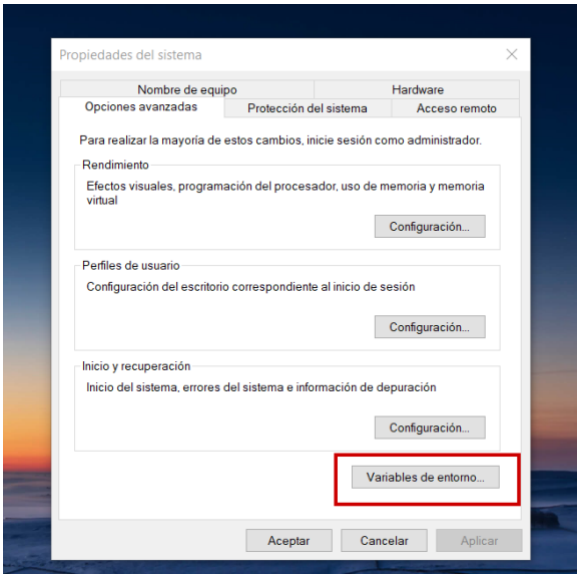
En la siguiente ventana, seleccionaremos la base de datos que instalamos, la cual será PostgreSQL 14. Daremos clic a "Siguiente", hasta que finalmente veremos que todo ha sido instalado correctamente, y le damos a "Finish".

Una vez que tenemos todo instalado, el siguiente paso será agregar el directorio Bbin de PostgreSQL a la variable path de Windows, de esta manera podremos ejecutar los binarios desde cualquier lugar, sin necesidad de escribir la ruta completa.

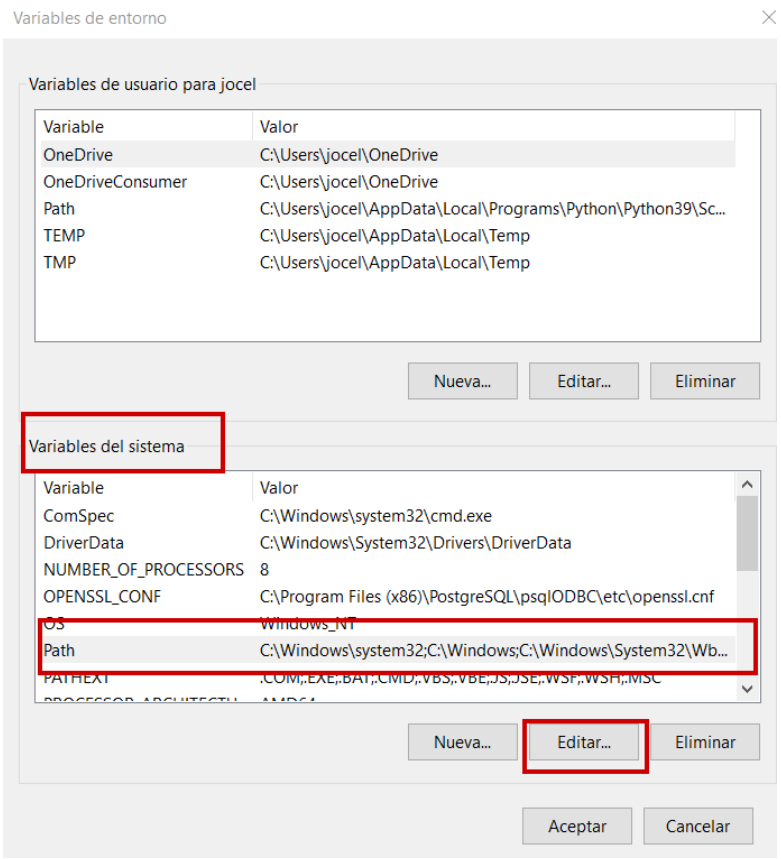
En el panel de control buscaremos "Sistema", y nos iremos a "Configuración avanzada del sistema".



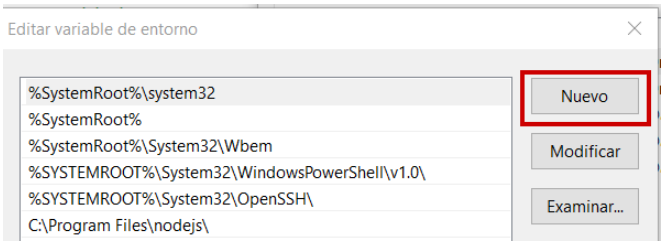

En la ventana que se abrirá, elegimos "Variables de entorno".



