Proyecto de lector de huella

Metodos Numericos

Grupo: 3402

Integrantes:

Francisco Daniel Gregorio Guevara
Emiliano Rivera Facio
Francisco David Colin Lira
Angel Sanchez Martinez
Angeles Joselyn Avila Gomez
Gabriel Acevedo Hernandez
Irbin Azael Verona Olivares

Objetivo:

IMPLEMENTAR UN DISPOSITIVO QUE CUMPLA CON LAS CARACTERÍSTICAS DE CAPTURAR LA HUELLA DACTILAR DEL USUARIO Y DE ACUERDO A LO QUE QUIERA REALIZAR LE OTORGUE PERMISO PARA MOVER EN ESTE CASO UN SERVOMOTOR A 90 GRADOS, TAMBIÉN VISUALIZAR TODOS LOS USUARIOS REGISTRADOS EN EL SISTEMA Y PODER ELIMINAR EL USUARIO CON EL ID

CÓDIGO DEL SERVIDOR:

Se utilizo node como tecnologia para el servidor en la cual le instalamos algunas dependencias importantes como lo son, mySql para la conexión con la base de datos para almacenar los datos, multer para permitir la carga de archivos en este caso las imágenes que queremos cargar en el servidor y almacenar en identificar de esta en la bd, serialPort para establecer la conexión con el dispositivo por medio de conexión serial y escuchar la data que envia el dispositivo, ws para enviarle data al cliente conectado, express, cors las cuales son fundamentales para iniciar nuestro servidor con node, entre otras

```
const express = require('express');
const http = require('http');
const cors = require('cors');
const WebSocket = require('ws');        You, 1 second ago • Uncommitted ch
const arduinoRouter = require('./routes/arduinoRouter');
const { parser } = require('./utils/arduinoConnection');
const { dataTMP } = require('./utils/dataTMP');
const {insertar, obtenerNumeroIds} = require('./models/modelBD');
const fs = require('fs');
const path = require('path');
const bodyParser = require('body-parser');
const app = express();
```

Se importan las dependencias y archivos de otros ficheros los cuales eran usados en la app inicial del servidor.

```
app.use(cors());


const port = 4321;
const server = http.createServer(app);

app.use(express.json());      You, 2 months ago • Create my API
app.use('/api/arduino', arduinoRouter);


const wss = new WebSocket.Server({ server });
app.set('wss', wss);
```

Configuración inicial del servidor donde creamos instancias de las dependencias a usar, también aqui declaramos un endpoint el cual será nuestro router de las peticiones

```
wss.on('connection', (ws) => {
    console.log('Nuevo cliente conectado a WebSocket');


    ws.on('close', () => {
        console.log('Cliente desconectado de WebSocket');
    });
});
```

Aquí escuchamos los eventos del webSocket de los clientes conectados al servidor

```
parser.on('data', async (data) => {

    let jsonData;
    try {
        jsonData = JSON.parse(data);


        console.log('Datos recibidos del Arduino:', jsonData);
        if(jsonData.found){
            const id_huellaFound = jsonData.id_huellaFound;
            console.log('Datos recibidos del id:', id_huellaFound);
            const urlGetData= `http://localhost:4321/api/arduino/getDataUser/${id_huellaFound}`;

            const response = await fetch(urlGetData);
            const dataUser = await response.json();
            console.log('Datos del usuario para login:', dataUser);




            jsonData = dataUser;
            wss.clients.forEach((client) => {
                if (client.readyState === WebSocket.OPEN) {
                    client.send(JSON.stringify({ event: 'found', data: dataUser }));
                }
            });


        }else if(jsonData.numFootprints){
            console.log('Datos recibidos del num:', jsonData);
            const response = await obtenerNumeroIds();
            const numFootprintsBD = response[0].numFootprints;
            console.log('Datos de la base de datos:', numFootprintsBD);
            const numFootprintsArduino = jsonData.numFootprints;
            if(numFootprintsBD == numFootprintsArduino){
                jsonData = {message: `Los datos en la base de datos y arduino son: ${numFootprintsBD}`};
            }else{
                jsonData = {message: `Los datos en la base de datos y arduino no coinciden: ${numFootprintsBD} y ${numFootprintsArduino} respectivamente`};
            }
            wss.clients.forEach((client) => {
                if (client.readyState === WebSocket.OPEN) {
                    client.send(JSON.stringify({ event: 'numFootprints', data: jsonData }));
                    console.log('Datos enviados al cliente:', jsonData);
                }
            });

        }
```

