1 creo una carpeta y la habro con el visual studio code

* Creo los paquetes
* Npm init : información de la api
* Git init : control de versiones

2 creo el index.ts

Tsc –init : archivo de validación tsconfig.json

Activo lo que necesito:

Outdir dist/

Compilacion auto guardado de tsc a jsc tsc –w esta por defecto su install si no esta instalado ejecutarlo

Compilacion auto npm install –g nodemon o confgurar pm2 si no esta instalado se debe de ejecutar el comando luego se puede levantar.

3 **dependencias necesarias: (208)**

npm install express: para el servisio de web o rest

body-parser: tranformar información en js ojebto y poder utilizarla

cors: para las peticiones con header http ip etz

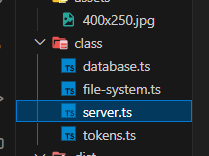
1. **Instalar las ayudas de tipado en modo dependencias de desarrollo –save-dev**

Npm install @types/express –save-dev

Npm install @types/bcrypt –save-dev

Npm install @types/cors –save-dev

1. **Crear carpeta class y archivo server.ts donde estarán las instancias como express y el port: (209)**

****

import express from "express";

// exporto por defecto el servidor de escucha

export default class Server {

    public app: express.Application;

    public port: number= 3000;

    // start the class with this constructor

    constructor() {

        this.app =express();

    }

    //end constructor

    // listen for port

    start(callback: ()=> void ) {

     this.app.listen(this.port, callback);

    }

// and listen

}

1. **Estructurar las rutas dentro de una carpeta llamada rutas creada**

* Carpeta Rutas, archivo usuario.ts
* Dentro de esta clase haremos la ruta para validar el usuario
* Se debe de hacer hash la contraseña que se recibe como md5
* Crear constante usando Router(); y se importa de express
* Se importa axios para hacer peticiones post xml a un webservice soap
* Creamos la ruta post definiendo la ruta, req, res y su => {res.json({})}
* Se exporta esta constante que contiene esta ruta
* Se importa en el index y la pasamos por el servidor serve.app.use(‘/use’, constante import);

Como vamos a realizar una solicitud SOAP desde una api en node js necesitamos módulos como

‘axios’ o ‘node-fetch’ con esto se puede realizar solicitudes POST con el cuerpo SOAP

1. Instalar npm install axios
2. import { Router, Request, Response } from "express";
3. import axios from "axios";
4. const crypto = require('crypto');
5. // para usar variabel de entorno
6. //require('dotenv').config();
7. // datos para realizar la peticion json
8. const UserRoutes = Router();
9. // url ruta donde se encuentra el servicio web lo pondre en la variable de entorno
10. // const url= process.env.TZ;
11. const url = 'http://pantera.goib.com/controlcargaws/movil.asmx';
12. // Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres
13. function generarHashMD5(cadena:any) {
14. const hash = crypto.createHash('md5');
15. hash.update(cadena);
16. return hash.digest('hex');
17. }
18. // inicio de proceso para el login de res a web
19. UserRoutes.post('/login',  async (req: Request, res: Response) =>{
21. try{
22. const { Codigo, Clave } = req.body; // estos datos vienen en la solicitud
24. // Generar el hash de la contraseña
25. const hashedClave = generarHashMD5(Clave);
26. // body de la peticion hacia el soap
27. const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
28. <soap:Body>
29. <Validar\_Usuario xmlns="http://tempuri.org/">
30. <Codigo>${Codigo}</Codigo>
31. <Clave>${hashedClave}</Clave>
32. </Validar\_Usuario>
33. </soap:Body>
34. </soap:Envelope>`;
36. const respuest = await axios.post(url, xmlBody, {
37. headers: {
38. 'Content-Type': 'text/xml; charset=utf-8',
39. SOAPAction: 'http://tempuri.org/Validar\_Usuario',
40. },
41. });
42. // Verificar si la respuesta contiene la información esperada
43. if (respuest.data) {
44. // Aquí puedes manejar la respuesta de la solicitud SOAP
45. res.status(200).json({
46. ok: true,
47. mensaje: 'Se logro conectar al servicio y esta es la respuesta:',
48. respuestaSOAP: respuest.data, // Puedes devolver la respuesta SOAP si es necesario
49. });
50. }
51. else {
52. res.status(500).json({
53. ok: false,
54. mensaje: 'No se recibió una respuesta válida del servicio SOAP',
55. });
56. }
57. } catch (error:any) {
58. res.status(500).json({
59. ok: false,
60. mensaje: 'Error al procesar la solicitud',
61. error: error.message,
62. });
64. }
65. });
66. export default UserRoutes;

