INSTITUTO POLITÉCNICO NACIONAL

upiita-ipn

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Aplicaciones Distribuidas

PRACTICA 4: API RESTFUL

Alumno: Mulato Romero Jazmin Haydee

Profesor: Romero Sierra Noe

CODIGO:

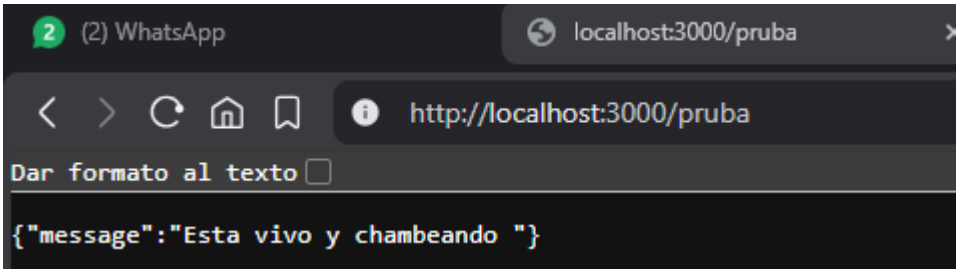
```
var express = require('express');
var app = express(); //Contenedor de Endpoints o WS Restful

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// --- Memoria para Ejercicio 3 ---
let tareas = [];

// --- EJERCICIO 1: Saludo ---
app.post("/saludo", (req, res) => {
    try {
        const { nombre } = req.body;
        if (!nombre) throw new Error("El nombre es requerido");

        res.json({ estado: "exito", mensaje: `Hola, ${nombre}` });
    } catch (error) {
        res.status(400).json({ estado: "error", mensaje: error.message });
    }
});

// --- EJERCICIO 2: Calculadora ---
app.post("/calculadora", (req, res) => {
    try {
        const { a, b, operacion } = req.body;
        let resultado;

        if (typeof a !== 'number' || typeof b !== 'number') throw new
Error("a y b deben ser números");

        switch (operacion) {
            case 'suma': resultado = a + b; break;
            case 'resta': resultado = a - b; break;
            case 'multiplicacion': resultado = a * b; break;
            case 'division':
                if (b === 0) return res.status(400).json({ estado: "error",
error: "División por cero" });
                resultado = a / b;
                break;
            default: throw new Error("Operación no válida");
        }
        res.json({ estado: "exito", resultado });
    } catch (error) {
        res.status(400).json({ estado: "error", mensaje: error.message });
```

