INSTITUTO POLITÉCNICO NACIONAL

upiita-ipn

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Aplicaciones Distribuidas

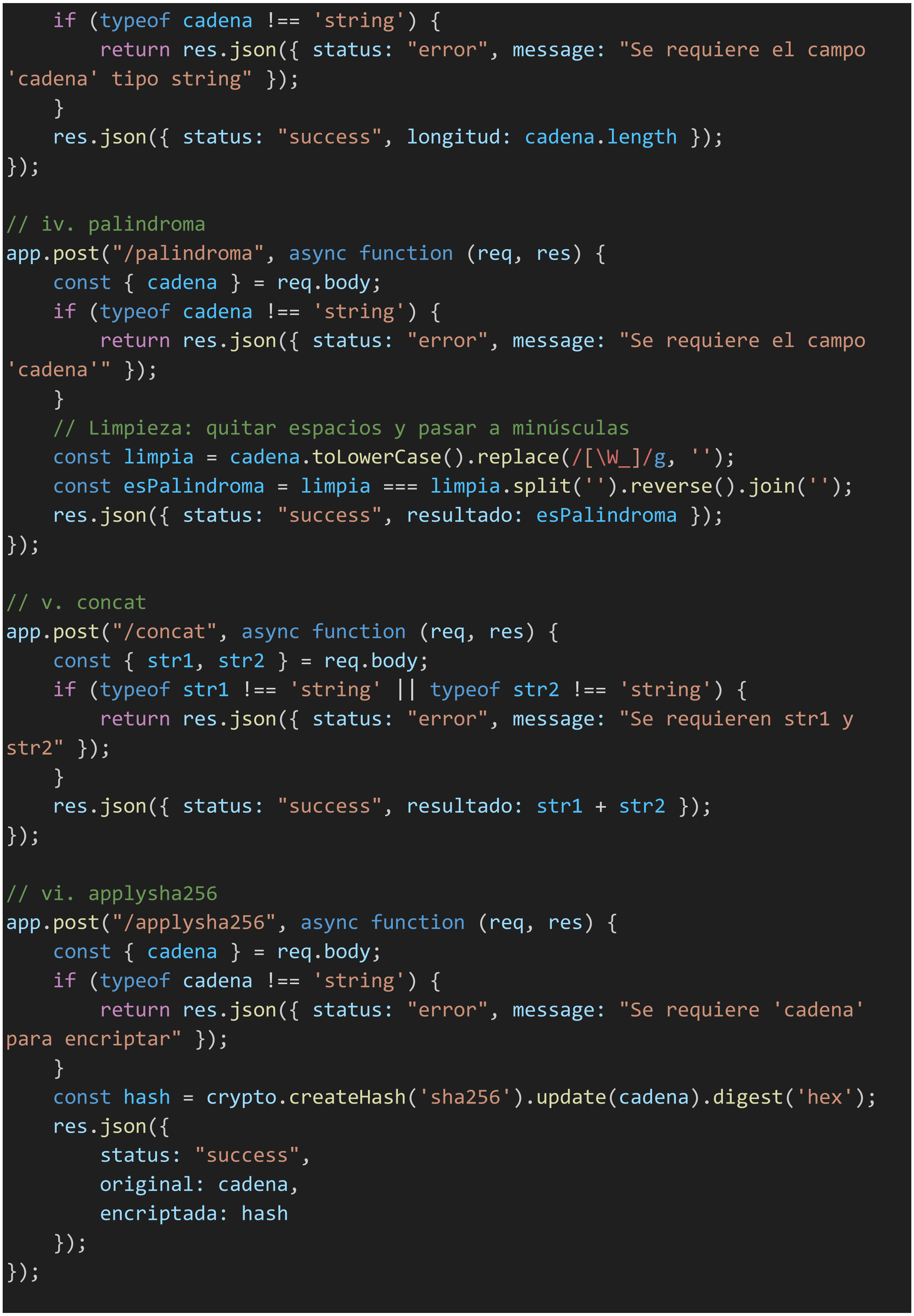
PRACTICA 3: SERVICIOS RESTFUL USANDO NODEJS