index.ts

import Server from "./class/server";

import UserRoutes from "./rutas/usuario";

import bodyParser from "body-parser";

//import axiosRoutes from "./rutas/usuario";

//A1 constante del server para usar la extancia de express servidor

const server = new Server;

//===============================fin A1======================

server.start(()=>{

    console.log(`El servidor esta corriendo en el puerto ${server.port}`);

})

//middleware with bodyparser pasa estos parametros primero estos middleware deben de ir de primero ya que cuando pase la ruta user deben de existir los datos para el registro

server.app.use(bodyParser.urlencoded({ extended:true}));

server.app.use(bodyParser.json());

//end middleware

// // user rutas

// server.app.use('/user', UserRoutes);

// // fin rutas

// user rutas

server.app.use('/user', UserRoutes);

// fin rutas

Activare en el index.ts el cors para eliminar la restrincion de mi petición desde cualquier fronter

Npm install cors

Instalar cors: para las peticiones con header http ip etz

Y su tipado

Npm install @types/cors –save-dev

import Server from "./class/server";

import UserRoutes from "./rutas/usuario";

import bodyParser from "body-parser";

import cors from "cors";

//A1 constante del server para usar la extancia de express servidor

const server = new Server;

//===============================fin A1======================

server.start(()=>{

    console.log(`El servidor esta corriendo en el puerto ${server.port}`);

})

//middleware with bodyparser pasa estos parametros primero estos middleware deben de ir de primero ya que cuando pase la ruta user deben de existir los datos para el registro

server.app.use(bodyParser.urlencoded({ extended:true}));

server.app.use(bodyParser.json());

//end middleware

// user Cors

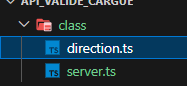
server.app.use(cors({ origin: true, credentials: true}));

// user rutas

server.app.use('/user', UserRoutes);

// fin rutas

Realizar ruta para consultar bodegas



// ruta principal webservices

const WebUrl = 'http://pantera.goib.com/controlcargaws/movil.asmx';

// dejo estas conectiones si en caso tal utilizare las bse de datos

//BASE DE DATOS PRUEBA PARA ANPR EN PANTERA

const dbpantera = {

    user: 'sa',

    password: 'root',

    port: 36934,

    server: '172.30.200.200',

    database: 'venus\_opp\_prueba\_02122020',

    options : {

    encrypt: true,

    trustServerCertificate: true,

    // trustedconnection : false,

    // enableArithAbort : true,

    },

};

// dejo estas conectiones si en caso tal utilizare las bse de datos

const dbccarga = {

    user: 'sa',

    password: 'root',

    port: 3341,

    server: 'ccarga.goib.com',

    database: 'venus\_opp',

    options : {

    encrypt: true,

    trustServerCertificate: true,

    // trustedconnection : false,

    // enableArithAbort : true,

    },

};

module.exports = WebUrl;

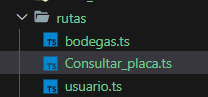
luego en cada clase de ruta la llamo

// llamando ruta principal de webservices

const urlweb = require('../class/direction')

const url = urlweb;

procedo a realizar la ruta para consultar placa en transito



import { Router, Request, Response } from "express";

import axios from "axios";

const urlwb = require('../class/direction');

const crypto = require('crypto');