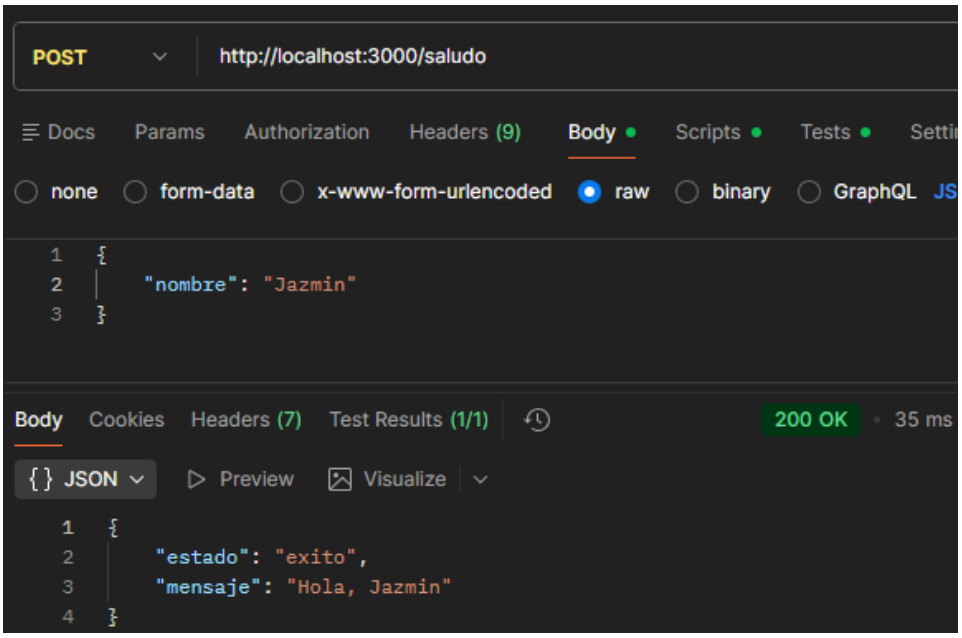
```
    }
});

// --- EJERCICIO 3: CRUD Tareas ---
app.post("/tareas", (req, res) => {
    try {
        const { id, titulo, completada } = req.body;
        tareas.push({ id, titulo, completada });
        res.json({ estado: "exito", mensaje: "Tarea creada" });
    } catch (error) {
        res.status(500).json({ estado: "error" });
    }
});

app.get("/tareas", (req, res) => {
    res.json({ estado: "exito", datos: tareas });
});

app.put("/tareas/:id", (req, res) => {
    try {
        const { id } = req.params;
        const index = tareas.findIndex(t => t.id == id);
        if (index === -1) throw new Error("Tarea no encontrada");

        tareas[index] = { ...tareas[index], ...req.body };
        res.json({ estado: "exito", mensaje: "Tarea actualizada" });
    } catch (error) {
        res.status(404).json({ estado: "error", mensaje: error.message });
    }
});

app.delete("/tareas/:id", (req, res) => {
    const { id } = req.params;
    tareas = tareas.filter(t => t.id != id);
    res.json({ estado: "exito", mensaje: "Tarea eliminada" });
});

// --- EJERCICIO 4: Validador de Password ---
app.post("/contra", (req, res) => {
    try {
        const { password } = req.body;
        let errores = [];

        if (password.length < 8) errores.push("Mínimo 8 caracteres");
        if (!/[A-Z]/.test(password)) errores.push("Al menos una mayúscula");
```

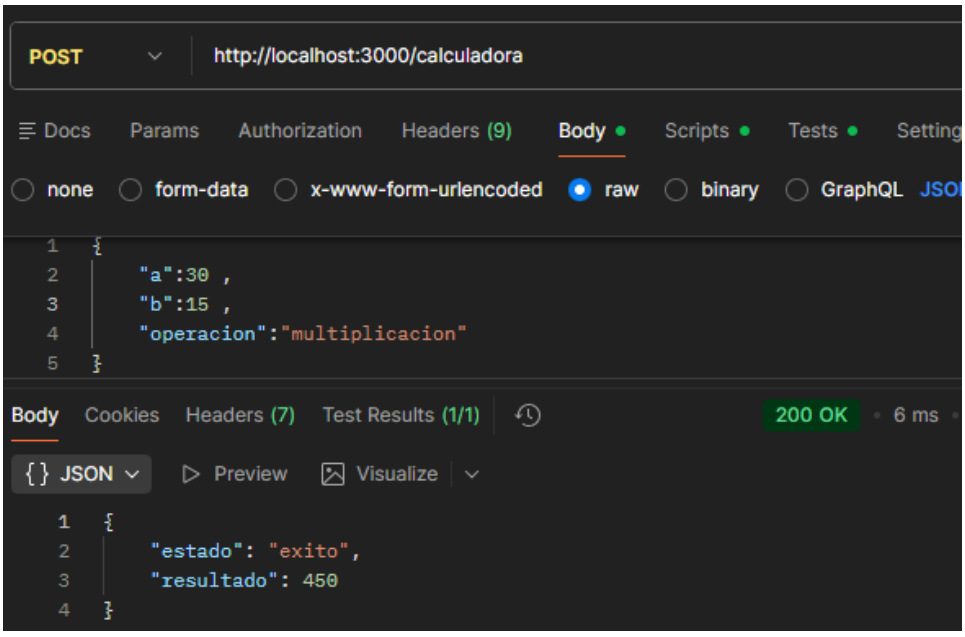
```
        if (!/[a-z]/.test(password)) errores.push("Al menos una minúscula");
        if (!/[0-9]/.test(password)) errores.push("Al menos un número");

        res.json({
            estado: "exito",
            esValida: errores.length === 0,
            errores: errores
        });
    } catch (error) {
        res.status(400).json({ estado: "error" });
    }
});

// --- EJERCICIO 5: Conversor Temperatura ---
app.post("/convtemp", (req, res) => {
    try {
        let { valor, desde, hacia } = req.body;
        let tempCelsius;

        // Normalizar a Celsius
        if (desde === 'C') tempCelsius = valor;
        else if (desde === 'F') tempCelsius = (valor - 32) * 5/9;
        else if (desde === 'K') tempCelsius = valor - 273.15;

        // Convertir a destino
        let resultado;
        if (hacia === 'C') resultado = tempCelsius;
        else if (hacia === 'F') resultado = (tempCelsius * 9/5) + 32;
        else if (hacia === 'K') resultado = tempCelsius + 273.15;

        res.json({
            estado: "exito",
            valorOriginal: valor,
            valorConvertido: Number(resultado.toFixed(2)),
            escalaOriginal: desde,
            escalaConvertida: hacia
        });
    } catch (error) {
        res.status(400).json({ estado: "error" });
    }
});

// --- EJERCICIO 6: Buscador en Array ---
app.post("/buscar", (req, res) => {
    try {
```

