Juarez Anguiano Mario Alexis 2022640543

Práctica de Aplicaciones Distribuidas

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

Código:

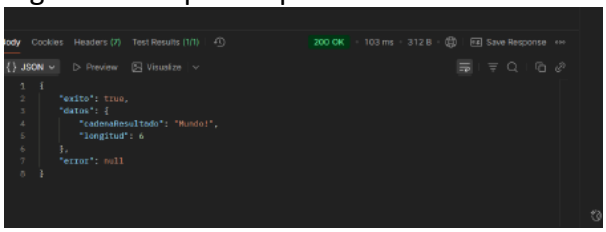
```
const express = require('express');
const crypto = require('crypto');

const app = express();
const PORT = 3000;

// Middleware para parsear JSON
app.use(express.json());

// Función auxiliar para crear respuestas
function crearRespuesta(exito, datos = null, error = null) {
  return {
    exito: exito,
    datos: datos,
    error: error
  };
}

// Función para aplicar SHA256
function aplicarSHA256(cadena) {
  return crypto.createHash('sha256').update(cadena).digest('hex');
}

// 1. Servicio: mascaracteres
app.post('/mascaracteres', (req, res) => {
  try {
    const { cadena1, cadena2 } = req.body;

    // Validación
    if (typeof cadena1 !== 'string' || typeof cadena2 !== 'string') {
      return res.json(crearRespuesta(false, null, 'Ambos parámetros deben ser cadenas'));
    }

    const resultado = cadena1.length >= cadena2.length ? cadena1 : cadena2;

    res.json(crearRespuesta(true, {
      cadenaResultado: resultado,
      longitud: resultado.length
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// 2. Servicio: menoscaracteres
app.post('/menoscaracteres', (req, res) => {
  try {
    const { cadena1, cadena2 } = req.body;

    // Validación
    if (typeof cadena1 !== 'string' || typeof cadena2 !== 'string') {
      return res.json(crearRespuesta(false, null, 'Ambos parámetros deben ser cadenas'));
    }
```

```
    const resultado = cadena1.length <= cadena2.length ? cadena1 : cadena2;

    res.json(crearRespuesta(true, {
      cadenaResultado: resultado,
      longitud: resultado.length
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// 3. Servicio: numcaracteres
app.post('/numcaracteres', (req, res) => {
  try {
    const { cadena } = req.body;

    // Validación
    if (typeof cadena !== 'string') {
      return res.json(crearRespuesta(false, null, 'El parámetro debe ser una cadena'));
    }

    res.json(crearRespuesta(true, {
      cadena: cadena,
      numeroCaracteres: cadena.length
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// 4. Servicio: palindroma
app.post('/palindroma', (req, res) => {
  try {
    const { cadena } = req.body;

    // Validación
    if (typeof cadena !== 'string') {
      return res.json(crearRespuesta(false, null, 'El parámetro debe ser una cadena'));
    }

    // Normalizar: eliminar espacios y convertir a minúsculas
    const cadenaLimpia = cadena.toLowerCase().replace(/\s/g, '');
    const cadenaInvertida = cadenaLimpia.split('').reverse().join('');
    const esPalindroma = cadenaLimpia === cadenaInvertida;

    res.json(crearRespuesta(true, {
      cadena: cadena,
      esPalindroma: esPalindroma
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// 5. Servicio: concat
app.post('/concat', (req, res) => {
```

```
  try {
    const { cadena1, cadena2 } = req.body;

    // Validación
    if (typeof cadena1 !== 'string' || typeof cadena2 !== 'string') {
      return res.json(crearRespuesta(false, null, 'Ambos parámetros deben ser cadenas'));
    }

    const resultado = cadena1 + cadena2;

    res.json(crearRespuesta(true, {
      cadena1: cadena1,
      cadena2: cadena2,
      concatenacion: resultado
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// 6. Servicio: applysha256
app.post('/applysha256', (req, res) => {
  try {
    const { cadena } = req.body;

    // Validación
    if (typeof cadena !== 'string') {
      return res.json(crearRespuesta(false, null, 'El parámetro debe ser una cadena'));
    }

    const cadenaEncriptada = aplicarSHA256(cadena);

    res.json(crearRespuesta(true, {
      cadenaOriginal: cadena,
      cadenaEncriptada: cadenaEncriptada
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// 7. Servicio: verifysha256
app.post('/verifysha256', (req, res) => {
  try {
    const { cadenaEncriptada, cadenaNormal } = req.body;

    // Validación
    if (typeof cadenaEncriptada !== 'string' || typeof cadenaNormal !== 'string') {
      return res.json(crearRespuesta(false, null, 'Ambos parámetros deben ser cadenas'));
    }

    // Aplicar SHA256 a la cadena normal
    const hashCalculado = aplicarSHA256(cadenaNormal);

    // Comparar con la cadena encriptada
    const coinciden = hashCalculado === cadenaEncriptada.toLowerCase();
```

