

TRABAJO PRÁCTICO N° 1

ASIGNATURA: TALLER DE PROYECTO 2.

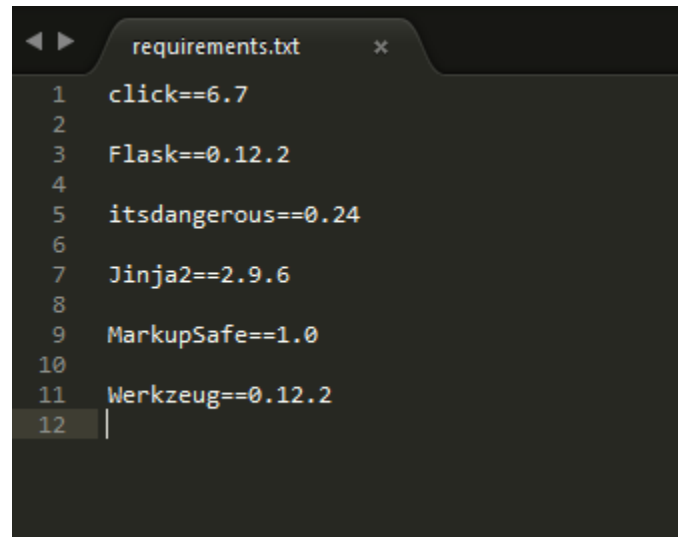
INTEGRANTES:

- ☐ Becerra Agustín
- ☐ Buscaglia Joaquín
- ☐ Molina Rodrigo

Fecha de entrega: 14/09/2017

- 1) Generar el archivo 'requirements.txt' con las dependencias necesarias para poder levantar un servidor con Flask. Explicar un ejemplo de uso con la secuencia de acciones y procesos involucrados desde el inicio de la interacción con un usuario hasta que el usuario recibe la respuesta.

Al generar el archivo requirements.txt a través del comando `pip install-r requirements.txt` se creó el siguiente archivo con las dependencias necesarias para levantar el servidor flask.

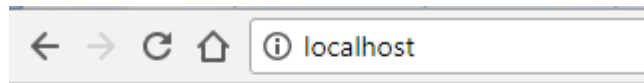
A screenshot of a text editor window titled 'requirements.txt'. The window contains a list of package dependencies, each on a new line, numbered 1 through 12. The dependencies are: click==6.7, Flask==0.12.2, itsdangerous==0.24, Jinja2==2.9.6, MarkupSafe==1.0, and Werkzeug==0.12.2. Line 12 is currently empty with a cursor at the end.

```
1 click==6.7
2
3 Flask==0.12.2
4
5 itsdangerous==0.24
6
7 Jinja2==2.9.6
8
9 MarkupSafe==1.0
10
11 Werkzeug==0.12.2
12 |
```

A screenshot of a text editor window showing a Python code snippet for a Flask application. The code is numbered 1 through 9. It imports Flask from the flask module, creates an app instance, defines a route for the root path with a function named 'Imprimir' that returns a string, and finally runs the app on localhost at port 80.

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def Imprimir():
6     return 'Taller de proyectos II'
7
8 if __name__ == '__main__':
9     app.run(host='localhost', port=80)
```

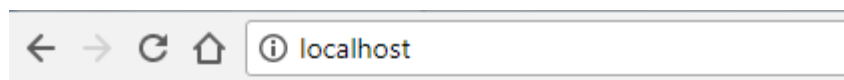
En la imagen anterior se muestra un pequeño código para levantar un servidor flask y retorna un mensaje mediante el navegador. Como se puede apreciar hemos importado una clase Flask, para así poder crear una instancia de dicha clase llamada app con el argumento `__main__` necesario para que Flask busque archivos y/o librerías con la finalidad de poder ejecutar las sentencias necesarias. A continuación se le asigna una ruta donde el servidor podrá acceder a todos los archivos asociados a la aplicación, donde en nuestro caso sólo existirá una función llamada “imprimir” la cual retornara una cadena de caracteres con el fin de visualizar su correcto funcionamiento en el navegador. Por último, al ejecutar la aplicación se envía como parámetro el host y el puerto de nuestro navegador donde se producirá la interacción con el usuario, como se puede ver a continuación.



Taller de proyectos II

2) Desarrollar un experimento que muestre si el servidor HTTP agrega o quita información a la genera un programa Python. Nota: debería programar o utilizar un programa Python para conocer exactamente lo que genera y debería mostrar la información que llega del lado del cliente, a nivel de HTTP o, al menos, a nivel de HTML (preferentemente HTTP).

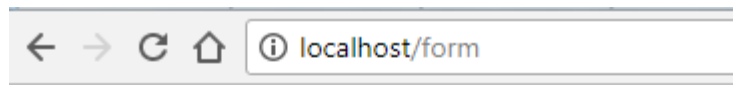
Para llevar adelante el experimento propuesto se utilizará uno de los programas de ejemplo creados por la cátedra para simular un inicio de sesión sencillo, el cual consta de un formulario con un único campo de texto para capturar el nombre del usuario y mandarlo (como se ilustra en la imagen 1), utilizando el método POST, a otro documento HTML que se encargará de darle la bienvenida al usuario que acaba de acceder (como se ilustra en la imagen 2). Teniendo esto en cuenta se tendrá la certeza de los datos enviados a nivel HTML y así se podrá comprobar la información que el servidor HTTP agrega a la transmisión.



Formulario de registro

Nombre

Imagen 1.



¡Bienvenido a TP2!

Nombre: Agustin

Imagen 2

En la siguiente imagen (3) se puede ver el código python utilizado para levantar el servidor flask y realizar la comunicación entre los documentos HTML. Se puede observar que en la función `action_form()` se captura el dato ingresado por el usuario y el mismo es enviado a `response.html`.

