ARTÍCULO DE INVESTIGACIÓN

Análisis sensado de temperatura para control de acceso con protocolo de bioseguridad a instituciones de educación superior

Sensed Temperature Analysis for Access Control with Biosafety Protocol to Higher Education Institutions

Baena, María Antonia; Hasper, Santiago; Puche, William S.

María Antonia Baena
maría_baena82141@elpoli.edu.co
Politécnico Colombiano Jaime Isaza Cadavid, Colombia
Santiago Hasper
santiago_hasper82141@elpoli.edu.co
Politécnico Colombiano Jaime Isaza Cadavid, Colombia
William S. Puche
wspuche@elpoli.edu.co
Politécnico Colombiano Jaime Isaza Cadavid, Colombia

Revista Ingenierías USBMed
Universidad de San Buenaventura, Colombia
ISSN-e: 2027-5846
Periodicidad: Semestral
vol. 13, núm. 2, 2022
ingenierias.usbmed@usb.edu.co

Recepción: 02 Marzo 2021
Aprobación: 02 Abril 2022

URL: http://portal.amelica.org/ameli/journal/536/5363548001/

DOI: https://doi.org/10.21500/20275846.5308

Resumen: El presente artículo plantea la implementación de un módulo prototipo para el control de acceso a instituciones de educación superior, en este caso el Politécnico Colombiano Jaime Isaza Cadavid. sede Poblado, a través del protocolo de bioseguridad sensado de temperatura. Esta herramienta permite la captura automática de temperatura y toma de datos personales para cada usuario que ingresa a la institución dada la situación que se presenta actualmente por el covid-19. Para la implementación se caracteriza el software y hardware de tecnologías ya existentes que se utilizan para capturar la temperatura de manera automatizada, a partir de esto se selecciona la tecnología más adecuada para la implementación utilizando librerías *open source* para modificar el código fuente según la necesidad puntual. De igual forma, se desarrolla una aplicación web para el ingreso de datos personales que reemplacen las planillas físicas por una base de datos digital.

Palabras clave: Sensado, temperatura, protocolo de bioseguridad, modulo, acceso, Raspberry Pi, aplicativo web, base de datos, IoT.

Abstract: The article proposes the implementation of a prototype module for the control of access to higher education institutions, in this case the Politecnico Colombiano Jaime Isaza Cadavid Poblado sectional through the temperature sensing biosafety protocol, the tool allows temperature capture and taking of personal data for each user who enters the institution automatically given the current situation due to Covid-19. For the implementation, the software and hardware of existing technologies that are used to capture the temperature in an automated way are characterized, from this, the most appropriate technology for the implementation is selected, using open-source libraries, to modify the source code according to the punctual need. Similarly, a web application is developed to enter personal data and replace the physical forms with a digital database.

Keywords: Temperature Sensing, Biosafety Protocol, Access Module, Raspberry Pi 3, Web Application, Database, IoT.

I. Introducción

La bioseguridad ha desempeñado un papel fundamental en la vida del ser humano, es por esto que los centros de investigación y comunidad académica estudian la mitigación del riesgo biológico laboral ante el virus covid-19 que en la actualidad ha producido cambios significativos a nivel mundial. Ante esta situación, tanto empresas como instituciones educativas buscan alternativas para controlar de manera anticipada cualquier síntoma que pueda afectar la población y así evitar la propagación entre las personas.

El Politécnico Colombiano Jaime Isaza Cadavid busca como estrategia, a través de herramientas tecnológicas de muy bajo costo, desarrollar prototipos para minimizar los riesgos en los empleados, estudiantes y directivas, atendiendo las recomendaciones del ministerio de salud. Es importante destacar que por ley, según el Ministerio de Salud y Protección Social, cada establecimiento debe tener un protocolo de bioseguridad, como lo indica la resolución número 000675 del 24 de abril de 2020.

Por lo tanto, la finalidad de este artículo es mostrar estrategias preventivas en las instituciones de educación superior, donde se evidencie parte del estado de salud de los empleados y estudiantes de acuerdo con los principales síntomas del covid-19 que se van a identificar por medio de un prototipo sensor de temperatura con una relación costo beneficio económica para la institución.

Gracias a los modelos tecnológicos de IoT (Internet of Things), la articulación de hardware y software permite solucionar de manera puntual y confiable las necesidades sobre protocolos de bioseguridad. Por lo cual, aunque se tienen algunas tecnologías para llevar estos protocolos de bioseguridad a cabo, con este prototipo se quiere mejorar el proceso de ingreso a la institución a muy bajo costo.

II. Problemática

La situación global que se vive actualmente frente a la pandemia covid-19 ha hecho que se deban tomar una serie de medidas de bioseguridad para poder regular la expansión del virus, dado que su nivel de contagio es muy alto y aún no se tiene una solución para mitigar su contagio o una vacuna segura en Colombia [1]. Debido a esto, la sociedad debe adaptarse al cambio con medidas de autocuidado y distanciamiento social.

El covid-19 puede presentar, en la persona contagiada, una serie de síntomas que pueden variar dependiendo de cada organismo, siendo los más comunes alta temperatura, tos seca y malestar general [1]. Por ello, en establecimientos o instituciones donde se pueda presentar un alto flujo de personas se tiene como protocolo principal de bioseguridad la toma de temperatura, la cual no debe sobrepasar los 37.5#C ya que esto indica una señal de alerta del organismo sobre una posible infección de un virus o una bacteria. De igual forma, la propuesta permitirá fortalecer la generación, producción, gestión y aplicación de conocimientos que contribuyan al desarrollo curricular en el marco de la investigación formativa por medio de trabajos realizados institucionalmente que transfieran sus resultados al desarrollo académico, científico, tecnológico, cultural y social de las facultades.

Actualmente, las instituciones de educación superior en Antioquia y Colombia tienen sus instalaciones cerradas debido a la cuarentena obligatoria declarada por el Gobierno Nacional. Para retomar las clases presenciales se debe continuar con los protocolos de bioseguridad dado el alto flujo de personas para ingresar a las instalaciones evitando aglomeraciones con la mejora del proceso de toma de datos y muestras de temperatura por persona; adicionalmente, las instituciones deben contar con un historial de las personas que ingresan a estas. El Politécnico Colombiano Jaime Isaza Cadavid, sede Poblado, será la muestra para la implementación del prototipo, el módulo presentará una mejora en el proceso de ingreso a la institución dado que se genera la alerta de manera automática para el acceso a las instalaciones, ya que con los datos obtenidos en tiempo real se puede tomar una decisión y también se evitará el diligenciamiento de formularios físicos para cumplir con los protocolos de bioseguridad, el distanciamiento social y ser amigables con el medio ambiente, a su vez cumpliendo con la relación costo-beneficio.

De lo anteriormente, surge la siguiente pregunta: ¿cómo se puede mejorar el ingreso a las a las instituciones de educación superior y, en nuestro caso, del Politécnico Colombiano Jaime Isaza Cadavid por medio de la toma de temperatura en una relación costo beneficio económica y confiable?

III. Propuesta de diseño

El objetivo consiste en implementar un prototipo funcional de un módulo de sensado de temperatura con un protocolo de bioseguridad para el ingreso a las instalaciones de educación superior, para este caso de muestra se tomará la sede Poblado del Politécnico Colombiano Jaime Isaza Cadavid. Para la realización de dicha labor se tienen en cuenta las siguientes consideraciones:

Identificar los requisitos y requerimientos con un planteamiento inicial del proyecto para brindarle una solución adecuada a la institución.

Caracterizar las diferentes implementaciones de visión artificial existentes, partiendo de la búsqueda en fuentes de investigaciones y artículos científicos enfocados en el control de acceso, reconocimiento de personas y sensado de temperatura.