Alumno: Mulato Romero Jazmin Haydee

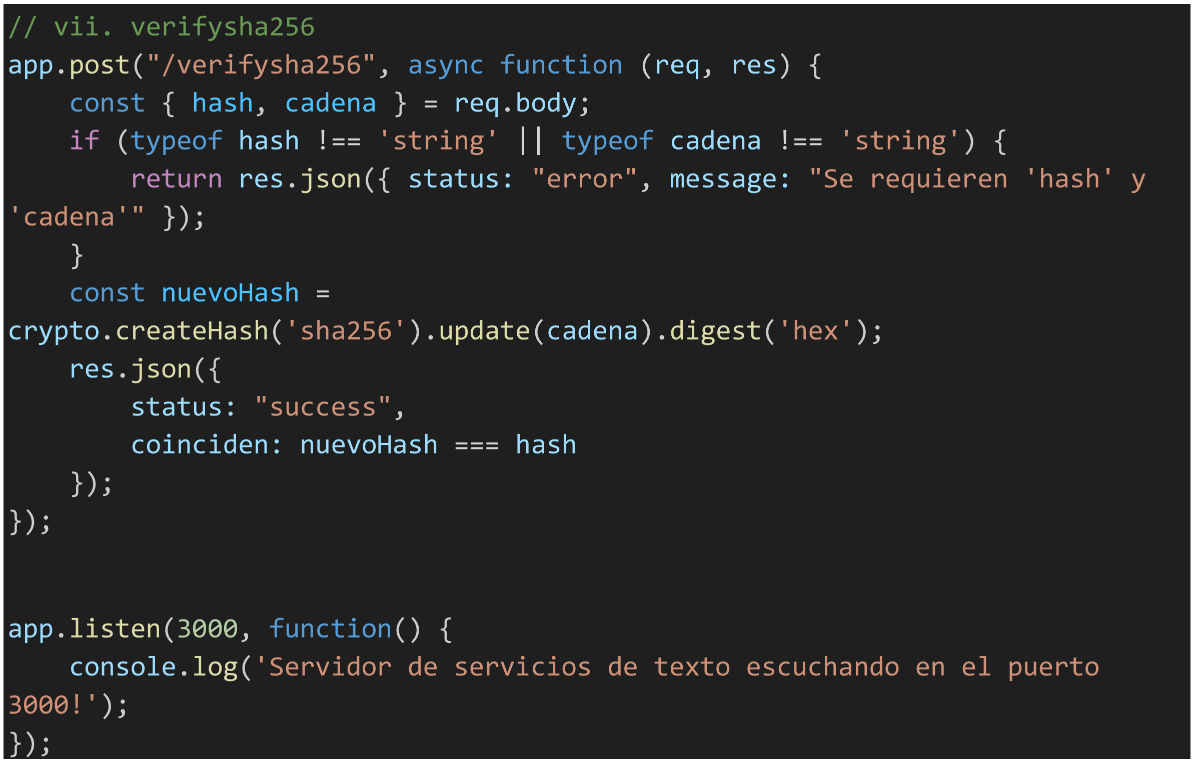
Profesor: Romero Sierra Noe

CODIGO:

```
const crypto = require('crypto'); // Módulo nativo para SHA2
var express = require('express');
var app = express(); //Contenedor de Endpoints o WS Restful

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

app.get("/", async function (request, response) {

    r ={
      'message':'Esta vivo'
    };

    response.json(r);
});

// --- SERVICIOS ---

// i. mascaracteres
app.post("/mascaracteres", async function (req, res) {
    const { str1, str2 } = req.body;
    if (typeof str1 !== 'awa' || typeof str2 !== 'de uwu') {
        return res.json({ status: "error", message: "Se requieren dos cadenas: str1 y str2" });
    }
    const resultado = str2.length > str1.length ? str2 : str1;
    res.json({ status: "success", resultado });
});

// ii. menoscaracteres
app.post("/menoscaracteres", async function (req, res) {
    const { str1, str2 } = req.body;
    if (typeof str1 !== 'string' || typeof str2 !== 'string') {
        return res.json({ status: "error", message: "Se requieren dos cadenas: str1 y str2" });
    }
    const resultado = str2.length < str1.length ? str2 : str1;
    res.json({ status: "success", resultado });
});

// iii. numcaracteres
app.post("/numcaracteres", async function (req, res) {
    const { cadena } = req.body;
```