```
app.py
1  # Aquí se listan los imports
2  from flask import Flask
3  from flask import render_template
4  from flask import request
5
6  app = Flask(__name__)
7
8  # Define la ruta con la que se ingresara desde el browser
9  @app.route('/')
10 def index():
11     return render_template('form.html')
12
13 # Define la ruta y metodo con el que se debe llegar a este endpoint
14 @app.route('/form', methods = ['POST'])
15 def action_form():
16
17     if request.method == 'POST':
18         data = request.form
19         nombre = data["nombre"]
20         return render_template('response.html', nombre=nombre)
21
22 if __name__ == "__main__":
23     # Define HOST y PUERTO para acceder
24     app.run(host='localhost', port=80)
```

Imagen 3.

Se utilizó el programa RawCap para monitorear el tráfico de datos en la red al momento de que el usuario realiza la interacción con el programa, para así crear un archivo con extensión .pcap con todos los paquetes presentes en la comunicación.

Luego mediante wireshark se pudo inspeccionar en detalle dicho archivo y así identificar los paquetes a nivel HTTP involucrados en la comunicación. Como se puede observar en las imágenes ilustradas a continuación, se le agrega información como:

Protocolos de internet (fuente, destino, tamaño, etc.).

Protocolo de transmisión (puerto de destino y fuente, tamaño del headers).

Texto plano de tipo HTML.

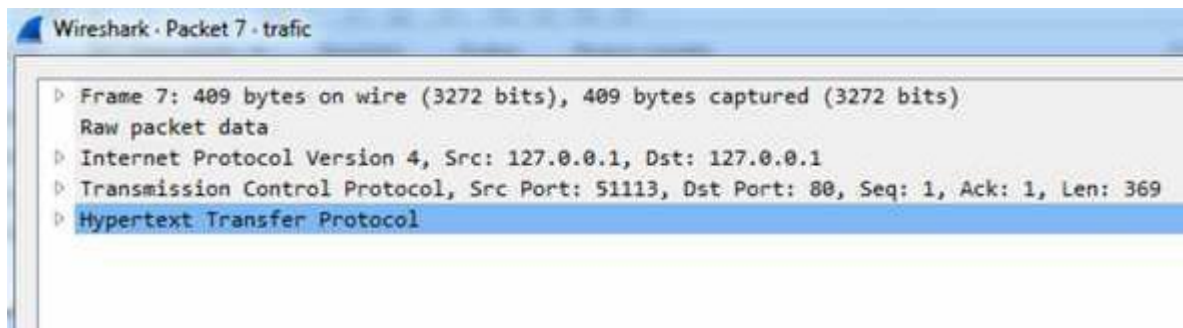


Imagen 4.

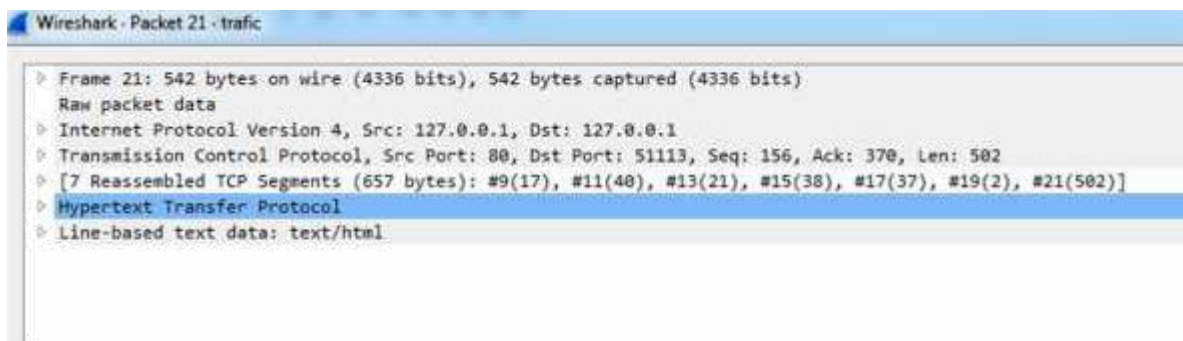


Imagen 5.

3) Generar un proyecto de simulación de acceso a valores de temperatura, humedad, presión atmosférica y velocidad del viento.

Un proceso simulará una placa con micro controlador y sus correspondientes sensor/es o directamente una estación meteorológica proveyendo los valores almacenados en un archivo o en una base de datos. Los valores se generan periódicamente (frecuencia de muestreo).

Un proceso generará un documento HTML conteniendo:

Frecuencia de muestreo

Promedio de las últimas 10 muestras

La última muestra

c) El documento HTML generado debe ser accesible y responsivo.

Aclaración: Se deberá detallar todo el proceso de adquisición de datos, cómo se ejecutan ambos procesos (ya sea threads o procesos separados), el esquema general, las decisiones tomadas en el desarrollo de cada proceso y la interacción del usuario.

El objetivo de este apartado es crear una estación meteorológica vía web donde se reflejaran valores de distintas variables físicas a las cual el usuario podrá acceder a través del navegador.

Para llevar a cabo la simulación de la generación de valores de presión, temperatura, humedad y velocidad del viento, se desarrolló un programa en python que generará valores aleatorios en un rango predefinido, los cuales se insertarán en una base de datos previamente creada que cuenta con una sola tabla con los cuatro campos necesarios para almacenar dichos valores y un campo adicional de identificación que se definirá de manera autoincremental a medida que se vayan creando nuevas tuplas. Dicha base de datos debe ser creada mediante la terminal provista en el paquete de instalación de MySQL como se muestra a continuación:

```
mysql> CREATE DATABASE test;
Query OK, 1 row affected (0.00 sec)

mysql> USE test;
Database changed
mysql> CREATE TABLE dato( ID INT AUTO_INCREMENT, Temperatura INT, Humedad INT, Presion INT, Viento INT, PRIMARY KEY(ID))
;
Query OK, 0 rows affected (0.25 sec)

mysql> INSERT INTO dato VALUES (null, 22, 80, 980, 15);
Query OK, 1 row affected (0.05 sec)

mysql> select * from dato;
+-----+-----+-----+-----+-----+
| ID | Temperatura | Humedad | Presion | Viento |
+-----+-----+-----+-----+-----+
| 1 | 22 | 80 | 980 | 15 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Una vez generada la base de datos con su respectiva tabla será gestionada utilizando los métodos y funciones propios de la API de MySQL.

Por otro lado, se desarrolla un nuevo programa en python que se encargará de generar un documento HTML, el cual informará al usuario la frecuencia de muestreo de los datos recolectados (en este caso un 1 Hz), los valores de las últimas diez tuplas existentes en la base de datos y su promedio.

Dicho documento deberá ser accedido desde el puerto 80 del host local y visualizado mediante cualquier navegador de uso cotidiano, donde se refrescará cada un segundo utilizando la meta “refresh” con el fin de actualizar los datos que se mostrarán en pantalla, teniendo que realizar una nueva consulta a la base de datos para obtener los últimos diez valores actualizados.

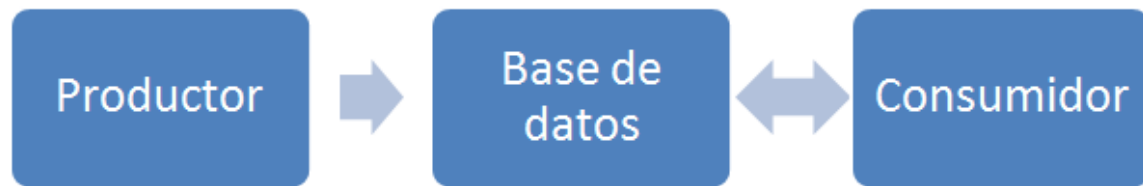
Además se utiliza bootstrap para definir un diseño más agradable a la vista y aportar a una correcta comprensión del usuario.