Diseñar un protocolo de acceso de personas con base en los protocolos de bioseguridad existentes para el ingreso a la institución.

Construir un prototipo del módulo que permite el sensado de la temperatura con el fin de restringir o autorizar el acceso a las instalaciones para el control de la propagación de covid-19.

Validar el prototipo a partir de una prueba que demuestre la mejora efectiva para que el proceso de acceso cumpla con los protocolos de bioseguridad.

Se debe tener presente que la información requerida será recolectada a través de artículos médicos y literatura académica con los temas relacionados, también se desea tener en cuenta la información suministrada por la Organización Mundial de la Salud (OMS) [2] sobre los protocolos de bioseguridad para tener los recursos necesarios para implementar el módulo de sensado de temperatura. (Error 3: La referencia debe estar ligada) (Error 4: El tipo de referencia es un elemento obligatorio) (Error 5: No existe una URL relacionada)

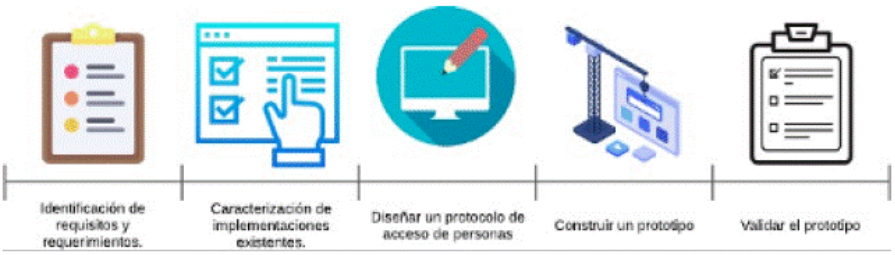

FIGURA 1.
Etapas del desarrollo.
Elaboración propia

IV. Propuesta de diseño

En esta sección se detalla el proceso realizado desde la caracterización de los dispositivos de hardware con los que ya cuenta la institución y la aplicación existente [3] e identificar los requisitos para su correcto funcionamiento con el fin de llevar a cabo la solución adecuada del problema. Por otra parte, los métodos realizados en este proyecto para los protocolos de software y hardware se enfilan a técnicas de servicios web y arquitecturas orientadas a servicios, como Ruby on Rail, frontend, React, PostgreSQL, API REST y lenguaje de programación Python; adicionalmente, para la priorización de las rutas, sensado y monitoreo del

prototipo, se implementaron una Raspberry PI 3, Camera PI y Sensor MLX90614, todo el proceso se explica a continuación:

A. Covid-19 y Protocolo de Bioseguridad

El panorama que se presenta en la actualidad es un reto sin precedentes en el manejo de las empresas para la toma de decisiones y el empleo de recursos en cuanto a material y equipos de protección en el contexto de la pandemia por covid-19, por lo cual hay que tomar en cuenta los lineamientos para el correcto uso de protocolos de bioseguridad enfocada en este caso en las empresas, el ingreso y egreso a las mismas, el uso del equipo de protección individual, uso correcto de mascarilla, toma de temperatura, correcto lavado de manos al ingreso de las instalaciones, distanciamiento social en las diferentes áreas de la compañía y de la zona de alimentación. El principal objetivo es seguir recomendaciones para mitigar el riesgo de contagio y educar al personal para que esté preparado para hacer frente a esta pandemia, se proponen diferentes medidas de protección al empleador con el fin de salvaguardar los derechos de los trabajadores, manteniendo así su contrato de trabajo.

En el caso del empleador, busca alternativas a la suspensión de labores o el despido, como el trabajo en casa, el teletrabajo, la jornada laboral flexible, las vacaciones anuales acumuladas anticipadas y colectivas, los permisos remunerados y el salario sin prestación del servicio, todo enfocado en prevenir que el empleado se contagie, ya que laborando desde casa se genera un mayor nivel de cuidado [4].

B. Sistemas de control de acceso

Los siguientes artículos hacen referencia a investigaciones e implementaciones relacionadas con sistemas para controlar el acceso.

Briones [5] diseñó e implementó un sistema para mantener el control del personal como un proceso integrado a la nómina de una empresa con el fin de controlar el ingreso y la salida de los empleados, también de validar la identidad de los mismos. Todo esto se implementó a través de un registro biométrico con la huella dactilar de cada empleado.

Rodríguez y Sánchez [6] desarrollaron un sistema domótico con el cual pueden controlar el acceso a las áreas vulnerable de una casa a través de una aplicación móvil. Esto se desarrolló con la ayuda de un Arduino. El sistema funciona mediante llamadas al celular del dueño de la casa cada vez que se active una alarma previamente configurada.

López y Ríos [7] implementaron un sistema de control de acceso para proyectos de domótica en el laboratorio LRDTBD de la Universidad Nacional del Nordeste, donde utilizaron JWT (JSON Web Token) para autenticar a los usuarios, también implementaron transferencia de archivos a través de un API REST.

C. Sistemas de control de acceso

Los siguientes artículos hacen referencia a investigaciones e implementaciones relacionadas con sistemas de medición de la temperatura utilizando diferentes herramientas y metodologías, cada uno de ellos con diferentes resultados.

Ortiz, Arias y Guerrero [8] plantearon un modelo matemático para un módulo didáctico de control donde se analiza la variable temperatura. Se simula el proceso y desarrollan estrategias de control que se aplican a un sistema de implementación real en la tecnología. La implementación del sistema presenta modelamientos matemáticos que permiten describir el comportamiento del proceso haciendo que las validaciones generen resultados confiables [8].

Núñez, González y Viloria (2012) [9] implementaron un sistema de adquisición de datos para el monitoreo de la temperatura utilizando el sensor LM35. Para lograr los objetivos de trabajo, también se utilizó una interfaz gráfica en C++ para evaluar el comportamiento del sensor. Se empleó un sistema constituido por las soluciones de NaCly H2O en estado sólido y una termocúpula utilizada como patrón para la calibración del sensor:

V. Limitaciones del prototipo

No es posible validar directamente si el visitante es empleado, estudiante ni a qué carrera pertenece, puesto que no hay un acceso a la base de datos del Politécnico [3].

La temperatura sensada por el MLX90614 no es del todo precisa ya que puede tener una precisión de ±0.5#C.

Este proceso depende de un personal que gestione el acceso a la institución.

VI. Análisis y diseño del prototipo para sensar la temperatura

Para el software se decide emplear un lenguaje open source basado en Javascript que permite realizar modificaciones más rápido, ya que el código es más ordenado y permite que los datos se asocien a las vistas, así el proceso de integración con el backend se facilitó al implementar React. En cuanto al backend, se decide implementar dos API REST, uno de ellos es el API para obtener la temperatura y el otro es el API para guardar y consultar los registros generados hasta la fecha en la base de datos [10].

A. Arquitectura

Para la implementación del sistema se utilizó la modalidad de cliente-servidor, separándolo por módulos, ya que se tiene un mejor ordenamiento de la arquitectura y mayor escalabilidad para futuros cambios y correcciones a fallas.