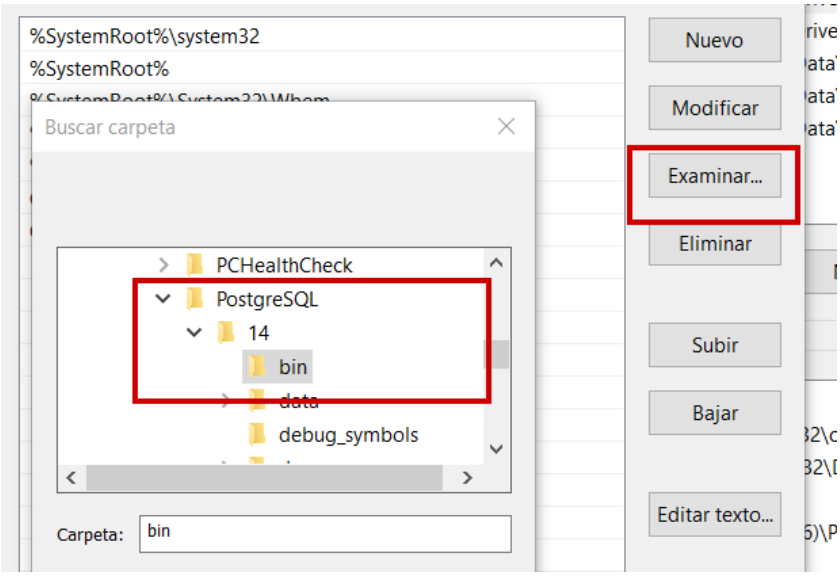
Nos aparecerá la ventana que vemos a continuación. En el apartado “variables de sistema”, nos dirigimos a “Path”, y presionaremos “Editar…”.



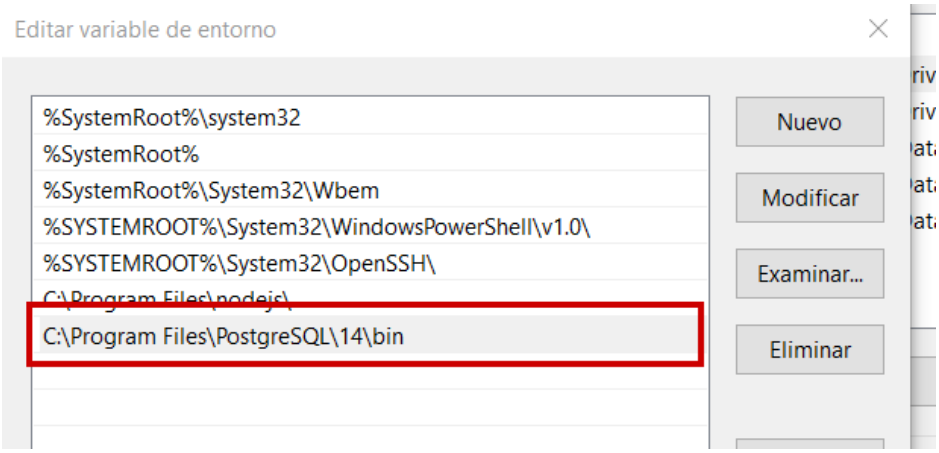

En la ventana que aparezca a continuación, seleccionamos “Nuevo”.



