



Transacciones y API REST

Registros (Parte II)

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?

- *Desarrollar una aplicación que ejecute operaciones de consulta y manipulación de datos utilizando el entorno Node de acuerdo a las buenas prácticas de la industria.*

¿Cuáles son las principales razones por las cuales se deben implementar funciones de actualización y eliminación de registros en una aplicación que interactúa con una base de datos?



/* Actualizando registros de una tabla */

Actualizando registros de una tabla

Ahora que sabes como insertar y consultar registros, llega el momento de la 3era letra del famoso CRUD, me refiero al UPDATE. Recordemos que en una API REST la actualización se acostumbra a relacionar con el verbo PUT, el cual actualiza de forma definitiva un recurso, no obstante, no está de más que sepas que la actualización podría ser solo parcial, pero lo correcto técnicamente sería ocupar el verbo PATCH para esto. En nuestro caso usaremos la actualización total y por eso ocuparemos el método PUT.



`/* Función editar */`

Función editar

Entonces necesitamos preparar una función asíncrona que llamaremos “editar” y una ruta que reciban los datos escritos en el formulario correspondiente y emitan la consulta SQL para actualizar ese ejercicio.



Ejercicio guiado: PUT & UPDATE (UPDATE)

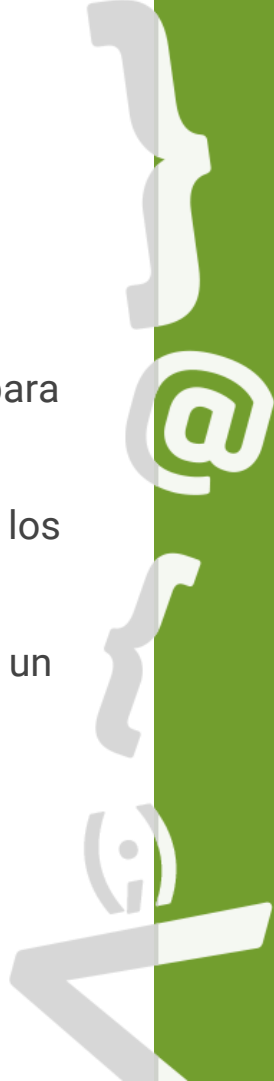


PUT & UPDATE

Ejercicio guiado

Agregar una función en el archivo “consultas.js” que reciba un arreglo con los nuevos valores del ejercicio que se desea editar, siendo el campo “nombre” el que usaremos para el condicional en la instrucción SQL.

- **Paso 1:** Crear una función asíncrona llamada editar que reciba como parámetros los datos.
- **Paso 2:** Preparar un objeto para hacer una consulta parametrizada, que actualice un registro utilizando el nombre como identificador en el comando “WHERE”. La propiedad “values” de este objeto debe ser igual al parámetro “datos”.
- **Paso 3:** Incluir la función “editar” dentro del objeto que estamos exportando en “consultas.js”



PUT & UPDATE

Ejercicio guiado

```
// Paso 1
const editar = async (datos) => {

  // Paso 2
  const consulta = {
    text: `UPDATE ejercicios SET
      nombre = $1,
      series = $2,
      repeticiones = $3,
      descanso = $4
    WHERE nombre = $1 RETURNING *`,
    values: datos,
  };
  const result = await pool.query(consulta);
  return result;
};

// Paso 3
module.exports = { insertar, consultar, editar };
```



/* Ruta PUT */

Ruta PUT

Bien, tenemos con esto la función “editar” creada, ahora vayamos con la ruta **PUT /ejercicios** en nuestro servidor. Lo que haremos básicamente será replicar la ruta **POST** cambiando solamente el método de la ruta y la función que ejecutaremos una vez tengamos la data enviada desde el cliente.

Prosigue con los siguientes pasos para el desarrollo de esta ruta:

- **Paso 1:** Importa la función editar en el script del servidor.
- **Paso 2:** Crear una ruta **PUT /ejercicios**.
- **Paso 3:** Almacenar los valores del cuerpo de la consulta usando el `Object.values()`.

