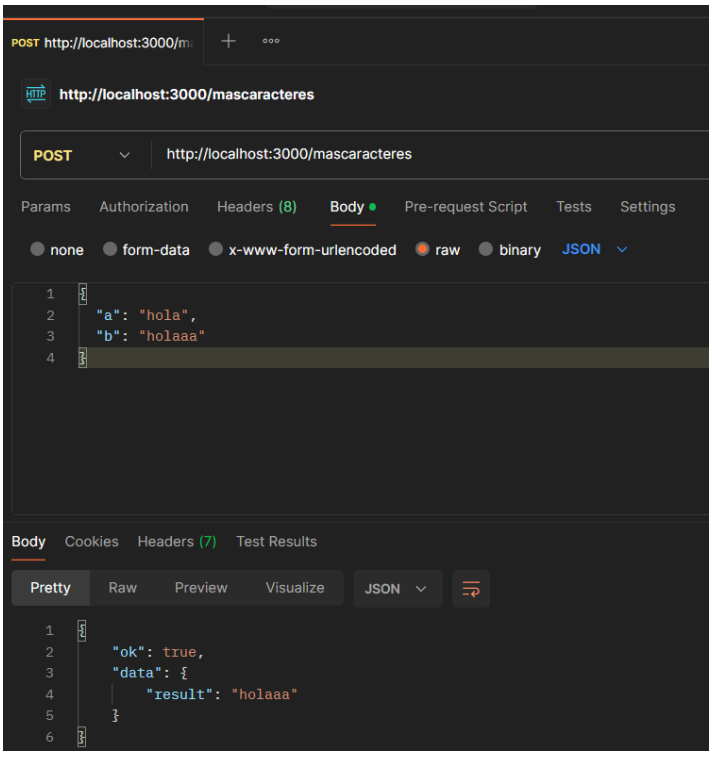
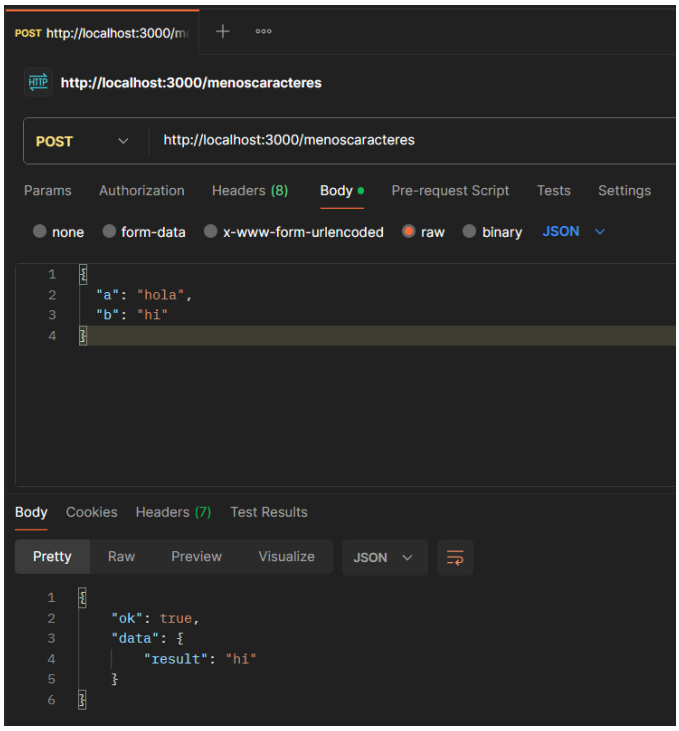
Práctica de Aplicaciones Distribuidas