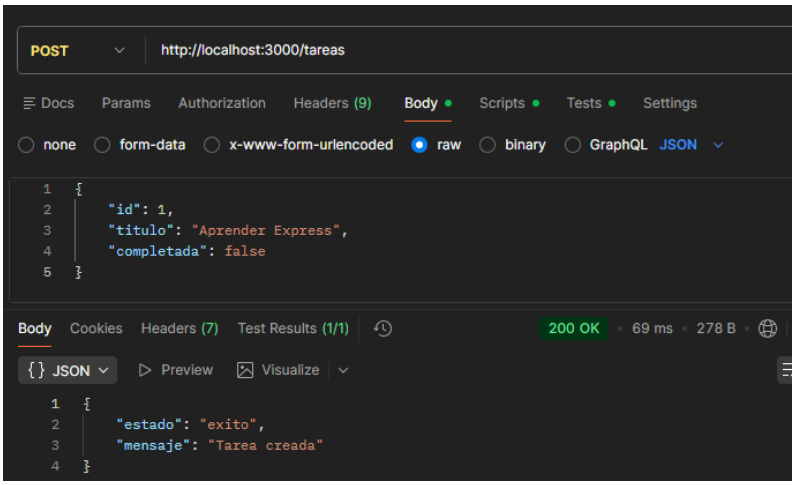
```
        const { array, elemento } = req.body;
        const indice = array.indexOf(elemento);
        res.json({
            estado: "exito",
            encontrado: indice !== -1,
            indice: indice,
            tipoElemento: typeof elemento
        });
    } catch (error) {
        res.status(400).json({ estado: "error" });
    }
});

// --- EJERCICIO 7: Contador de Palabras ---
app.post("/contar-palabras", (req, res) => {
    try {
        const { texto } = req.body;
        const palabras = texto.trim().split(/\s+/);
        const unicas = new Set(palabras.map(p => p.toLowerCase()));

        res.json({
            estado: "exito",
            totalPalabras: palabras[0] === "" ? 0 : palabras.length,
            totalCaracteres: texto.length,
            palabrasUnicas: palabras[0] === "" ? 0 : unicas.size
        });
    } catch (error) {
        res.status(400).json({ estado: "error" });
    }
});

app.listen(3000, () => {
    console.log('Servidor API corriendo en puerto 3000');
});
```

