24-6-2021

PROYECTO 1

ANÁLISIS Y DISEÑO DE SISTEMAS 1

MANUAL TÉCNICO

Objetivos

- Alcances del sistema
- Explicación técnica
- Guía de uso
- Sección de solución de problemas

Requerimientos mínimos para la app

- Windows 10, Linux 16.04 o Linux 20.04
- Navegador Firefox, Microsoft Edge, brave, Google Chrome, opera.
- Base de datos (mysql v 08.0)
- Procesador Procesador de x86 o x64 bits de doble núcleo de 1,9 gigahercios (GHz) o más con el conjunto de instrucciones SSE2
- 4 gb de RAM
- Servidor nodejs
- Para el desarrollo de nodejs tener instalado visual studio y la carga de trabajo de desarrollo de nodejs
- Para la app móvil se requiere android 8.0 o versiones más recientes.

Servidor (Back-end)

En este apartado se desarrolla la conexión cliente servidor, donde el cliente se comunica con el api de nodejs y este se comunica con la base de datos que es mysql.
Node.js utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente. Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud http.

- Instalar nodejs
- Verificacion de la versión de ***node -v*** si no está instalado usar el siguiente comando ***install nodejs***
- Verificación versión de ***npm -v*** si no está instalado usar el siguiente comando para instalar ***npm install -g npm*** este comando sirve para instalar globalmente
- Crear un directorio donde desea ubicar el proyecto de nodejs
- Instalar librerías necesarias para la creación del proyecto o

  Npm install express –save
  Npm install body-parser –save
  Npm install nodemon –save

Comunicación de node.js con la base de datos (mysql)

```
or > src > database > TS database.ts > ...
import mysql from 'promise-mysql';
import config from './config';
const pool = mysql.createPool(config.database);

pool.getConnection()
    .then(connection =>{
        pool.releaseConnection(connection);
        console.log('Conectado a la base de datos');
    });
export default pool;
```

Para utilizar el código import mysql from 'promise-mysql' es necesario descargar e instalar algunos módulos de mysql desde nodejs, este comando npm install mysql se ejecuta en la cosola (cmd o consola visual stuido).

Crear conexión

Ahora se realiza la conexión a la base de datos, utilizando el nombre y la contraseña de la base de datos de MySql.

```
import pg, { Pool } from 'pg'

export const pool=new Pool(
    {
        user: 'szikcpxrgtvilb',
        host:'ec2-3-212-75-25.compute-1.amazonaws.com',
        password:'ee37b621e61edfcfaa6fccba8087818924fd96caaf4d002e589b70719bc804db',
        database:'d3pflou1n0s24i',
        port:5432,
        ssl:{
            rejectUnauthorized:false
        }
    }
);
```

Consulta a la base de datos

Utilizando sentencias SQL para crear, actualizar, insertar y eliminar se utilizaron los siguientes comandos desde un archivo creado en nodejs. Como se muestra en la siguiente figura.

```
/** METODO PARA RETORNAR TODOS LOS USUARIOS */
export const getUsuarios = async (req:Request,res:Response):Promise<Response>=>{
    try{
        const response:QueryResult = await pool.query('SELECT * FROM usuario');
        return res.status(200).json(response.rows);
    }catch(e){
        console.log(e);
        return res.status(500).json('Internal Server error');
    }
}
/** METODO PARA RETORNAR UN USUARIO DEPENDIENO EL ID */
export const getUsuarioPorId = async (req:Request,res:Response):Promise<Response> =>{
    const id = parseInt(req.params.id);
    const response = await pool.query('SELECT * FROM usuario WHERE idUsuario=$1',[id]);
    return res.json(response.rows);
}
```

```
/** METODO PARA RETORNAR TODOS LOS EMPLEADOS*/
export const getEmpleados = async (req:Request,res:Response):Promise<Response>=>{
    try{
        const response:QueryResult = await pool.query('SELECT * FROM empleado');
        return res.status(200).json(response.rows);
    }catch(e){
        console.log(e);
        return res.status(500).json('Internal Server error');
    }
}
/** METODO PARA RETORNAR UN EMPLEADO DEPENDIENDO DEL ID */
export const getEmpleadoPorId = async (req:Request,res:Response):Promise<Response> =>{
    const id = parseInt(req.params.id);
    const response = await pool.query('SELECT * FROM empleado WHERE idEmpleado=$1',[id]);
    return res.json(response.rows);
}
```

```
/** METODO PARA EL LOGIN */
export const login = async (req:Request,res:Response)=>{
    const {usuario,password} = req.body;
    console.log(usuario,password);
    const response:QueryResult = await pool.query('SELECT * FROM usuario WHERE usuario=$1 AND password=$2',[usuario,password]);
    console.log(response.rows);
    if(response.rows.length>0){
        return res.status(200).json(response.rows);
    }else{
        return res.status(201).json(response.rows);
    }
}
/** METODO PARA EL LOGIN EMPLEADO */
export const loginEmpleado = async (req:Request,res:Response)=>{
    const {usuario,password} = req.body;
    console.log(usuario,password);
    const response:QueryResult = await pool.query('SELECT * FROM empleado WHERE usuario=$1 AND password=$2',[usuario,password]);
    console.log(response.rows);
    if(response.rows.length>0){
        return res.status(200).json(response.rows);
    }else{
        return res.status(201).json(response.rows);
    }
}
/** RETORNAR REPORTES */
export const getReportes = async (req:Request,res:Response):Promise<Response>=>{
    const response:QueryResult = await pool.query('SELECT * FROM reporte');
    return res.status(200).json(response.rows);
}
```

```
/** RETORNAR REPORTES POR ID */
export const getReportesPorId = async (req:Request,res:Response):Promise<Response>=>{
    const id = parseInt(req.params.id);
    const response = await pool.query('SELECT * FROM reporte WHERE idUsuario=$1',[id]);
    return res.status(200).json(response.rows);
}
/** RETORNAR REPORTES POR ID */
export const getReportesPorIdReporte = async (req:Request,res:Response):Promise<Response>=>{
    const id = parseInt(req.params.id);
    const response = await pool.query('SELECT * FROM reporte WHERE idReporte=$1',[id]);
    return res.status(200).json(response.rows);
}
/** GUARDAR REPORTE */
export const crearReporte = async (req:Request,res:Response)=>{
    try{
    const {zona,fechaReporte,horaReporte,fechaProblema,horaProblema,descripcion,idTipoProblema,idUsuario} = req.body;
    const response:QueryResult = await pool.query('INSERT INTO reporte (zona,fechaReporte,horaReporte,fechaProblema,horaProblema
                                                  +'VALUES($1,$2,$3,$4,$5,$6,$7,$8)',
    [zona,fechaReporte,horaReporte,fechaProblema,horaProblema,descripcion,idTipoProblema,idUsuario]);
    const retornar = await pool.query('SELECT MAX(idReporte) as idReporte FROM reporte');
    //console.log(retornar.rows[0].idreporte);
    const idReporte = retornar.rows[0].idreporte;
    return res.json({
        idReporte:idReporte,
        estado:true});
    }catch(e){
        return res.status(400).json("No se creo el reporte");
    }

}
```

EndPoints (Servicio de Angular)

El servicio que a continuación se muestra contiene los métodos que hacen peticiones por medio de endpoints.

```
private API:string = "https://proyecto1-ayd1.herokuapp.com";

constructor(private http:HttpClient,private route:Router) { }

public getAllEmpleados():Promise<Empleado>{
  return this.http.get<Empleado>(`${this.API}/empleados`).toPromise()
}

public Loguin(user:string,password:string):Promise<Empleado[]>{
  const body = {usuario:user,password:password}
  return this.http.post<Empleado[]>(`${this.API}/login/empleado`,body).toPromise()
}

public GetTipoProblemas():Promise<TipoProblema[]>{
  return this.http.get<TipoProblema[]>(`${this.API}/tipoProblemas`).toPromise()
}

public GetAllReportes():Promise<any[]>{
  return this.http.get<any[]>(`${this.API}/detalleProblema`).toPromise()
}

public GetOneReport(id:string):Promise<Reporte[]>{
  return this.http.get<Reporte[]>(`${this.API}/reportePorId/${id}`).toPromise()
}

public GetOneUserVecino(id:string):Promise<any[]>{
  return this.http.get<any[]>(`${this.API}/usuario/${id}`).toPromise()
}

public GetImageOfReport(id:string):Promise<string[]>{
  return this.http.get<string[]>(`${this.API}/imagenReporteId/${id}`).toPromise()
}
```