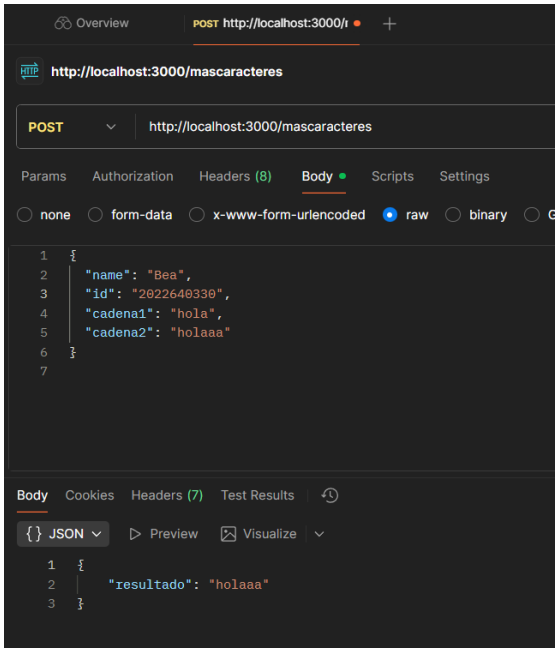
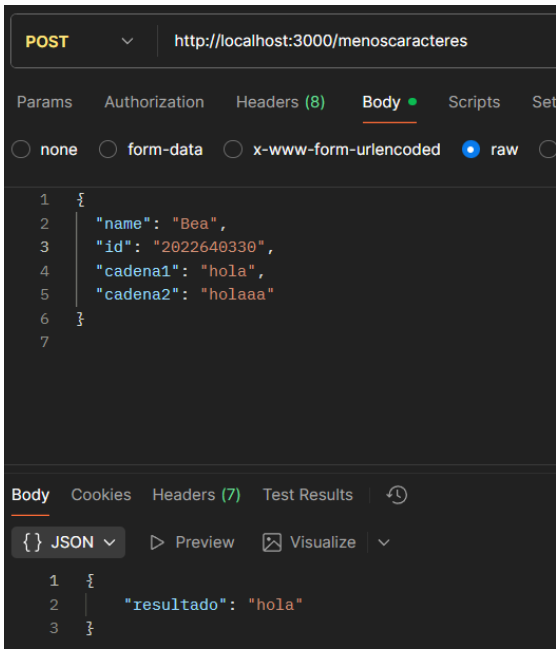
Práctica de Aplicaciones Distribuidas

Beatriz Cruz Carrillo

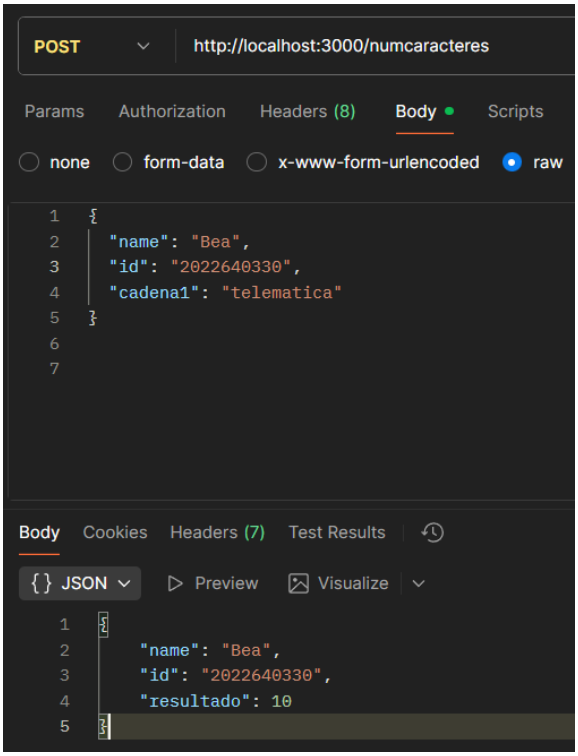
1. Crear una serie de servicios web en la plataforma de RepLit, usando un servidor NodeJS que realicen lo siguiente:
    a. Requerimientos
        i. Todos los servicios reciben sus parámetros en una estructura JSON
        ii. Todos los servicios responden en una estructura JSON
        iii. Todos los 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



