Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Ingeniería Electrónica y Telecomunicaciones
Desarrollo de aplicaciones para plataformas ubicuas

Docente: Juan Pablo Ruiz Rosero jpabloruiz@unicauca.edu.co

5. Python Flask web framework

Una aplicación web es un programa cliente-servidor, con el cual los usuarios pueden acceder a ciertas herramientas disponibles en un servidor. Un web framework es un software framework diseñado para soportar el desarrollo de aplicaciones web. Flask es un micro web framework escrito en Python, que no requiere herramientas o librerías particulares.

1. Instale flask:

   Desde Windows, acceda a la carpeta `C:\Python27\Scripts` y luego instale con pip, el paquete flask:

   ```
   cd C:\Python27\Scripts
   pip install flask
   ```

   Desde Ubuntu, instale primero python-pip y luego flask:

   ```
   sudo apt-get install python-pip
   sudo pip install flask
   ```

2. Cree la carpeta guia5 donde copie los archivos adjuntos:

   - insertValues.py
   - dbFunctions.py
   - globals.py
   - dataWeather.csv

3. Cree el archivo app1.py con el siguiente contenido, el cual se creerá una aplicación web que retorna un saludo de hola mundo.

   ```
   from flask import Flask
   app = Flask(__name__)  # Create the object app from Flask

   @app.route("/")  # For app, we are going to route the home page
   def main():     # This python function is routed to home page
     return "Hola mundo, flask"

   if __name__ == "__main__": # If this file is run directly (not imported)
     app.run()           # Start the web server
   ```

4. Ejecute el script app1.py

   ```
   python app1.py
   ```

5. En su navegador web entre a la dirección `http://127.0.0.1:5000`, si el mensaje de esta página es *Hola mundo, flask* su aplicación web esta funcionando correctamente.

6. Detenga la aplicación web app1.py desde la consola donde se este ejecutando la misma, mediante las teclas `Ctrl+C`. Esto liberará el puerto 5000 para la ejecución de las siguientes aplicaciones web.

7. Cree un archivo app2.py que defina un diccionario con una serie de datos los cuales serán presentados en una página web mediante el template HTML definido en index2.html:

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/")
def main():
    dataPage = {}
    dataPage["title"] = "Web application 2"
    dataPage["nombre"] = "Marcos"
    dataPage["apellidos"] = "Torres"
    dataPage["edad"] = 30

    return render_template('index2.html',dataPage=dataPage)

if __name__ == "__main__":
    app.run()
```

8. Dentro de la carpeta `guia5` cree la carpeta `templates`. Dentro de la carpeta `templates` cree el template index2.html con el siguiente contenido

```
<html>
  <head>
    <title>{{ dataPage.title }} - microblog</title>
  </head>
  <body>
      <h1>Datos de usuario:</h1>
      <p>Nombre: {{ dataPage.nombre }}</p>
      <p>Apellidos: {{ dataPage.apellidos }}</p>
      <p>Edad: {{ dataPage.edad }}</p>
  </body>
</html>
```

9. Ejecute el script app2.py y en su navegador web entre a la dirección `http://127.0.0.1:5000`, detenga la aplicación web al finalizar.

10. Cree el archivo app3.py en donde, mediante un diccionario, se defina el título de la página y una lista con los nombres, apellidos y edades de 3 usuarios:

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/")
def main():
    dataPage = {}
    dataPage["title"] = "Web page 2"
    dataPage["users"] = []

    user = {'nombre' : 'Julio', 'apellidos': 'Rodriguez','edad' : 19}
    dataPage["users"].append(user)

    user = {'nombre' : 'Roberto', 'apellidos': 'Contreras','edad' : 20}
    dataPage["users"].append(user)

    user = {'nombre' : 'Mario', 'apellidos': 'Marquez','edad' : 21}
    dataPage["users"].append(user)

    return render_template('index3.html',dataPage=dataPage)

if __name__ == "__main__":
    app.run()
```

11. Dentro de la carpeta `templates` cree el template index3.html donde mediante un ciclo for se impriman los datos de los usuarios registrados en app3.py

```
<html>
  <head>
    <title>{{ dataPage.title }} - microblog</title>
  </head>
  <body>
    <h1>Usuarios:</h1>
    {% for user in dataPage.users %}
    <div><p>{{ user.nombre }} {{ user.apellidos }}, edad: {{ user.edad }} </p></div>
    {% endfor %}
  </body>
</html>
```

12. Ejecute el script app3.py y en su navegador web entre a la dirección `http://127.0.0.1:5000`, detenga la aplicación web al finalizar.

13. Cree el archivo app4.py en donde mediante la función `dbFunctions.getAllEvents()` del archivo dbFunctions.py adjunto, se lean todos los eventos registrados en los sensores de su base de datos de la guía anterior `sensors.db3`. Igualmente, cree en la lista `sensorEvents` un nuevo key `"dateHum"` para almacenar la fecha con datos humanamente leíbles:

```
from flask import Flask, render_template
import dbFunctions
import time

app = Flask(__name__)

@app.route("/")
def main():
    sensorEvents = dbFunctions.getAllEvents()
    for event in sensorEvents:
        event["dateHum"] = time.strftime(" %Z - %Y/%m/%d, %H:%M:%S", time.localtime(event["date"]))

    return render_template('index4.html', sensorEvents=sensorEvents)

if __name__ == "__main__":
    app.run()
```

14. Dentro de la carpeta `templates` cree el template index4.html donde mediante un ciclo for se impriman los datos de los eventos de los sensores.

```
<html>
  <head>
    <title>Eventos sensores</title>
  </head>
  <body>
    <h1>Eventos:</h1>
    {% for event in sensorEvents %}
    <div><p>{{ event.dateHum }}, {{ event.sensorID }} </p></div>
    {% endfor %}
  </body>
</html>
```

15. Ejecute el script app4.py y en su navegador web entre a la dirección `http://127.0.0.1:5000`, detenga la aplicación web al finalizar.

16. Añada otro servicio a la aplicación app4.py, el cual utilice el sensorID ingresado en la dirección de la página web para mostrar los eventos solo de dicho sensor:

```
@app.route("/sensor/<sensorID>")  # Get <sensorID> as input variable for the function sensor
def sensor(sensorID):
    sensorEvents = dbFunctions.getAllEventsFrom(sensorID)
    for event in sensorEvents:
        event["dateHum"] = time.strftime(" %Z - %Y/%m/%d, %H:%M:%S", time.localtime(event["date"]))

    return render_template('index4.html', sensorEvents=sensorEvents)
```

17. Ejecute el script app4.py y en su navegador web entre a la dirección `http://127.0.0.1:5000/sensor/sensor1`, remplace sensor1 por diversos sensores que tenga en su base de datos, no detenga la aplicación web.

18. Ejecute el script adjunto insertValues.py, inserte 3 sensores iguales y 3 diferentes. Corrobore en la aplicación web que estos nuevos sensores estén reportados.

19. Detenga la aplicación web.

20. Cree el archivo `ConvertWeatherCSVtoDB.py` con el cual lea los datos del archivo `dataWeather.csv` y los guarde en el archivo base de datos `dataWeather.db3`. Añada funciones a la librería `dbFunctions.db` si lo considera necesario.

21. Cree una aplicación web `app5.py` y un template `index5.html` la cual al acceder a la dirección `http://127.0.0.1:5000` despliegue los datos climáticos de todas las ciudades, y al acceder a la dirección `http://127.0.0.1:5000/ciudad/Popayan` despliegue los datos climáticos de la ciudad Popayán o de la ciudad que se introduzca en la dirección web.