Actividades para configuración del backend y frontend

Configuración del backend y frontend del proyecto usando **Node.js** y **React.js**, asegurando la correcta conexión con la base de datos SQL Server, y resolviendo varios problemas relacionados con la sincronización entre el servidor (backend) y la aplicación de React (frontend).

1. **Configuración del servidor backend con Node.js:**
    - Creamos el archivo server.js para manejar las peticiones a la base de datos SQL Server.
    - Configuramos la conexión con SQL Server usando la biblioteca **mssql**.
    - Creamos una API en Node.js que expone los datos de los proyectos desde la base de datos.
    - Probamos la ruta http://localhost:3001/api/proyectos para verificar que los datos de los proyectos se recuperen correctamente.
2. **Integración del frontend con React.js**:
    - Configuramos el frontend en React para mostrar la lista de proyectos obtenidos de la API.
    - Usamos la librería **Axios** para realizar peticiones HTTP desde el frontend hacia el backend.
    - Ajustamos el archivo App.js en React para cargar los datos de los proyectos y mostrarlos en el navegador.
3. **Resolución de errores de conexión y proxy:**
    - Resolvimos el problema del puerto entre React y Node.js, asegurando que el backend corriera en localhost:3001 y el frontend en localhost:3000.
    - Configuramos correctamente el archivo package.json de React para que las peticiones hacia /api/proyectos fueran proxy hacia el puerto 3001.
    - Aseguramos que ambos servidores (React y Node.js) estuvieran corriendo en paralelo.

Pasos para replicar este proceso:

1. Backend (Node.js):

1. **Instalar las dependencias necesarias** (ejecutar en consola el comando):
```
npm install express mssql cors
```

Crear el archivo server.js con el siguiente código:
```
const express = require('express');
const cors = require('cors');
```

```
const { poolPromise, sql } = require('./db'); // Conexión a la DB

const app = express();
const port = process.env.PORT || 3001;

app.use(cors());
app.use(express.json());

app.get('/api/proyectos', async (req, res) => {
    try {
        const pool = await poolPromise;
        const result = await pool.request().query('SELECT * FROM
Proyectos');
        res.json(result.recordset);
    } catch (err) {
        res.status(500).send({ message: err.message });
    }
});

app.listen(port, () => {
    console.log(`Servidor ejecutándose en http://localhost:${port}`);
});
```

2. **Configurar la conexión a SQL Server** en un archivo db.js:

```
const sql = require('mssql');

const config = {
    user: 'sa',
    password: 'tu_password',
    server: 'localhost',
    database: 'IDProjectGASCHSOFT',
    options: {
        encrypt: true,
        trustServerCertificate: true
    }
};

const poolPromise = new sql.ConnectionPool(config)
    .connect()
    .then(pool => {
        console.log('Conectado a SQL Server');
        return pool;
    })
    .catch(err => console.log('Error en la conexión: ', err));

module.exports = { sql, poolPromise };
```

3. Ejecutar el backend:
   `node server.js`

2. Frontend (React.js):

1. A Instalar la librería Axios en React:
   `npm install axios`
2. Modificar el archivo App.js para cargar los proyectos:

```
import React, { useEffect, useState } from 'react';

import axios from 'axios';
```