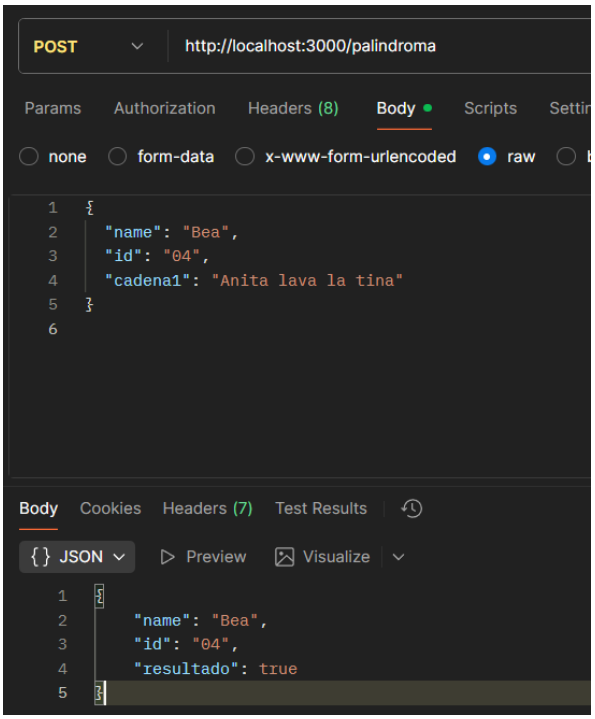
        ii. menoscaracteres: recibe dos cadenas y regresa la que tenga menos caracteres. Si son iguales, regresa la del primer parámetro



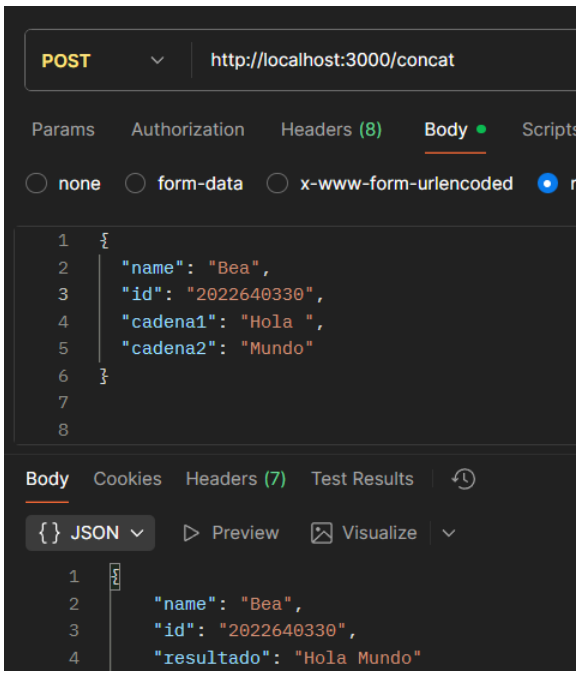
iii. numcaracteres: recibe una cadena y regresa el número de caracteres que la cadena tiene



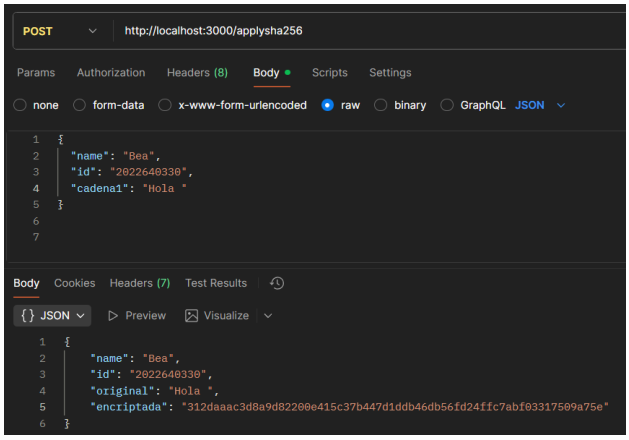
iv. palindroma: recibe una cadena y regresa true si la cadena es una palindroma, y false en caso contrario



