Transacciones y API REST

Levantando un servidor con conexión a PostgreSQL

{desafío} latam_

Implementar operaciones transaccionales en una base de datos para mantener la consistencia de los datos utilizando el entorno Node.js.

Implementar la capa de acceso a datos utilizando un ORM con entidades no relacionadas para realizar operaciones CRUD.

Utilizar asociaciones uno a uno, uno a muchos y muchos a muchos en la definición de las relaciones entre las entidades de un modelo que resuelven un problema.

{desafío} latam_

- Unidad 1:
  Implementación y gestión de una base de datos
- Unidad 2:
  Transacciones y API REST
- Unidad 3:
  Trabajo práctico

Te encuentras aquí

¿Qué aprenderás en esta sesión?

- *Preparación del servidor para establecer conexiones y configuración.*
- *Capacidad para configurar la conexión del servidor a PostgreSQL.*
- *Habilidad para procesar y utilizar los resultados de las consultas desde el servidor.*

{desafío} latam_

¿Cuáles son algunas de las consideraciones clave que debemos tener en cuenta al instalar y configurar PostgreSQL en un servidor antes de comenzar a trabajar con él?

{desafío} latam_

Ejercicio guiado:
Preparando mi servidor

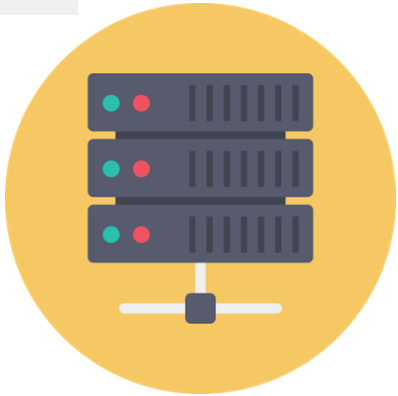
{desafío}
latam_

Preparando mi servidor

Ejercicio guiado

Construir un servidor en Express que utilice el paquete pg para consultar la fecha actual emitida desde PostgreSQL y devuelta a un cliente a través de una ruta.

```
npm i express
```



{desafío} latam_

Preparando mi servidor

Ejercicio guiado

Prosigue con los siguientes pasos para la realización de este ejercicio:

- **Paso 1:** Importar Express y levantar un servidor en el puerto 3000.
- **Paso 2:** Crea una ruta raíz / GET.

```
// Paso 1
const express = require('express');
const app = express();

app.listen(3000, console.log("Server ON"))

// Paso 2
app.get("/", (req, res) => {
    res.send("Servidor funcionando =D !");
})
```

Preparando mi servidor

Ejercicio guiado

Ahora abre tu navegador y entra a tu servidor a través de la ruta: http://localhost:3000/, deberás ver lo que te muestro en la siguiente imagen:

¡Perfecto!, ya hemos levantado el servidor sin problemas, aprovecho para recordarte que la práctica hace al maestro, si hoy en día has logrado levantar un servidor sin problemas y además entiendes como funciona, se debe a que ya lo hemos hecho varias veces y en cada repetición adquirimos más dominio en las tecnologías y herramientas que estemos usando.

/* Consultando con el paquete pg desde el servidor */

Consultando con el paquete pg desde el servidor

Ahora que tenemos nuestro servidor, podemos enfocarnos en la consulta a PostgreSQL con el paquete “pg”, para esto debes instalar el paquete con siguiente comando:

```
npm i pg
```

Una vez instalado, procedemos con la lógica a consultar al motor de bases de datos, pero hagámoslo de una forma modular y organizada.

Ejercicio guiado:
Probando conexión

Probando conexión

Ejercicio guiado

Crear un archivo nuevo llamado “consultas.js”, en donde escribiremos una función asíncrona que al ser ejecutada realice una consulta SQL con la instrucción “SELECT NOW()”, la cual recordemos procesa una petición al motor de base de datos, que devuelve un dato de tipo Date con la fecha actual.

Prosigue los siguientes pasos para la realización de este ejercicio:

- **Paso 1:** Importar el paquete pg y crear una instancia de la clase Pool, definiendo las propiedades básicas para una consulta. No es necesario definir ninguna base de datos.

- **Paso 2:** Crear una función asíncrona que devuelva el objeto result de una consulta SQL con la instrucción “SELECT NOW()”.

Probando conexión

Ejercicio guiado

- **Paso 3:** Exportar un módulo en forma de objeto que contenga la función asíncrona creada en el paso anterior.

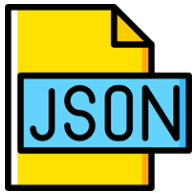
```
// Paso 1
const { Pool } = require("pg");

const pool = new Pool({
  user: "postgres",
  host: "localhost",
  password: "postgres",
  port: 5432,
});

// Paso 2
const getDate = async () => {
  const result = await pool.query("SELECT NOW()");
  return result;
};

// Paso 3
module.exports = { getDate };
```

Probando conexión

Ejercicio guiado

Muy bien, con este código escrito en el archivo “consultas.js”, ahora debemos importarlo en nuestro servidor, por lo que deberás realizar los siguientes cambios al archivo principal en donde desarrollamos el servidor:

- **Paso 1:** Importar la función asíncrona getDate del archivo “consultas.js”.
- **Paso 2:** Dentro de la ruta raíz, almacena en una constante el resultado de la función asíncrona importada.
- **Paso 3:** Devolver al cliente a través del método “json()” del parámetro “res” un JSON con el objeto result de la función asíncrona.



Probando conexión

Ejercicio guiado

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

app.listen(3000, console.log("Server ON"))

// Paso 1
const { getDate } = require("./consultas");

app.get("/", async (req, res) => {
    //   Paso 2
    const result = await getDate();
    //   Paso 3
    res.json(result);
})
```

{desafío} latam_

Probando conexión

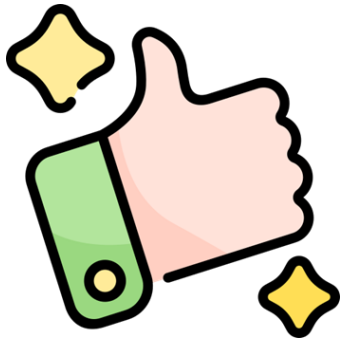
Ejercicio guiado

Ahora abre tu navegador y entra al servidor. Deberás recibir algo como lo que te muestro en la siguiente imagen:

Probando conexión

Ejercicio guiado

Ahí lo tenemos, estamos recibiendo el objeto result enviado desde PostgreSQL a nuestro servidor, disponibilizado a través de la ruta raíz. Con esto has logrado un importante paso, porque podremos de ahora en adelante utilizar los mecanismos de un servidor con Express, para programar una API REST y ofrecerle a un cliente los métodos para la gestión de datos en nuestras bases de datos.



En resumen, ¿qué pasos específicos debemos seguir para establecer una conexión exitosa desde un servidor a una base de datos PostgreSQL y ejecutar consultas SQL?

{desafío} latam_

Próxima sesión...

- *Desafío guiado - Mi Banco*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales