Nombre del proyecto: Parkimaniacos

Codigo Api Rest Parquimetros

Equipo de trabajo:

José Ángel García Arce	NC: 17130786
Luis Emmanuel Méndez Barrios	NC: 17130804
Enrique Antonio Belmarez Meraz	NC: 17130765
José Baltazar Martínez De La Rosa	NC: 17130049

```
/*-----------------------------------------------------------------
----------------
:*                                        MANIACORP
:*                                  Fecha: AGO-DIC/2020
:*
:*     Api Rest encargada de manejar las conexiones con la BD y los Pa
rquimetros
:*
:*  Archivo     : parquimetros.js
:*  Autor       : Maniacorp Team
:*  Fecha       : 11/11/2020
:*  Compilador  : Node js
:*  Descripci□n : Api Encargada de realizar las conexiones con la bd
y los parquimetros
:*                 Se declararon varias rutas las cuales tiene un prop
osito en especifico
:*  Ultima modif:
:*  Fecha       Modific□            Motivo
:*================================================================
================
:*  11/11/2020  Luis       Se inicializo el proyecto
:*  15/11/2020  Luis       se declararon las rutas /parquimetros, /act
ualizar
:*  20/11/2020  Luis       se declararon las rutas /usuarios /Login /q
r_estado /checar
:*  4/12/2020   Luis       se declararon las rutas /realizar_cobro se
declararon los metodos validaciones,eliminar multa, verificarQr
:*  10/12/2020  Luis       se declararon las rutas /obtener multas /ch
ecar
:*  10/12/2020  Luis       Documentacion de codigo y refactorizacion
:*-----------------------------------------------------------------
----------------------*/

const express = require("express");
const route = express.Router();
const mysqlConnection = require("../database");

//Obtener Informacion sobre los parquimetros
route.get("/parquimetros",( res ) => {
    mysqlConnection.query("select * from parquimetros",(err,row,fields
) => {
        if(!err){
            res.json(row);
        }else{
            res.json("Se encontro un error al realizar la consulta");
```

```
        }
    });
});

//obtener informacion sobre las multas
route.get("/obtener_multas",( res ) => {
    mysqlConnection.query("select * from multas",(err,row,fields) => {
        if(!err){
            res.json(row);
        }else{
            res.json("Se encontro un error al realizar la consulta");
        }
    });
});



//Cambiar el estado del parquimetro
route.get("/actualizar", ( req , res ) =>{
    let id = req.query.id;
    let estado = req.query.estado;
    console.log(id)
    console.log(estado)
    if( id == null || estado==null){  //validar que el parquimetro env
ie datos a la api
          res.send("false");
    }else{
    let valoresAceptados =  new RegExp("[0-
9]+"); //validar que los datos enviados son validos
    if( id.match(valoresAceptados) ){
        if( estado == "1" || estado == "0"){
            mysqlConnection.query("update parquimetros set estado = ?
where id_ubicacion = ?",[estado,id],(err) =>{
                    if(!err){
                       if(estado == 0) {
                          eliminar_multa(id);
                       }
                       res.send("true");
                    }else{
                       res.send("false");
                    }
            });
          }
       }
   }
```

```
});


//verificar que el usuario haya leido el qr
function verificarQr(dato){
    return new Promise ( (resolve,reject) =>{
        setTimeout(() => {
            mysqlConnection.query("select * from parquimetro_cliente w
here ubicacion=?",[dato],(err,rows)=>{
                if(err){
                    throw err
                }else{
                if(rows.length<=0){
                    resolve(false);
                }else{
                    resolve(true);  // indicar al parquimetro que leyo
 el qr
                }
            }
            });
         }, 100); //tiempo despues de detectar un auto
        }
    )
}

//elimina los parquimetros que no estan siendo ocupados
function eliminar_multa(id_parquimetro){
     mysqlConnection.query("CALL delete_Data(?)",[id_parquimetro],(err
)=>{
         if(err)
             throw err;
     });
}


//validar si el usuario o correo ya existe
function validaciones(correos,user){
        return new Promise((resolve) =>{
            var estado = 0
            mysqlConnection.query("select correo from clientes where c
orreo=?",[correos],(err,rows) =>{
                if(!err){
                    if(rows.length > 0){
                        resolve(-2)
                    }else{
```

```