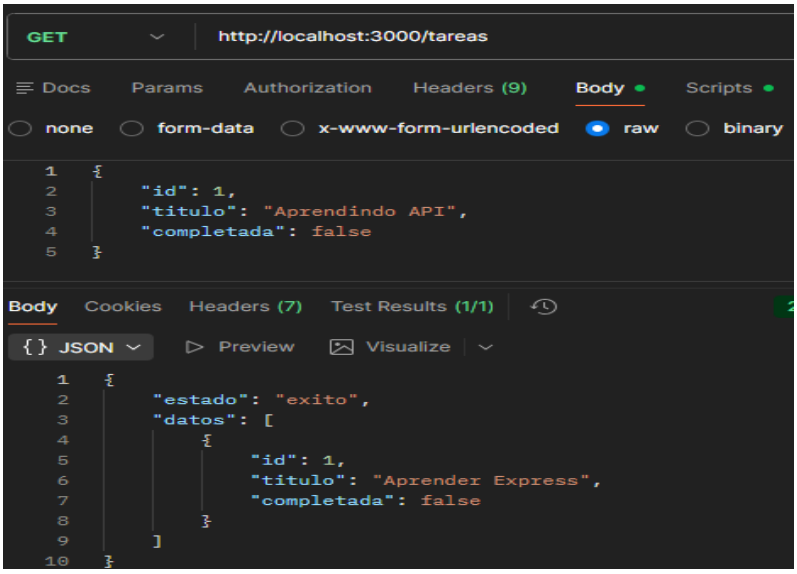
PRUEBAS:

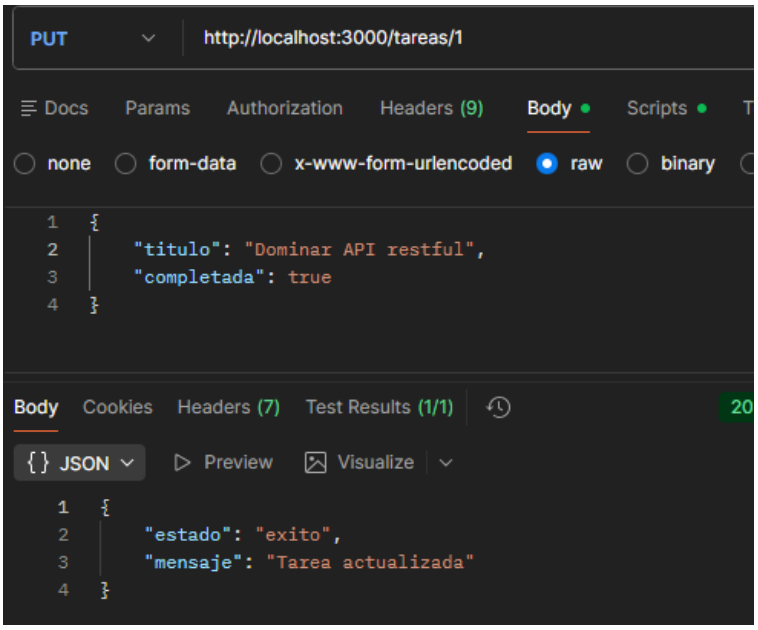

(2) WhatsApp
localhost:3000/pruba
http://localhost:3000/pruba
Dar formato al texto
{"message":"Esta vivo y chambeando "}