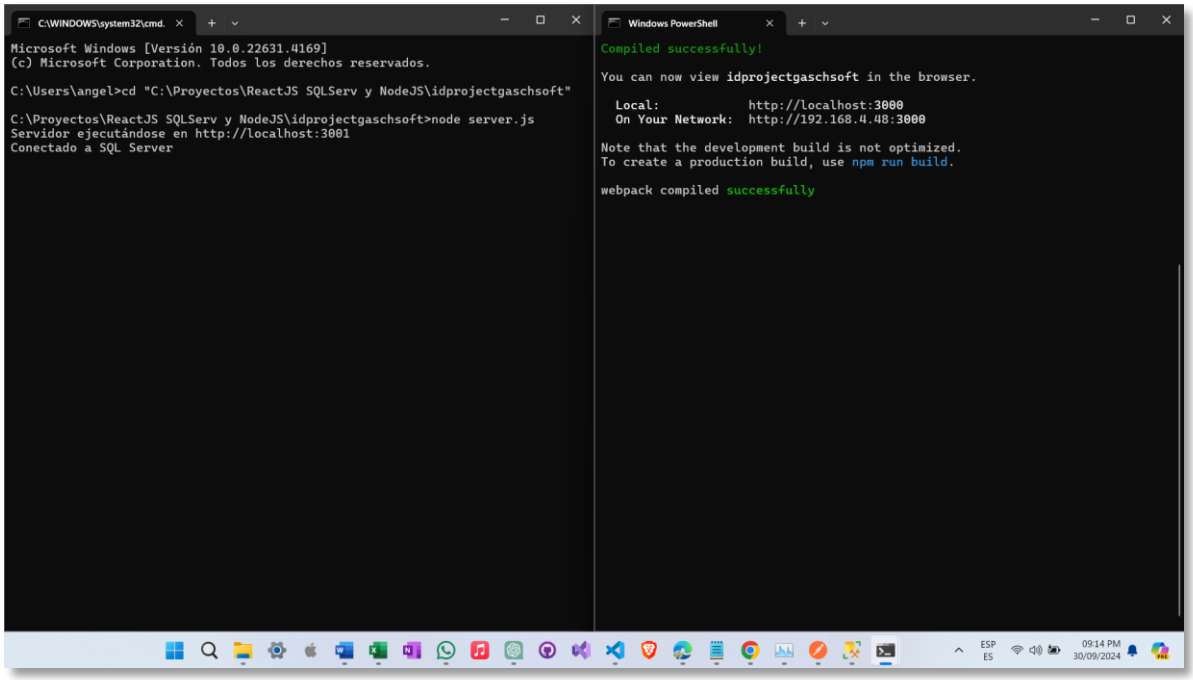
```
function App() {
  const [proyectos, setProyectos] = useState([]);

  useEffect(() => {
    axios.get('/api/proyectos')
      .then(res => setProyectos(res.data))
      .catch(err => console.log('Error al obtener los proyectos: ',
err));
  }, []);

  return (
    <div>
      <h1>Gestión de Proyectos</h1>
      <h2>Lista de Proyectos</h2>
      <ul>
        {proyectos.map(proyecto => (
          <li key={proyecto.idProyecto}>
            <h3>{proyecto.nombreProyecto}</h3>
            <p><strong>Descripción:</strong> {proyecto.descripcion}</p>
            <p><strong>Estado:</strong> {proyecto.estado}</p>
            <p><strong>Fecha Inicio:</strong> {new
Date(proyecto.fechaInicio).toLocaleDateString()}</p>
            <p><strong>Fecha Fin:</strong> {new
Date(proyecto.fechaFin).toLocaleDateString()}</p>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

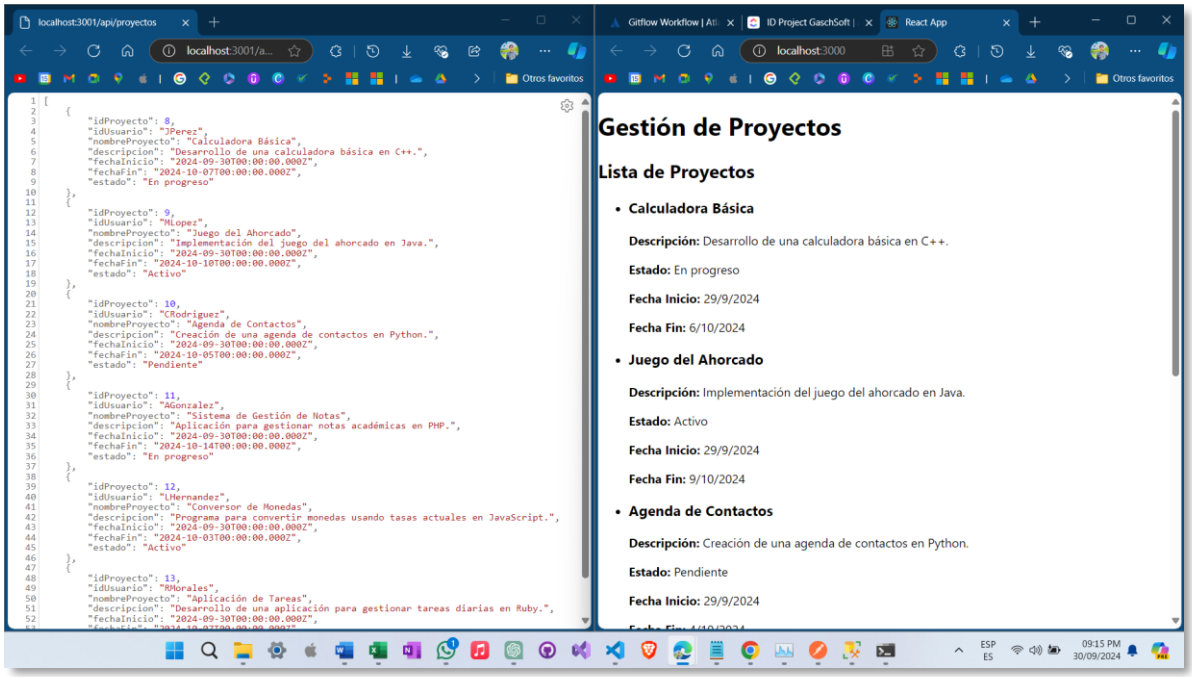
3. **Configurar el proxy en React** en el archivo package.json:

```
{
  "name": "idprojectgaschsoft",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "axios": "^1.7.7",
    "cors": "^2.8.5",
    "express": "^4.18.1",
    "mssql": "^11.0.1",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "proxy": "http://localhost:3001"
}
```

4. Ejecutar el frontend:

```
npm start
```





Manejo de errores comunes y soluciones alternativas:

1. **Error: Proxy no funciona o muestra ECONNREFUSED:**
    - **Causa:** El backend no estaba corriendo o el puerto del servidor backend era incorrecto.

- **Solución:** Asegurarse de que el backend esté corriendo en el puerto 3001 y que la configuración del proxy en React sea correcta.

2. **Error: Axios no puede obtener datos:**
    - **Causa:** El servidor Node.js no estaba devolviendo correctamente los datos.
    - **Solución:** Verificar que la conexión a la base de datos esté activa y que la ruta API esté bien configurada.
3. **Confusión entre puertos de React y Node.js:**
    - **Causa:** Intentar ejecutar tanto el frontend como el backend en el mismo puerto.
    - **Solución:** Asegurarse de que el backend corra en 3001 y React en 3000, usando el proxy para redirigir las solicitudes.

Conclusión:

- Configuramos y ejecutamos exitosamente Node.js y React.js para gestionar proyectos y conectarlos con SQL Server.
- A lo largo del proceso, corregimos problemas de conexión, sincronización y proxy entre backend y frontend.