Ruta PUT

- **Paso 4:** Utiliza la función editar y retorna a la aplicación cliente la respuesta.

```
// Paso 1
const { insertar, consultar, editar } = require('./consultas')
```

```
// Paso 2
app.put("/ejercicios", async (req, res) => {
  try {
    // Paso 3
    const datos = Object.values(req.body)
    // Paso 4
    const respuesta = await editar(datos)
    res.json(respuesta)
  } catch (error) {
    res.status(500).send("Algo salió mal :/ ...")
  }
})
```

Ruta PUT

Ok, es momento de probar nuestra ruta para actualizar un registro en la tabla ejercicios basado en el campo “nombre”. Para esto ocupa la interfaz en HTML, específicamente en el formulario de leyenda “Editar ejercicio” con los datos que te muestro en la siguiente imagen:

Editar ejercicio

Nombre: Abdominales

Series: 4

Repeticiones: 50

Descanso: 40

Editar

Ruta PUT

Ahora presionando el botón deberías ver en la tabla que se cambiaron los datos de “Abdominales”, así como te muestro en la siguiente imagen.

Agregar nuevo ejercicio

Nombre:

Series:

Repeticiones:

Descanso:

Agregar

Editar ejercicio

Nombre:

Series:

Repeticiones:

Descanso:

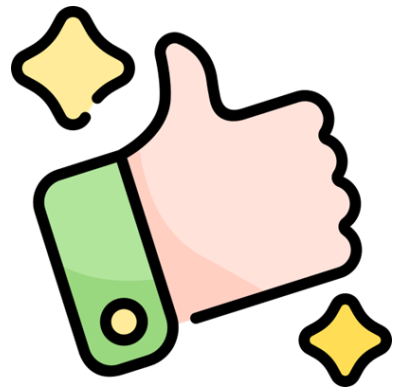
Editar

Eliminar ejercicio

Nombre:

Eliminar

Nombre	Series	Repeticiones	Descanso
Abdominales	4	50	40 segundos



/* Eliminando registros de una tabla */

Eliminando registros de una tabla

Has llegado a la última etapa de la construcción de nuestra API REST y esta se refiere al método DELETE, el cual comparte su nombre con la última letra de las siglas del CRUD.



Ejercicio guiado: DELETE & DELETE (DELETE)



DELETE & DELETE (DELETE)

Ejercicio guiado

Para eliminar un registro de la tabla **ejercicios** necesitamos solamente 1 dato para identificar al ejercicio que queremos eliminar, este dato será en nuestra temática de gimnasio el campo “nombre”, es decir que con solo el nombre del ejercicio podremos emitir la consulta SQL y eliminar ese registro de nuestra base de datos.



`/* Función eliminar */`

Función eliminar

Ejercicio guiado

Ya que solo necesitamos el nombre, podremos ocupar las query strings para recibir este campo como parámetro de la consulta **DELETE**. Prosigue los siguientes pasos para el desarrollo de una función “eliminar” en nuestro archivo “consultas.js”.

- **Paso 1:** Crear una función asíncrona llamada “eliminar” que reciba un parámetro llamado “nombre”.
- **Paso 2:** Realizar una consulta SQL que tenga por interpolación el parámetro nombre para eliminar el registro de un ejercicio.



Función eliminar

Ejercicio guiado

- **Paso 3:** Incluir la función “eliminar” en el objeto que estamos exportando del archivo “consultas.js”.

```
// Paso 1
const eliminar = async (nombre) => {
  // Paso 2
  const result = await pool.query(
    `DELETE FROM ejercicios WHERE nombre = '${nombre}'`
  );
  return result;
};

// Paso 3
module.exports = { insertar, consultar, editar, eliminar };
```

/* Ruta DELETE */

Ruta DELETE

Con la función “eliminar” lista, ahora solo queda desarrollar la ruta en el servidor que reciba como query string un parámetro “nombre” en la petición emitida desde el cliente. Para el desarrollo e inclusión de esta ruta y última etapa de nuestra API REST, sigue los siguientes pasos.

- **Paso 1:** Incluir la función eliminar.
- **Paso 2:** Crear una ruta **DELETE /ejercicios**.
- **Paso 3:** Enviar como argumento de la función asíncrona “eliminar” el nombre extraído de las query strings y retornar la respuesta a la aplicación cliente.

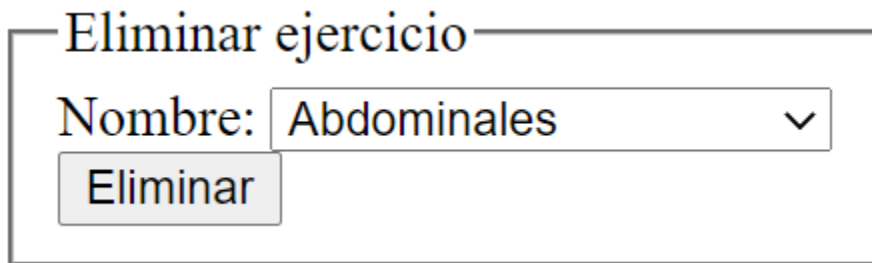
Ruta DELETE

```
// Paso 1
const { insertar, consultar, editar, eliminar } = require('./consultas')
```

```
// Paso 2
app.delete("/ejercicios", async (req, res) => {
  try {
    // Paso 3
    const { nombre } = req.query
    const respuesta = await eliminar(nombre)
    res.json(respuesta)
  } catch (error) {
    res.status(500).send("Algo salió mal :/ ...")
  }
})
```

