

Proyecto #3 Interfaces Gráficas

INFO 1128 By Alberto Caro

Gabriel Muñoz, Benjamín Sanhueza
gmunoz2021@alu.uct.cl, bsanhueza2021@alu.uct.cl

INTRODUCCION

En esta entrega tenemos el objetivo de implementar los conceptos de interfaces graficas en cuanto a la conexión usuario-pagina.

Para esto se deberán desarrollarse los conocimientos de desarrollo de un proyecto, tanto en como identificar los requisitos, las herramientas a utilizar como, el operarlas, las investigaciones de nuevos procedimientos, las pruebas del software creado, optimización entre otros.

Definido por el autor, Saenz et al. quien referencia a la guía ISO 31000 del 2009, en este sentido, gestionar proyectos, exige de la implementación efectiva de procesos de planificación, ejecución y control. La precisión de cumplimientos en tiempos específicos estipulados y el hacer uso racional de recursos, es esencial en estos procesos de gestión, donde se apuesta a costos racionales a nivel de las operaciones desarrolladas y riesgos impredecibles.

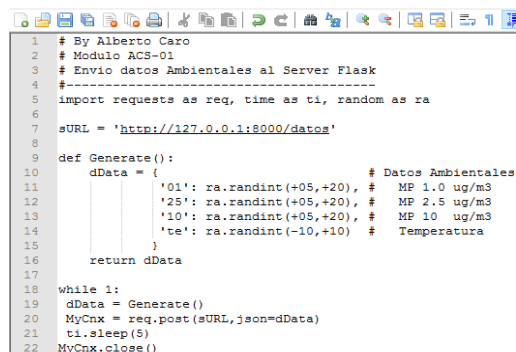
Aplicando los procedimientos con las problemáticas planteadas se deberá realizar procesos de traficación de un set de datos del sensor de material particulado, lo cual debe ser dinámico, ya que se actualizará cada cierto tiempo, así como esta grafica se debe enviar a través de la plataforma de comunicación telegram, a los cuales se utilizarán herramientas como, flask, telegram bot, entre otros.

MARCO TEORICO

- **Time:** Este módulo proporciona varias funciones relacionadas con el tiempo, proporciona un conjunto de funciones para trabajar con fechas y/o horas. Además de estas funciones hay otras relacionadas en los módulos datetime y calendar.
- **TelegramBot:** Para usar la API de Telegram Bot, primero debe obtener una cuenta de bot chateando con BotFather quien te dará un token, para poder comenzar a usar Telepot y acceder a la cuenta del bot.
- **Json:** librería para manipular archivos json asi como codificar, decodificar; El uso de objetos JavaScript (JSON) es una poderosa herramienta para intercambiar datos rápidamente entre varias plataformas de programación. Ya sea para almacenar datos o crear una API, convertir sus datos en JSON los hace reutilizables, independientemente de la tecnología que acceda a ellos.
- **Flask:** un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2.

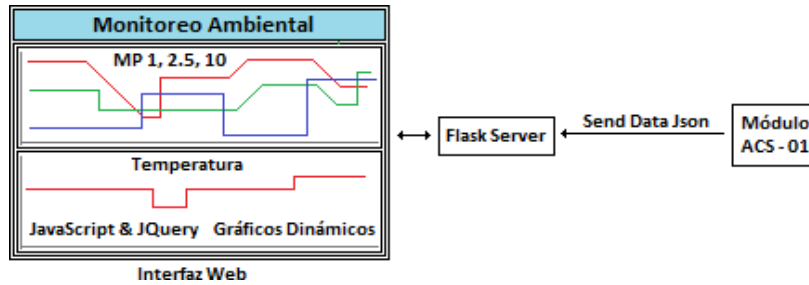
DESARROLLO

Utilizando el siguiente script resuelva los problemas planteados.

