

**APLICACIONES DISTRIBUIDAS**

Taller # 2

***PONER EN PRÁCTICA LOS CONOCIMIENTOS ADQUIRIDOS EN LA UNIDAD 2 SOBRE***

***FUNDAMENTOS DE LAS LLAMADAS RPC***

ESTUDIANTE:

PEDRO VIDAL ECHEVERRÍA ALBURQUEQUE

AÑO LECTIVO

2025 – 2026

# Introducción

La Llamada a Procedimiento Remoto (RPC) permite que un programa ejecute funciones ubicadas en otro equipo o servidor como si fueran locales.   
Este modelo facilita la comunicación entre sistemas distribuidos, haciendo posible que diferentes aplicaciones se conecten y compartan recursos sin necesidad de conocer sus detalles internos.

En esta práctica se implementó un ejemplo funcional de RPC utilizando Node.js en el servidor y HTML + Bootstrap como cliente, simulando la interacción entre ambos mediante peticiones JSON.

# Objetivos Generales

Implementar una llamada a procedimiento remoto (RPC) utilizando Node.js y una interfaz web, comprendiendo su funcionamiento y estructura.

## 2.1 Objetivos específicos

Comprender el concepto y funcionamiento de las RPC.

Desarrollar un servidor que permita ejecutar funciones remotas.

Diseñar un cliente web que invoque dichas funciones a través de peticiones HTTP.

Simular la comunicación entre cliente y servidor utilizando JSON como formato de intercambio.

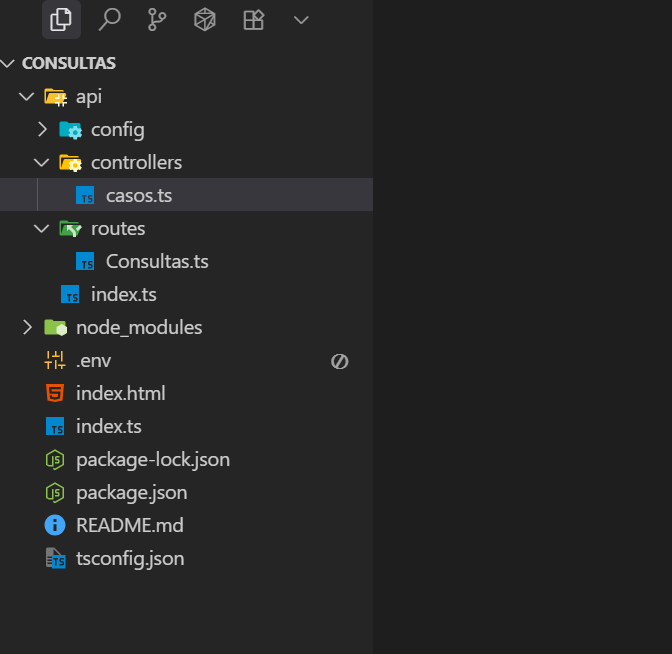
# Herramientas utilizadas

* Node.js (v18 o superior)
* Express.js – para crear el servidor RPC
* Bootstrap 5 – para el diseño de la interfaz
* Visual Studio Code – entorno de desarrollo
* Navegador web (Chrome, Edge o Firefox)

