

**Curso:** ST0251 - Proyecto Integrador 1

**Taller:** 1. Servicios Web

**Caso:** *EverGreen* – Plataforma para la Productividad de Cultivos Agrícolas

**Plataforma:** - IDE: Visual Studio Code 1.37.0

- Lenguaje: Python 3.7.4

- Framework: Flask 1.1.1

- Pruebas: Postman 7.5.0

**Objetivos:** - Aplicar las dinámicas de trabajo con servicios web.

- Probar un conjunto de servicios web.

- Apropiar el uso de herramientas para el desarrollo con servicios web.

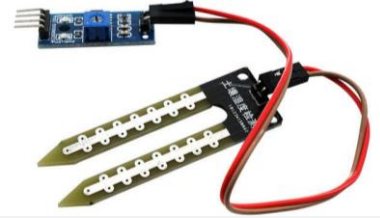
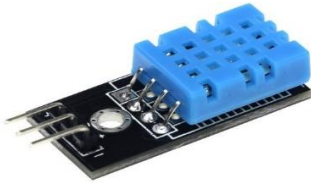

**Modalidad:** Realización del taller individual.

### **Contexto del Caso:**

*EverGreen* es un proyecto que apoya la labor de los productores agrícolas a través del análisis de la información que afecta la productividad de sus cultivos. Uno de los frentes del proyecto realiza el monitoreo de los cultivos, para ello es necesario que desde un dispositivo de IoT que esté conectado a unos sensores en terreno y que utilice como plataforma Raspberry, haga la invocación de servicios web que permitan registrar el valor obtenido de una medición.

### **Numerales del Taller:**

- 1) Escribir en Python usando el *framework* Flask el servicio que haga el POST para registrar una nueva medición; para tal propósito cada estudiante debe basarse en mediciones que provengan de un sensor de la tabla de sensores que se muestra a continuación, de acuerdo con el último dígito de su cédula.

VARIABLE	NOMBRE/ <u>REFERENCIA</u>	IMAGEN	PRECIO AROX.	SALIDA	ÚLTIMO DÍGITO CÉDULA
PH	Aquarium Hydroponic <u>PH0-14</u> controller meter sensor		\$95	Numérico (0 – 14)	Caso para el ejemplo en clase
TEMPERATURA AMBIENTE	Sensor Digital De Temperatura <u>DS18B20</u> Impermeable		\$13	Numérico (Centígrados Fahrenheit)	0 y 5
HUMEDAD TIERRA	<u>FC28</u> soil hygrometer humidity sensor		\$7	Numérico (% agua por volumen de tierra)	1 y 6
HUMEDAD AIRE	<u>DHT11</u> temperature and humidity sensor		\$9	Numérico (% agua en el aire o vapor)	2 y 7
DISTANCIA	Ultrasonic sensor <u>HC-SR04</u>		\$8	Numérico (Centímetros)	3 y 8
LUZ SOLAR	<u>TSL2561</u> Digital Luminosity Light Sensor,		\$14	Numérico (Lux)	4 y 9

- 2) Escribir el servicio que haga el GET para obtener una respuesta basada en las mediciones registradas; para tal propósito cada estudiante debe obtener la respuesta esperada de la tabla que se muestra a continuación, de acuerdo con el penúltimo dígito de su cédula.

Penúltimo dígito de la cédula	Nombre del Método	Parámetro(s) a enviar	Respuesta esperada
Caso para el ejemplo en clase 0 y 9	getByFecha	fecha	Todas las mediciones que se realizaron en fecha
	getMayoresA	fecha o valor	En caso de fecha, las mediciones después de fecha. En caso de valor, las mediciones mayores a valor. ** Se debe enrutar de acuerdo con el tipo de parámetro
1 y 8	getMayores	porcentaje	El porcentaje de valores mayores de las mediciones (ej: si hay 10 mediciones y porcentaje es 0.3, trae los 3 mayores valores)
2 y 7	getMedia	--	La media de las mediciones.
3 y 6	getMediana	--	La mediana de las mediciones.
4 y 5	getModa	--	La moda de las mediciones.

- 3) Escribir el servicio que haga el GET con el método `getAll()` que devuelva todas las mediciones registradas (**Bonus:** escribir los métodos PUT y DELETE que considere apropiados).
- 4) Probar en Postman todos los métodos implementados desde una misma *Collection*.

#### Condiciones de la Entrega:

- Enviar la URL del repositorio público en Github en donde esté alojado el \*.py del código.
- Ejecutar cada invocación en Postman y adjuntar a la entrega en Interactiva los archivos JSON de las respuestas obtenidas (cada archivo de respuesta debe tener el nombre del método que se está invocando).
- Exportar la *Collection* usando la opción de exportación para la versión 2.1 (Recomendada), y adjuntar a la entrega en Interactiva el archivo JSON resultante a la entrega.

#### Criterios de Calificación:

- Contexto: uso apropiado del lenguaje, el *framework* y conceptos de servicios; teniendo en cuenta: variables, nombres de métodos, paso de parámetros y técnicas de enrutamiento.
- Adecuación: carga de un vector inicial con diccionarios que correspondan a casos apropiados para el sensor que le concierna a cada estudiante.
- Originalidad: considerando el amplio rango de posibilidad que brinda el taller, los valores de las variables y parámetros no deben coincidir con los de otros estudiantes.
- Ejecución: el archivo \*.py debe abrir en Visual Studio Code 1.37.0 y ejecutar sin problemas.
- Entrega: que los archivos de la entrega cumplan con las condiciones planteadas.
- Correspondencia: que las *responses* de las invocaciones correspondan al funcionamiento y los parámetros enviados en cada *request*.