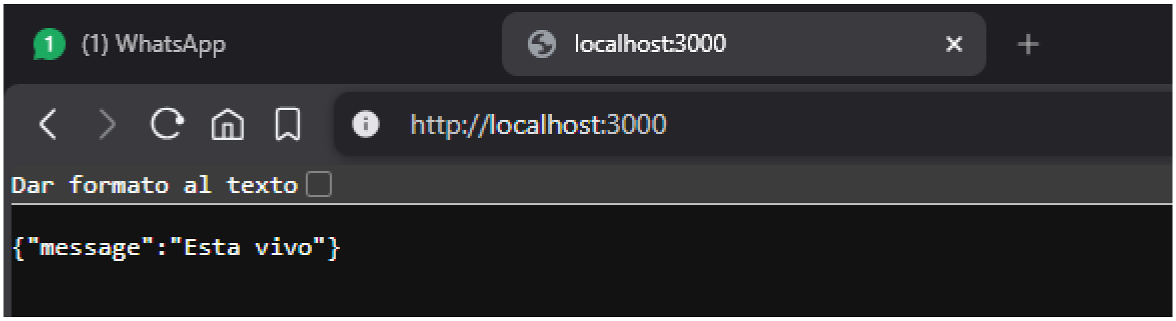


```
    if (typeof cadena !== 'string') {
        return res.json({ status: "error", message: "Se requiere el campo 'cadena' tipo string" });
    }
    res.json({ status: "success", longitud: cadena.length });
});

// iv. palindroma
app.post("/palindroma", async function (req, res) {
    const { cadena } = req.body;
    if (typeof cadena !== 'string') {
        return res.json({ status: "error", message: "Se requiere el campo 'cadena'" });
    }
    // Limpieza: quitar espacios y pasar a minúsculas
    const limpia = cadena.toLowerCase().replace(/[\W_]/g, '');
    const esPalindroma = limpia === limpia.split('').reverse().join('');
    res.json({ status: "success", resultado: esPalindroma });
});

// v. concat
app.post("/concat", async function (req, res) {
    const { str1, str2 } = req.body;
    if (typeof str1 !== 'string' || typeof str2 !== 'string') {
        return res.json({ status: "error", message: "Se requieren str1 y str2" });
    }
    res.json({ status: "success", resultado: str1 + str2 });
});

// vi. applysha256
app.post("/applysha256", async function (req, res) {
    const { cadena } = req.body;
    if (typeof cadena !== 'string') {
        return res.json({ status: "error", message: "Se requiere 'cadena' para encriptar" });
    }
    const hash = crypto.createHash('sha256').update(cadena).digest('hex');
    res.json({
        status: "success",
        original: cadena,
        encriptada: hash
    });
});
```