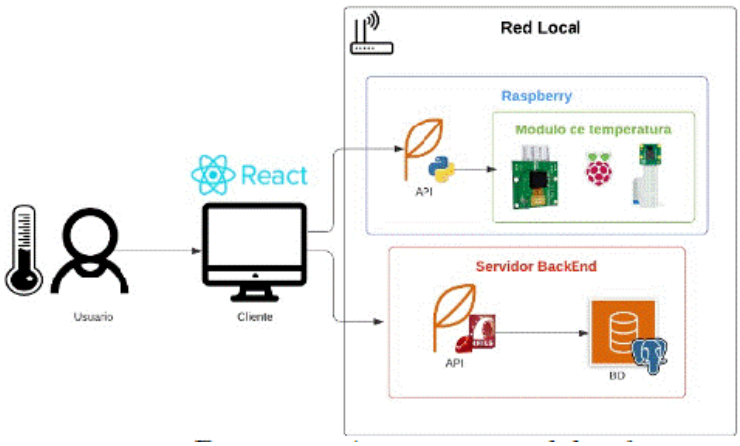

FIGURA 2.
Diagrama Arquitectura del software.
Elaboración propia

B. Hardware

Para la implementación del módulo de temperatura se decide usar diferentes herramientas como se observa en la Figura 3. Estas son de programación de bajo nivel en las cuales están la Raspberry Pi 3 que es un mini computador de una sola placa, Pi camera es un periférico diseñado especialmente para la Raspberry Pi ya que contiene un puerto especial para ello y el sistema operativo de Raspbian contiene las librerías necesarias para la configuración de la cámara, solo es necesario activar la interfaz. Por último, el sensor MLX90614 lee patrones de luz infrarroja (luz no visible para el ser humano) que permite calcular niveles de temperatura [11].

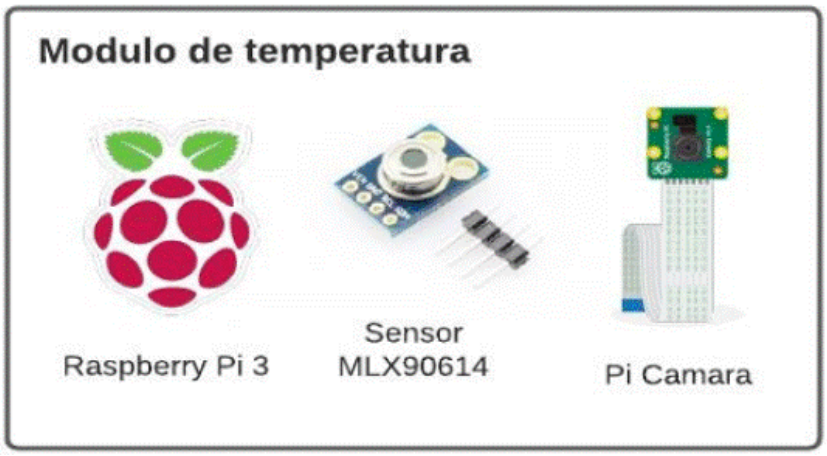

FIGURA 3.
Herramientas del módulo de temperatura
Elaboración propia

C. Software

Como se observa en la Figura 4, para el desarrollo frontend se decide utilizar React como framework, ya que este cumple con los objetivos para la integración e implementación del módulo de temperatura y la base de datos, con este lenguaje permite hacer la aplicación más dinámica y fácil de modificar. Para la base de datos se utilizó PostgreSQL. Al ser una base de datos relacional permite administrar el almacenamiento físico de datos sin afectar el acceso a esos datos como una estructura lógica, también provee buenas herramientas gráficas y de consola para ejecutar scripts, realizar consultas y respectivas modificaciones; además, es gratuita y de código abierto. En el backend se desarrollaron dos API REST con funciones completamente diferentes, para una se utilizó Python, dado que permite la integración con el módulo de temperatura de una manera más simple por la gran cantidad de documentación y desarrollos que existen en este lenguaje para el tipo de hardware que se utilizó, para la otra API se usó Rails, esta es una gema de Ruby que facilita la creación de un modelo de manera lógica y permite la configuración de la conexión con la base de datos más rápido e intuitivo, siendo esta herramienta open source [12].

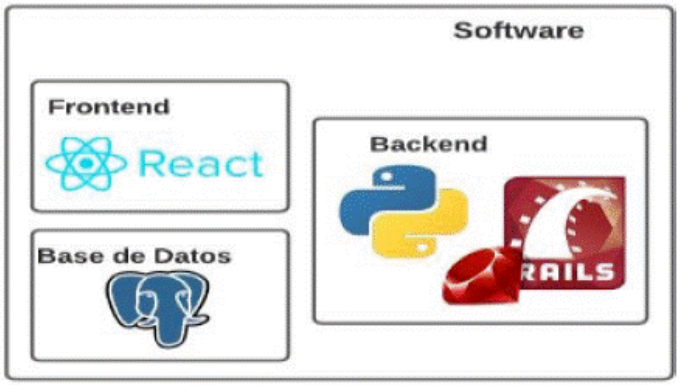

FIGURA 4.
Lenguajes de desarrollo aplicado al proyecto.
Elaboración propia

VII. Construcción del prototipo para sensar la temperatura

A. Base de datos

Para la base de datos se planteó PostgreSQL como motor, pues permite la integración fácil con cualquier backend y un modelo entidad-relación como se observa en la Figura 5.

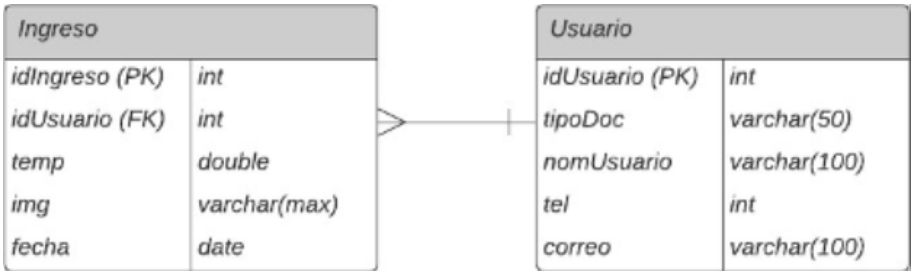

FIGURA 5.
Modelo entidad relación de la base de datos.
Elaboración propia

La creación y configuración de la base de datos se realizó a través de Rails y se ejecutaron los siguientes comandos:

sudo apt install postgresql postgresql-contrib

Se crea y se configura el usuario

sudo -u postgres createuser -s postgres -P sudo -u postgres psql postgres-# \password

postgres postgres=# create database controltest; postgres=# \q

B. API: Módulo de temperatura

Para habilitar el sensado de temperatura se debe configurar un entorno de desarrollo en Python como sigue:

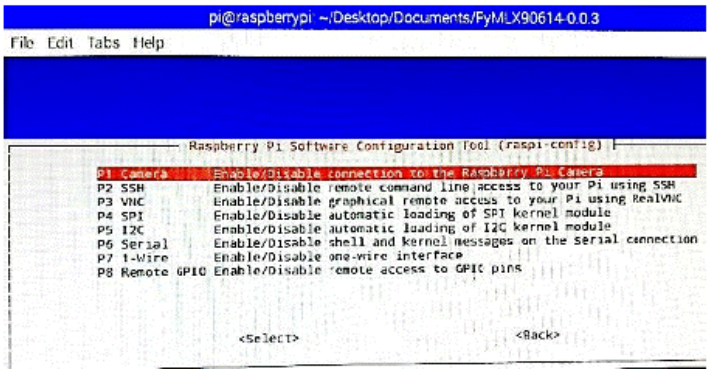

FIGURA 6.
Habilitar interfaces de la Pi Camera y i2c
Elaboración propia

Habilitar las interfaces de la Pi Camera y i2c: Ejecutar el comando "sudo raspi-config". Este comando mostrará una interfaz de configuración de la Raspberry (ver Figura 6).

Ir a la opción "Interfacing options" o en español "Opciones de interfaz" (ver Figura 7).

