Solución Prueba - Banco Solar

Para desarrollar esta prueba deberás seguir el siguiente paso a paso en donde estaremos abordando todos los requerimientos:

- ***Paso 1:*** Crear el servidor y devolver la aplicación cliente en la ruta raíz. (*Requerimiento abordados: 4 y 5)*

index.js

```
const http = require("http");
const fs = require("fs");

http
  .createServer(async (req, res) => {
    if (req.url == "/" && req.method == "GET") {
      fs.readFile("index.html", (err, data) => {
        if (err) {
          res.statusCode = 500;
          res.end();
        } else {
          res.setHeader("Content-type", "text/html");
          res.end(data);
        }
      });
    }
  })
  .listen(3000, console.log("SERVER ON"));
```

- ***Paso 2:*** Crear la ruta **/usuario POST** para registrar nuevos usuarios. *(Requerimiento abordados: 1, 3, 4 y 5)*

index.js

```
  if (req.url == "/usuario" && req.method == "POST") {
      let body = "";
      req.on("data", (chunk) => {
        body = chunk.toString();
      });
      req.on("end", async () => {
        const usuario = JSON.parse(body);
        try {
          const result = await guardarUsuario(usuario);
          res.statusCode = 201;
          res.end(JSON.stringify(result));
        } catch (e) {
          res.statusCode = 500;
          res.end("Ocurrió un problema en el servidor... " + e);
        }
      });
    }
```

consultas.js

```
const { Pool } = require("pg");

const pool = new Pool({
  user: "postgres",
  host: "localhost",
  password: "postgres",
  database: "bancosolar",
  port: 5432,
});

const guardarUsuario = async (usuario) => {
  const values = Object.values(usuario);
  const consulta = {
    text: "INSERT INTO usuarios (nombre,balance) values ( $1, $2)",
    values,
  };
```

```
  const result = await pool.query(consulta);
  return result;
};
```

- ***Paso 3:*** Crear la ruta **/usuarios GET** para devolver todos los usuarios registrados en PostgreSQL. (*Requerimiento abordados: 1, 3, 4 y 5)*

index.js

```
 if (req.url == "/usuarios" && req.method == "GET") {
      try {
        const usuarios = await getUsuarios();
        res.end(JSON.stringify(usuarios));
      } catch (e) {
        res.statusCode = 500;
        res.end("Ocurrió un problema en el servidor... " + e);
      }
    }
```

consultas.js

```
const getUsuarios = async () => {
  const result = await pool.query("SELECT * FROM usuarios");
  return result.rows;
};
```

- ***Paso 4:*** Crear la ruta **/usuario PUT** para modificar los datos de un usuario registrado en PostgreSQL. (*Requerimiento abordados: 1, 3, 4 y 5)*

index.js

```
  if (req.url.startsWith("/usuario") && req.method == "PUT") {
      let body = "";
      let { id } = url.parse(req.url, true).query;

      req.on("data", (chunk) => {
        body = chunk.toString();
      });
      req.on("end", async () => {
        const usuario = JSON.parse(body);

        try {
          const result = await editUsuario(usuario, id);
          res.statusCode = 200;
          res.end(JSON.stringify(result));
        } catch (e) {
          res.statusCode = 500;
          res.end("Ocurrió un problema en el servidor... " + e);
        }
      });
    }
```

consultas.js

```
const editUsuario = async (usuario, id) => {
  const values = Object.values(usuario).concat([id]);
  const consulta = {
    text:
      "UPDATE usuarios SET nombre = $1, balance = $2 WHERE id = $3
RETURNING *",
    values,
  };
  const result = await pool.query(consulta);
  return result;
};
```

- ***Paso 5:*** Crear la ruta **/usuario DELETE** para eliminar un usuario registrado en PostgreSQL. (*Requerimiento abordados: 1, 3, 4 y 5*)

index.js

```
    if (req.url.startsWith("/usuario?id") && req.method == "DELETE") {
      try {
        let { id } = url.parse(req.url, true).query;
        await eliminarUsuario(id);
        res.end("Usuario eliminado");
      } catch (e) {
        res.statusCode = 500;
        res.end("Ocurrió un problema en el servidor... " + e);
      }
    }
```

consultas.js

```
const eliminarUsuario = async (id) => {
  const result = await pool.query(`DELETE FROM usuarios WHERE id =
${id}`);
  return result.rows;
};
```

- ***Paso 6:*** Crear la ruta **/transferencia POST** para agregar una nueva transferencia en PostgreSQL. (*Requerimiento abordados: 1, 2, 4 y 5*)

index.js

```
    if (req.url == "/transferencia" && req.method == "POST") {
      let body = "";
      req.on("data", (chunk) => {
        body += chunk.toString();
      });
      req.on("end", async () => {
        try {
          const transferencia = JSON.parse(body);
          const result = await registrarTransferencia(transferencia);
          res.statusCode = 201;
          res.end(JSON.stringify(result));
        } catch (e) {
          res.statusCode = 500;
          res.end("Ocurrió un problema en el servidor..." + e);
        }
      });
    }
```

consultas.js

```
const registrarTransferencia = async ({ emisor, receptor, monto }) => {
  const { id: emisorId } = (
    await pool.query(`SELECT * FROM usuarios WHERE nombre =
'${emisor}'`)
  ).rows[0];
  const { id: receptorId } = (
    await pool.query(`SELECT * FROM usuarios WHERE nombre =
'${receptor}'`)
  ).rows[0];

  const registrarTransferenciaQuery = {
    text:
      "INSERT INTO transferencias (emisor, receptor, monto, fecha)
values ( $1, $2, $3, NOW())",
    values: [emisorId, receptorId, monto],
  };
```

```
  const actualizarBalanceEmisor = {
    text: "UPDATE usuarios SET balance = balance - $1 WHERE nombre =
$2",
    values: [monto, emisor],
  };

  const actualizarBalanceReceptor = {
    text: "UPDATE usuarios SET balance = balance + $1 WHERE nombre =
$2",
    values: [monto, receptor],
  };

  try {
    await pool.query("BEGIN");
    await pool.query(registrarTransferenciaQuery);
    await pool.query(actualizarBalanceEmisor);
    await pool.query(actualizarBalanceReceptor);
    await pool.query("COMMIT");
    return true;
  } catch (e) {
    console.log(e);
    await pool.query("ROLLBACK");
    throw e;
  }
};
```

- ***Paso 7:*** Crear la ruta **/transferencias GET** para devolver todas las transferencias almacenadas en PostgreSQL. (*Requerimiento abordados: 1, 4 y 5)*

index.js

```
if (req.url == "/transferencias" && req.method == "GET") {
     try {
       const historial = await getTransferencias();
       res.end(JSON.stringify(historial));
     } catch (e) {
       res.statusCode = 500;
       res.end("Ocurrió un problema en el servidor..." + e);
     }
   }
```

consultas.js

```
const getTransferencias = async () => {
  const consulta = {
    text: `SELECT * FROM transferencias`,
    rowMode: "array",
  };
  const result = await pool.query(consulta);
  return result.rows;
};
```