// para usar variabel de entorno

//require('dotenv').config();

// datos para realizar la peticion json

const PlacasRoutes = Router();

// url ruta donde se encuentra el servicio web lo pondre en la variable de entorno

// const url= process.env.TZ;

const url =  urlwb;

// Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres

function generarHashMD5(cadena:any) {

    const hash = crypto.createHash('md5');

    hash.update(cadena);

    return hash.digest('hex');

}

// inicio de proceso para el login de res a web

PlacasRoutes.post('/placa',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave, Placa } = req.body; // estos datos vienen en la solicitud

    // Generar el hash de la contraseña

    const hashedClave = generarHashMD5(Clave);

    // body de la peticion hacia el soap

    const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <soap:Body>

    <Consultar\_Placa xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        <Placa>${Placa}</Placa>

       </Consultar\_Placa>

       </soap:Body>

     </soap:Envelope>`;

        const respuest = await axios.post(url, xmlBody, {

            headers: {

                'Content-Type': 'text/xml; charset=utf-8',

                SOAPAction: 'http://tempuri.org/Consultar\_Placa',

              },

        });

         // Verificar si la respuesta contiene la información esperada

         if (respuest.data) {

            // Aquí puedes manejar la respuesta de la solicitud SOAP

            res.status(200).json({

                ok: true,

                mensaje: 'Se logro conectar al servicio y esta es la respuesta:',

                respuestaSOAP: respuest.data, // Puedes devolver la respuesta SOAP si es necesario

            });

        }

            else {

                res.status(500).json({

                    ok: false,

                    mensaje: 'No se recibió una respuesta válida del servicio SOAP',

                });

            }

  } catch (error:any) {

    res.status(500).json({

        ok: false,

        mensaje: 'Error al procesar la solicitud',

        error: error.message,

    });

}

});

export default PlacasRoutes;

index.ts

// user Cors

server.app.use(cors({ origin: true, credentials: true}));

//rutas validar usuario

server.app.use('/user', UserRoutes);

//rutas ver bodegas

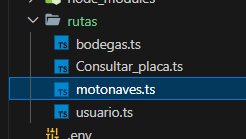
server.app.use('/user', BodegasRoutes);

//rutas consultar placas

server.app.use('/user', PlacasRoutes);

// fin rutas

Ruta de motonaves



import { Router, Request, Response } from "express";

import axios from "axios";

const urlwb = require('../class/direction');

const crypto = require('crypto');

// para usar variabel de entorno

//require('dotenv').config();

// datos para realizar la peticion json

const MotonavesRoutes = Router();

// url ruta donde se encuentra el servicio web lo pondre en la variable de entorno

// const url= process.env.TZ;

const url =  urlwb;

// Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres

function generarHashMD5(cadena:any) {

    const hash = crypto.createHash('md5');

    hash.update(cadena);

    return hash.digest('hex');

}

// inicio de proceso para el login de res a web

MotonavesRoutes.post('/motonaves',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave } = req.body; // estos datos vienen en la solicitud

    // Generar el hash de la contraseña

    const hashedClave = generarHashMD5(Clave);

    // body de la peticion hacia el soap

    const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <soap:Body>

    <Cargar\_Arribos\_de\_Motonaves xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        </Cargar\_Arribos\_de\_Motonaves>

        </soap:Body>

      </soap:Envelope>`;

        const respuest = await axios.post(url, xmlBody, {

            headers: {

                'Content-Type': 'text/xml; charset=utf-8',

                SOAPAction: 'http://tempuri.org/Cargar\_Arribos\_de\_Motonaves',

              },

        });

         // Verificar si la respuesta contiene la información esperada

         if (respuest.data) {

            // Aquí puedes manejar la respuesta de la solicitud SOAP

            res.status(200).json({

                ok: true,

                mensaje: 'Se logro conectar al servicio y esta es la respuesta:',

                respuestaSOAP: respuest.data, // Puedes devolver la respuesta SOAP si es necesario