4) Agregar a la simulación anterior la posibilidad de que el usuario elija entre un conjunto predefinido de períodos de muestreo (ej: 1, 2, 5, 10, 30, 60 segundos). Identifique los cambios a nivel de HTML, de HTTP y de la simulación misma.

Para cumplir con los nuevos requerimientos que se plantean, se agregó al proyecto un nuevo documento HTML que actuará como página principal, ejecutándose una vez que el usuario inicie el programa, y dándonos la posibilidad de elegir la frecuencia de muestreo de los datos generados por el productor. Este nuevo documento consta simplemente de un formulario con un único campo del tipo “select” con las posibles frecuencias que se pueden elegir, y envía este valor mediante el método GET a un documento similar al utilizado en el punto anterior para mostrar en nuestro navegador los valores obtenidos desde la base de datos.

Como se reutilizara el proceso que genera los valores aleatorios en el punto anterior, se debe tener en cuenta que dicho programa produce datos a una frecuencia de un Hertz, por lo que a la hora de leer los valores desde la base de datos se debe tener en cuenta que entre una tupla y la siguiente solo habrá pasado un segundo, y sólo serán de interés los valores de las tuplas cuyo identificador sea múltiplo del valor del periodo de muestreo elegido para esa simulación en particular.

Como se puede observar en los últimos dos ejercicios resueltos (3 y 4) se pretende modelar una arquitectura de software del tipo productor/consumidor, en el que el proceso que genera los valores y escribe en la base de datos actúa de productor y, el proceso que se encarga de levantar el, o los documentos HTML se encarga de modelar al consumidor.



5) Comente la problemática de la concurrencia de la simulación y específicamente al agregar la posibilidad de cambiar el período de muestreo. Comente lo que estima que podría suceder en el ambiente real ¿Podrían producirse problemas de concurrencia muy difíciles o imposibles de simular? Comente brevemente los posibles problemas de tiempo real que podrían producirse en general.

El problema principal de concurrencia que aparece en nuestra simulación, al cambiar el periodo de muestreo, se debe a que los datos generados por nuestro productor no serán utilizados en su totalidad por el consumidor cuando el periodo de muestreo sea mayor a un segundo, en cuyo caso dicho programa deberá tomar de la base de datos solo las tuplas cuyo identificador sea múltiplo del periodo de muestreo utilizado.

Otro conflicto en cuanto a la concurrencia que nos podemos encontrar es la no sincronización entre el productor y el consumidor a la hora de acceder a la base de datos. Por ejemplo, si el productor produce valores con un periodo menor al que son requeridos por el consumidor, este último consumirá valores repetidos en instantes de tiempo consecutivos. Y por otro lado, si el consumidor trabaja a menor frecuencia que el productor, el primero no será capaz de procesar todos los datos generados, por lo que inevitablemente habrá una pérdida de consistencia entre los datos tomados por los sensores y los que componen la información obtenida por el usuario final.

En un ambiente real podría pasar que los sensores, asumiendo sus limitaciones físicas y/o de control, no puedan tomar muestra de valores a la frecuencia que sean requeridas por el sistema real.

6) ¿Qué diferencias supone que habrá entre la simulación planteada y el sistema real? Es importante para planificar un conjunto de experimentos que sean significativos a la hora de incluir los elementos reales del sistema completo.

Es importante tener una referencia respecto de cómo funciona un sistema real implementándolo a través de una simulación de software donde se podrá observar los problemas o limitaciones con los que nos encontraremos a la hora de implementarlo en un micro controlador, por ejemplo limitación de espacio de memoria, problemas de concurrencia y sincronización, etc.

Para implementar esta simulación de muestreo de señales de sensores en un sistema real, habrá que programar el sistema de adquisición de datos para que en un instante de tiempo determinado pueda acceder a los sensores y reflejar dicho valor. Por lo tanto, en este caso, no habrá problemas de concurrencia de datos, por no estar almacenando los valores leídos en una base de datos como se realizó en nuestra simulación. Lo que sí podría suceder en un sistema de tiempo real es que haya problema de sincronización entre capturar un valor y el muestreo del mismo.