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

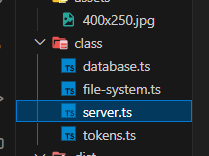
1. **Instalar las ayudas de tipado en modo dependencias de desarrollo –save-dev**

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

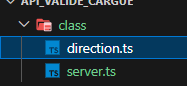
server.app.use(cors({ origin: true, credentials: true}));

// 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;

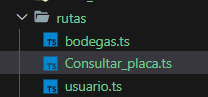
luego en cada clase de ruta la llamo

// 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

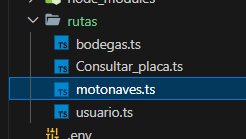
server.app.use('/user', BodegasRoutes);

//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);

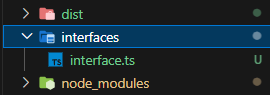
//rutas consultar turnos\_Bodegas

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

* Agregar la interfase de usuario
* export interface Usuario {
* Codigo: string;
* Clave: String;
* Valido: String;
* }
* Importamos esta interfaces en la ruta usuario y modificamos el código tal cual esta aquí

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Usuario } from '../interfaces/interface';

// llamando ruta principal de webservices

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

// componente para encryptar clave

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});

// datos para realizar la peticion json

const UserRoutes = Router();

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

// const url= process.env.TZ;

const url = urlweb;

// 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

UserRoutes.post('/login',  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>

      <Validar\_Usuario xmlns="http://tempuri.org/">

        <Codigo>${Codigo}</Codigo>

        <Clave>${hashedClave}</Clave>

      </Validar\_Usuario>

    </soap:Body>

    </soap:Envelope>`;

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

            headers: {

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

                SOAPAction: 'http://tempuri.org/Validar\_Usuario',

              },

        });

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

        let user: Usuario[] = [];

        // 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 Validar\_UsuarioResult = result['soap:Envelope']['soap:Body']['Validar\_UsuarioResponse']['Validar\_UsuarioResult'];

        if (Array.isArray(Validar\_UsuarioResult)) {

            // Si hay múltiples elementos, iterar sobre ellos

            Validar\_UsuarioResult.forEach((info: any) => {

                const codigo = info.Codigo;

                const clave = info.Clave;

                const valido = info.Valido;

                // Crear un objeto de tipo usuario y agregarlo al arreglo 'user'

                user.push({ Codigo: codigo, Clave: clave, Valido: valido });

            });

        } else if (Validar\_UsuarioResult) {

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

            const codigo = Validar\_UsuarioResult.Codigo;

            const clave = Validar\_UsuarioResult.Clave;

            const valido = Validar\_UsuarioResult.Valido;

            user.push({ Codigo: codigo, Clave: clave, Valido: valido });

        }

    });

    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,

            user,

        });

    }

        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,

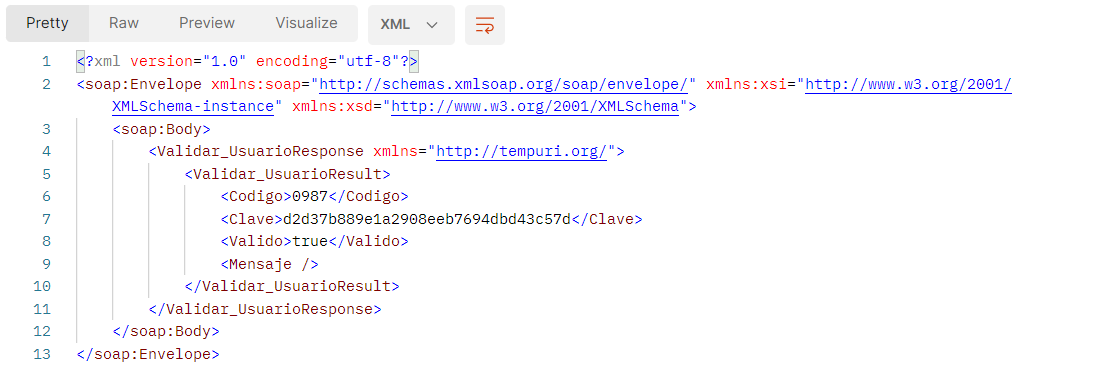
                });

                }

           });

             export default UserRoutes;