            });

        }

            else {

                res.status(500).json({

                    ok: false,

                    mensaje: 'No se recibió una respuesta válida del servicio SOAP',

                });

            }

  } catch (error:any) {

    res.status(500).json({

        ok: false,

        mensaje: 'Error al procesar la solicitud',

        error: error.message,

    });

}

});

export default MotonavesRoutes;

// user Cors

server.app.use(cors({ origin: true, credentials: true}));

//rutas validar usuario

server.app.use('/user', UserRoutes);

//rutas ver bodegas

server.app.use('/user', BodegasRoutes);

//rutas consultar placas

server.app.use('/user', PlacasRoutes);

//rutas consultar placas

server.app.use('/user', MotonavesRoutes);

// fin rutas

Ruta patios

import { Router, Request, Response } from "express";

import axios from "axios";

const urlwb = require('../class/direction');

const crypto = require('crypto');

// para usar variabel de entorno

//require('dotenv').config();

// datos para realizar la peticion json

const PatiosRoutes = Router();

// url ruta donde se encuentra el servicio web lo pondre en la variable de entorno

// const url= process.env.TZ;

const url =  urlwb;

// Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres

function generarHashMD5(cadena:any) {

    const hash = crypto.createHash('md5');

    hash.update(cadena);

    return hash.digest('hex');

}

// inicio de proceso para el login de res a web

PatiosRoutes.post('/patios',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave } = req.body; // estos datos vienen en la solicitud

    // Generar el hash de la contraseña

    const hashedClave = generarHashMD5(Clave);

    // body de la peticion hacia el soap

    const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

      <soap:Body>

    <Cargar\_Patios xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        </Cargar\_Patios>

  </soap:Body>

</soap:Envelope>`;

        const respuest = await axios.post(url, xmlBody, {

            headers: {

                'Content-Type': 'text/xml; charset=utf-8',

                SOAPAction: 'http://tempuri.org/Cargar\_Patios',

              },

        });

         // Verificar si la respuesta contiene la información esperada

         if (respuest.data) {

            // Aquí puedes manejar la respuesta de la solicitud SOAP

            res.status(200).json({

                ok: true,

                mensaje: 'Se logro conectar al servicio y esta es la respuesta:',

                respuestaSOAP: respuest.data, // Puedes devolver la respuesta SOAP si es necesario

            });

        }

            else {

                res.status(500).json({

                    ok: false,

                    mensaje: 'No se recibió una respuesta válida del servicio SOAP',

                });

            }

  } catch (error:any) {

    res.status(500).json({

        ok: false,

        mensaje: 'Error al procesar la solicitud',

        error: error.message,

    });

}

});

export default PatiosRoutes;

server.app.use('/user', PlacasRoutes);

//rutas consultar motonaves

server.app.use('/user', MotonavesRoutes);

//rutas consultar partios

server.app.use('/user', PatiosRoutes);

cargar turnos aproches

import { Router, Request, Response } from "express";

import axios from "axios";

const urlwb = require('../class/direction');

const crypto = require('crypto');

// para usar variabel de entorno

//require('dotenv').config();

// datos para realizar la peticion json

const Turnos\_AprochesRoutes = Router();

// url ruta donde se encuentra el servicio web lo pondre en la variable de entorno

// const url= process.env.TZ;

const url =  urlwb;

// Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres

function generarHashMD5(cadena:any) {

    const hash = crypto.createHash('md5');

    hash.update(cadena);

    return hash.digest('hex');

}

// inicio de proceso para el login de res a web

Turnos\_AprochesRoutes.post('/turn\_aproches',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave, Bodega } = req.body; // estos datos vienen en la solicitud

    // Generar el hash de la contraseña

    const hashedClave = generarHashMD5(Clave);