Aqui se ejecuta la data que manda el arduino y en la cual hacemos comprobaciones de lo que envia, entre ellas como se logra apreciar es si encontró la data que buscaba entonces procedemos a mandarlo a la una ruta de la api para extraer la información del usuario y que le permita iniciar sesión al usuario y le muestre los datos en su panel

Tambien hay otra comprobación la cual el arduino manda los números de huellas registrados en su memoria del lector, luego extrae la información de la base de datos de cuantos usuarios hay y si esta no coincide lo informa

```
        jsonData = { message: cleanedData };

        console.log('Datos recibidos del Arduinos:', jsonData);
        if(jsonData.message== "Huella guardada"){

            const userDataTmp = new dataTMP();
            const dataUser = userDataTmp.getData()[0];
            console.log('Datos del usuario:', dataUser);
            const {id_huella, nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, fotoUser} = dataUser;
            console.log('Datos del usuario:', nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, fotoUser);


            const imageName = fotoUser;


            const response = await insertar(id_huella, nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, imageName);
            console.log('Datos insertados:', response);
            userDataTmp.deleteData();
        }else if(jsonData.message== "No se pudieron encontrar caracteristicas en la huella" || jsonData.message== "Error desconocido" || jsonData.message==
            const userDataTmp = new dataTMP();
            const data = userDataTmp.getData()[0];
            const {fotoUser} = data;
            const imageName = fotoUser;
            const pathImage = path.join(__dirname, 'public/uploads', imageName);
            fs.unlinkSync(pathImage);

        }
    }
    let JsonString= JSON.stringify(jsonData);
    wss.clients.forEach((client) => {
        if (client.readyState === WebSocket.OPEN) {
            client.send(JsonString);
        }
    });
    console.log('Datos enviados al cliente:', JsonString);
});
```

Aqui se sigue validando si la información no se pudo tratar como un archivo JSON, entonces la primera comprobación es si la huella se guardo entonces extraemos la data del usuario la cual se guardo temporalmente por medio de unas instancias de clases, luego de eso la insertamos en la base de datos ya que fue comprobada la información.

En otra comprobación si la data contiene un mensaje de error entonces extraemos la data por medio de la instancia y lo que hacer es obtener el contenido de la imagen y borrarlo del fichero de archivos de imágenes ya que hubo un error y no debemos guardarlo luego al final enviamos la información al cliente conectado.

```
arduinoConnection.js > ...
const { SerialPort } = require('serialport');
const Readline = require('@serialport/parser-readline');


const arduinoPort = new SerialPort({
    path: '/dev/cu.usbserial-1130',
    baudRate: 9600,
});

      You, 1 second ago • Uncommitted changes
const parser = arduinoPort.pipe(new Readline.ReadlineParser({ delimiter: '\n' }));



module.exports = { arduinoPort, parser};
```

Aqui se hace lo conexión con el arduino por medio de la serial en la cual indicamos el puerto por el cual se tiene que establecer la conexión, y los ciclos de reloj de transmisión que estos deben ser los mismos del arduino, luego se delimita hasta cuando se lee la información que es hasta que exista un salto de linea.

```
dataTMP.js > dataTMP > getData
You, 2 months ago | 1 author (You)
class dataTMP {
    constructor() {
        if (!dataTMP.sharedData) {
            dataTMP.sharedData = [];
        }
    }

    addData(data) {
        dataTMP.sharedData.push(data);
    }

    getData() {      You, 2 months ago • add class to save data
        return dataTMP.sharedData;
    }

    deleteData() {
        dataTMP.sharedData = [];
    }
}

module.exports = {
    dataTMP
};
```