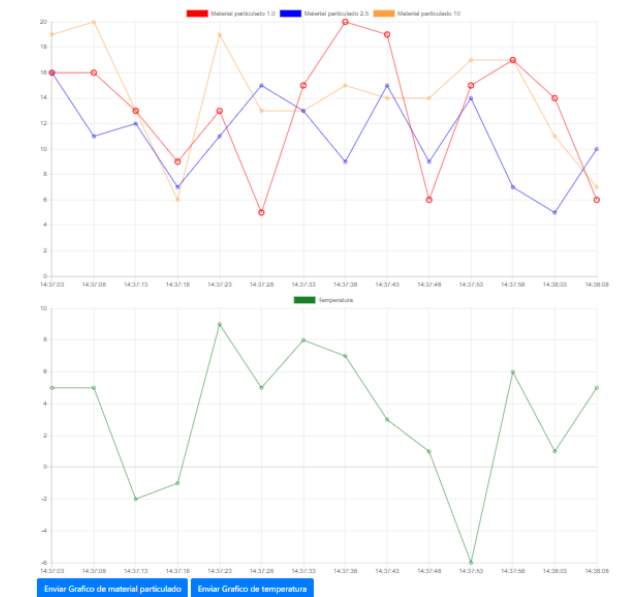


```
1 # By Alberto Caro
2 # Modulo ACS-01
3 # Envio datos Ambientales al Server Flask
4 #-----
5 import requests as req, time as ti, random as ra
6
7 sURL = 'http://127.0.0.1:8000/datos'
8
9 def Generate():
10     dData = {
11         '01': ra.randint(+05,+20), # MP 1.0 ug/m3
12         '25': ra.randint(+05,+20), # MP 2.5 ug/m3
13         '10': ra.randint(+05,+20), # MP 10 ug/m3
14         'te': ra.randint(-10,+10) # Temperatura
15     }
16     return dData
17
18 while 1:
19     dData = Generate()
20     MyCnx = req.post(sURL,json=dData)
21     ti.sleep(5)
22     MyCnx.close()
```

1.- Desarrolle una aplicación web que grafique los niveles de contaminación ambiental de material particulado **1.0, 2.5 y 10 ug/m3** y la temperatura de manera dinámica. Debe programar un **Web Server Flask** con la librería **Chart.js**. Haga los cambios que estime necesario. Sea creativo. Utilice **JSON, JQuery y Java Script**. **[35 puntos]**



Proyecto IV 5 de interfaces graficas



MODULO ACS-01

Para el módulo se utilizó el script facilitado

```
import requests as req, time as ti, random as ra

sURL = 'http://127.0.0.1:5000/data'

def Generate():
    dData = {
        '01': ra.randint(+5,+20),
        '25': ra.randint(+5,+20),
        '10': ra.randint(+5,+20),
        'te': ra.randint(-10,+10),
    }
    return dData

while 1:
    dData = Generate()
    print(dData)
    rq = req.post(sURL,data=dData)
    ti.sleep(5)
    rq.close()
    print(rq.txt)
```

RUTA DE DATA

Para que el modulo acs-01 este habilitado, se genera una ruta en la aplicación principal, deacuerdo a los métodos get y post, se obtiene la data generada la cual se almacena en una variable global para que sea accesible en la aplicación.

```
@app.route("/data",methods=['GET', 'POST'])
def hello():

    if request.method == 'POST':
        if len(Dtime)> 50:
            Dtime.pop(0)
            data.pop(0)
            print(request.data)
            data.append(request.form)
            Dtime.append({
                "time": time.strftime("%H:%M:%S")
            })
            print(Dtime)

    return {"success": True}
```

RUTA PRINCIPAL

Como antes se definió la ruta de data se debe tener una ruta de acceso para visualizar el grafico, por lo cual con los datos almacenados en los data se utilizan, al ser renderizados por el archivo html.

```
@app.route("/getdata",methods=['GET', 'POST'])
def getdata():

    if request.method == 'POST':
        return [data, Dtime]
    else:
        return render_template('test.html')
```