listo ya se cambio la estructura

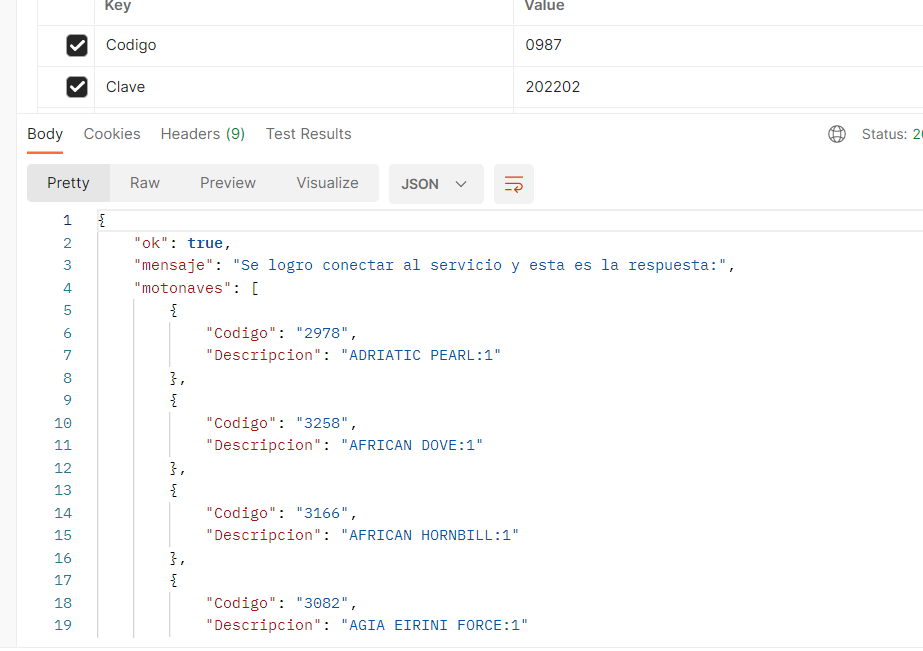
de esto 

a esto



Mapear la información de motonaves

* Agregar la interfase de motonaves
* export interface Usuario {
* Codigo: string;
* Clave: String;
* Valido: String;
* }
* Importamos esta interfaces en la ruta motonaves y modificamos el código tal cual esta aquí
* import { Router, Request, Response } from "express";
* import axios from "axios";
* import { Parser } from 'xml2js';
* import { Motonave } 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});
* // 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',
* },
* });
* //varalbe de tipo arreglo para agregar los datos como los quiero en mi interface map
* let motonaves: Motonave[] = [];
* // 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');
* }
* //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan
* const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_Arribos\_de\_MotonavesResponse']['Cargar\_Arribos\_de\_MotonavesResult']['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'
* motonaves.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;
* motonaves.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,
* motonaves,
* });
* }
* 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;



mapear la información de consultar placa

primero su interface

export interface Placa {

    Cedula: string;

    Nombre\_Conductor: String;

    Orden: String;

    Valido: String;

    Transportadora: String;

    Concepto: String;

    Articulo: String;

    Arribo\_Motonave: String;

    Bodega: String;

    Bodega\_Tipo: String;

    Bodega\_Propia: String;

    Modalidad: String;

    Empaque: String;

    Nombre\_Motonave: String;

    Deposito: String;

    Nombre\_Articulo: String;

    Configuracion\_Vehicular: String;

    Observaciones: String;

    Fecha\_Entrada: String;

    Unidades: String;

    Escotilla: String;

    Tara: String;

    Peso\_a\_Cargar: String;

    Peso\_a\_Cambiar: String;

    Peso\_Maximo: String;

    Saldo\_al\_BL: String;

    Saldo\_Solicitud: String;

    De\_Compartido: String;

    Urbano\_Directo\_Controlado: String;

    Documentacion\_Validada: String;

}

Ahora en la ruta importo la interface y la modifico de la siguiente manera

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Placa } 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});

// 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',

              },

        });

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

 let placa: Placa[] = [];

 // 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');

 }

 //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

 const Consultar\_PlacaResult = result['soap:Envelope']['soap:Body']['Consultar\_PlacaResponse']['Consultar\_PlacaResult'];

 if (Array.isArray(Consultar\_PlacaResult)) {

     // Si hay múltiples elementos, iterar sobre ellos

     Consultar\_PlacaResult.forEach((info: any) => {

        const Cedula= info.Cedula;

        const Nombre\_Conductor= info.Nombre\_Conductor;

        const Orden= info.Orden;

        const Valido= info.Valido;

        const Transportadora= info.Transportadora;

        const Concepto= info.Concepto;

        const Articulo= info.Articulo;

        const Arribo\_Motonave= info.Arribo\_Motonave;

        const Bodega= info.Bodega;

        const Bodega\_Tipo= info.Bodega\_Tipo;

        const Bodega\_Propia= info.Bodega\_Propia;

        const Modalidad= info.Modalidad;

        const Empaque= info.Empaque;

        const Nombre\_Motonave= info.Nombre\_Motonave;

        const Deposito= info.Deposito;

        const Nombre\_Articulo= info.Nombre\_Articulo;

        const Configuracion\_Vehicular= info.configuracion\_Vehicular;

        const Observaciones= info.Observaciones;

        const Fecha\_Entrada= info.Fecha\_Entrada;

        const Unidades= info.Unidades;

        const Escotilla= info.Escotilla;

        const Tara= info.Tara;

        const Peso\_a\_Cargar= info.Peso\_a\_Cargar;

        const Peso\_a\_Cambiar= info.Peso\_a\_Cambiar;

        const Peso\_Maximo= info.Peso\_Maximo;

        const Saldo\_al\_BL= info.Saldo\_al\_BL;

        const Saldo\_Solicitud= info.Saldo\_Solicitud;

        const De\_Compartido= info.De\_Compartido;

        const Urbano\_Directo\_Controlado= info.Urbano\_Directo\_Controlado;

        const Documentacion\_Validada= info.Documentacion\_Validada;

         // Crear un objeto de tipo placa y agregarlo al arreglo 'placa'

         placa.push({

             Cedula: Cedula, Nombre\_Conductor: Nombre\_Conductor,

             Orden: Orden,

             Valido: Valido,

             Transportadora: Transportadora,

             Concepto: Concepto,

             Articulo: Articulo,

             Arribo\_Motonave: Arribo\_Motonave,

             Bodega: Bodega,

             Bodega\_Tipo: Bodega\_Tipo,

             Bodega\_Propia: Bodega\_Propia,

             Modalidad: Modalidad,

             Empaque: Empaque,

             Nombre\_Motonave: Nombre\_Motonave,

             Deposito:Deposito,

             Nombre\_Articulo: Nombre\_Articulo,

             Configuracion\_Vehicular: Configuracion\_Vehicular,

             Observaciones: Observaciones,

             Fecha\_Entrada: Fecha\_Entrada,

             Unidades: Unidades,

             Escotilla: Escotilla,

             Tara: Tara,

             Peso\_a\_Cargar: Peso\_a\_Cargar,

             Peso\_a\_Cambiar: Peso\_a\_Cambiar,

             Peso\_Maximo: Peso\_Maximo,

             Saldo\_al\_BL: Saldo\_al\_BL,

             Saldo\_Solicitud: Saldo\_Solicitud,

             De\_Compartido: De\_Compartido,

             Urbano\_Directo\_Controlado: Urbano\_Directo\_Controlado,

             Documentacion\_Validada: Documentacion\_Validada

         });