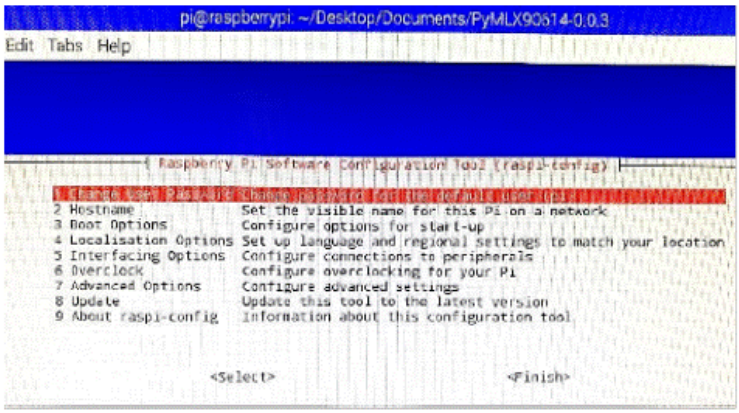

FIGURA 7.
Configuración interfaces de la Pi Camera y i2c.
Elaboración propia

Habilitar la opción "P1 Camera' ' y 'P5 IC2" (ver Figura 8).

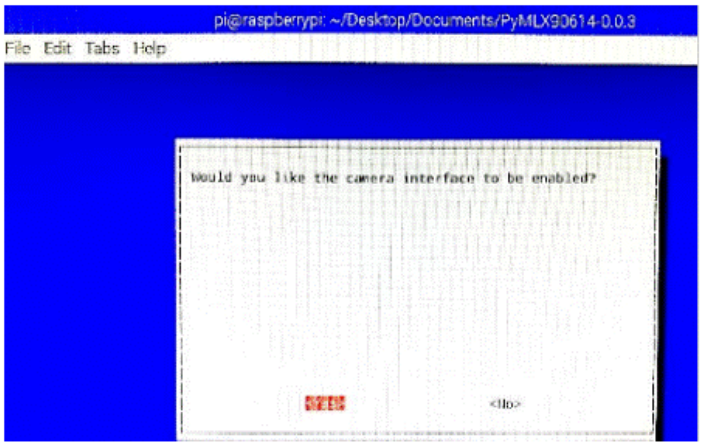

FIGURA 8.
Confirmación de la configuración de las interfaces de la PiCamera y i2c.
Elaboración propia

Descargar la librería para el sensor de temperatura MLX90614 desde el link:
https://files.pythonhosted.org/packages/67/8a/443af31ff99cca1e30304dba28a60d3f07d247c8d410822411054e170c9c/PyMLX906140.0.3.tar.gz
Ejecutando el comando:
wget https://files.pythonhosted.org/packages/67/8a/443af31ff99cca1e30304dba28a60d3f07d247c8d410822411054e170c9c/PyMLX90614-0.0.3.tar.gz
Descomprimir el archivo TAR:
tar -xf PyMLX90614-0.0.3.tar.gz
Ingresar a la carpeta ejecutando el comando:
cd PyMLX90614-0.0.3
Ejecutando el comando:
sudo python setup.py install
Después de instalar el setup.py, ejecutamos el comando de la interfaz i2c para verificar que el puerto este habilitado:
sudo apt-get install -y i2c-tools i2cdetect -y 1
Luego, descargar e instalar la librería smbus2 con los comandos:
wget https://files.pythonhosted.org/packages/97/00/47ed0ae68da93e1186fd45dbed1102469eef49dc20871ab537b69b8bcb7/smbus2-0.2.0.tar.gz
tar -x smbus2-0.2.0.tar.gz cd smbus2-0.2.0/
sudo python setup.py install
Ejecutar el siguiente comando:
Sudo su c 'echo "Y" >
/sys/module/i2c_bcm2708/parameters/combine'
Instalar request:
pip install request
Instalar Flask:
pip install Flask
Instalar Python3-venv:

sudo apt-get install python3-venv

C. Desarrollo del código

Crear dentro de la carpeta donde descomprimimos PyMLX90614-0.3.0 un archivo .py. En este caso el archivo se llamará app.py. Importamos las siguientes librerías y tenemos el siguiente código:

from flask import Flask, jsonify from picamera import PiCamera from smbus2 import SMBus from mlx90614 import MLX90614 import requests

import base64

app = Flask(name)

def leerTemperatura(variable): bus = SMBus (1)

sensor = MLX90614(bus, address=0x5A) pasa = 0

camara = PiCamera () camara.resolution = (220, 180)

x = 1

if(variable == 1):

camara.rotation = 270

camara.capture('/home/pi/Desktop/Documents/Py MLX90 614-0.0.3/captura.png')

image = open('/home/pi/Desktop/Documents/PyMLX90 614-0.0.3/captura.png', 'rb')

#open binary file in read mode image_read =

image.read()image_64_encode = base64.encodestring(image_read) if (sensor.get_object_1()>=37): pasa = 0 else: pasa = 1

params =

{'temperature':sensor.get_object_1(), 'pasa':pasa, 'img':image_64_encode}

response = requests.post('http://pythonscraping.com/pages/pro cessing.php', data = params)

senal=0 response.status_code = 0 bus.close() return params

@app.route('/temperatura') def obtener():

respuesta = jsonify(leerTemperatura(1)) ##respuesta = return respuesta

if name == ' main ': app.run(host='192.168.20.28',port=5000)

Esta última ip es la que arroja como local dentro de la red la Raspberry (192.168.20.28), con esto se tiene completamente desarrollado el módulo de temperatura exponiendo un servicio con flask en ip de red local.

D. API: Servicios de base de datos

Para los consumos a la base de datos se implementó Ruby on Rails donde se configuró la base de datos y se realizó toda la conexión, para comenzar con el desarrollo se requería previamente tener instalado Ruby (v 2.7.2p137), la gema de Rails (v rails-6.0.3.4), la gema de Postgres (v pg-1.2.3) y Nodejs (v 10.19.0) para finalmente configurar el proyecto y comenzar con el desarrollo.

Los comandos para instalar Ruby, la gema de Rails, la gema de Postgres y Nodejs son:

sudo apt-get update

sudo apt-get install ruby-full gem install rails sudo apt-get install nodejs gem install pg

Luego se crean el proyecto y los modelos, se configura la base de datos:

Se ejecuta el siguiente comando para crear el proyecto y se indica la base de datos que va a usar:

rails new prueba -d postgresql

El siguiente comando de rails para crear las bases de datos se establece por defecto para los entornos de desarrollo, producción y pruebas:

rake db:create:all

Se crean los modelos para la base de datos:

rails generate model user iduser:int tipodoc:string numdoc:int nomuser:string tel:int correo:string rails generate model entries identrie:int iduser: int temp:double img:text fecha:datetime

Se ejecuta el comando para configurar el postgres para que cualquier ip pueda acceder y el puerto (ver Figura 9): sudonano/etc/postgresql/12/main/postgresql.con f

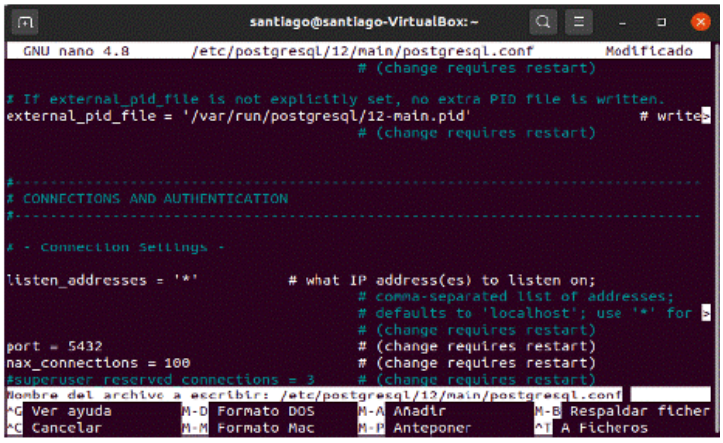

FIGURA 9
Confirmación de la configuración del postgresql.conf.
Elaboración propia