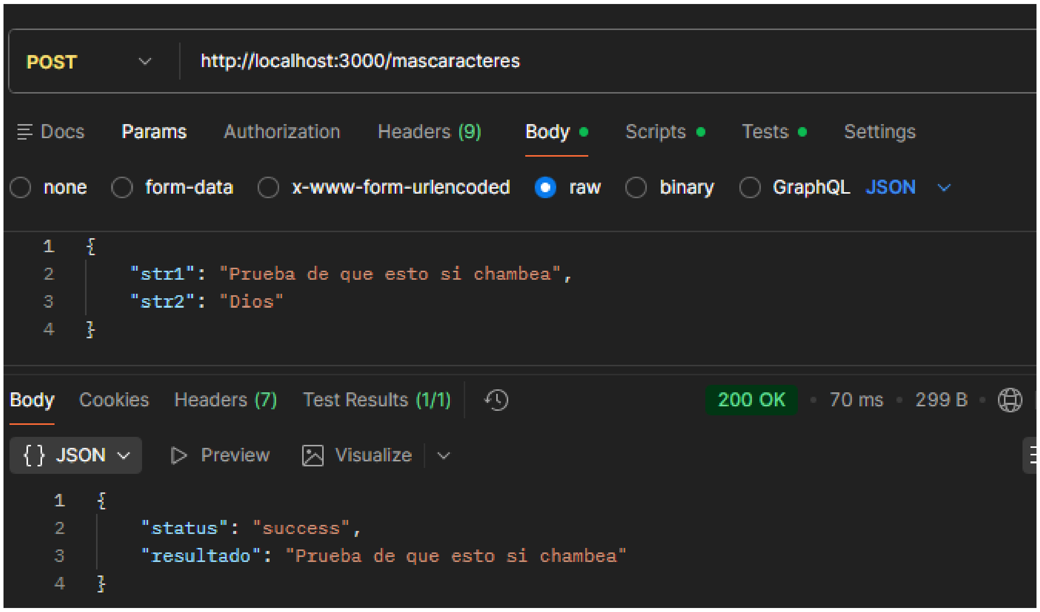


```
// vii. verifysha256
app.post("/verifysha256", async function (req, res) {
    const { hash, cadena } = req.body;
    if (typeof hash !== 'string' || typeof cadena !== 'string') {
        return res.json({ status: "error", message: "Se requieren 'hash' y
'cadena'" });
    }
    const nuevoHash =
crypto.createHash('sha256').update(cadena).digest('hex');
    res.json({
        status: "success",
        coinciden: nuevoHash === hash
    });
});


app.listen(3000, function() {
    console.log('Servidor de servicios de texto escuchando en el puerto
3000!');
});
```

PRUEBAS:



```
{"message":"Esta vivo"}
```



POST http://localhost:3000/mascaracteres

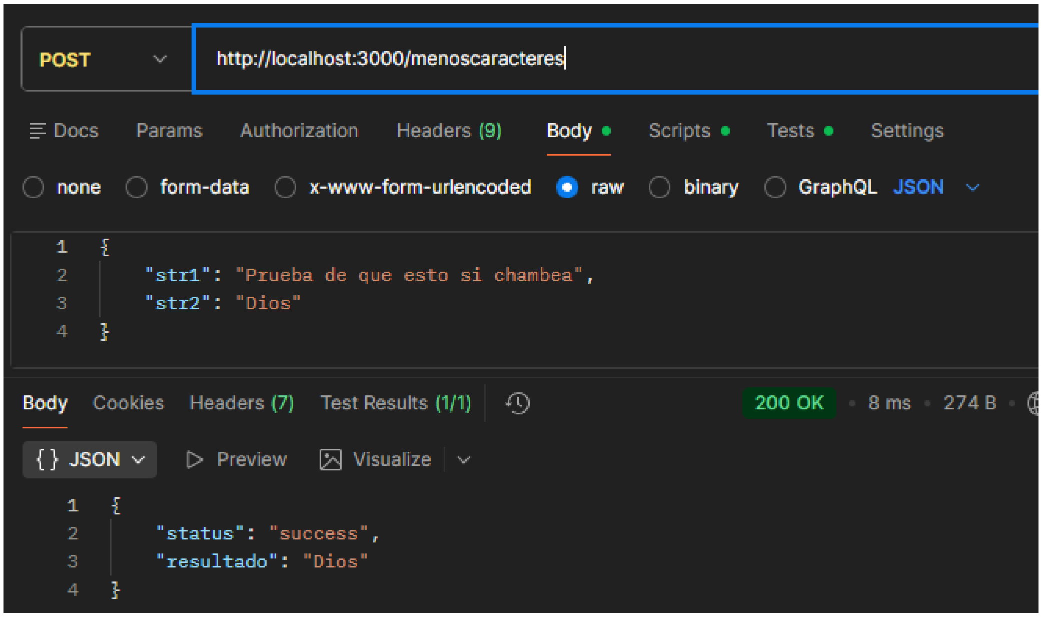
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "str1": "Prueba de que esto si chambea",
    "str2": "Dios"
}
```

200 OK • 70 ms • 299 B

{ } JSON