HTML

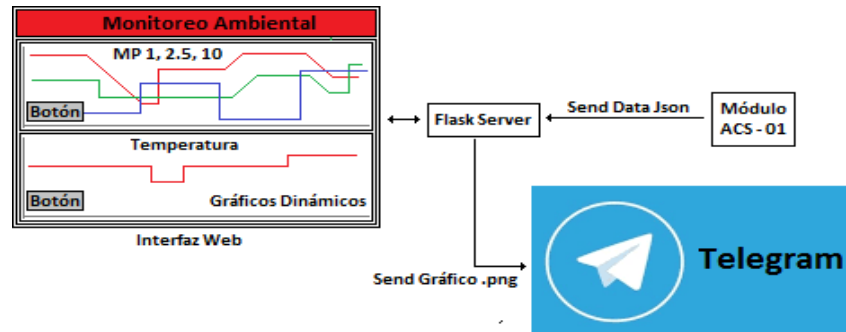
En cuanto a lo anteriormente definido, obtenemos los datos anteriormente generados y enrutados, verificamos y realizamos la gráfica, así separando los datos obtenidos en variables de almacenado, a lo cual de acuerdo a la estructura del char se realiza, para ser mostrado por pantalla.

```
<script>
    var Last;
    function delay(milliseconds){
        return new Promise(resolve => {
            setTimeout(resolve, milliseconds);
        });
    }
    var backgroundColor = 'white';
    Chart.plugins.register({
        beforeDraw: function(c) {
            var ctx = c.chart.ctx;
            ctx.fillStyle = backgroundColor;
            ctx.fillRect(0, 0, c.chart.width, c.chart.height);
        }
    });
    async function init(){
        while(true){
            fetch('/getdata',{
                method: 'POST',
                credentials: 'include',
                headers: new Headers({
                    'Content-Type': 'application/json',
                })
            }).then(response =>{
                return response.json();
            }).then( data =>{
                console.log(data)
                if(data[0][0]){
                    if(Last != data[1][data[1].length-1]["time"]){
                        console.log("true")
                        graficar(data);
                    }
                    Last = data[1][data[1].length-1]["time"];
                }else{
                    if(Last != 0){
                        graficar(data);
                        Last=0;
                        console.log("true")
                    }
                }
            }).catch(error =>console.error(error));
            await delay(5000);
        }
    }
    init();
```

```
function graficar(data){
    const $grafica = document.querySelector("#grafica");
    const $temperatura = document.querySelector("#temperatura");
    let etiquetas = [];
    let d01 = [];
    let d25 = [];
    let d10 = [];
    let dte = [];
    for (i in data[0]) {
        etiquetas.push(data[1][i]["time"].toString());
        d01.push(parseInt(data[0][i]["01"]));
        d25.push(parseInt(data[0][i]["25"]));
        d10.push(parseInt(data[0][i]["10"]));
        dte.push(parseInt(data[0][i]["te"]));
    }
    const P01 = {
        label: "Material particulado 1.0",
        data: d01,
        backgroundColor: 'transparent', // Color de fondo
        borderColor: 'rgba(255, 0, 0, 1)', // Color del borde
        borderWidth: 1, // Ancho del borde
        lineTension: 0,
        pointRadius: 6,
        pointBorderWidth: 2,
        fill: false,
    };
    const P25 = {
        label: "Material particulado 2.5",
        data: d25, // La data es un arreglo que debe tener la misma
        backgroundColor: 'transparent', // Color de fondo
        borderColor: 'rgba(0, 0, 255, 1)', // Color del borde
        borderWidth: 1, // Ancho del borde
        lineTension: 0,
        fill: false,
    };
    const P10 = {
        label: "Material particulado 10",
        data: d10, // La data es un arreglo que debe tener la misma
        backgroundColor: 'transparent', // Color de fondo
        borderColor: 'rgba(255, 159, 64, 1)', // Color del borde
        borderWidth: 1, // Ancho del borde
        lineTension: 0,
        fill: false,
    };
    const Te = {
        label: "Temperatura",
        data: dte, // La data es un arreglo que debe tener la misma
        backgroundColor: 'transparent', // Color de fondo
        borderColor: 'rgba(23, 129, 40, 1)', // Color del borde
        borderWidth: 1, // Ancho del borde
        lineTension: 0,
        fill: false,
    };
    const chart = new Chart($grafica, {
        type: 'line',
        data: {
            labels: etiquetas,
            datasets: [
                P01,
                P25,
                P10,
            ],
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    },
                }],
            },
        },
    });
    const Tchart = new Chart($temperatura, {
        type: 'line',
        data: {
            labels: etiquetas,
            datasets: [
                Te
            ],
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    },
                }],
            },
        },
    });
}
```

```
const chart = new Chart($grafica, {
    type: 'line',
    data: {
        labels: etiquetas,
        datasets: [
            P01,
            P25,
            P10,
        ],
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                },
            }],
        },
    },
});
const Tchart = new Chart($temperatura, {
    type: 'line',
    data: {
        labels: etiquetas,
        datasets: [
            Te
        ],
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                },
            }],
        },
    },
});
}
```