Se reinicia el postgres:
sudo systemctl restart postgresql
Se ejecuta el comando para configurar las direcciones ip (ver Figura 10):
sudo nano /etc/postgresql/12/main/pg_hba.conf

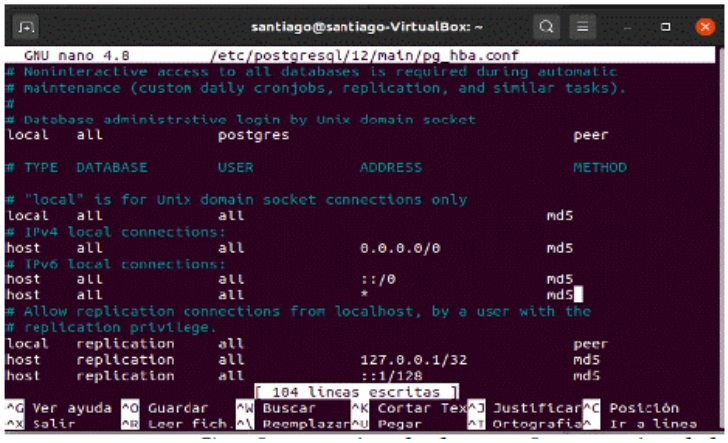

FIGURA 10.
Confirmación de la configuración del pg_hba.conf

Se reinicia el postgres:
sudo systemctl restart postgresql
Para construir el proyecto se ejecuta el siguiente comando:
rake db:migrate
Se desarrollaron 2 controladores
El controlador de usuarios (ver Figuras 11 y 12).
El controlador de entradas (ver Figuras 13 y 14).

users_controller.rb

```ruby
class UsersController < ApplicationController
    before_action :set_user, only: [:show, :edit, :update, :destroy]

    # GET /users
    # GET /users.json
    def index
      @users = User.all
      render json: @users
    end

    # GET /users/1
    # GET /users/1.json
    def show
      render json: @user
    end

    # GET /users/new
    def new
      @users = User.new
    end

    # GET /users/1/edit
    def edit
    end

    # POST /users
    # POST /users.json
    def create
        @users = User.new(user_params)
        if @users.save
            render :show, status: :created, location: @users
```

FIGURA 11.
Controlador de usuarios parte 1.
Elaboración propia

users_controller.rb

```ruby
    # POST /users.json
    def create
        @users = User.new(user_params)
        if @users.save
            render :show, status: :created, location: @users
        else
            render json: @users.errors, status: :unprocessable_entity
        end
    end

    # PATCH/PUT /users/1
    # PATCH/PUT /users/1.json
    def update
        if @users.update(user_params)
            render :show, status: :ok, location: @users
        else
            render json: @users.errors, status: :unprocessable_entity
        end
    end

    # DELETE /users/1
    # DELETE /users/1.json
    def destroy
        @users.destroy
    end

    private
      # Use callbacks to share common setup or constraints between actions.
      def set_user
        @user = User.find(params[:id])
      end
```

FIGURA 12
Controlador de usuarios parte 2.
Elaboración propia

users_controller.rb

```ruby
def show
  render json: @entry
end

# POST /entries
# POST /entries.json
def create
  @entry = Entry.new(entry_params)

      if @entry.save
          render :show, status: :created, location: @entry
      else
          render json: @entry.errors, status: :unprocessable_entity
      end
end

# PATCH/PUT /entries/1
# PATCH/PUT /entries/1.json
def update
    if @entry.update(entry_params)
      render :show, status: :ok, location: @entry
    else
      render json: @entry.errors, status: :unprocessable_entity
    end
end

private
  # Use callbacks to share common setup or constraints between actions.
  def set_entry
    @entry = Entry.find(params[:id])
  end
```

FIGURA 13.
Controlador de entradas parte 1.
Elaboración propia

users_controller.rb

```ruby
def show
  render json: @entry
end

# POST /entries
# POST /entries.json
def create
  @entry = Entry.new(entry_params)

      if @entry.save
          render :show, status: :created, location: @entry
      else
          render json: @entry.errors, status: :unprocessable_entity
      end
end

# PATCH/PUT /entries/1
# PATCH/PUT /entries/1.json
def update
    if @entry.update(entry_params)
      render :show, status: :ok, location: @entry
    else
      render json: @entry.errors, status: :unprocessable_entity
    end
end

private
  # Use callbacks to share common setup or constraints between actions.
  def set_entry
    @entry = Entry.find(params[:id])
  end
```

FIGURA 14.
Controlador de usuarios parte 2.
Elaboración propia

Para ejecutar el api se ejecuta con el comando:
rails s -p 4000 --binding=0.0.0.0

E. API: Frontend

Para el desarrollo del frontend se definió usar ReactJS. Para comenzar con el desarrollo se debe instalar el framework Create React App, un intérprete de consola (CLI) que nos va a permitir instalar React fácilmente, así que se ejecuta el siguiente comando:

npm install -g create-react-app

Luego, para crear el proyecto se ejecuta el siguiente comando:

create-react-app ProyectoTemperatura

Por último, para ejecutar el proyecto se deben ejecutar en la raíz del proyecto estos comandos:

cd ProyectoTemperatura npm start

En el desarrollo se usó la estructura base de React (ver Figura 15) y se crearon los components para el diseño de las pantallas, los helpers para crear el consumo de los servicios, los global para las constantes que se usan en el desarrollo, los styles que son los estilos para el proyecto, entre otros.

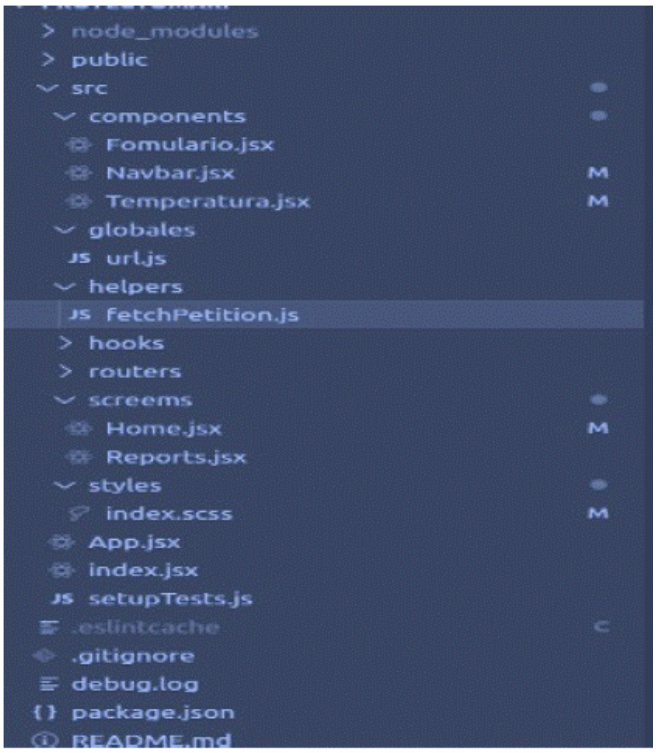

FIGURA 15.
Estructura del proyecto en React.
Elaboración propia

Los components son los 3 componentes principales de la pantalla principal como el navbar el cual representa la pantalla superior de la página, el formulario está ubicado en la parte izquierda de la pantalla, sirve para diligenciar la información básica de la persona que desee ingresar y el formulario de la temperatura que está ubicado en la parte derecha de la pantalla, este permite obtener la temperatura corporal y una foto de la persona (ver Figura 16). Para el desarrollo se usó la estructura base de React (ver Figura 15) y se crearon los components para el diseño de las pantallas, los helpers para crear el consumo de los servicios, los global para las constantes que se usan en el desarrollo, los styles que son los estilos para el proyecto, entre otros.