# 4. Desarrollo de la práctica

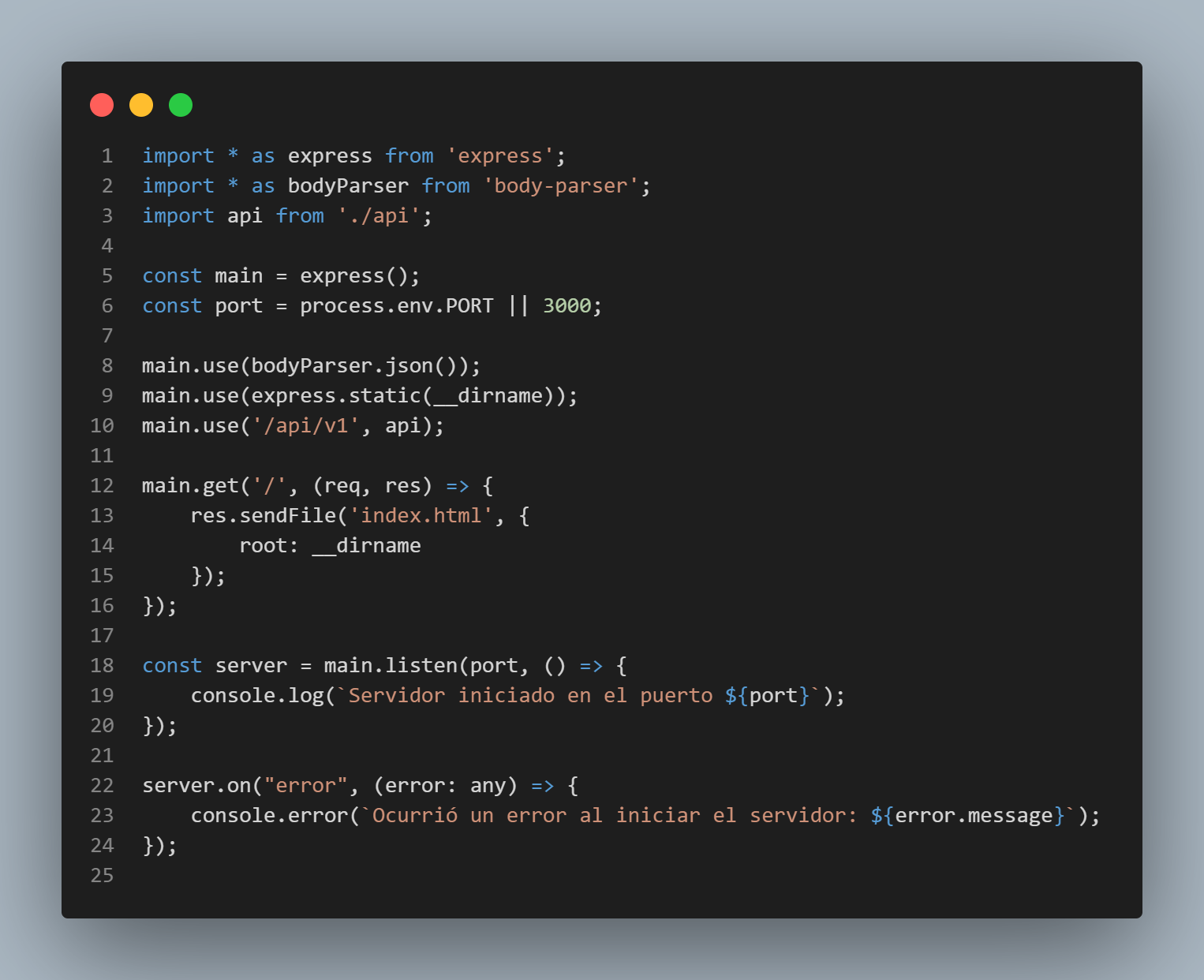
## 4.1. Estructura del proyecto

Se creó una carpeta con la siguiente estructura:



## 4.2. Configuración del servidor (Node.js)

En el archivo **server.js** se desarrolló el servidor usando Express.



Esto manejará las rutas que se encargarán de recibir las llamadas RPC desde el cliente