                        mysqlConnection.query("select usuario from cli
entes where usuario=?",[user],(err,rows) =>{
                            if(!err){
                            if(rows.length > 0){
                                resolve(-1)
                            }else{
                                resolve(1)
                            }}
                    })
                    }
                }
        })

    })
}



//Registrar Usuario
route.post("/usuarios", (req,res) =>{
    const {user,password,nombre,apellido,correo} = req.body;
    if(user == null || password == null || nombre == null || nombre==n
ull || apellido == null || correo==null){
        res.json({"estado":"Datos Incompletos","flag":"false"})
    }else if(user.trim().length < 1 || password.trim().length < 1 || n
ombre.trim().length < 1 || apellido.trim().length < 1 || correo.trim()
.length < 1){
        res.json({"estado":"Datos Incompletos","flag":"false"})
    }
    else{
        (async ()=>{
            const valor = await validaciones(correo,user)
              if(valor >=0){
                  const query = `
                     CALL registroUsuario(?,?,?,?,?)`;
                     mysqlConnection.query(query,[user,password,nombre
,apellido,correo],(err) =>{
                         if(!err){
                             res.json({"estado":"Registro Realizado","
flag":"true"});
                             console.log("registro exitoso")
                         }
                         });
                    }else if(valor == -2){
```

```
                        res.json({"estado":"el correo ya esta registr
ado","flag":"false"})
                    }else if (valor == -1){
                        res.json({"estado":"el usuario ya existe","fl
ag":"false"})
                    }
          })()
    }
});

//Iniciar Sesion en la aplicacion
route.post("/Login",(req,res) =>{
    let usuario  = req.body.usuario;
    let password  = req.body.password;
    console.log(usuario,password)
    if( usuario==null || password==null){
        res.json({"estado":"Datos Incorrectos"})
    }else if (usuario.trim().length < 1 || password.trim().length < 1)
{
        res.json({"estado":"Datos Incorrectos"})
    }
    else{
            const query =  `
            CALL loginUsuario(?,?)`;
            mysqlConnection.query(query,[usuario,password],(err,rows,f
ields) => {
            if(err){
                res.json({"estado":"Error en el servidor"});
            }else{
                if(rows.length<=2){
                    res.json({"estado":"El usuario o la contraseña son
 incorrectos"})
                }else{
                    res.json({"nombre":rows[0][0].nombre,
                    "apellido":rows[0][0].apellido,
                    "estado": "Bienvenido"
                    });
                }
        }
    })
}});


//cambiar el estado del parquimetro si el usuario leyó el qr
route.post("/qr_estado",(req,res) =>{
```

```
    let usuario = req.body.usuario;
    let id_parquimetro = req.body.id_parquimetro;
    if( usuario==null || id_parquimetro==null){
        res.json({"estado":"Errores en los Datos","flag":"false"})
    }else if (usuario.trim().length < 1 || id_parquimetro.trim().lengt
h < 1){
        res.json({"estado":"Errores en los Datos","flag":"false"})
    }else{
        let promise =  new Promise(resolve =>{
            let estado = 0
             mysqlConnection.query("select estado from parquimetros wh
ere id_ubicacion=?",[id_parquimetro],(err,rows)=>{
                 try{
                     resolve(rows[0].estado)
                 }catch(err) {
                     console.log("Error Parquimetro no registrado")
                 }
             });
        });
    (async ()=>{
         let estado =  await promise
         if(estado == 0){
            res.json({"estado":"Estacione el auto","flag":"false"})
         }else{
             //validar que el parquimetro no este siendo usado por otr
o usuario
            const query = `
            CALL uso_parquimetro(?,?)`;
            mysqlConnection.query(query,[usuario,id_parquimetro],(err,
rows)=>{
                if(err)
                   throw err;
                else{
                    if(rows[0][0].estado == "false"){
                        res.json({"estado":"El parquimetro ya esta ocu
pado por otro usuario","flag":"true"})
                    }else{
                        res.json({"estado":"Estacionado","flag":"true"
})
                    }
                }
           })