     });

 } else if (Consultar\_PlacaResult) {

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

        const codigo = Consultar\_PlacaResult.Codigo;

        const descripcion = Consultar\_PlacaResult.Descripcion;

        const Cedula= Consultar\_PlacaResult.Cedula;

        const Nombre\_Conductor= Consultar\_PlacaResult.Nombre\_Conductor;

        const Orden= Consultar\_PlacaResult.Orden;

        const Valido= Consultar\_PlacaResult.Valido;

        const Transportadora= Consultar\_PlacaResult.Transportadora;

        const Concepto= Consultar\_PlacaResult.Concepto;

        const Articulo= Consultar\_PlacaResult.Articulo;

        const Arribo\_Motonave= Consultar\_PlacaResult.Arribo\_Motonave;

        const Bodega= Consultar\_PlacaResult.Bodega;

        const Bodega\_Tipo= Consultar\_PlacaResult.Bodega\_Tipo;

        const Bodega\_Propia=Consultar\_PlacaResult.Bodega\_Propia;

        const Modalidad= Consultar\_PlacaResult.Modalidad;

        const Empaque= Consultar\_PlacaResult.Empaque;

        const Nombre\_Motonave= Consultar\_PlacaResult.Nombre\_Motonave;

        const Deposito= Consultar\_PlacaResult.Deposito;

        const Nombre\_Articulo= Consultar\_PlacaResult.Nombre\_Articulo;

        const Configuracion\_Vehicular= Consultar\_PlacaResult.configuracion\_Vehicular;

        const Observaciones=Consultar\_PlacaResult.Observaciones;

        const Fecha\_Entrada= Consultar\_PlacaResult.Fecha\_Entrada;

        const Unidades= Consultar\_PlacaResult.Unidades;

        const Escotilla= Consultar\_PlacaResult.Escotilla;

        const Tara= Consultar\_PlacaResult.Tara;

        const Peso\_a\_Cargar= Consultar\_PlacaResult.Peso\_a\_Cargar;

        const Peso\_a\_Cambiar= Consultar\_PlacaResult.Peso\_a\_Cambiar;

        const Peso\_Maximo= Consultar\_PlacaResult.Peso\_Maximo;

        const Saldo\_al\_BL= Consultar\_PlacaResult.Saldo\_al\_BL;

        const Saldo\_Solicitud= Consultar\_PlacaResult.Saldo\_Solicitud;

        const De\_Compartido= Consultar\_PlacaResult.De\_Compartido;

        const Urbano\_Directo\_Controlado= Consultar\_PlacaResult.Urbano\_Directo\_Controlado;

        const Documentacion\_Validada= Consultar\_PlacaResult.Documentacion\_Validada;

     placa.push({

        Cedula: Cedula, Nombre\_Conductor: Nombre\_Conductor,

        Orden: Orden,

        Valido: Valido,

        Transportadora: Transportadora,

        Concepto: Concepto,

        Articulo: Articulo,

        Arribo\_Motonave: Arribo\_Motonave,

        Bodega: Bodega,

        Bodega\_Tipo: Bodega\_Tipo,

        Bodega\_Propia: Bodega\_Propia,

        Modalidad: Modalidad,

        Empaque: Empaque,

        Nombre\_Motonave: Nombre\_Motonave,

        Deposito:Deposito,

        Nombre\_Articulo: Nombre\_Articulo,

        Configuracion\_Vehicular: Configuracion\_Vehicular,

        Observaciones: Observaciones,

        Fecha\_Entrada: Fecha\_Entrada,

        Unidades: Unidades,

        Escotilla: Escotilla,

        Tara: Tara,

        Peso\_a\_Cargar: Peso\_a\_Cargar,

        Peso\_a\_Cambiar: Peso\_a\_Cambiar,

        Peso\_Maximo: Peso\_Maximo,

        Saldo\_al\_BL: Saldo\_al\_BL,

        Saldo\_Solicitud: Saldo\_Solicitud,

        De\_Compartido: De\_Compartido,

        Urbano\_Directo\_Controlado: Urbano\_Directo\_Controlado,

        Documentacion\_Validada: Documentacion\_Validada

     });

 }

});

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,

     placa,

 });

