Práctica de Aplicaciones Distribuidas

Contreras Jiménez Mariana Montserrat

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 index.js.

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

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

const PORT = process.env.PORT || 3000;

// ======================
// SERVICIOS WEB
// ======================

// mascaracteres
app.post("/mascaracteres", (req, res) => {
  const { cad1, cad2 } = req.body;

  if (!cad1 || !cad2) {
    return res.json({ ok: false, error: "Faltan parámetros" });
  }
  const resultado = cad1.length >= cad2.length ? cad1 : cad2;
  res.json({ ok: true, resultado });
});

// menoscaracteres
app.post("/menoscaracteres", (req, res) => {
  const { cad1, cad2 } = req.body;

  if (!cad1 || !cad2) {
    return res.json({ ok: false, error: "Faltan parámetros" });
  }

  const resultado = cad1.length <= cad2.length ? cad1 : cad2;
  res.json({ ok: true, resultado });
});

// numcaracteres
app.post("/numcaracteres", (req, res) => {
  const { cadena } = req.body;

  if (!cadena) {
    return res.json({ ok: false, error: "No se envió cadena" });
  }

  res.json({ ok: true, resultado: cadena.length });
});

// palindroma
app.post("/palindroma", (req, res) => {
  const { cadena } = req.body;

  if (!cadena) {
    return res.json({ ok: false, error: "No se envió cadena" });
  }

  const limpia = cadena.toLowerCase().replace(/[^a-z0-9]/g, "");
  const invertida = limpia.split("").reverse().join("");

  res.json({ ok: true, resultado: limpia === invertida });
});

// concat
app.post("/concat", (req, res) => {
  const { cad1, cad2 } = req.body;

  if (!cad1 || !cad2) {
    return res.json({ ok: false, error: "Faltan parámetros" });
  }

  res.json({ ok: true, resultado: cad1 + cad2 });
});

// applysha256
app.post("/applysha256", (req, res) => {
  const { cadena } = req.body;

  if (!cadena) {
    return res.json({ ok: false, error: "No se envió cadena" });
  }

  const hash = crypto.createHash("sha256").update(cadena).digest("hex");

  res.json({
    ok: true,
    original: cadena,
    sha256: hash,
  });
});

// verifysha256
app.post("/verifysha256", (req, res) => {
  const { original, encriptada } = req.body;

  if (!original || !encriptada) {
    return res.json({ ok: false, error: "Faltan parámetros" });
  }

  const hash = crypto.createHash("sha256").update(original).digest("hex");

  res.json({
    ok: true,
    resultado: hash === encriptada,
  });
});

// =======================

app.listen(PORT, () => {
  console.log("Servidor activo en puerto " + PORT);
});
```

Código package.json.