    // body de la peticion hacia el soap

    const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <soap:Body>

    <Cargar\_Turnos\_Aproche xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        <Bodega>${Bodega}</Bodega>

        </Cargar\_Turnos\_Aproche>

        </soap:Body>

      </soap:Envelope>`;

        const respuest = await axios.post(url, xmlBody, {

            headers: {

                'Content-Type': 'text/xml; charset=utf-8',

                SOAPAction: 'http://tempuri.org/Cargar\_Turnos\_Aproche',

              },

        });

         // Verificar si la respuesta contiene la información esperada

         if (respuest.data) {

            // Aquí puedes manejar la respuesta de la solicitud SOAP

            res.status(200).json({

                ok: true,

                mensaje: 'Se logro conectar al servicio y esta es la respuesta:',

                respuestaSOAP: respuest.data, // Puedes devolver la respuesta SOAP si es necesario

            });

        }

            else {

                res.status(500).json({

                    ok: false,

                    mensaje: 'No se recibió una respuesta válida del servicio SOAP',

                });

            }

  } catch (error:any) {

    res.status(500).json({

        ok: false,

        mensaje: 'Error al procesar la solicitud',

        error: error.message,

    });

}

});

export default Turnos\_AprochesRoutes;

//rutas consultar turnos\_Aproches

server.app.use('/user', Turnos\_AprochesRoutes);

turnos bodega

import { Router, Request, Response } from "express";

import axios from "axios";

const urlwb = require('../class/direction');

const crypto = require('crypto');

// para usar variabel de entorno

//require('dotenv').config();

// datos para realizar la peticion json

const Turnos\_BodegaRoutes = Router();

// url ruta donde se encuentra el servicio web lo pondre en la variable de entorno

// const url= process.env.TZ;

const url =  urlwb;

// Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres

function generarHashMD5(cadena:any) {

    const hash = crypto.createHash('md5');

    hash.update(cadena);

    return hash.digest('hex');

}

// inicio de proceso para el login de res a web

Turnos\_BodegaRoutes.post('/turn\_bodega',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave, Bodega, Motonave, Articulo } = req.body; // estos datos vienen en la solicitud

    // Generar el hash de la contraseña

    const hashedClave = generarHashMD5(Clave);

    // body de la peticion hacia el soap

    const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <soap:Body>

    <Cargar\_Turnos\_Bodega xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        <Bodega>${Bodega}</Bodega>

      <Arribo\_Motonave>${Motonave}</Arribo\_Motonave>

      <Articulo>${Articulo}</Articulo>

        </Cargar\_Turnos\_Bodega>

       </soap:Body>

          </soap:Envelope>`;

        const respuest = await axios.post(url, xmlBody, {

            headers: {

                'Content-Type': 'text/xml; charset=utf-8',

                SOAPAction: 'http://tempuri.org/Cargar\_Turnos\_Bodega',

              },

        });

         // Verificar si la respuesta contiene la información esperada

         if (respuest.data) {

            // Aquí puedes manejar la respuesta de la solicitud SOAP

            res.status(200).json({

                ok: true,

                mensaje: 'Se logro conectar al servicio y esta es la respuesta:',

                respuestaSOAP: respuest.data, // Puedes devolver la respuesta SOAP si es necesario

            });

        }

            else {

                res.status(500).json({

                    ok: false,

                    mensaje: 'No se recibió una respuesta válida del servicio SOAP',

                });

            }

  } catch (error:any) {

    res.status(500).json({

        ok: false,

        mensaje: 'Error al procesar la solicitud',

        error: error.message,

    });

}

});

export default Turnos\_BodegaRoutes;

index

//rutas consultar turnos\_Aproches

server.app.use('/user', Turnos\_AprochesRoutes);

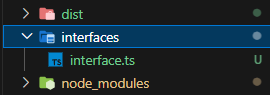
//rutas consultar turnos\_Bodegas

server.app.use('/user', Turnos\_BodegaRoutes);

ahora voy a convertir los datos xml a json y luego crear una interfaz de cada dato mapeando esta información y tener mas control con ella