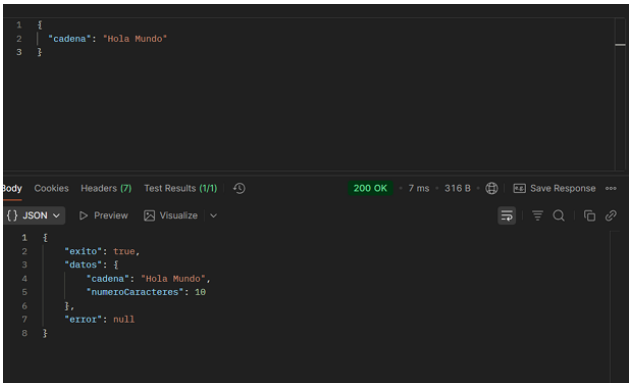
```
    res.json(crearRespuesta(true, {
      cadenaNormal: cadenaNormal,
      cadenaEncriptada: cadenaEncriptada,
      hashCalculado: hashCalculado,
      coinciden: coinciden
    }));
  } catch (error) {
    res.json(crearRespuesta(false, null, 'Error en el servidor: ' + error.message));
  }
});

// Ruta raíz para verificar que el servidor está funcionando
app.get('/', (req, res) => {
  res.json({
    mensaje: 'Servidor de Servicios Web Distribuidos',
    servicios: [
      'POST /mascaracteres',
      'POST /menoscaracteres',
      'POST /numcaracteres',
      'POST /palindroma',
      'POST /concat',
      'POST /applysha256',
      'POST /verifysha256'
    ]
  });
});

// Iniciar el servidor
app.listen(PORT, () => {
  console.log(`Servidor ejecutándose en puerto ${PORT}`);
  console.log(`URL: http://localhost:${PORT}`);
});
```

i. mascaracteres: recibe dos cadenas y regresa la que tenga más caracteres. Si son iguales, regresa la del primer parámetro

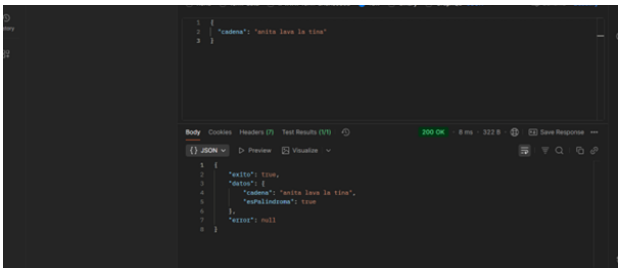


ii. menoscaracteres: recibe dos cadenas y regresa la que tenga menos caracteres. Si son iguales, regresa la del primer parámetro

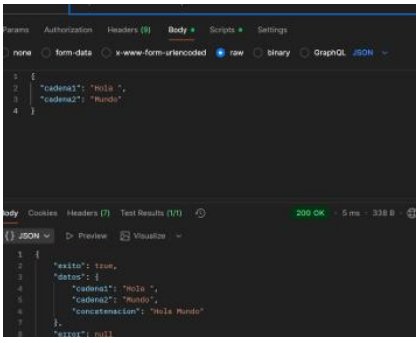
iii. numcaracteres: recibe una cadena y regresa el número de caracteres que la cadena tiene



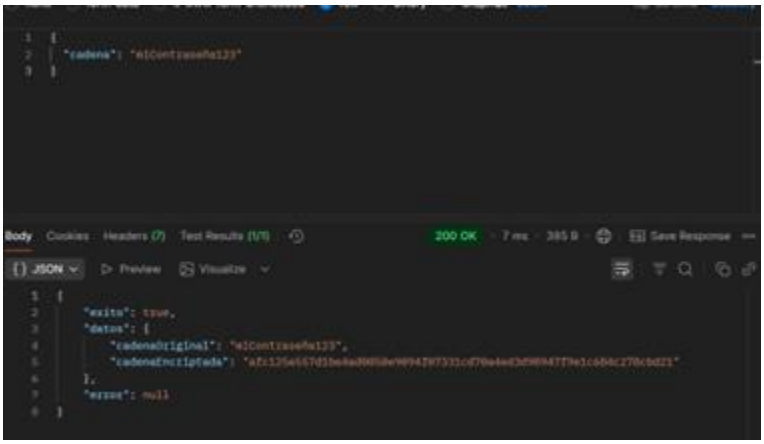
iv. palindroma: recibe una cadena y regresa true si la cadena es una palindroma, y false en caso contrario



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