```
{
  "name": "rest-express",
  "version": "1.0.0",
  "license": "MIT",
  "scripts": {
    "start": "node server/index.js"
  },
  "dependencies": {
    "@hookform/resolvers": "^3.10.0",
    "@jridgewell/trace-mapping": "^0.3.25",
    "@radix-ui/react-accordion": "^1.2.4",
    "@radix-ui/react-alert-dialog": "^1.1.7",
    "@radix-ui/react-aspect-ratio": "^1.1.3",
    "@radix-ui/react-avatar": "^1.1.4",
    "@radix-ui/react-checkbox": "^1.1.5",
    "@radix-ui/react-collapsible": "^1.1.4",
    "@radix-ui/react-context-menu": "^2.2.7",
    "@radix-ui/react-dialog": "^1.1.7",
    "@radix-ui/react-dropdown-menu": "^2.1.7",
    "@radix-ui/react-hover-card": "^1.1.7",
    "@radix-ui/react-label": "^2.1.3",
    "@radix-ui/react-menubar": "^1.1.7",
    "@radix-ui/react-navigation-menu": "^1.2.6",
    "@radix-ui/react-popover": "^1.1.7",
    "@radix-ui/react-progress": "^1.1.3",
    "@radix-ui/react-radio-group": "^1.2.4",
    "@radix-ui/react-scroll-area": "^1.2.4",
    "@radix-ui/react-select": "^2.1.7",
    "@radix-ui/react-separator": "^1.1.3",
    "@radix-ui/react-slider": "^1.2.4",
    "@radix-ui/react-slot": "^1.2.0",
    "@radix-ui/react-switch": "^1.1.4",
    "@radix-ui/react-tabs": "^1.1.4",
    "@radix-ui/react-toast": "^1.2.7",
    "@radix-ui/react-toggle": "^1.1.3",
    "@radix-ui/react-toggle-group": "^1.1.3",
    "@radix-ui/react-tooltip": "^1.2.0",
    "@tanstack/react-query": "^5.60.5",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "cmdk": "^1.1.1",
    "connect-pg-simple": "^10.0.0",
    "date-fns": "^3.6.0",
    "drizzle-orm": "^0.39.3",
    "drizzle-zod": "^0.7.0",
    "embla-carousel-react": "^8.6.0",
    "express": "^4.21.2",
    "express-session": "^1.18.1",
    "framer-motion": "^11.13.1",
    "input-otp": "^1.4.2",
    "lucide-react": "^0.453.0",
    "memorystore": "^1.6.7",
    "next-themes": "^0.4.6",
    "passport": "^0.7.0",
    "passport-local": "^1.0.0",
    "pg": "^8.16.3",
    "react": "^18.3.1",
    "react-day-picker": "^8.10.1",
    "react-dom": "^18.3.1",
    "react-hook-form": "^7.55.0",
    "react-icons": "^5.4.0",
    "react-resizable-panels": "^2.1.7",
    "recharts": "^2.15.2",
    "tailwind-merge": "^2.6.0",
    "tailwindcss-animate": "^1.0.7",
    "tw-animate-css": "^1.2.5",
    "vaul": "^1.1.2",
    "wouter": "^3.3.5",
    "ws": "^8.18.0",
    "zod": "^3.24.2",
    "zod-validation-error": "^3.4.0"
```

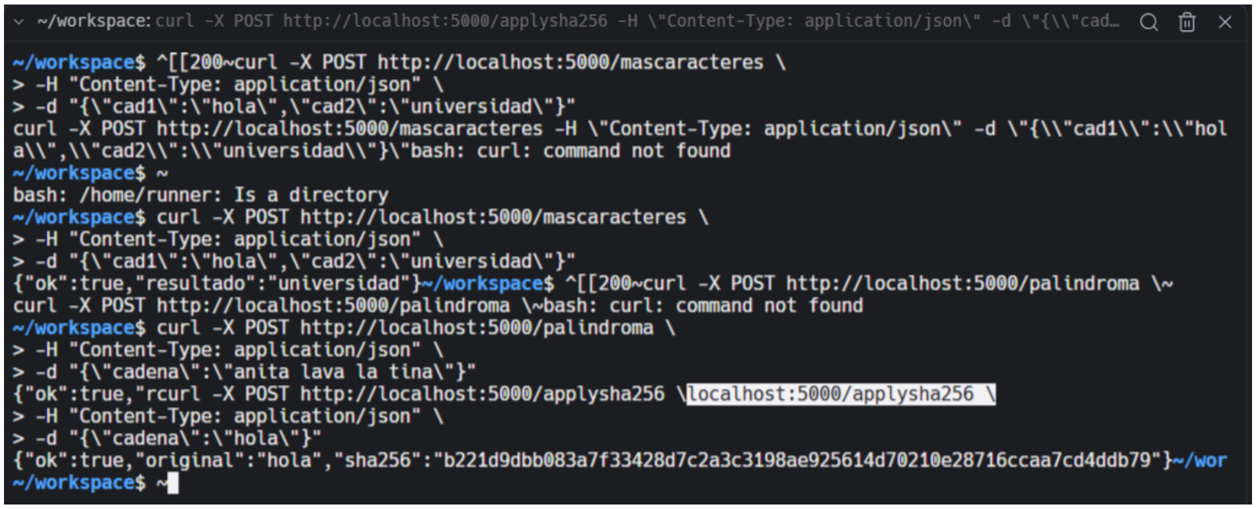
```
  },
  "devDependencies": {
    "@replit/vite-plugin-cartographer": "^0.4.4",
    "@replit/vite-plugin-dev-banner": "^0.1.1",
    "@replit/vite-plugin-runtime-error-modal": "^0.0.3",
    "@tailwindcss/typography": "^0.5.15",
    "@tailwindcss/vite": "^4.1.18",
    "@types/connect-pg-simple": "^7.0.3",
    "@types/express": "4.17.21",
    "@types/express-session": "^1.18.0",
    "@types/node": "20.19.27",
    "@types/passport": "^1.0.16",
    "@types/passport-local": "^1.0.38",
    "@types/react": "^18.3.11",
    "@types/react-dom": "^18.3.1",
    "@types/ws": "^8.5.13",
    "@vitejs/plugin-react": "^4.7.0",
    "autoprefixer": "^10.4.20",
    "drizzle-kit": "^0.31.8",
    "esbuild": "^0.25.0",
    "postcss": "^8.4.47",
    "tailwindcss": "^3.4.17",
    "tsx": "^4.20.5",
    "typescript": "5.6.3",
    "vite": "^7.3.0"
  },
  "overrides": {
    "drizzle-kit": {
      "@esbuild-kit/esm-loader": "npm:tsx@^4.20.4"
    }
  },
  "optionalDependencies": {
    "bufferutil": "^4.0.8"
  }
}
```

Código .replit.