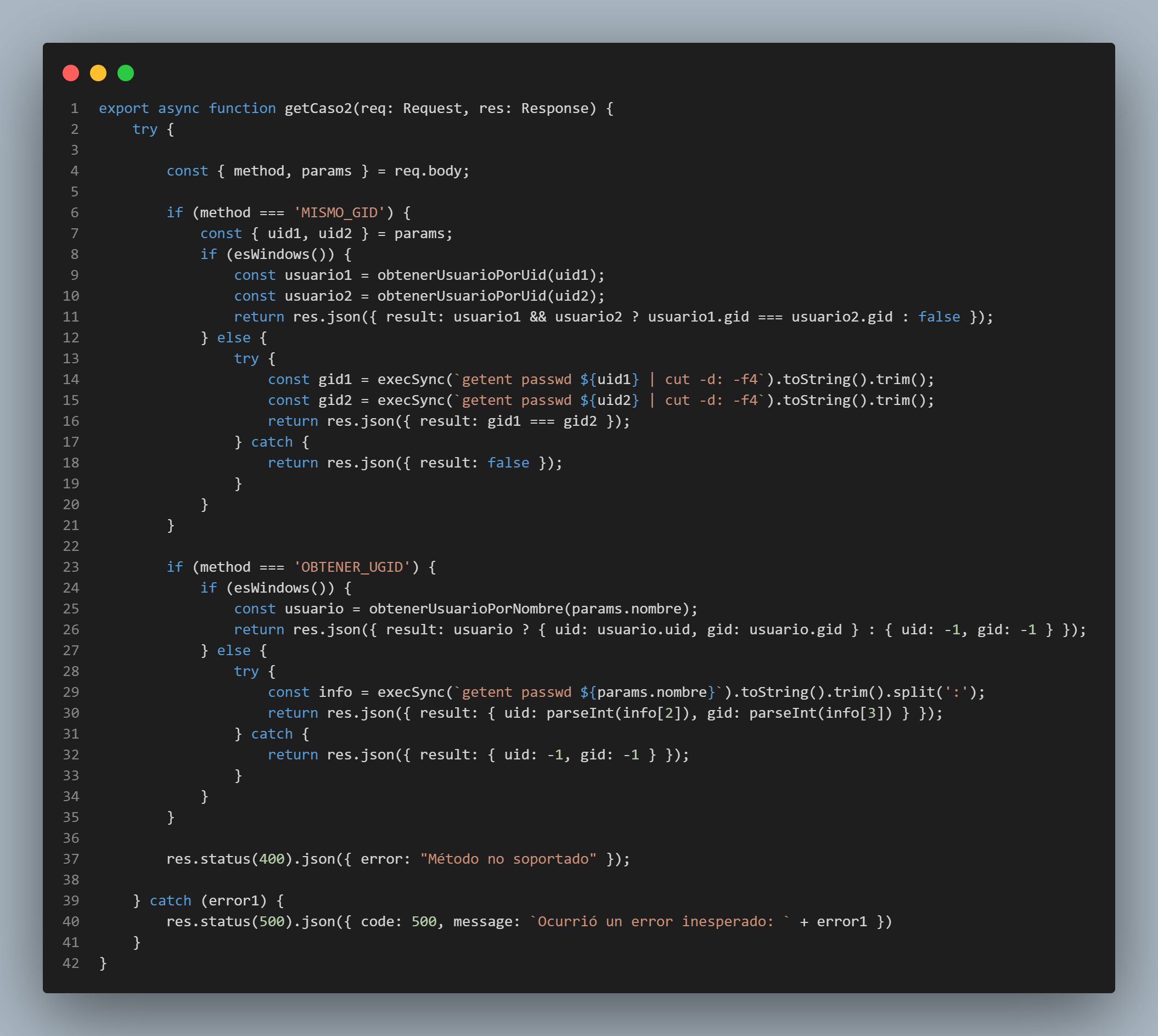


Aquí procesaremos los datos y devolver el resultado.