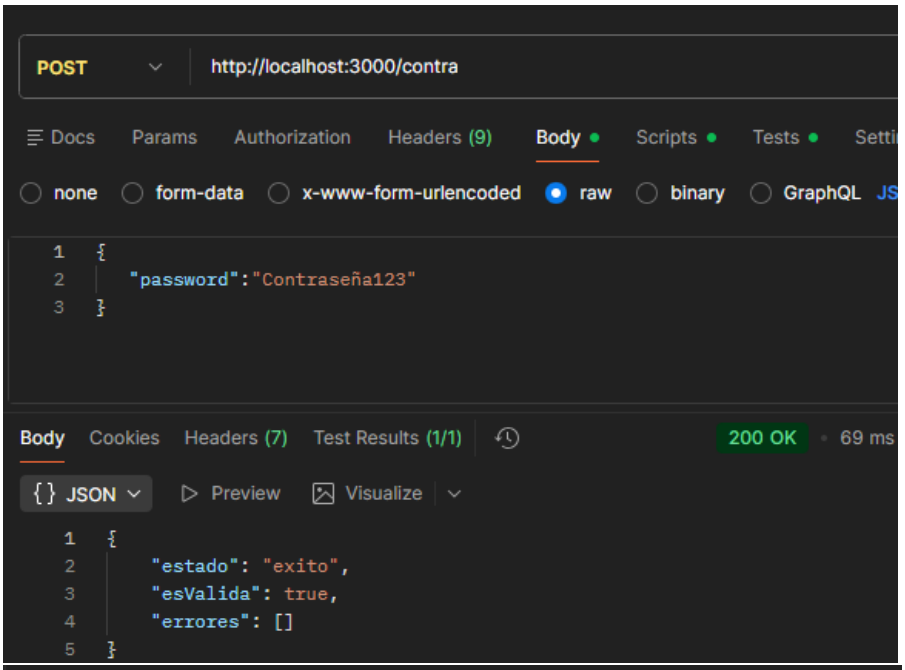

POST
http://localhost:3000/saludo
Docs Params Authorization Headers (9) Body Scripts Tests Setti
none form-data x-www-form-urlencoded raw binary GraphQL JS
{
"nombre": "Jazmin"
}
Body Cookies Headers (7) Test Results (1/1)
200 OK 35 ms
{} JSON Preview Visualize
{
"estado": "exito",
"mensaje": "Hola, Jazmin"
}

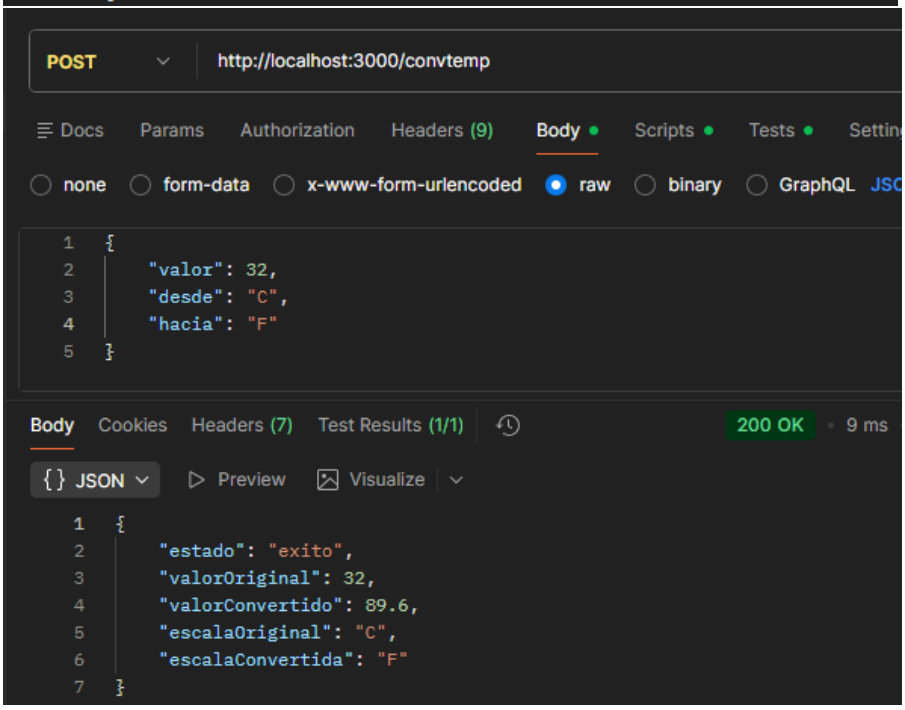

POST
http://localhost:3000/calculadora
Docs Params Authorization Headers (9) Body Scripts Tests Setting
none form-data x-www-form-urlencoded raw binary GraphQL JSO
{
"a":30 ,
"b":15 ,
"operacion":"multiplicacion"
}
Body Cookies Headers (7) Test Results (1/1)
200 OK 6 ms
{} JSON Preview Visualize
{
"estado": "exito",
"resultado": 450
}