La clase temporal donde guardo la data temporalmente para darle uso después

Conexión con la bd indicandole credenciales para establecer la conexión

Archivo de las routes aqui se importan las dependencias a usar

Configuracion de multer para poder hacer la carga de imágenes donde se esta indicando donde las tiene que guardar lo cual le indicamos la ruta y con que extension debe guardar y nombre y todo lo cual se obtiene del archivo que se mando a guardar, aqui se le da el formato de guardado y asi, al final indicamos que muleta use esa configuración para guardar

```
router.post('/register', upload.single('fotoUser'), arduinoController.signUp);


router.get('/getFingers', arduinoController.getIdFingers);

router.delete('/deleteFinger/:id', arduinoController.deleteFinger);

router.get('/getAllData', arduinoController.getAllData);

router.get('/verifyFinger', arduinoController.logIn);
                            import arduinoController
router.get('/getDataUser/:id', arduinoController.getDataUser);

//router.post('/uploadImage', upload.single('image'), arduinoController.uploadImage);

router.get('/getImages/:image', arduinoController.getImages);

module.exports = router;
```

Rutas de los endpoint de la api, cada una con su controlador, pero podemos observar que el endpoint /registar tiene un midlewere el cual intercepta el archivo y lo procesa y ahi es la función del multer

```
You, 60 minutes ago | 1 author (You)
const { arduinoPort } = require('../utils/arduinoConnection');
const { dataTMP }= require('../utils/dataTMP');
const {eliminar, obtener, obtenerPorId} = require('../models/modelBD');

const path = require('path');
const fs = require('fs');



const getIdFingers= async (req, res)=>{
    console.log('Comando enviado al Arduino: getHuellasId');
    const data = { command: 'getHuellasId' };
    const jsonData = JSON.stringify(data);
    arduinoPort.write(jsonData);
    res.json({ response: 'ok', message: `Enviado correctamente.` });

}
```

Aqui este archivo es el de los controladores en la cual se incluyen dependencias y luego tenemos una función de getFingers la cual manda un comando al arduino el cual es para saber cuantos usuarios hay registrados en memoria y luego se compara como al inicio de la app

```
const signUp = async (req, res) => {
    const file = req.file;
    console.log('Archivo:', file);
    const data = req.body;
    console.log('Datos enviados:', data);

    console.log('Datos enviados:', data);
    const id_huella = data.id_huella;

    console.log('ID del usuario:', id_huella);

    const {nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, command} = data;

    console.log('Datos del usuario:', nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, command);

    const userDuplicate = await obtenerPorId(id_huella);
    const huella_id_Duplicate = userDuplicate && userDuplicate.id_huella;


    if (!data) {
        res.status(400).json({ response: 'error', message: 'Los campos son requeridos, has enviado datos vacios' });
    }else if (nombre == '' || apellidoPaterno == '' || apellidoMaterno == '' || fechaNacimiento == '' || carrera == '' || correoInstituc
        res.status(400).json({ response: 'error', message: 'Los campos del usuario son requeridos' });
    }else if(huella_id_Duplicate==id_huella){
        res.status(400).json({ response: 'error', message: 'El id ya existe'});
    }else {
        const userDataTmp = new dataTMP();
        const imageName = file.fieldname + '-' + id_huella + '-' + correoInstitucional + `.${file.originalname.split('.').pop()}`;
        console.log('foto nombre:', imageName);

        userDataTmp.addData({id_huella: id_huella, nombre: nombre, apellidoPaterno: apellidoPaterno,
                apellidoMaterno: apellidoMaterno, fechaNacimiento: fechaNacimiento,
                carrera: carrera, correoInstitucional: correoInstitucional, fotoUser: imageName});

        const jsonData = JSON.stringify({command: command, huella_id: id_huella});
        arduinoPort.write(jsonData);
        console.log('Comando enviado al Arduino:', jsonData);
        res.json({ response: 'ok', message: `Enviado correctamente.` });
    }

};
```