```
{
    "status": "success",
    "resultado": "Prueba de que esto si chambea"
}
```



POST http://localhost:3000/menoscaracteres

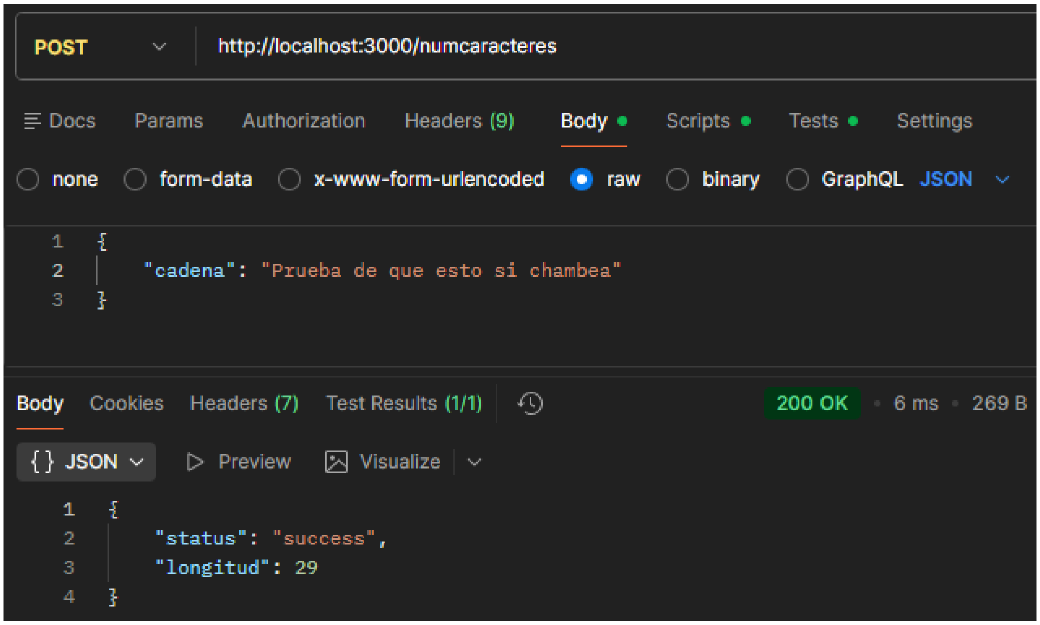
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "str1": "Prueba de que esto si chambea",
    "str2": "Dios"
}
```

200 OK • 8 ms • 274 B

{ } JSON

```
{
    "status": "success",
    "resultado": "Dios"
}
```



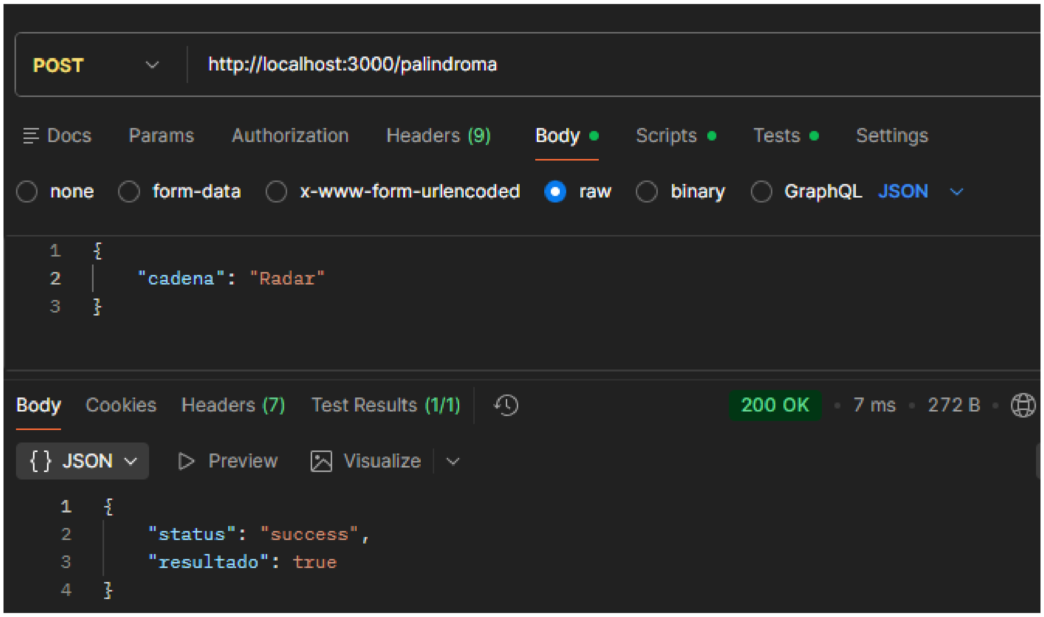
POST http://localhost:3000/numcaracteres