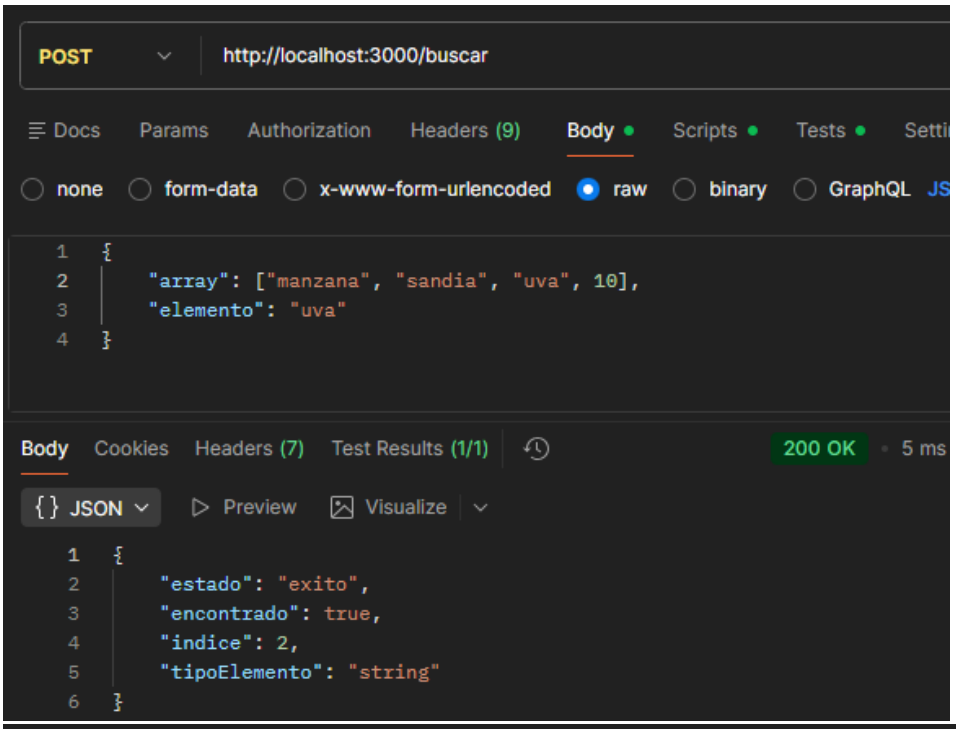

POST
http://localhost:3000/tareas
Docs Params Authorization Headers (9) Body Scripts Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"id": 1,
"titulo": "Aprender Express",
"completada": false
}
Body Cookies Headers (7) Test Results (1/1)
200 OK 69 ms 278 B
{} JSON Preview Visualize
{
"estado": "exito",
"mensaje": "Tarea creada"
}