}

 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',

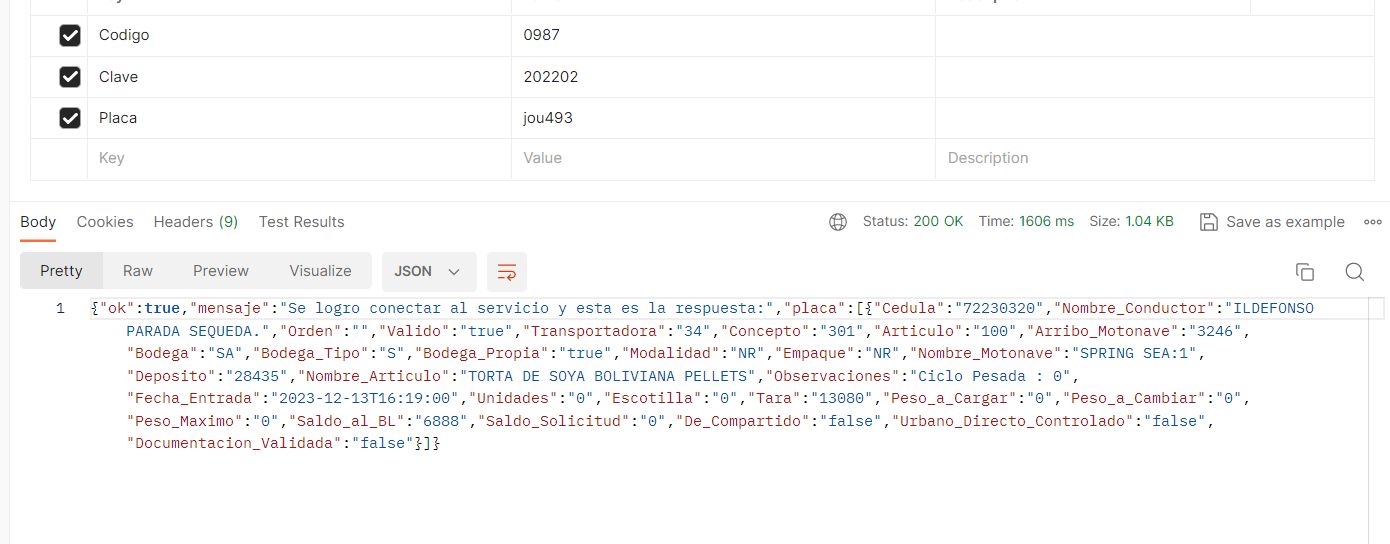
                error: error.message,

            });

             }

});

export default PlacasRoutes;



Mapear la ruta de los patios

Interface

export interface Patio {

    Codigo: string;

    Descripcion: String;

}

Tura mapeo

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Patio } 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});

// 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',

              },

        });

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

         let patios: Patio[] = [];

       // 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');

        }

        //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

        const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_PatiosResponse']['Cargar\_PatiosResult']['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'

                patios.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;

            patios.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,

            patios,

        });

    }

        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;

listo



Cargar silos es la misma configuración de las bodegas lo único que cambia es el tipo

Entonces debo de enviar el campo tipo y automáticamente carga los silos si es b son bodegas si es m son muelles y si es s son silos

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;

//este campo se debe de cambiar ya que el fornterd debe de enviar la clave encriptada despues de haber iniciado section

// 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, Tipo } = 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>${Tipo}</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');

        }

        //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

        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',

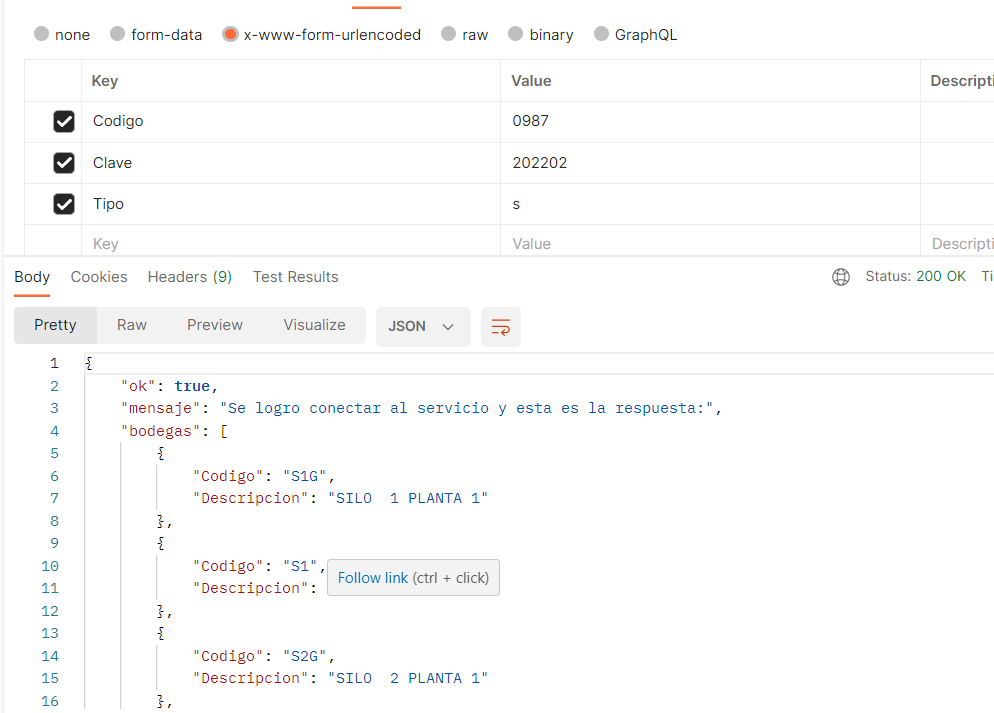
                       error: error.message,

                   });

                    }

                   });

           export default BodegasRoutes;



Ruta para asignar turno

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

import axios from "axios";

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

const crypto = require('crypto');

// datos para realizar la peticion json

const Asignar\_turno\_Routes = Router();

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

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

Asignar\_turno\_Routes.post('/Asignar\_turno',  async (req: Request, res: Response) =>{

    try{

    const { Codigo, Clave, Placa, Concepto, Bodega, Modalidad, Empaque, Unidades, Escotilla } = 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>

    <Asignar\_Turno\_Vehiculo xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        <Placa>${Placa}</Placa>

        <Concepto>${Concepto}</Concepto>

        <Bodega>${Bodega}</Bodega>

        <Modalidad>${Modalidad}</Modalidad>

        <Empaque>${Empaque}</Empaque>

        <Unidades>${Unidades}</Unidades>

        <Escotilla>${Escotilla}</Escotilla>

      </Asignar\_Turno\_Vehiculo>

       </soap:Body>

     </soap:Envelope>`;