```
modules = ["nodejs-20", "web", "postgresql-16"]
run = "npm start"
hidden = [".config", ".git", "generated-icon.png", "node_modules", "dist"]

[nix]
channel = "stable-24_05"

[[ports]]
localPort = 5000
externalPort = 80

[env]
PORT = "5000"

[deployment]
deploymentTarget = "autoscale"
run = ["node", "./dist/index.cjs"]
```

```
build = ["npm", "run", "build"]

[workflows]
runButton = "Project"

[[workflows.workflow]]
name = "Project"
mode = "parallel"
author = "agent"

[[workflows.workflow.tasks]]
task = "workflow.run"
args = "Start application"

[[workflows.workflow]]
name = "Start application"
author = "agent"

[[workflows.workflow.tasks]]
task = "shell.exec"
args = "npm start"
waitForPort = 5000
```

Pruebas.



```
~/workspace: curl -X POST http://localhost:5000/applysha256 -H \"Content-Type: application/json\" -d \"{\\"cad…

~/workspace$ ^[[200~curl -X POST http://localhost:5000/mascaracteres \
> -H "Content-Type: application/json" \
> -d "{\"cad1\":\"hola\",\"cad2\":\"universidad\"}"
curl -X POST http://localhost:5000/mascaracteres -H \"Content-Type: application/json\" -d \"{\\"cad1\\":\\"hol
a\\",\\"cad2\\":\\"universidad\\"}\"bash: curl: command not found
~/workspace$ ~
bash: /home/runner: Is a directory
~/workspace$ curl -X POST http://localhost:5000/mascaracteres \
> -H "Content-Type: application/json" \
> -d "{\"cad1\":\"hola\",\"cad2\":\"universidad\"}"
{"ok":true,"resultado":"universidad"}~/workspace$ ^[[200~curl -X POST http://localhost:5000/palindroma \~
curl -X POST http://localhost:5000/palindroma \~bash: curl: command not found
~/workspace$ curl -X POST http://localhost:5000/palindroma \
> -H "Content-Type: application/json" \
> -d "{\"cadena\":\"anita lava la tina\"}"
{"ok":true,"rcurl -X POST http://localhost:5000/applysha256 \localhost:5000/applysha256 \
> -H "Content-Type: application/json" \
> -d "{\"cadena\":\"hola\"}"
{"ok":true,"original":"hola","sha256":"b221d9dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79"}~/wor
~/workspace$ ~
```