FIGURA 16
Componentes principales de la aplicación web.
Elaboración propia

El componente navbar es muy básico ya que solo contiene el navbar con el color de la institución, el logo y una redirección a la misma página de home, así como se observa en la Figura 17.

```jsx
src > components > Navbar.jsx > Navbar
import React from "react";
import { Link } from "react-router-dom";

const Navbar = () => {
  return (
    <>
      <ul className="menu displayFlex">
        <div>
          <img src="./assets/logo.png" alt="logo" />
        </div>
        <div className="displayFlex displayCenter">
          <li className="nav-item">
            <Link className="nav-link" to="/">
              Home
            </Link>
          </li>
        </div>
      </ul>
    </>
  );
};

export default Navbar;
```

FIGURA 17.
Componente navbar.
Elaboración propia

FIGURA 18.
Métodos del componente "Formulario" parte 1.

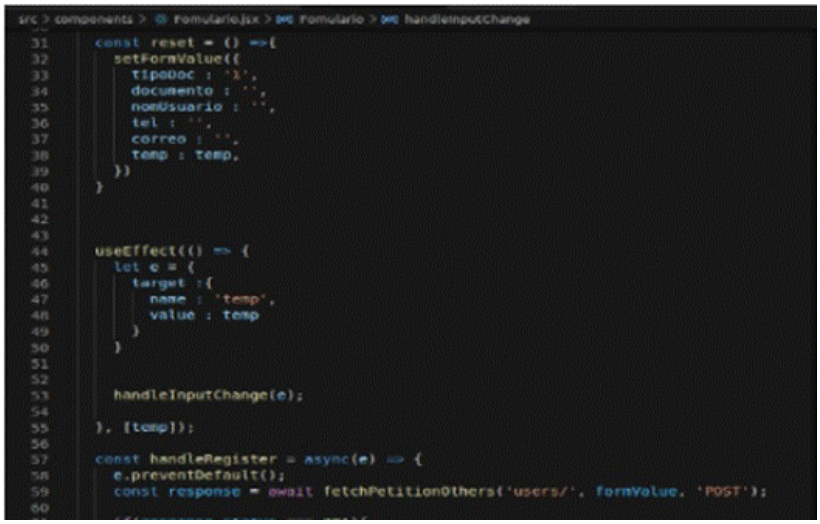

FIGURA 19.
Métodos del componente "Formulario" parte 2.
Elaboración propia

Luego, tiene la confirmación de la creación del registro y las respectivas validaciones al consultar un usuario existente, así como se observa en la Figura 20.

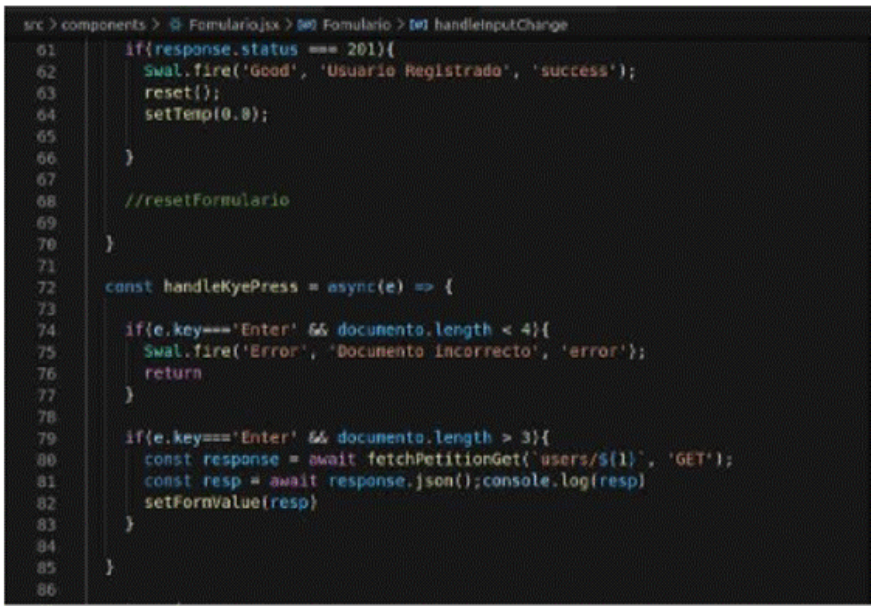

FIGURA 20.
Métodos que consume el API.
Elaboración propia

Para finalizar, contiene cómo se debe visualizar el formulario ya sea por una consulta de un usuario existente o de un usuario nuevo, este formulario tiene las respectivas validaciones en cada campo (ver Figura 21, Figura 22 y Figura 23).