Controlador de registro en el cual se reciben los datos del archivo enviado y se hacen las validaciones correspondiente y si todo va bien pues guardamos la data del usuario en temporal y se manda el comando al arduino indicándole que queremos registrar la data, para después escucharla en la app como anteriormente se presento.

```
const deleteFinger = async (req, res) => {
    const {id} = req.params;
    const idExist = await obtenerPorId(id);
    console.log('Datos del usuario:', idExist);
    console.log('id enviado a eliminar al Arduino:', id);
    if (!id || id == null || id == undefined || id == '' || id < 0) {
        res.status(400).json({ response: 'error', message: 'Los campos enviados no son validos' });
        return;
    }else if (!idExist || idExist == null || idExist == undefined || idExist == '' || idExist < 0) {
        res.status(404).json({ response: 'error', message: 'El id no existe' });
        return;       You, 60 minutes ago • Uncommitted changes
    } else {
        console.log('3')
        const jsonData = JSON.stringify({ command: 'deleteFinger', huella_id: id });
        const dataImage = await obtenerPorId(id);
        console.log('Datos del usuario:', dataImage);
        const imageName = dataImage.fotoUser;
        const response = await eliminar(id);
        console.log('Datos eliminados en la bd:', response);

                            (alias) namespace path
                            (alias) const path: PlatformPath
                            import path

                            Click to show 2 definitions.

        const pathImage = path.join(__dirname, '../public/uploads', imageName);
        fs.unlinkSync(pathImage);


        arduinoPort.write(jsonData);
        res.json({ response: 'ok', message: `Enviado correctamente.` });
    }
}
```

Controlador de eliminar la informaciones la cual se obtiene l id y se hace las validaciones y después si todo va bien entonces de borra la información del fichero donde guarda en la cual le indicamos que nombre es el que debe borrar y luego lo mandamos a borrar al arduino la huella.

```
const logIn = (req, res) => {
    const data= {"command": "verifyFinger"};
    console.log('Comando enviado al Arduino:', data);
    if (!data || data == null) {
        res.status(400).json({ response: 'error', message: 'Los campos son requerido' });
    } else {
        const jsonData = JSON.stringify(data);
        arduinoPort.write(jsonData);
        res.json({ response: 'ok', message: `Enviado correctamente.` });
    }
};
```

El controlador de inicio de sesión en el cual se manda el comando al arduino y este lo recibe y ejecuta la función, ya solo escuchamos lo que responda en la app

```
const getAllData = async (req, res) => {
    const response = await obtener();
    console.log('Datos de la base de datos:', response);
    res.json(response);
};


const getDataUser = async (req, res) => {
    const { id } = req.params;
    console.log('ID del usuario:', id);

    const userData = await obtenerPorId(id);
    console.log('Datos del usuario:', userData);



    if (!userData) {
        res.status(404).json({ response: 'error', message: 'No se encontraron datos' });
    } else {
        res.json(userData);
    }
}
```

Controladores para obtener la información de la base de datos asi como toda la información o por id

```
const getImages = async (req, res) => {
    console.log('Imagen solicitada:', req.params.image);
    const imageName = req.params.image;
    res.sendFile(path.join(__dirname, `../public/uploads/${imageName}`));
}
module.exports = {
    getIdFingers,
    signUp,
    deleteFinger,
    logIn,
    getAllData,
    getDataUser,
    uploadImage,
    getImages
};
```

Controlador para obtener todas las imágenes y responderlas a la solicitud y abajo estamos exportando las funciones para usarlas en el router.

```
const conexion = require("../utils/dbConection");

module.exports = {
    insertar(id_huella, nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, fotoUser) {
        return new Promise((resolve, reject) => {
            conexion.query(`
                INSERT INTO Usuarios
                (id_huella, nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, fotoUser)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?)`,
                [id_huella, nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, fotoUser],
                (err, resultados) => {
                    if (err) reject(err);
                    else resolve(resultados.insertId);
                    console.log('Datos insertados:', resultados, err);
                }
            );
        });
    },
    obtener() {
        return new Promise((resolve, reject) => {
            conexion.query(`
                SELECT nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, id_huella, fotoUser,
                TIMESTAMPDIFF(YEAR, fechaNacimiento, CURDATE()) AS EDAD
                FROM Usuarios`,
                (err, resultados) => {
                    if (err) reject(err);
                    else resolve(resultados);
                }
            );
        });
    },
    obtenerPorId(id_huella) {       You, 2 months ago • add dependencies
        return new Promise((resolve, reject) => {
            conexion.query(`
                SELECT id_huella, nombre, apellidoPaterno, apellidoMaterno, fechaNacimiento, carrera, correoInstitucional, fotoUser,
                TIMESTAMPDIFF(YEAR, fechaNacimiento, CURDATE()) AS EDAD
                FROM Usuarios
                WHERE id_huella = ?`,
                [id_huella],
                (err, resultados) => {
                    if (err) reject(err);
                    else resolve(resultados[0]);
                }
            );
        });
    },
```

```
    eliminar(id_huella) {
        return new Promise((resolve, reject) => {
            conexion.query(`
                DELETE FROM Usuarios
                WHERE id_huella = ?`,
                [id_huella],
                (err) => {
                    if (err) reject(err);
                    else resolve(`Eliminado correctamente en bd: ${id_huella} `);
                }
            );
        });
    },
    obtenerUltimoId() {
        return new Promise((resolve, reject) => {
            conexion.query(`
                SELECT MAX(id_huella) AS ultimoId
                FROM Usuarios`,
                (err, resultados) => {
                    if (err) reject(err);
                    else {
                        const ultimoId = resultados[0].ultimoId == null ? resultados[0].ultimoId : 0;
                        resolve(ultimoId);
                    }
                }
            );
        });
    },
    obtenerNumeroIds() {
        return new Promise((resolve, reject) => {
            conexion.query(`
                SELECT COUNT(id_huella) AS totalIds
                FROM Usuarios`,
                (err, resultados) => {
                    if (err) reject(err);
                    else resolve(resultados[0].totalIds);
                }
            );
        });
    }
};
```