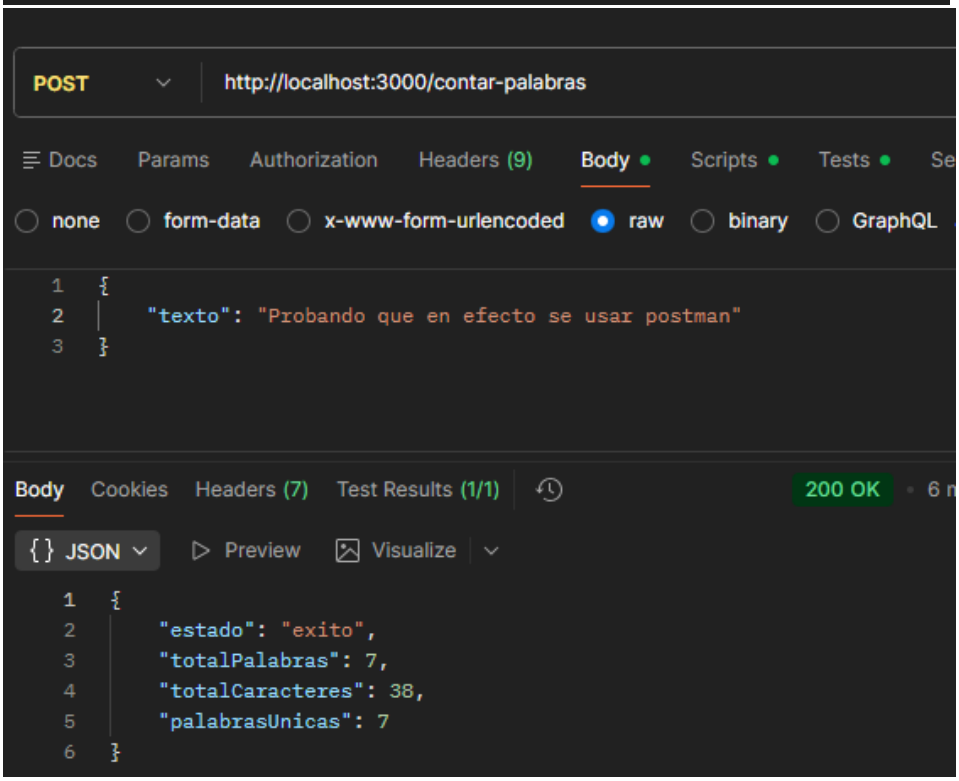

GET
http://localhost:3000/tareas
Docs Params Authorization Headers (9) Body Scripts
none form-data x-www-form-urlencoded raw binary
{
"id": 1,
"titulo": "Aprendindo API",
"completada": false
}
Body Cookies Headers (7) Test Results (1/1)
{} JSON Preview Visualize
{
"estado": "exito",
"datos": [
{
"id": 1,
"titulo": "Aprender Express",
"completada": false
}
]
}


PUT
http://localhost:3000/tareas/1
Docs Params Authorization Headers (9) Body Scripts
none form-data x-www-form-urlencoded raw binary
{
"titulo": "Dominar API restful",
"completada": true
}
Body Cookies Headers (7) Test Results (1/1)
{} JSON Preview Visualize
{
"estado": "exito",
"mensaje": "Tarea actualizada"
}


POST
http://localhost:3000/contra
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
{
"password":"Contraseña123"
}
Body
Cookies
Headers (7)
Test Results (1/1)
200 OK
69 ms
JSON
Preview
Visualize
{
"estado": "exito",
"esValida": true,
"errores": []
}


POST
http://localhost:3000/convtemp
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
{
"valor": 32,
"desde": "C",
"hacia": "F"
}
Body
Cookies
Headers (7)
Test Results (1/1)
200 OK
9 ms
JSON
Preview
Visualize
{
"estado": "exito",
"valorOriginal": 32,
"valorConvertido": 89.6,
"escalaOriginal": "C",
"escalaConvertida": "F"
}


POST
http://localhost:3000/buscar
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
{
"array": ["manzana", "sandia", "uva", 10],
"elemento": "uva"
}
Body
Cookies
Headers (7)
Test Results (1/1)
200 OK
5 ms
JSON
Preview
Visualize
{
"estado": "exito",
"encontrado": true,
"indice": 2,
"tipoElemento": "string"
}


POST
http://localhost:3000/contar-palabras
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
{
"texto": "Probando que en efecto se usar postman"
}
Body
Cookies
Headers (7)
Test Results (1/1)
200 OK
JSON
Preview
Visualize
{
"estado": "exito",
"totalPalabras": 7,
"totalCaracteres": 38,
"palabrasUnicas": 7
}