Nombre: Cruz Laparra Jose Julian

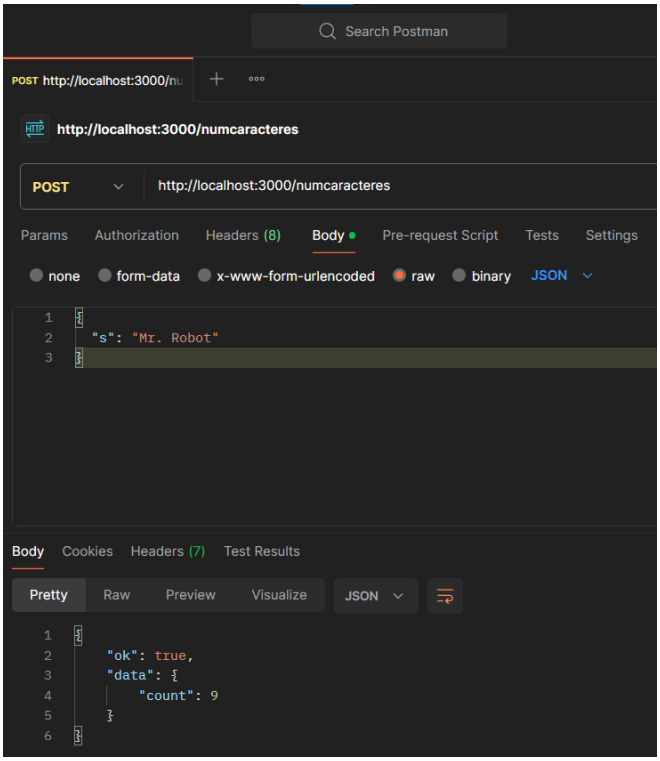
1. Crear una serie de servicios web en la plataforma de RepLit, usando un servidor NodeJS que realicen lo siguiente:
    a. Requerimientos
        i. Todos los servicios reciben sus parámetros en una estructura JSON
        ii. Todos los servicios responden en una estructura JSON
        iii. Todos los servicios deben validar los parámetros y el resultado de la operación, e incluir un atributo en el JSON de respuesta que indique el resultado o un campo de error indicando el problema
    b. Tareas a implementar
        i. mascaracteres: recibe dos cadenas y regresa la que tenga más caracteres. Si son iguales, regresa la del primer parámetro



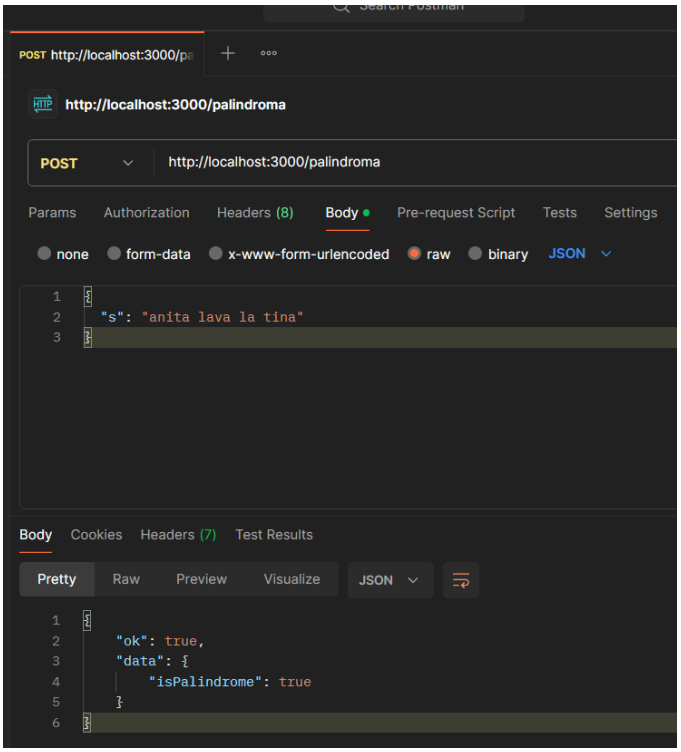
ii. menoscaracteres: recibe dos cadenas y regresa la que tenga menos caracteres. Si son iguales, regresa la del primer parámetro