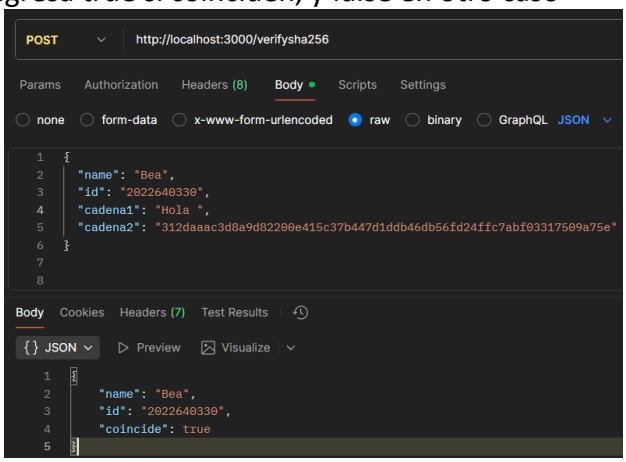
v. concat: recibe dos cadenas y regresa la concatenación iniciando con el primer parámetro



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



Creación el entorno para trabajar con PostMan:

```
C:\Users\bea_c\OneDrive\Documentos\26-1\aplicaciones_distrbuidas\practica_nodejs>npm init -y
Wrote to C:\Users\bea_c\OneDrive\Documentos\26-1\aplicaciones_distrbuidas\practica_nodejs\package.json:

{
  "name": "practica_nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



C:\Users\bea_c\OneDrive\Documentos\26-1\aplicaciones_distrbuidas\practica_nodejs>npm install express

added 68 packages, and audited 69 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\bea_c\OneDrive\Documentos\26-1\aplicaciones_distrbuidas\practica_nodejs>echo > index.js
```

Código utilizado en el index.js:

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

app.use(express.json());

function sha256(text) {
  return crypto.createHash('sha256').update(text).digest('hex');
}

// Función de validación general
function validarCampos(req, res, campos) {
  for (let campo of campos) {
    if (!req.body[campo]) {
      return res.json({
        name: req.body.name || null,
        id: req.body.id || null,
        error: `Falta el parámetro: ${campo}`,
      });
    }
  }
  return null;
}

// --- mascaracteres ---
app.post('/mascaracteres', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1', 'cadena2']);
  if (error) return;

  const { name, id, cadena1, cadena2 } = req.body;
  const resultado = cadena1.length >= cadena2.length ? cadena1 : cadena2;

  res.json({ name, id, resultado });
});

// --- menoscaracteres ---
app.post('/menoscaracteres', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1', 'cadena2']);
  if (error) return;

  const { name, id, cadena1, cadena2 } = req.body;
  const resultado = cadena1.length <= cadena2.length ? cadena1 : cadena2;

  res.json({ name, id, resultado });
});

// --- numcaracteres ---
app.post('/numcaracteres', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1']);
  if (error) return;

  const { name, id, cadena1 } = req.body;
  res.json({ name, id, resultado: cadena1.length });
});

// --- palindroma ---
app.post('/palindroma', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1']);
```

```
  if (error) return;

  const { name, id, cadena1 } = req.body;
  const limpia = cadena1.toLowerCase().replace(/[^a-z0-9]/g, '');
  const esPalindroma = limpia === limpia.split('').reverse().join('');

  res.json({ name, id, resultado: esPalindroma });
});

// --- concat ---
app.post('/concat', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1', 'cadena2']);
  if (error) return;

  const { name, id, cadena1, cadena2 } = req.body;
  res.json({ name, id, resultado: cadena1 + cadena2 });
});

// --- applysha256 ---
app.post('/applysha256', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1']);
  if (error) return;

  const { name, id, cadena1 } = req.body;
  const encriptada = sha256(cadena1);

  res.json({ name, id, original: cadena1, encriptada });
});

// --- verifysha256 ---
app.post('/verifysha256', (req, res) => {
  const error = validarCampos(req, res, ['name', 'id', 'cadena1', 'cadena2']);
  if (error) return;

  const { name, id, cadena1, cadena2 } = req.body; // cadena1 = texto normal, cadena2 = hash
  const coincide = sha256(cadena1) === cadena2;

  res.json({ name, id, coincide });
});

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