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

            headers: {

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

                SOAPAction: 'http://tempuri.org/Asignar\_Turno\_Vehiculo',

              },

        });

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,

 });

}

 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 Asignar\_turno\_Routes;

index

//ruta agsirnar turno

server.app.use('/user', Asignar\_turno\_Routes);

cargar conceptos

interface

export interface Concepto {

    Codigo: string;

    Descripcion: String;

}

Ruta

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Concepto } 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 ConceptoRoutes = Router();

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

// const url= process.env.TZ;

const url =  urlwb;

//este campo se debe de cambiar ya que el fornterd debe de enviar la clave encriptada despues de haber iniciado section

// 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

ConceptoRoutes.post('/conceptos',  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\_Conceptos xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        </Cargar\_Conceptos>

        </soap:Body>

      </soap:Envelope>`;

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

            headers: {

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

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

              },

        });

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

         let conceptos: Concepto[] = [];

       // 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');

        }

        //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

        const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_ConceptosResponse']['Cargar\_ConceptosResult']['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'

                conceptos.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;

            conceptos.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,

            conceptos,

        });

    }

        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,

                   });

                    }

                   });

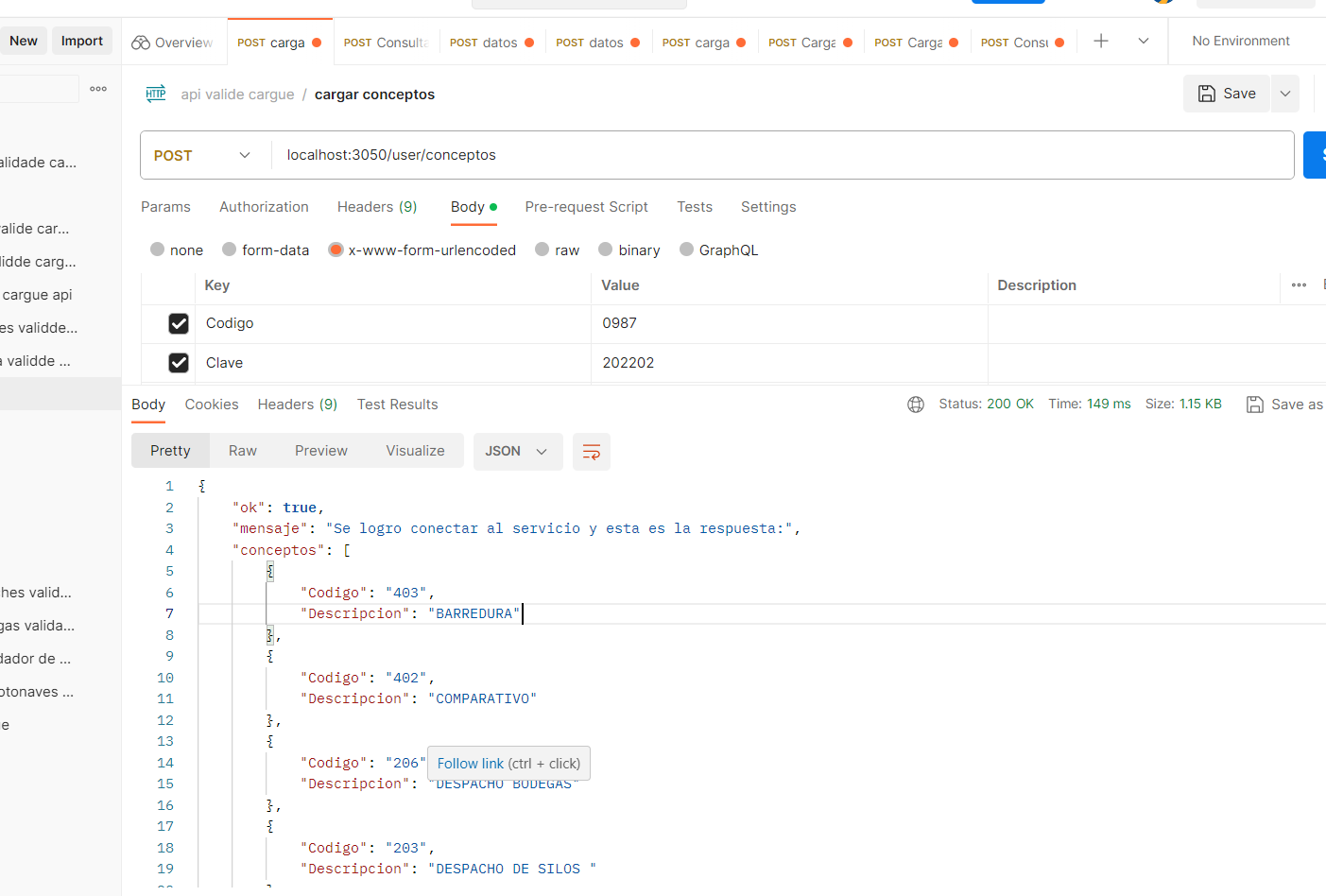
           export default ConceptoRoutes;

index

//ruta consultar conceptos

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

resultado



Ruta configuración vehicular

Interface

export interface Con\_veh {

    Codigo: string;

    Descripcion: String;

}

Ts

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Con\_veh } 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 ConfivehiRoutes = Router();

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

// const url= process.env.TZ;

const url =  urlwb;

//este campo se debe de cambiar ya que el fornterd debe de enviar la clave encriptada despues de haber iniciado section

// 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

ConfivehiRoutes.post('/configuracion\_veh',  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\_Configuracion\_Vehicular xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

      </Cargar\_Configuracion\_Vehicular>

        </soap:Body>

      </soap:Envelope>`;