1. instalar xml2js nom install xml2js
2. luego su tipado de mode dev
3. npm i --save-dev @types/xml2js

se crea la carpeta y clase interface



export interface Bodega {

    Codigo: string;

    Descripcion: String;

}

Importamos en la clase ruta bodegas

import { Router, Request, Response } from "express";

import axios from "axios";

import { Parser } from 'xml2js';

import { Bodega } from "../interfaces/interface";

const urlwb = require('../class/direction');

const crypto = require('crypto');

//empiezo a utilizar la dependencia para parsear xml a json

// configuracion para parsear el xml a json

const xmlparser = new Parser({ explicitArray: false, ignoreAttrs:true});

// para usar variabel de entorno

//require('dotenv').config();

// datos para realizar la peticion json

const BodegasRoutes = Router();

// url ruta donde se encuentra el servicio web lo pondre en la variable de entorno

// const url= process.env.TZ;

const url =  urlwb;

// Función para generar un hash de una cadena usando el algoritmo md5 por que ccarga usa este pero se recomienda SHA256 ya que son mas caracteres

function generarHashMD5(cadena:any) {

    const hash = crypto.createHash('md5');

    hash.update(cadena);

    return hash.digest('hex');

}

// inicio de proceso para el login de res a web

BodegasRoutes.post('/bodegas',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave } = req.body; // estos datos vienen en la solicitud

    // Generar el hash de la contraseña

    const hashedClave = generarHashMD5(Clave);

    // body de la peticion hacia el soap

    const xmlBody =`<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

    <soap:Body>

    <Cargar\_Bodegas xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        <Tipo>B</Tipo>

      <Propia>true</Propia>

        </Cargar\_Bodegas>

        </soap:Body>

      </soap:Envelope>`;

        const respuest = await axios.post(url, xmlBody, {

            headers: {

                'Content-Type': 'text/xml; charset=utf-8',

                SOAPAction: 'http://tempuri.org/Cargar\_Bodegas',

              },

        });

         //varalbe de tipo arreglo para agregar los datos como los quiero en mi interface map

         let bodegas: Bodega[] = [];

       // Verificar si la respuesta contiene la información esperada

       xmlparser.parseString(respuest.data, (err:any, result:any)=>{

        if (err) {

            throw new Error('Error al parsear la respuesta XML');

        }

        const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_BodegasResponse']['Cargar\_BodegasResult']['Informacion\_Movil\_General'];

        if (Array.isArray(informacionMovilGeneral)) {

            // Si hay múltiples elementos, iterar sobre ellos

            informacionMovilGeneral.forEach((info: any) => {

                const codigo = info.Codigo;

                const descripcion = info.Descripcion;

                // Crear un objeto de tipo Bodega y agregarlo al arreglo 'bodegas'

                bodegas.push({ Codigo: codigo, Descripcion: descripcion });

            });

        } else if (informacionMovilGeneral) {

            // Si es un solo elemento, tratarlo como un arreglo de un solo elemento

            const codigo = informacionMovilGeneral.Codigo;

            const descripcion = informacionMovilGeneral.Descripcion;

            bodegas.push({ Codigo: codigo, Descripcion: descripcion });

        }

    });

     if (respuest.data) {

        // Aquí puedes manejar la respuesta de la solicitud SOAP

        res.status(200).json({

            ok: true,

            mensaje: 'Se logro conectar al servicio y esta es la respuesta:',

        // respuestaSOAP: respuest.data,

            bodegas,

        });

    }

        else {

            res.status(500).json({

                ok: false,

                mensaje: 'No se recibió una respuesta válida del servicio SOAP',

            });

        }

} catch (error:any) {

res.status(500).json({

    ok: false,

    mensaje: 'Error al procesar la solicitud',

    error: error.message,

});

}

});

export default BodegasRoutes;



Mapear las Demas rutas

User login