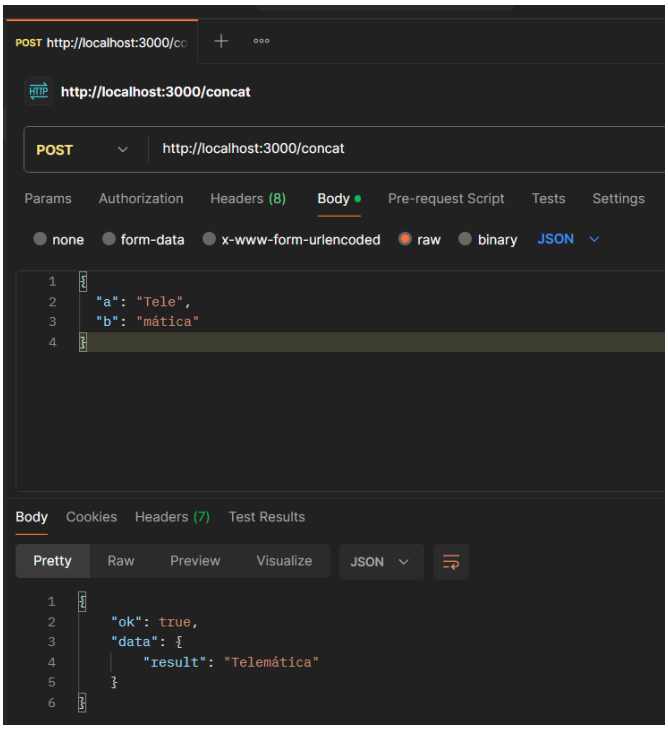
iii. numcaracteres: recibe una cadena y regresa el número de caracteres que la cadena tiene



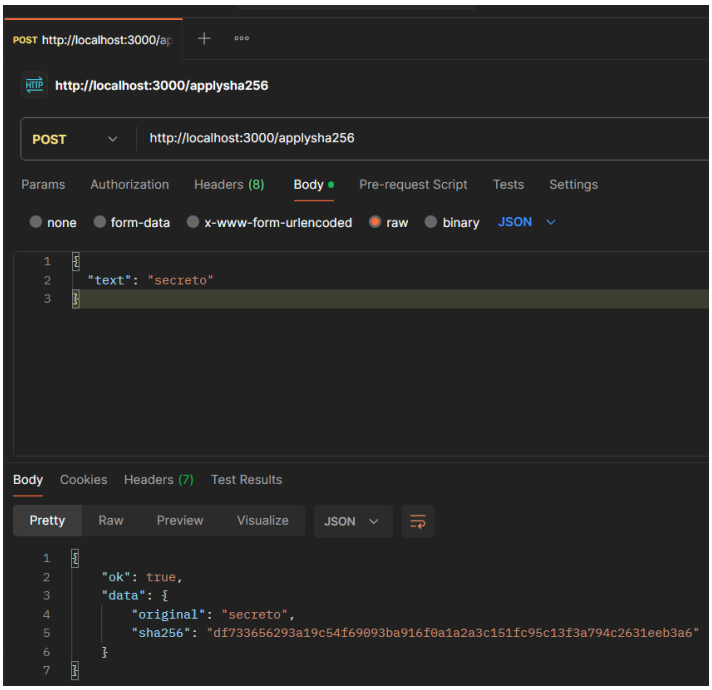
iv. palindroma: recibe una cadena y regresa true si la cadena es una palindroma, y false en caso contrario



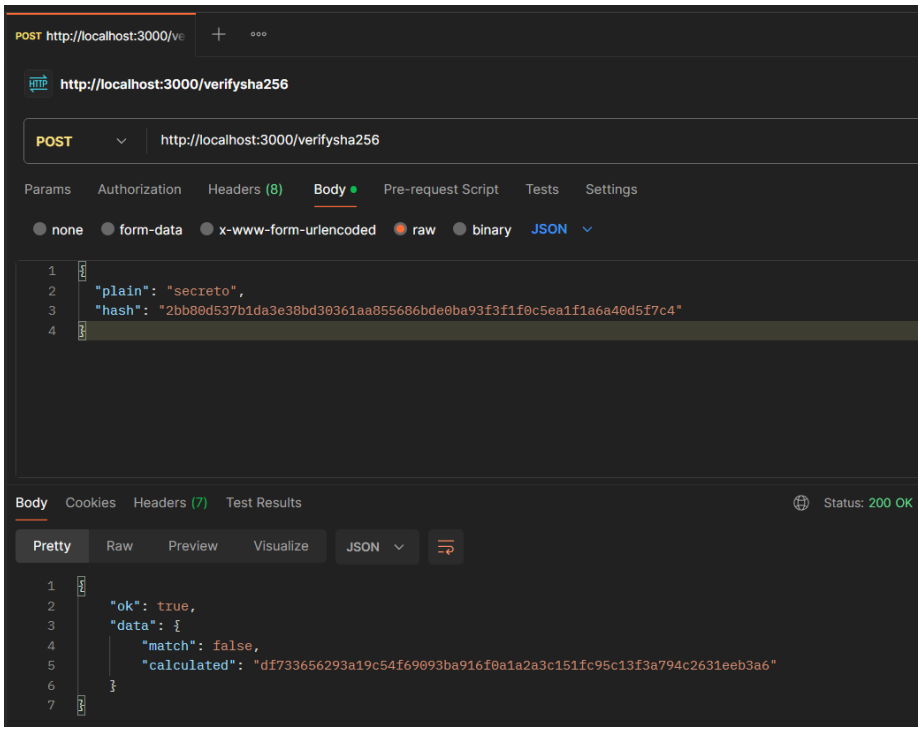
v. concat: recibe dos cadenas y regresa la concatenación iniciando con el primer parámetro