Modelos de las bases de datos para insertar, extraer, eliminar información en la BD por medio de consultas.

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <Servo.h>
#include <ArduinoJson.h>

SoftwareSerial fingerprintSerial(2, 3);  // RX, TX
Adafruit_Fingerprint fingerprint = Adafruit_Fingerprint(&fingerprintSerial);
Servo servoMotor;
```

Código del arduino en el cual importamos Bibliotecas a utilizar entre ellas para el tratado de información es el ArduinoJson, Servo para mover el servomotor de acuerdo a como le indiquemos, La de software serial para indicar los pines a usar y por ultimo la de adafruit fingerprint para manejar el tratado de la huella dactilar y las validaciones.

```
void enviarExitoF(const char* message){
  JsonDocument docExito;
  docExito["response"]= "success";
  docExito["message"]= message;


  serializeJson(docExito, Serial);
  Serial.println();
}

void enviarErrorF(const char* message){
  JsonDocument docError;
  docError["response"]= "error";
  docError["message"]= message;


  serializeJson(docError, Serial);
  Serial.println();
}
void enviarExito(const char* message){
  Serial.println(message);
}
void enviarError(const char* message){
  Serial.println(message);
}
```

Funciones para enviar errores o éxito por medio de Json o por el serial los cuales eran escuchados por la app del servidor.

```
void setup() {
  servoMotor.attach(8);
  servoMotor.write(0);
  Serial.begin(9600);
  while (!Serial);


  fingerprintSerial.begin(57600);
  delay(5);

  if (fingerprint.verifyPassword()) {
   enviarExitoF("Lector de huellas detectado");
  } else {
    enviarErrorF("Dispositivo no detectado, revise las conexiones o reinicie");
    while (1) delay(1);
  }
}
```

Se inicializa la conexione con el lector de huellas y si este no es detectado se manda la información y en dado caso que si solo se informa y ademas aqui se inicializa todas las configuraciones de la placa y dispositivos alternos como en este caso el lector de huella y el servo indicando que pines usar.

```
void loop() {
  if (Serial.available() > 0){

    String command = Serial.readStringUntil('\n');

    procesarComando(command);

  }

  delay(50);
}
```

Aqui en la función de loop mientras el puerto este disponible esperara un comando por el serial y si llega un comando lo procesa, después espera 50 milisegundos y reinicia el proceso.

```
void procesarComando(String command) {
  JsonDocument doc;
  DeserializationError error = deserializeJson(doc, command);


  const char* comando = doc["command"];

  if (error) {
    enviarErrorF("Error al deserializar el objeto json");
    return;
  }

  if (strcmp(comando, "verifyFinger") == 0) {
    verifyFinger();
  } else if (strcmp(comando, "signUp") == 0) {
    registrarHuella(doc["huella_id"]);
  } else if (strcmp(comando, "getHuellasId") == 0) {
    numeroHuellas();
  } else if(strcmp(comando, "deleteFinger")==0){
    deleteHuellaId(doc["huella_id"]);
  }else {
    enviarErrorF("Comando desconocido");
  }
}
```

Aqui en la funcion de procesar el comando se lee por medio de un documento Json el cual lo parseamos por asi decirlo para poderlo leer y después se hace una validaciones para ver que función debe ejecutar el comando y los comandos que existen, aqui se usan las funciones que al inicio dimos para enviar los resultados.