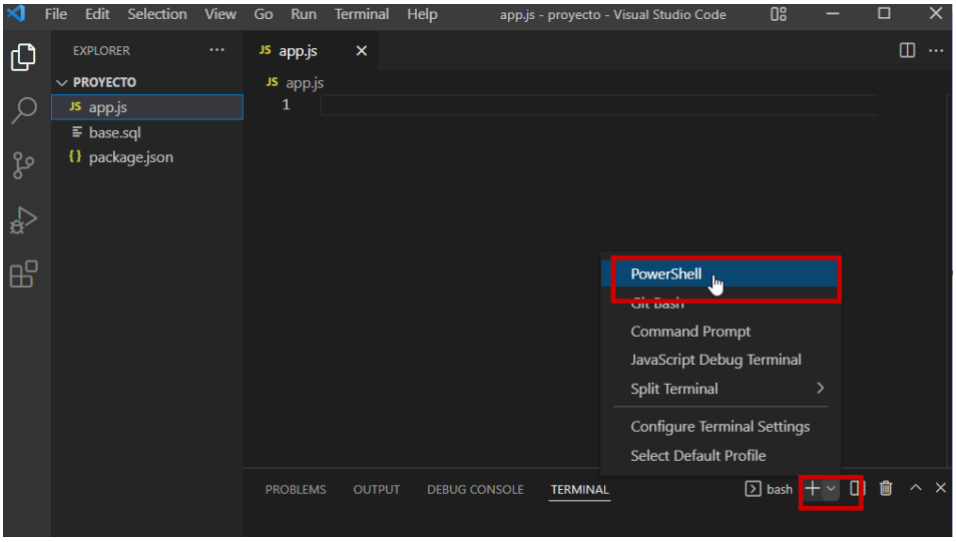

Luego seleccionamos “Examinar”, y se nos abrirá una pequeña ventana donde tendremos que buscar la carpeta en la que fue instalado PostgreSQL/bin. En este caso es: c:\program Files\PostgreSQL\14\bin.



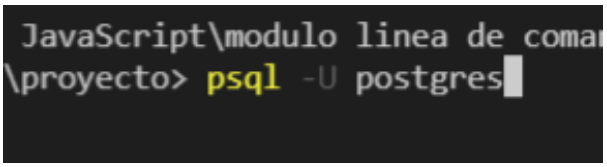

Quedando de la siguiente manera, y por último seleccionamos “Aceptar” a todo.



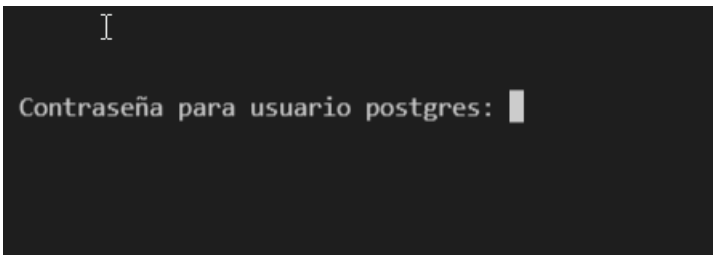

Ahora, para utilizar la base de datos que acabamos de instalar, y confirmar que todo se hizo correctamente, abriremos nuestro proyecto Node con Visual Studio Code, y para conectarnos a postgres a través de su programa de consola o terminal, nos dirigimos al signo + del terminal de Visual Studio Code, y presionaremos PowerShell.



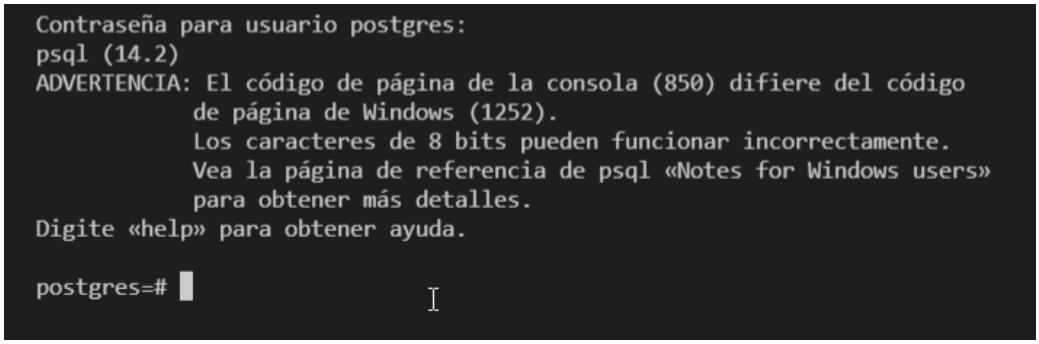

En la nueva terminal que se nos abra, escribiremos el comando para conectarnos a Postgres: `psql -U postgres`.




Al presionar Enter, nos solicitará ingresar la contraseña que creamos al instalar el programa.