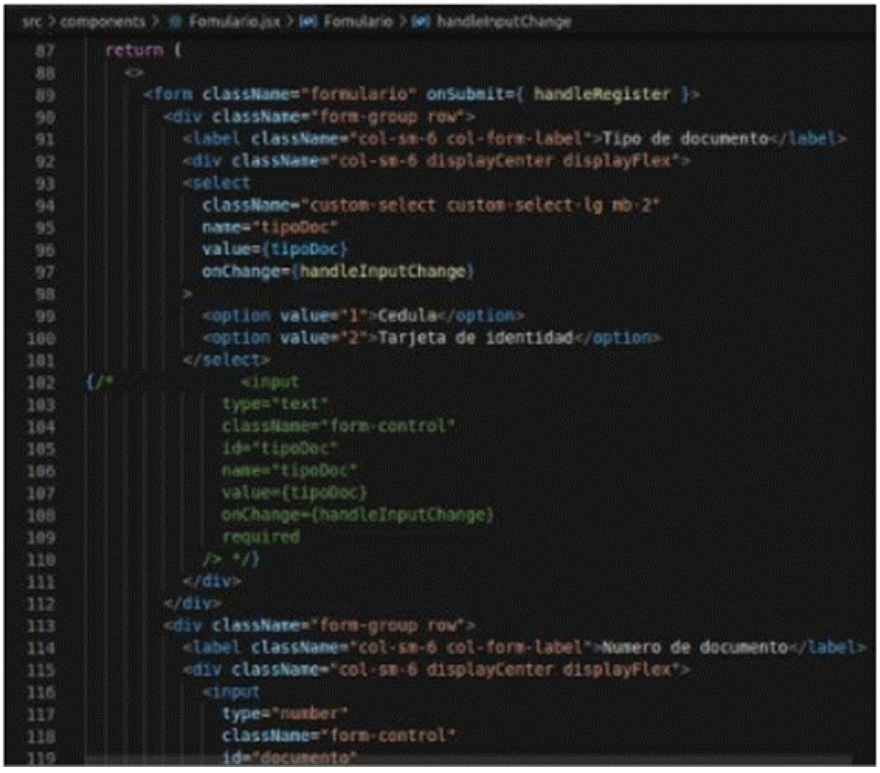

FIGURA 21.
Diseño del componente del Formulario parte 1.
Elaboración propia

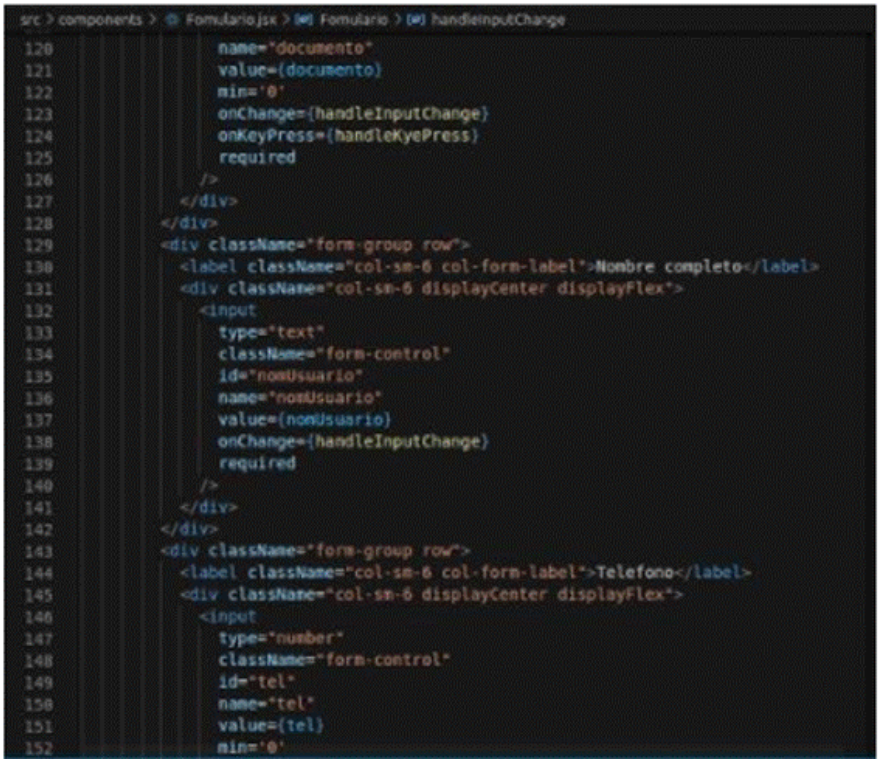

FIGURA 22.
Diseño del componente del Formulario parte 2.
Elaboración propia

El componente de temperatura comienza con la creación del modelo y el consumo del servicio para ob-

FIGURA 23.
Diseño del componente del Formulario parte 3.
Elaboración propia

tener la temperatura (ver Figura 24) y termina con el diseño de los campos que se encuentran en ese compo nente (ver Figura 25).

src > components > Temperatura.jsx > Temperatura

```jsx
import React, { useState } from "react";
import { fetchPetitionGetTemp } from "../helpers/fetchPetition";

const Temperatura = ({setTemp, temp}) => {

  const [temperature, setTemperature] = useState({
    img : null,
    pasa : 1,
    temperature : 0.0
  })

  const  {pasa, img} = temperature;

  const hanldeTemperature = async(e) => {
    e.preventDefault();
    const response = await fetchPetitionGetTemp('temperatura', 'GET');

    const resp = await response.json();

    const { img, pasa, temperature } = resp;

    console.log(resp);

    setTemperature({
      ...temperature,
      img,
      pasa,
      temperature
    })
```

FIGURA 24.
Modelo y consumo del API de temperatura.
Elaboración propia

FIGURA 25.
Diseño del componente del Temperatura.
Elaboración propia

En los helpers existe un archivo para consumir las dos API's, así que se crearon 3 métodos 1 GET para consumir el API del módulo de temperatura, 1 GET para consultar si el usuario ya está creado y 1 POST para crear el usuario nuevo que ingresa, esta información se observa en la Figura 26 y Figura 27.

FIGURA 26.
Métodos GET's para consumir servicios de las API's.
Elaboración propia

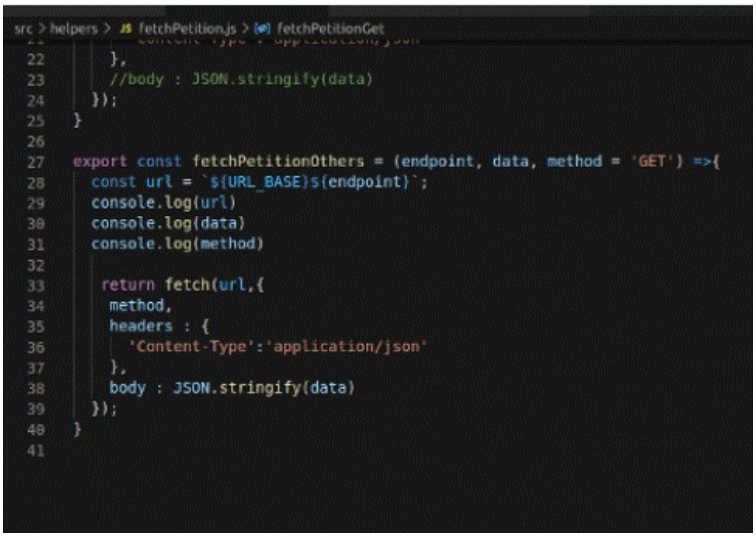

FIGURA 27.
Método POST para crear los nuevos registros en la base de datos.
Elaboración propia

En el global hay un archivo creado especialmente para las constanteS globales que se pueden consumir desde cualquier archivo con un "import" de la variable y el nombre del archivo donde se creó la variable (ver Figura 28).

```
src > globales > JS url.js > [@] URL_BASE
1  export const URL_BASE = "http://192.168.20.41:4000/";
2  export const URL_TEMP = "http://192.168.20.28:5000/";
3
```

FIGURA 28.
Constantes globales.
Elaboración propia

VIII. Análisis de resultados

En esta etapa se realizó la integración del frontend con las API's, todo esto conectado a la misma red para que el consumo fuera un poco más fácil sin necesidad de exponer los servicios en una nube, para comenzar con las pruebas y validaciones se requiere levantar todos los servicios con los siguientes comandos:

Python app.ty (API temperatura)

rails s -p 4000 --binding=0.0.0.0 (API con conexión a la base de datos) npm start (ejecutar el frontend)

Para el paso siguiente se realizan las respectivas validaciones desde la página web que se ejecuta desde http://localhost:3000, al ingresar a esa url se puede comenzar con el proceso de control de temperatura (ver Figura 29).

FIGURA 29.
Pantalla inicial para el control de temperatura.
Elaboración propia

Al oprimir obtener temperatura consume un servicio GET que está ubicado en la API de Python el cual retorna un JSON con la foto en formato base 64, la información sobre si la persona puede o no acceder dependiendo de la temperatura y la temperatura corporal, ya con esta respuesta se visualiza en la pantalla la información obtenida y el color de la temperatura es dinámico, si la persona no puede ingresar la temperatura se pone roja (ver Figura 30) de lo contrario es verde (ver Figura 31). Después de finalizar ese proceso, se comienza a completar la información requerida en el formulario para así poder generar un registro nuevo a la base de datos.

FIGURA 30.
Control del acceso basado en la temperatura en este caso NO puede acceder.
Elaboración propia

FIGURA 31.
Control del acceso basado en la temperatura en este caso sí puede acceder.
Elaboración propia

Finalmente, para terminar con todo el proceso se confirma la creación de los registros en la base de datos (ver Figura 32 y Figura 33).