```
void numeroHuellas(){
  JsonDocument doc;
  fingerprint.getTemplateCount();
  doc["response"]= "success";
  doc["numFootprints"] = fingerprint.templateCount;
  serializeJson(doc, Serial);
  Serial.println();
}
```

Función de numero de huellas la cual extrae el numero de huellas registradas en el lector y se guardan en formato Json y se mandan a parsear como Json y se envían.

```
void verifyFinger() {
  JsonDocument doc;
  enviarExito("Coloque su dedo para verificar");
  delay(3000);

  uint8_t p = fingerprint.getImage();
  if (p != FINGERPRINT_OK) return;

  p = fingerprint.image2Tz();
  if (p != FINGERPRINT_OK) return;

  p = fingerprint.fingerFastSearch();
  if (p != FINGERPRINT_OK) {
    enviarError("Huella no registrada, registra una nueva huella");
    return;
  }
  doc["response"]="success";
  doc["found"] = true;
  doc["id_huellaFound"]= fingerprint.fingerID;
  doc["coincidences"]= fingerprint.confidence;
  serializeJson(doc, Serial);
  Serial.println();
  abrirPuerta();
}
```

Función de verificar la huella en la cual se le indica que coloque el dedo, luego que obtiene la imagen por medio del dato uint8_t el cual solo admite numeros de 1-127 entonces lo que hace después de obtenerlo es hacer validaciones de si la imagen de tomo correctamente y si es asi manda por un Json en que id fue encontrada y con esto el servidor busca la data y aqui se abre la puerta, si en dado caso que no este registrada entonces manda que no lo esta y acaba la función.

```
void abrirPuerta() {
  int pos;
  for(pos=90; pos>0; pos--){
    servoMotor.write(pos);
    delay(2);
  }

  enviarExito("Puerta Abierta");
  delay(5000);
  for(pos=0; pos<90; pos++){
    pos+=2;
    servoMotor.write(pos);
    delay(2);
  }
  delay(1000);
}
```

La función de abrir la puerta en la cual solo se declara la posición inicial del arduino y esta la movemos con ciclos for indicándole que como cambia y espera 5 segundos y regresa el movimiento a la posición del servo.

```
void registrarHuella(uint8_t id) {
  uint8_t p=-1;
  /*
  enviarExito("Coloque su dedo para verificar primero su huella");
  p = fingerprint.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
    // La huella ya está registrada en otro ID
    enviarError("La huella ya ha sido registrada antes");
    return;
  }
  */

  p= -1;
  while (p != FINGERPRINT_OK){
    p = fingerprint.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      enviarExito("Imagen tomada");
      delay(2000);
      break;
    case FINGERPRINT_NOFINGER:
      enviarExito("Coloque su dedo el el lector");
      delay(3000);
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      enviarError("Error de comunicacion");
      break;
    case FINGERPRINT_IMAGEFAIL:
      enviarError("Error en la imagen");
      break;
    default:
      enviarError("Error desconocido");
      break;
    }
  }
```

La función de registrar la huella lo hace indicando el tipo de dato uint8_t como -1 para que este siga asi mientras no se obtiene la imagen, entonces mientras la imagen no se obtenga y se lea sigue mandando que coloque el dedo si no esta puesto en el lector, si esta puesto solo se manda que la imagen se tomo o en dado caso que existiera un error de lectura se indica este mismo.

```
p = fingerprint.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    enviarExito("Imagen convertida");
    delay(3000);
    break;
  case FINGERPRINT_IMAGEMESS:
    enviarError("Imagen muy desordenada");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    enviarError("Error de comunicacion");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    enviarError("No se pudieron encontrar caracteristicas en la huella");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    enviarError("No se pudieron encontrar caracteristicas en la huella");
    return p;
  default:
    enviarError("Error desconocido");
    return p;
}
```

Se convierte la imagen de los bytes que tomo los cuales los analiza y convierte a algo fácil de analizar los cuales son los rasgos de la imagen los convierte en array de bytes indicando las características de la imagen de la huella