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

            headers: {

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

                SOAPAction: 'http://tempuri.org/Cargar\_Configuracion\_Vehicular',

              },

        });

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

         let configuracionv: Con\_veh[] = [];

       // 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');

        }

        //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

        const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_Configuracion\_VehicularResponse']['Cargar\_Configuracion\_VehicularResult']['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'

                configuracionv.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;

            configuracionv.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,

            configuracionv,

        });

    }

        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,

                   });

                    }

                   });

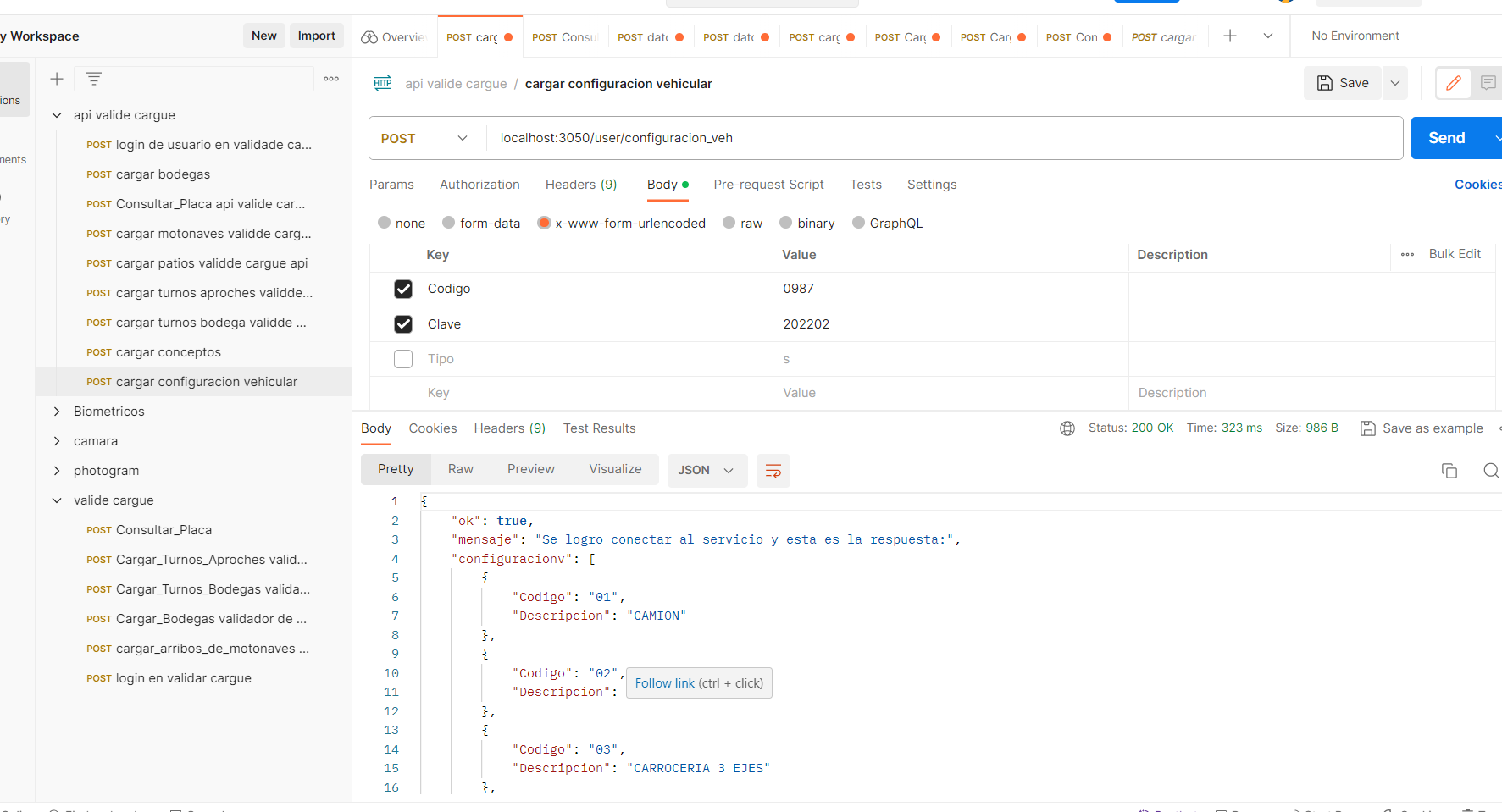
           export default ConfivehiRoutes;

index

//ruta consultar configuracion vehicular

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

resultado



RUTA cargar empaques

Interface

export interface Empaques {

    Codigo: string;

    Descripcion: String;

}

ruta

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Empaques } 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 EmpaquesRoutes = Router();

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

// const url= process.env.TZ;

const url =  urlwb;

//este campo se debe de cambiar ya que el fornterd debe de enviar la clave encriptada despues de haber iniciado section

// 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

EmpaquesRoutes.post('/empaques',  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\_Empaques xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        </Cargar\_Empaques>

        </soap:Body>

      </soap:Envelope>`;

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

            headers: {

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

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

              },

        });

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

         let empaques: Empaques[] = [];

       // 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');

        }

        //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

        const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_EmpaquesResponse']['Cargar\_EmpaquesResult']['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'

                empaques.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;

           empaques.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,

            empaques,

        });

    }

        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,

                   });