vi. applysha256: recibe una cadena, le aplica una encriptación SHA256 y regresa como resultado la cadena original y la encriptada



vii. verifysha256: recibe una cadena encriptada, una cadena normal, a la cadena normal le aplica SHA256, la compara con la cadena encriptada y regresa true si coinciden, y false en otro caso



Anexo código en VS Code

Preámbulo:

- crear un index.js
- npm init -y
- npm install express

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

const app = express();
app.use(express.json());

// helpers
const isStr = v => typeof v === 'string';
const sha256 = t => crypto.createHash('sha256').update(t, 'utf8').digest('hex');
const ok = data => ({ ok: true, data });
const fail = msg => ({ ok: false, error: msg });

// i) mascaracteres
app.post('/mascaracteres', (req, res) => {
  const { a, b } = req.body || {};
  if (!isStr(a) || !isStr(b)) return res.status(400).json(fail('a y b deben ser string'));
  if (a.length === b.length) return res.json(ok({ result: a }));
  res.json(ok({ result: a.length > b.length ? a : b }));
});

// ii) menoscaracteres
app.post('/menoscaracteres', (req, res) => {
  const { a, b } = req.body || {};
  if (!isStr(a) || !isStr(b)) return res.status(400).json(fail('a y b deben ser string'));
  if (a.length === b.length) return res.json(ok({ result: a }));
  res.json(ok({ result: a.length < b.length ? a : b }));
});

// iii) numcaracteres
app.post('/numcaracteres', (req, res) => {
  const { s } = req.body || {};
  if (!isStr(s)) return res.status(400).json(fail('s debe ser string'));
  res.json(ok({ count: s.length }));
});

// iv) palindroma  (ignora espacios, mayúsculas y acentos)
app.post('/palindroma', (req, res) => {
  const { s } = req.body || {};
```

```
  if (!isStr(s)) return res.status(400).json(fail('s debe ser string'));
  const norm = s.toLowerCase()
    .normalize('NFD').replace(/[\u0300-\u036f]/g, '') // quita acentos
    .replace(/\s+/g, ''); // quita espacios
  const isPalindrome = norm === [...norm].reverse().join('');
  res.json(ok({ isPalindrome }));
});

// v) concat
app.post('/concat', (req, res) => {
  const { a, b } = req.body || {};
  if (!isStr(a) || !isStr(b)) return res.status(400).json(fail('a y b deben ser string'));
  res.json(ok({ result: a + b }));
});

// vi) applysha256
app.post('/applysha256', (req, res) => {
  const { text } = req.body || {};
  if (!isStr(text)) return res.status(400).json(fail('text debe ser string'));
  res.json(ok({ original: text, sha256: sha256(text) }));
});

// vii) verifysha256  (acepta "encrypted" o "hash")
app.post('/verifysha256', (req, res) => {
  const { plain, encrypted, hash } = req.body || {};
  const target = encrypted || hash;
  if (!isStr(plain) || !isStr(target)) {
    return res.status(400).json(fail('plain y encrypted/hash deben ser string'));
  }
  const calculated = sha256(plain);
  res.json(ok({ match: calculated === target.toLowerCase(), calculated }));
});

// salud y 404
app.get('/', (_, res) => res.json(ok({ message: 'API lista' })));
app.use((req, res) => res.status(404).json(fail('Ruta no encontrada')));

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Servidor en http://localhost:${PORT}`));
```