**Caso 1:** *Servicios con tipos de datos simples*



**Caso 2:** *Tipos de datos compuestos*



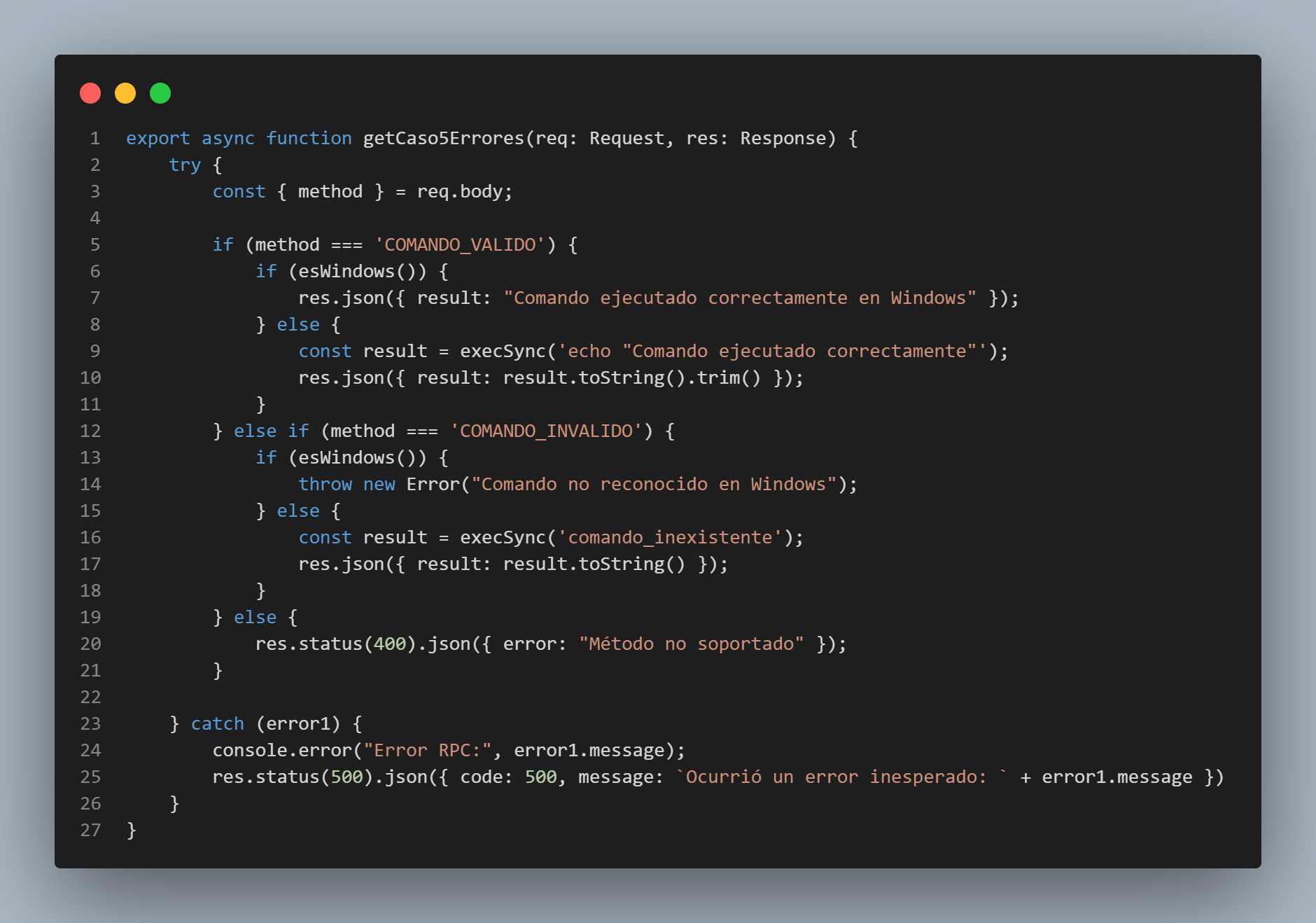
**Caso 3:** *Uso de tipo union (XDR)*



**Caso 4:** *Vectores de tamaño variable*