                    }

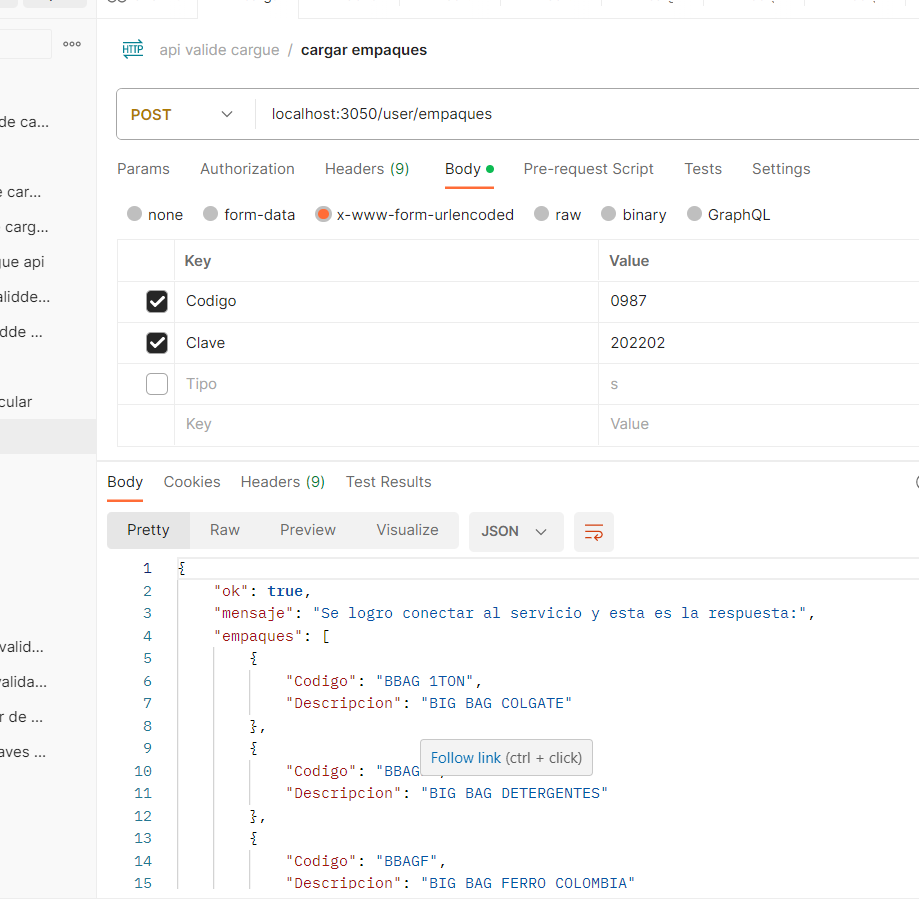
                   });

           export default EmpaquesRoutes;

index

//ruta consultar empaques

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



Ruta cargar modalidades

export interface Modalidades {

    Codigo: string;

    Descripcion: String;

}

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

import axios from "axios";

import { Parser } from 'xml2js';

import { Modalidades } 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 ModalidadesRoutes = Router();

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

// const url= process.env.TZ;

const url =  urlwb;

//este campo se debe de cambiar ya que el fornterd debe de enviar la clave encriptada despues de haber iniciado section

// 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

ModalidadesRoutes.post('/modalidades',  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\_Modalidades xmlns="http://tempuri.org/">

        <Usuario>${Codigo}</Usuario>

        <Clave>${hashedClave}</Clave>

        </Cargar\_Modalidades>

        </soap:Body>

      </soap:Envelope>`;

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

            headers: {

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

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

              },

        });

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

         let modalidades: Modalidades[] = [];

       // 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');

        }

        //esta es la esctructura de cada valor en el xml hasta llegar a los campos que se necesitan

        const informacionMovilGeneral = result['soap:Envelope']['soap:Body']['Cargar\_ModalidadesResponse']['Cargar\_ModalidadesResult']['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'

                modalidades.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;

           modalidades.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,

            modalidades,

        });

    }

        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,

                   });