Ruta DELETE

¿Cómo probar esta ruta? El tercer formulario HTML tiene un selector en el que encontrarás el ejercicio “Abdominales”, selecciónalo así como te muestro en la siguiente imagen.

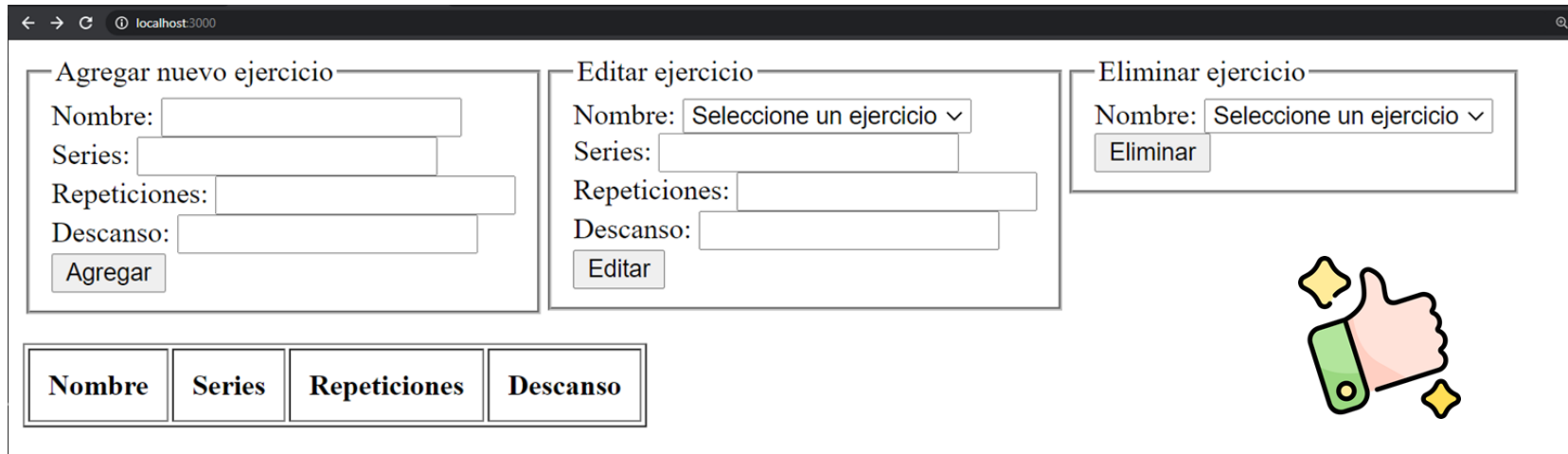


Eliminar ejercicio

Nombre:

Ruta DELETE

Ahora solo queda presionar el botón “eliminar” y tu tabla de ejercicios quedará de nuevo totalmente vacía, así como te muestro en la siguiente imagen:



The screenshot shows a web browser at localhost:3000. The interface has three main panels:

- Agregar nuevo ejercicio:** Contains input fields for 'Nombre:', 'Series:', 'Repeticiones:', and 'Descanso:', followed by an 'Agregar' button.
- Editar ejercicio:** Contains a dropdown menu for 'Nombre:' (labeled 'Seleccione un ejercicio'), and input fields for 'Series:', 'Repeticiones:', and 'Descanso:', followed by an 'Editar' button.
- Eliminar ejercicio:** Contains a dropdown menu for 'Nombre:' (labeled 'Seleccione un ejercicio') and an 'Eliminar' button.

Below these panels is a table with the following columns:

Nombre	Series	Repeticiones	Descanso
--------	--------	--------------	----------



¡Excelente!, en el ejercicio fue eliminada de la tabla y tú has logrado terminar tu primer CRUD usando una API REST desarrollada en un servidor Node ocupando PostgreSQL como motor de base de datos.

¿cómo aplicarías estos
conocimientos en un
proyecto real?





Próxima sesión...

- *Desafío evaluado - Mi repertorio*

{desafío}
latam_

*Academia de
talentos digitales*