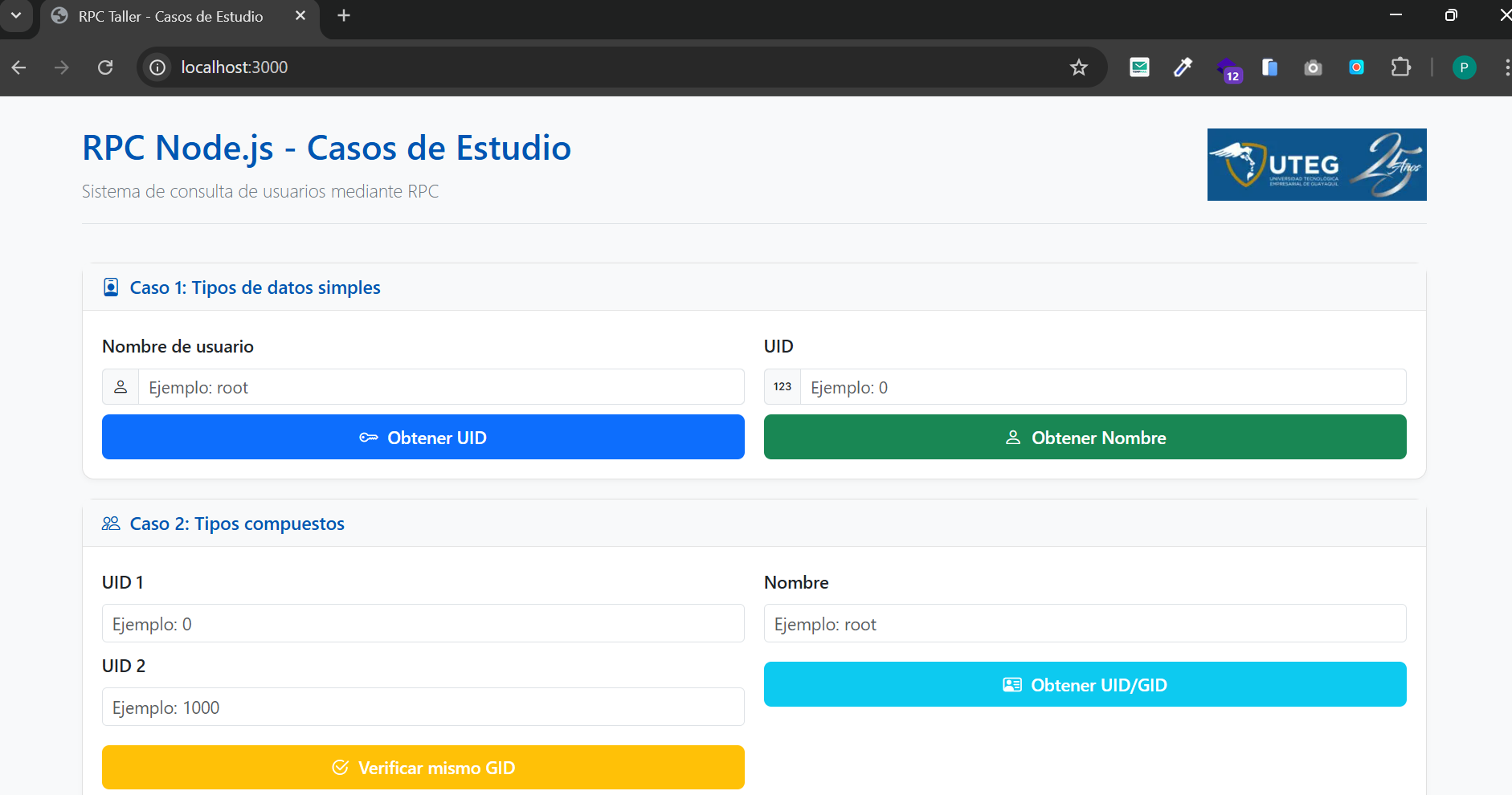


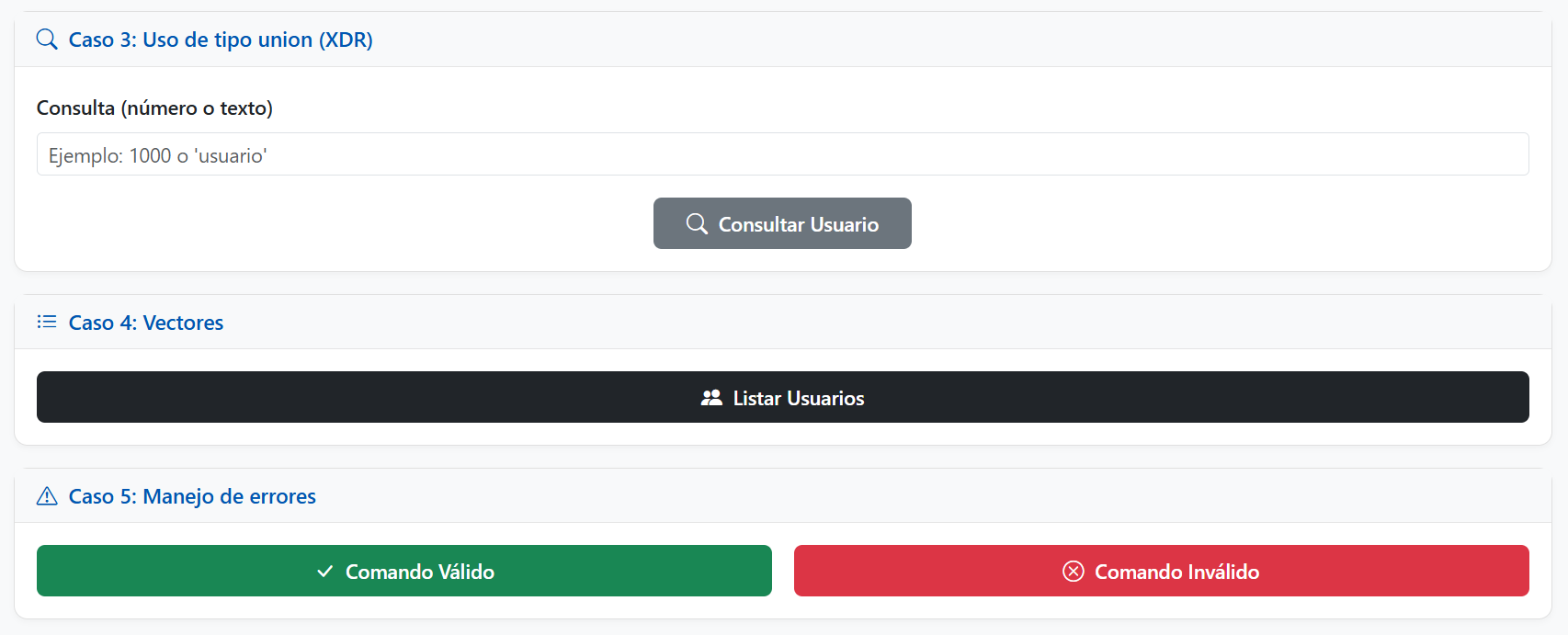
**Caso 5:** *Manejo de errores y xdr\_free*