```
enviarExito("Quite el dedo del lector");
delay(2000);
p = 0;
while(p != FINGERPRINT_NOFINGER){
  p= fingerprint.getImage();
}

p = -1;
//enviarExito("Coloque su dedo nuevamente en el lector");

while (p != FINGERPRINT_OK) {
  p = fingerprint.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    enviarExito("Imagen tomada");
    delay(3000);
    break;
  case FINGERPRINT_NOFINGER:
    enviarExito("Coloque su dedo nuevamente en el lector");
    delay(3000);
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    enviarError("Error de comunicacion");
    break;
  case FINGERPRINT_IMAGEFAIL:
    enviarError("Error en la imagen");
    break;
  default:
    enviarError("Error desconocido");
    break;
  }
}

  p = fingerprint.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    enviarExito("Imagen convertida");
    delay(3000);
    break;
  case FINGERPRINT_IMAGEMESS:
    enviarError("Imagen muy desordenada");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
     enviarError("Error de comunicacion");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    enviarError("No se pudieron encontrar caracteristicas en la huella");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    enviarError("No se pudieron encontrar caracteristicas en la huella");
    return p;
  default:
     enviarError("Error desconocido");
    return p;
}
```

Se indica que quite el dedo del lector y que lo vuelva a colocar para volver a analizarlo y este se vuelve a convertir.

```
enviarExito("Creando modelo");
delay(3000);

p = fingerprint.createModel();

if (p == FINGERPRINT_OK) {
  enviarExito("Existen coincidencias en las capturas de huella");
  delay(3000);
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  enviarError("Error de comunicacion");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  enviarError("No existen coincidencias en las capturas de huella");
  return p;
} else {
  enviarError("Error desconocido");
  return p;
}


p = fingerprint.storeModel(id);
if (p == FINGERPRINT_OK) {
  enviarExito("Huella guardada");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  enviarError("Error de comunicacion");
  return p;
} else if (p == FINGERPRINT_BADLOCATION) {
  enviarError("No se pudo almacenar en la locacion indicada");
  return p;
} else if (p == FINGERPRINT_FLASHERR) {
  enviarError("Error al escribir en flash");
  return p;
} else {
  enviarError("Error desconocido");
  return p;
}
}
```

Se indica que se esta creando el modelo de la huella el cual es estar guardando los datos de los array de bytes de información de la huella y después de guarda en el id indicado y si hay un error en el proceso se indica este mismo.

```
void deleteHuellaId(uint8_t id){
  uint8_t p = -1;

  p = fingerprint.deleteModel(id);

  if (p == FINGERPRINT_OK) {
    enviarExito("Huella eliminada");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    enviarError("Error de comunicacion");
  } else if (p == FINGERPRINT_BADLOCATION) {
    enviarError("No se pudo borrar en esa locacion");
  } else if (p == FINGERPRINT_FLASHERR) {
    enviarError("Error al escribir en flash");
  } else {
   enviarError("Error desconocido");
  }
}
```

La función de eliminar la huella como modelo el cual recibe como parámetros un id de tipo uint8_t que es de 1-127 y después lo elimina de los modelos y si existiera un error en el proceso de indica.

```
document.addEventListener('DOMContentLoaded', function(){
    let conectado = false;


    function mostrarModal(opcion) {
        const optionMsg=['Conexion Establecida con el Servidor WebSocket', 'Conexión cerrada con el servidor WebSocket'];
        const titleMsg=['Conexion Exitosa', 'Conexion Fallida'];
        var msg = '';
        var title = '';
        msg= opcion ? optionMsg[0] : optionMsg[1];
        title= opcion ? titleMsg[0] : titleMsg[1];
        Swal.fire({
            position: "bottom-end",
            icon: opcion ? "success" : "error",      You, 2 hours ago • Uncommitted changes
            title: title,
            text: msg,
            showConfirmButton: false,
            timer: 1500
          });
    }

    function conectarArduino(e) {
        e.preventDefault();
        if (!conectado) {
            ws = new WebSocket('ws://localhost:4321');
            ws.onopen = function() {
                mostrarModal(true);
                document.getElementById('conectarBtn').textContent = 'Desconectar de Arduino';
                document.getElementById('functions').textContent = 'Funciones disponibles';
                conectado = true;
                localStorage.setItem('conectado', conectado);