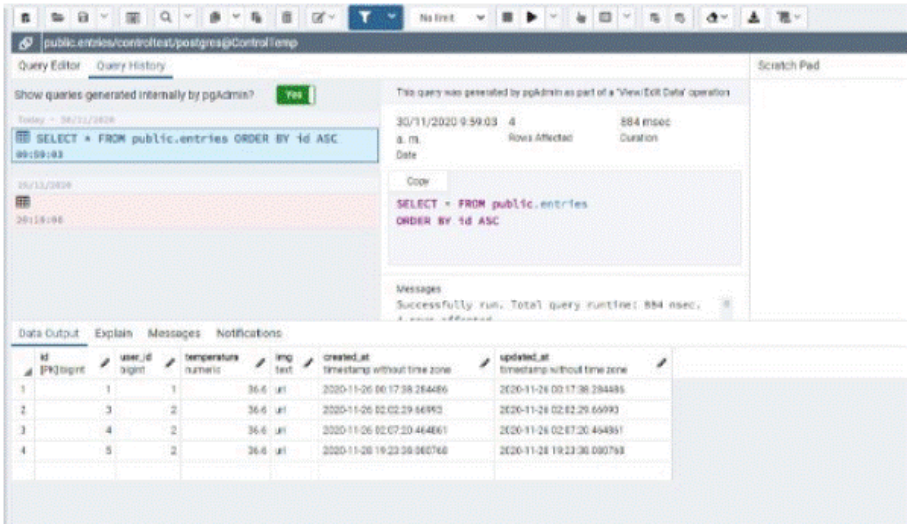

FIGURA 32.
Tabla "Entradas" base de datos PostgreSQL.
Elaboración propia

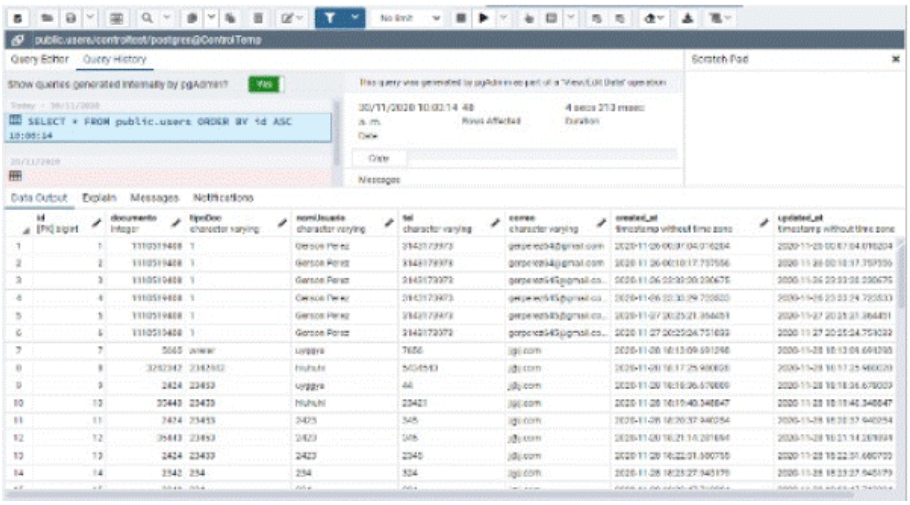

FIGURA 33.
Tabla "Usuarios" base de datos PostgreSQL.
Elaboración propia

De acuerdo a la pregunta problematizadora, finalmente, se responde con los resultados obtenidos en la validación del prototipo, donde se demostró que es posible mejorar el control del acceso a las instalaciones con ayuda de un hardware muy simple y un desarrollo escalable. Es importante resaltar que en las validaciones realizadas encontramos que para la toma de la temperatura es importante estar a 1 cm de distancia del sensor para que tenga una mejor precisión y así el porcentaje de error seria de 0.5#C, de lo contario si la distancia es mayor o menor de la recomendada el porcentaje de error puede variar y seguramente será mayor a lo establecido anteriormente teniendo en cuenta que la finalidad de nuestro prototipo es costo beneficio.

IX. Conclusiones

Se cumplieron todos los objetivos del proyecto, los cuales aportan de manera positiva a la automatización del protocol

Se logró implementar una estructura adecuada para el módulo de sensado de temperatura y la aplicación web para ejecutar el protocolo de bioseguridad de manera correcta resolviendo la pregunta problematizadora. La arquitectura del proyecto permite tener futuras mejoras de las características de la aplicación.

Con base en la literatura, las herramientas y procesos que se utilizaron, dieron los resultados esperados en los módulos de protocolo diseñados, ya que cada interfaz se pudo ejecutar de manera correcta y con precisión.

Se cumplieron todos los objetivos del proyecto, los cuales aportan de manera positiva a la automatización del protocolo de bioseguridad de sensado de temperatura para el ingreso a las instalaciones del Politécnico Colombiano Jaime Isaza Cadavid.

Por último, se logra obtener la temperatura e imagen del usuario en tiempo real, guardando su histórico en la base de datos, aportando al medio ambiente quitando los formularios físicos y automatizando los procesos de manera más ágil, lo cual cumple con el objetivo general del proyecto.

Para desarrollos futuros se recomienda utilizar un hardware de más alto nivel para que la precisión sea mayor independiente de la distancia a la cual se captura la temperatura.

Para finalizar, se logra establecer una visualización de registro de visitas en tiempo real que aporta a la seguridad y facilita el control en las instalaciones.

X. Agradecimientos

Este trabajo es soportado y adscrito a los proyectos de investigación sede central microcuantia SMC 7224, 2019, denominado "Empleo de las Tics para el monitoreo térmico avícola en la granja Román Gómez Gómez del Politécnico Colombiano Jaime Isaza Cadavid, sede Marinilla, Antioquia, Colombia", financiado por la Vicerrectoría de Docencia e Investigación del Politécnico Colombiano Jaime Isaza Cadavid.

Referencias

[1] A. M. Otoya-Tono, M. García-Chabur, C. Jaramillo-Moncayo y Á. Mahecha, "COVID-19: Generalidades, comportamiento epidemiológico y medidas adoptadas en medio de la pandemia en Colombia," *Acta de Otorrinolaringología & Cirugíade Cabeza y Cuello*, vol. 48, n.o 1, págs. 93-102, 2020.

[2] Organización Mundial Salud OMS, 2021. https://www.who.int/es.

[3] W. S. Puche y J. P. P. Duque, "Módulo de control de acceso a zonas restringidas de la granja Román Gómez Gómez del Politécnico Colombiano Jaime Isaza Cadavid sede Marinilla," Ingenierías USBMed, vol. 12, n. 2, pp. 17–32, 2021. https: //doi.org/10.21500/20275846.5071.

[4] Ministerio de Trabajo, "Circular 0021, Circular externa 0022", 2020.

[5] J. O. Briones Calvache, "Análisis y Diseño de un sistema que permita controlar el acceso y asistencia del personal para la Empresa Human Trend," Tesis de pregrado, Escuela Politécnica Nacional, Ecuador, 2010

[6] J. Rodríguez Caruajulca y J. Sánchez Flores, "Sistema domótico para controlar el acceso de áreas vulnerables de una casa mediante sensores, a través de una aplicación móvil," Tesis de pregrado, Facultad de Ciencias Físicas y Matemáticas, Uni versidad Nacional de Trujillo, Perú, 2018

[7] M. E. A. López y L. J. Rios, "Gestión del control de acceso con tecnología open source en proyectos de domótica," en XX Workshop de Investigadores en Ciencias de la Computación, Universi dad Nacional del Nordeste, 2018

[8] P. Ortiz, A. Arias y D. Guerrero, "Modelo matemático no lineal en un sistema de temperatura para un recinto cerrado," ITECKNE: Innovación e Investigación en Ingeniería, vol. 7, n.# 2, pp. 165–174, 2010

[9] B. Nuñez, J. González y P. Viloria, "Sistema de monitoreo en tiempo real para la medición de temperatura," Scientia et technica, vol. 17, n. # 50, pp. 128–131, 2012

[10] D. Atencio Flores y D. Mamani Machaca, "Interconectividad basada en API REST en aplicaciones de la Municipalidad Provincial de Lampa," Tesis de pregrado, Facultad de Ingeniería Estadística e Informática, Universidad Nacional del Altiplano, Perú, 2017

[11] J. Peris Martínez, "Sistema de monitorización inalámbrica de temperatura mediante sensor de infrarrojos y microcontrolador," Tesis de Maestría, Escuela Técnica Superior de Ingeniería del Diseño, Universitat Politècnica de València, España, 2020

[12] S. Flores Larsen y E. Hongn, "Termografía infrarroja en la edificación: aplicaciones cualitativas; Asociación Argentina de Energía Solar," Avances En Energías Renovables y Medio Ambiente, vol. 16, n. #, pp.25–32, 2012