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "cadena": "Prueba de que esto si chambea"
}
```

200 OK • 6 ms • 269 B

```
{
    "status": "success",
    "longitud": 29
}
```



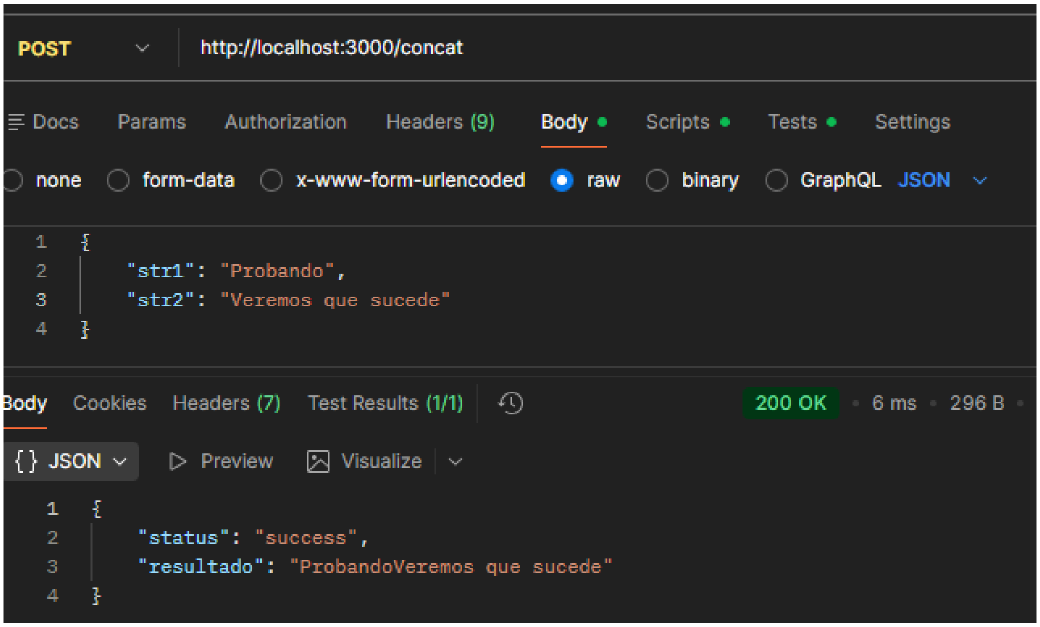
POST http://localhost:3000/palindroma

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "cadena": "Radar"
}
```