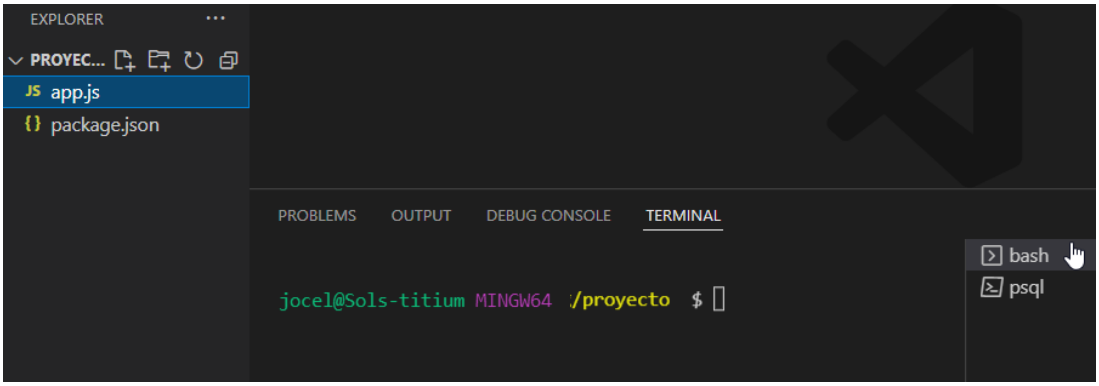


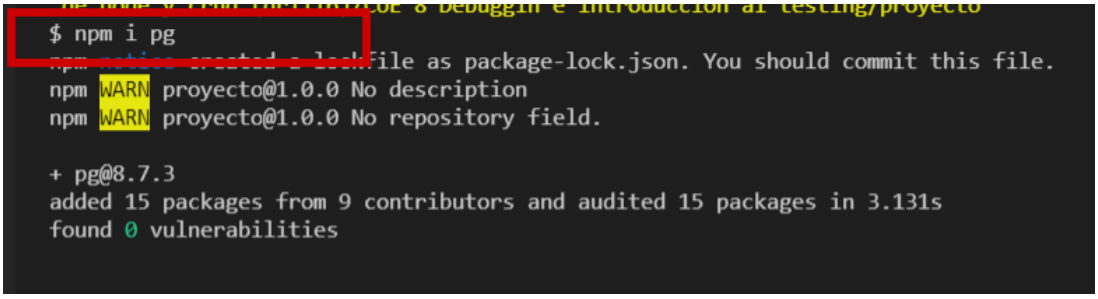

Nos dará la bienvenida con una advertencia de que algunos caracteres se podrían mostrar de manera distinta, esto es algo que no tiene mucha importancia.



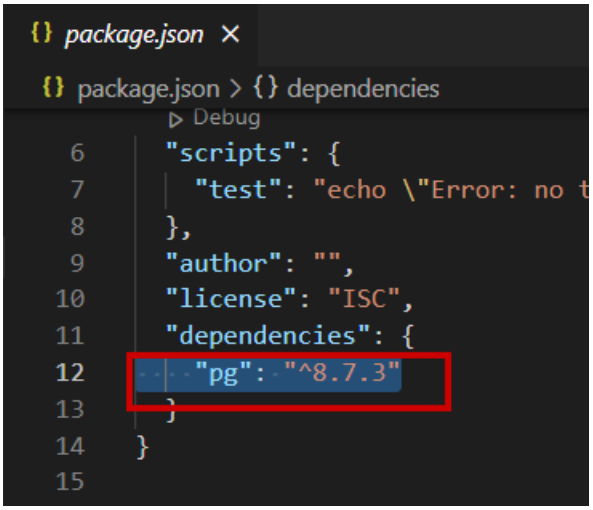

Finalmente, ya tenemos la conexión a Postgres, y podríamos empezar a crear una base de datos.

Volvemos a la terminal de nuestro proyecto donde inicializamos con `npm init`, y nos conectaremos a PostgreSQL por medio de la biblioteca `pg`. Para esto utilizaremos el comando `npm install pg` (podemos conocer más sobre esta biblioteca en su página oficial: https://node-postgres.com)




Una vez instalada la biblioteca, podemos confirmar si está incluida en nuestro proyecto. Para esto nos dirigimos al archivo `package.json`, y podremos ver en `dependencies` la biblioteca que acabamos de instalar.




Ya con esto tenemos las bases para conectar una base de datos a nuestro proyecto Node.js.