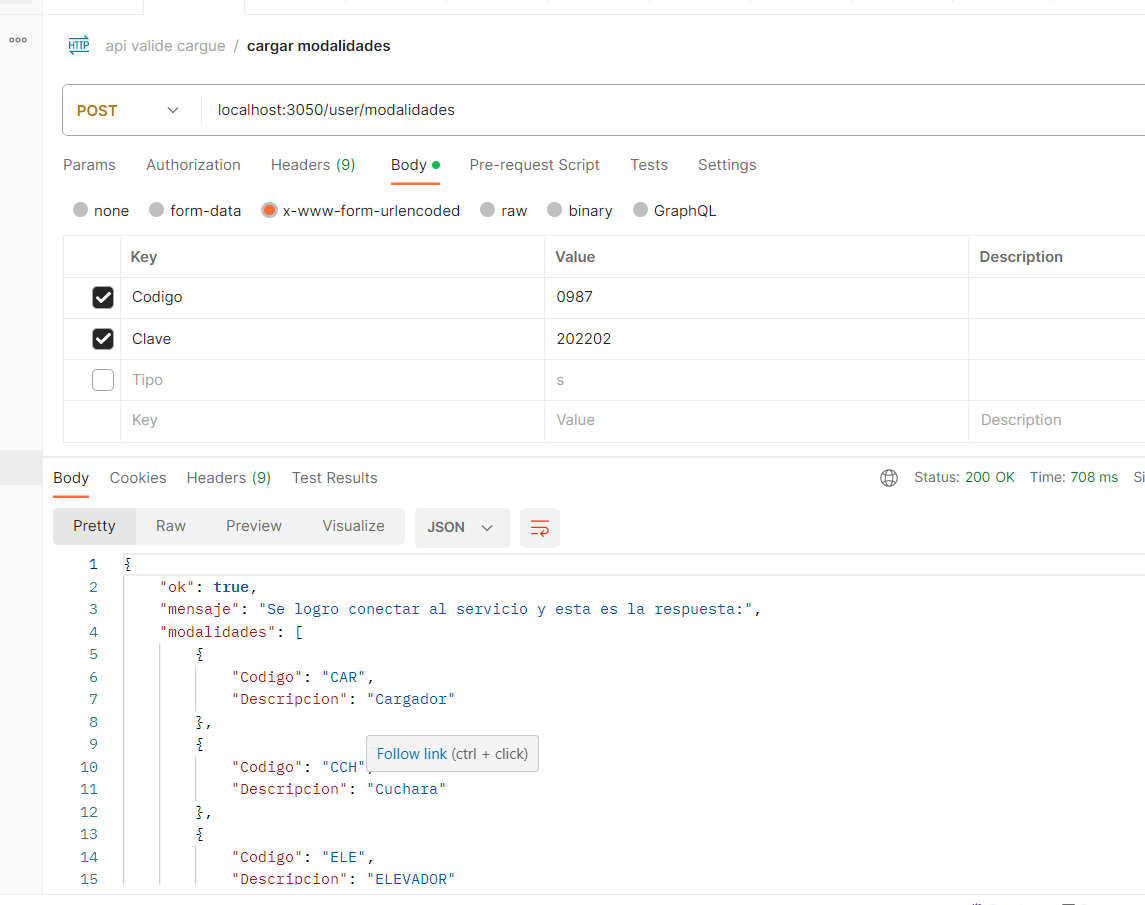
                    }

                   });

           export default ModalidadesRoutes;

//ruta consultar modalidades

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



Mapear asignar turno

Implementar el uso de variables de entorno en node js .env

01 instalamos el paquete dotenv

* npm install dotenv

02 creamos el archivo .env y agregamos las rutas

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

03 lo usamos en la aplicacion en la ruta para que inicialice las variables de entorno

//inicializao esta dependencia para utilizar las variables de entorno

require('dotenv').config();

04 acedo alas variables de entorno de la siguiente manera

// inicializo la variable de entorno

const urlwb1 = process.env.WebUrl;

//aqui paso esa variable con la dependencia de que utilizara axio para las peticiones soap

const urlwb = `${urlwb1}?wsdl`;

listo este caso es especial ya que como es un consumo de soap se debe de concatenar de esa manera

pero de por si las rutas ya se pueden importar

verificar las variables de entorno

console.log(process.env);

realizar registros a una DB msql

instalar paquete npm install mssql

en la .env debemos de tener la conexión para luego uzarla

DB\_USER=sa

DB\_PASSWORD=root

DB\_PORT=36934

DB\_SERVER=pantera.goib.com

DB\_DATABASE=estado\_hechos

DB\_ENCRYPT=true

DB\_TRUST\_SERVER\_CERTIFICATE=true

Conexión.ts

//BASE DE DATOS PRUEBA PARA ANPR EN PANTERA

const dbpantera = {

    user: process.env.DB\_USER,

    password: process.env.DB\_PASSWORD,

    port: process.env.DB\_PORT,

    server: process.env.DB\_SERVER,

    database: process.env.DB\_DATABASE,

    options : {

    encrypt: process.env.DB\_ENCRYPT === 'true' || true,

    trustServerCertificate: process.env.DB\_TRUST\_SERVER\_CERTIFICATE === 'true' || true,

    // trustedconnection : false,

    // enableArithAbort : true,

    },

Inicializo la ruta dirección require

const con = require('../classes/Config');

listo en la ruta que va a utilizar la conexión a la DB se inicializa esta dependencia

const sql = require('mssql');

en una carpeta servicio a DB colocamos este archivo con el siguiente código

//requeriendo las dependencias de mssql

const sql = require('mssql');

//requeriendo las crednciales y rutas a la DB

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

//realizar la consulta////////////////////////////////////////////////////////////////////////////

export async function verturnos(codigo:any) {

    try {

//crear la instancia de la conexión con ..class/direction

const pool = await sql.connect(con);

//console.log(codigo.Codigo)

//luego realizar la consulta

const result = await pool.request().query("select \* from shifts where code = '"+codigo.Codigo+"' ")

//imprimir result

//console.log(result);

 // Close connection después de recibir y procesar la respuesta

sql.close();

// Aquí podrías devolver el resultado o hacer cualquier otro procesamiento necesario

return result;

    } catch (err:any){

        //manejo los errores

        console.log(err);

    }

}

Y en la ruta llamamos ese método aplicando el async await

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

const ethDB = require("../serviciosdb/asignar\_turno\_DB");

const Asignar\_turno\_eth\_Routes = Router();

// inicio de proceso para asignar\_turno de res a web

Asignar\_turno\_eth\_Routes.post('/Asignar\_turno\_eth',  async (req: Request, res: Response) =>{

    try{

        const Codigo  = req.body; // estos datos vienen en la solicitud

    const result= await ethDB.verturnos(Codigo);

        if (result) {

            // 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,

            result,

            });

            }

            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 Asignar\_turno\_eth\_Routes;

en la direction que carga las variables de entorno toco modificar la ruta de port como int parse int

//BASE DE DATOS PRUEBA PARA ANPR EN PANTERA

const dbpantera = {

    user: process.env.DB\_USER,

    password: process.env.DB\_PASSWORD,

 // port: process.env.DB\_PORT,

    port: process.env.DB\_PORT ? parseInt(process.env.DB\_PORT, 10) : undefined, // Convierte el valor del puerto a un número

    server: process.env.DB\_SERVER,

    database: process.env.DB\_DATABASE,

    options : {

    encrypt: process.env.DB\_ENCRYPT === 'true' || true,

    trustServerCertificate: process.env.DB\_TRUST\_SERVER\_CERTIFICATE === 'true' || true,

    // trustedconnection : false,

    // enableArithAbort : true,

    },

Para que funcionara