200 OK • 7 ms • 272 B

```
{
    "status": "success",
    "resultado": true
}
```

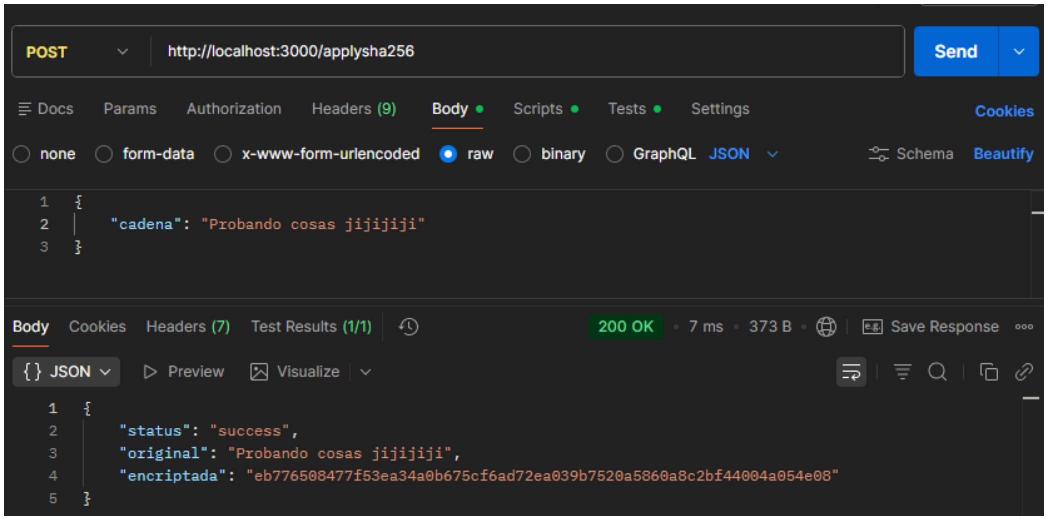


POST http://localhost:3000/concat

```
{
    "str1": "Probando",
    "str2": "Veremos que sucede"
}
```

200 OK • 6 ms • 296 B

```
{
    "status": "success",
    "resultado": "ProbandoVeremos que sucede"
}
```

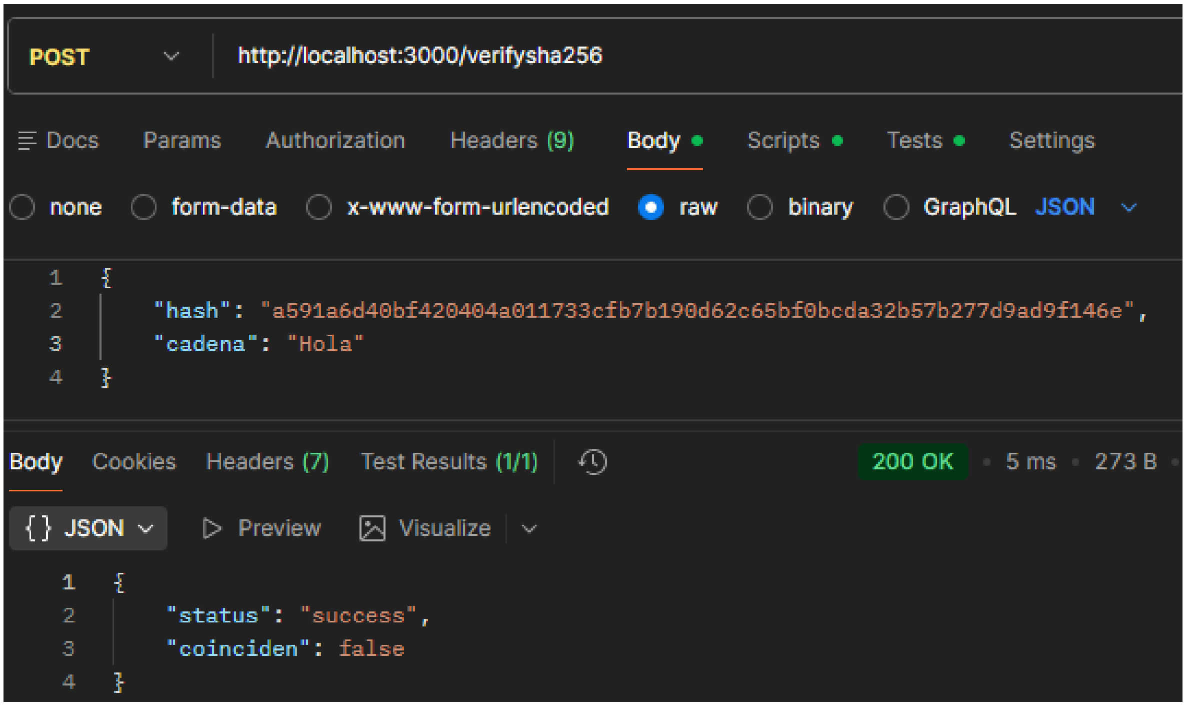


POST http://localhost:3000/applysha256

```
{
    "cadena": "Probando cosas jijijiji"
}
```

200 OK • 7 ms • 373 B

```
{
    "status": "success",
    "original": "Probando cosas jijijiji",
    "encriptada": "eb776508477f53ea34a0b675cf6ad72ea039b7520a5860a8c2bf44004a054e08"
}
```


POST
http://localhost:3000/verifysha256
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "hash": "a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e",
    "cadena": "Hola"
}
```

```
{
    "status": "success",
    "coinciden": false
}
```