         }
    })();
   }
```

```
});


//obtener informacion sobre el parquimetro sobre si leyó el qr o no lo
 leyó
route.get("/checar",(req,res)=>{
    console.log("entro")
    let parquimetro = req.query.id;
    if(parquimetro == null){
        res.send("false");
    }else{

        (async ()=>{
            let resultado = await verificarQr(parquimetro);
            console.log(resultado);
            if(resultado){
                eliminar_multa(parquimetro);
                res.send("true");
            }else{
                res.send("false");
            }
        })();
    }
});


//realizar pago y finalizar el uso del parquimetro
route.post("/realizar_cobro",(req,res)=>{

     let parquimetro = req.body.id_parquimetro ;
     let usuario = req.body.usuario;
     let cantidad = req.body.cantidad;

     console.log(parquimetro);
     console.log(usuario);
     console.log(cantidad);

     mysqlConnection.query("insert into pagos_realizados(usuario,id_pa
rquimetro,cantidad) values(?,?,?)",[usuario,parquimetro,cantidad],(err
)=>{
                  if(err){
                      res.json({estado:"Problemas de Conexion ",flag:"
false"});
                  }else{
                    res.json({estado:"Pago Realizado",flag:"true"});
```

```
                }
    });
});


module.exports = route;


/*-------------------------------------------------------------------
----------------
:*                                        MANIACORP
:*                                   Fecha: AGO-DIC/2020
:*
:*     Api Rest encargada de manejar las conexiones con la BD y los Pa
rquimetros
:*
:*  Archivo     : database.js
:*  Autor       : Maniacorp Team
:*  Fecha       : 11/11/2020
:*  Compilador  : Node js
:*  Descripci□n : Api Encargada de realizar las conexiones con la bd
y los parquimetros
:*                  Se declararon varias rutas las cuales tiene un prop
osito en especifico
:*  Ultima modif:
:*  Fecha       Modific□            Motivo
:*=================================================================
================
:*  10/11/2020  Luis  Declaracion de la conexión y el método connect
:*  10/12/2020  Luis  Documentacion de codigo y refactorizacion
:*-------------------------------------------------------------------
----------------------*/

//importar modulo mysql
const mysql = require("mysql");

//Crear conexión a la base de datos
const mysqlConnection = mysql.createConnection({
    host:"localhost",
    user:"root",
    password:"&Root.312",
    database: "parquimetro"
});

//realizar la conexion a la base de datos
mysqlConnection.connect(function(err){
```

```
    if(err){
        console.log(err);
    } else {
        console.log("Connected to DB");
    }
});

//exportar modulo
module.exports = mysqlConnection;



/*-------------------------------------------------------------------
----------------
:*                                        MANIACORP
:*                                  Fecha: AGO-DIC/2020
:*
:*     Api Rest encargada de manejar las conexiones con la BD y los Pa
rquimetros
:*
:*  Archivo     : database.js
:*  Autor       : Maniacorp Team
:*  Fecha       : 11/11/2020
:*  Compilador  : Node js
:*  Descripci?n : Api Encargada de realizar las conexiones con la bd
y los parquimetros
:*                 Se declararon varias rutas las cuales tiene un prop
osito en especifico
:*  Ultima modif:
:*  Fecha       Modific?            Motivo
:*===================================================================
================
:*  10/11/2020  Luis  Configuracion del sevidor
:*  10/12/2020  Luis  Documentacion de codigo y refactorizacion
:*-------------------------------------------------------------------
--------------------*/

//importar modulos a utilizar
const express = require("express");
const app = express();

// Configuracion del Servidor
app.set( "port" , process.env.PORT || 3000);

//Middlewares
```

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

//Rutas
app.use(require("./routes/parquimetros"));

//Iniciar el Servidor
app.listen(3000,() => {
    console.log("Server on port ",app.get("port"))
});
```

Estándares utilizados

El estándar utilizado para el desarrollo de esta api fue ECMAScript que es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO/IEC 22275:2018.

Anexos

Utilizamos el framework llamado node js para desarrollar la api rest así como también utilizamos mysql como gestor de base de datos.

El código se encuentra en github el link es el siguiente:

`https://github.com/LEMB1999/Api_rest_parquimaniaco.git`