2.- Utilizando la solución anterior haga los cambios para que al hacer click en los botones se envíe el gráfico dinámico asociado a su **SmartPhone** cuenta **Telegram**. Utilice la librería **Canvas.js** y **Telepot** de python junto con **Web Server Flask**. Investigue y sea creativo. Utilice **JSON**, **Jquery** y **Java Script**. **[35 puntos]**



BOT DE TELEGRAM

En html definimos el elemento cual es el chart o gráfico, el cual de la data se transforma a imagen en formato png, luego es enviada a la función creada para el bot, por lo cual, en la ruta principal, se obtiene la imagen por get, a lo cual se guarda, posteriormente en la variable bot, definimos el bot de telegram teniendo el token para acceder a él, así para finalmente a través del id del usuario a enviar el mensaje, enviamos la imagen anteriormente guardada abriéndola en modo lectura binaria.

```

@app.route('/Telebot',methods=['POST'])
def Telebot():
    if request.method == 'POST':
        img = request.files.get('imagen')
        img.save('test.png')
        bot = telepot.Bot('5890117421:AAH4SL5Do8ZuXvMR7z9poV51f4dT_1C0JzE')

        bot.sendPhoto(1552297914, photo=open('test.png', "rb"))
        bot.sendPhoto(5212849344, photo=open('test.png', "rb"))

        return {"success": True}

<script>

function Enviar(element){
    var canvas = document.getElementById(element);
    console.log(canvas)

    canvas.toBlob((blob) => {

        const Fdata = new FormData();
        Fdata.append("imagen", blob, "imagen.png");
        console.log(Fdata.get("imagen"))
        fetch('/Telebot',{
            method : 'POST',
            //credentials : "include",
            body: Fdata,
        }).then(response =>{
            console.log(response);
            return response.json()
        })
    });
}

</script>

```

CONCLUSION

Por lo anteriormente expuesto, se puede concluir que el utilizar herramientas como json, request entre otros, facilitaron los enrutamientos, así como generar funciones en base a algún parámetro; flask por otro lado resulto ser un entorno minimalista ya que no necesita una estructura compleja para realizar una pagina web, permitiendo en codificación tener más libertad para realizar procedimientos; por consiguiente telegram bot, es bastante especifico en cuanto a sus funcionalidades por lo cual realizando la investigación correspondiente se realizó la comunicación según lo previsto.

Finalizando, con lo anteriormente desarrollado, la curva de aprendizaje es favorable ya que, en la capacidad de desarrollar aplicaciones, flask es un framewrok que proporciona una amplia gama de posibilidades para el desarrollo de alguna problemática.