                habilitarBotones(true);
                window.ws=ws;
            };

            ws.onclose = function() {
                mostrarModal(false);
                document.getElementById('conectarBtn').textContent = 'Conectar con Arduino';
                document.getElementById('functions').textContent = 'Funciones no disponibles hasta conectar con Arduino'
                conectado = false;
                localStorage.setItem('conectado', conectado);

                habilitarBotones(false);
            };
        } else {
            ws.close();
```

Archivo index del panel de control en el front en el cual se utiliza el websocket y se indica la conexión a usar hacia el servidor y se establece y solo se muestran mensajes de lo ocurrido.

Lo mas importante aqui es mirar como se registra un usuario el cual se establece la conexión con el websocket y después de que el usuario ingreso los datos se guardan en un formData el cual se le hace una instancia y se le agregan los datos del usuario asi como el archivo, ya que este es el formato que admite multer para el tratado de archivos.

```
registrarData.addEventListener('click', async (e) => {
    e.preventDefault();

    const formData = new FormData();


    const nombre = document.getElementById('nombre').value;
    const id_huella = document.getElementById('id_huella').value;
    const apellidoPaterno = document.getElementById('apellidoPaterno').value;
    const apellidoMaterno = document.getElementById('apellidoMaterno').value;
    const correoInstitucional = document.getElementById('correoInstitucional').value;
    const carrera = document.getElementById('carrera').value;
    const fechaNacimiento = document.getElementById('fechaNacimiento').value;

    const fotoUser = document.getElementById('fotoUser')

    formData.append('command', 'signUp');
    formData.append('nombre', nombre);
    formData.append('id_huella', id_huella);
    formData.append('apellidoPaterno', apellidoPaterno);
    formData.append('apellidoMaterno', apellidoMaterno);
    formData.append('correoInstitucional', correoInstitucional);
    formData.append('carrera', carrera);
    formData.append('fechaNacimiento', fechaNacimiento);
    formData.append('fotoUser', fotoUser.files[0]);

    console.log('dataUser:', formData.get('fotoUser'));
    const res= await fetch('http://localhost:4321/api/arduino/register', {
        method: 'POST',
        body: formData,
    })
    const resp= await res.json();
    console.log(resp)

});
```

Ya los demás archivos js de las paginas son los mismo son solo peticiones a la API en donde se le pasan el dato que espere.

Aprendizajes en el transcurso del proyecto:

- Se aprendió a colaborar en equipo ya que algunos tenían otras ideas sobre el proyecto y al final se tomo la mejor decisión para plantear de forma adecuada el proyecto y hacerlo de la mejor manera.
- Manejo de tiempos, con esto nos referimos a que el tiempo lo manejamos de manera adecuada para acabar el proyecto antes de tiempo y tener tiempo para hacer otras cosas ya que algunos compañeros se dedican a trabajar o otras cosas importantes en su vida.
- Aprendimos a usar tecnologias como lo son multer que a lo mejor ninguno de nosotros había tratado y aqui se utilizo y se pudo ver el potencial de esta tecnología.
- Diseño de las paginas en el cual muchos de nosotros teníamos cosas en mente y al final se trato de plasmar algo sencillo pero agradable a la vista del usuario.
- Se aprendió bastante con este proyecto, cada quien puso su esfuerzo y dedicación para aprender cosas y comentarlas al equipo para que esto se pudiera llevar a cabo y concluirlo ya que lo tomamos como reto y este reto lo logramos.

Conclusion general:

Con la finalización de este proyecto tuvimos muchas experiencias y aprendizajes nuevos, ya que fue una amplia investigación de temas nuevos con los que no estábamos familiarizados, fue muy compleja ya que la realización de las distintas actividades llevo mucho tiempo y empeño para conocer todas las bases de el programa realizado; el lector de huellas realizado nos tiene como finalidad leer y guardar huellas dactilares para la creación de perfiles de usuarios y guardarlo en una base de datos que nos almacena los distintos datos ingresados del usuario y así realizar distintas acciones para el beneficio común o laboral; con la finalidad de este proyecto quedamos satisfechos por lograr con éxito el objetivo principal de este lector de huellas, que es el aprendizaje de nuevas cosas siendo autodidactas con el amplio repertorio de investigaciones que se realizaron para finalizar este proyecto.