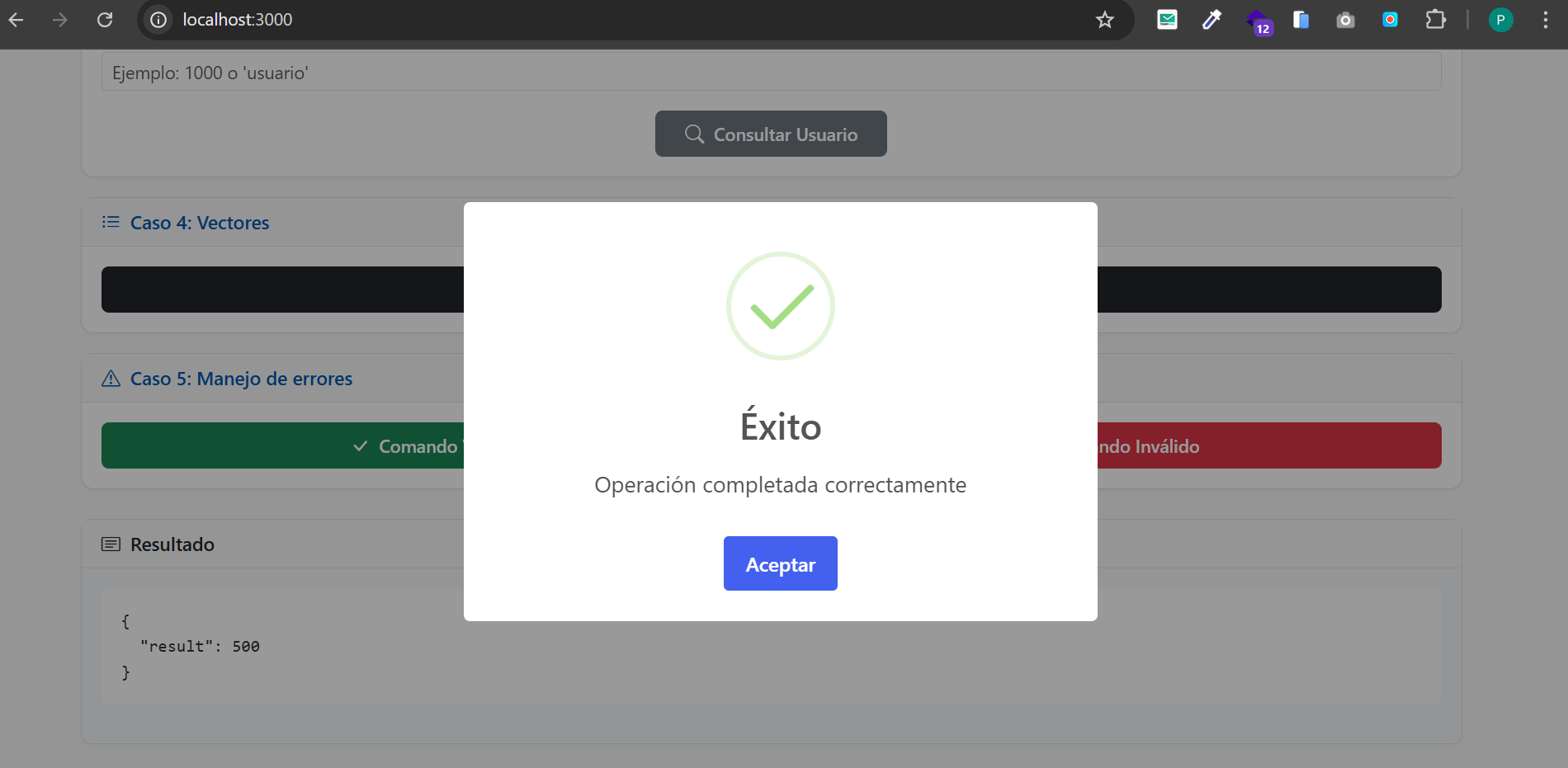


# Interfaz del cliente (HTML + Bootstrap)

Se diseñó una página web para que el usuario pueda ingresar un **nombre de usuario** o un **UID** y consultar el dato contrario mediante el servidor RPC.







Peticiones al servidor desde el cliente

***Repositorio GitHub:*** [*https://github.com/P3droVidal/Llamadas-RPC.git*](https://github.com/P3droVidal/Llamadas-RPC.git)



# Conclusiones

En esta práctica se implementó un sistema RPC (Remote Procedure Call) usando Node.js como servidor y HTML + Bootstrap como cliente.

El objetivo fue simular cómo un cliente puede ejecutar funciones que están en otro equipo o servidor, sin necesidad de conocer cómo están implementadas, solo invocándolas de forma remota.

***Lo que se realizó:***

* Crear un servidor RPC en Node.js (Express):
* Definir funciones remotas como:
* OBTENER\_UID(nombre) → devuelve el UID del usuario.
* OBTENER\_NOMBRE(uid) → devuelve el nombre del usuario.
* El servidor recibe las peticiones RPC en formato JSON y responde con los resultados.
* Diseñamos un cliente web con HTML y Bootstrap:
* Una interfaz con campos para ingresar nombre o UID.
* Botones que, al hacer clic, envían peticiones RPC al servidor usando fetch().

**Simular la comunicación RPC:**

* El cliente llama funciones que realmente se ejecutan en el servidor.
* Se manejan tipos de datos simples (string, int) y se devuelven resultados en formato JSON.

**Se aprendió los principios del modelo RPC:**

* **Abstracción:** el cliente no sabe cómo se ejecuta la función, solo la invoca.
* **Comunicación remota:** el procedimiento se ejecuta en otro sistema.
* **Serialización:** los datos viajan en formato JSON (similar a XDR en el RPC clásico de Sun).