Universidad Nacional de La Plata

**Facultad de Informática**

**Trabajo Práctico**

# Estación Meteorológica

Taller de Proyecto II

Integrantes

Iglesias Ramiro

Ramallo Juan Manuel

14 de septiembre de 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.**

Para poder levantar un servidor con el microframework Flask fue necesario crear un proyecto (carpeta) con los siguientes archivos:

- App.py: contiene el código Python encargado de procesar datos del lado del servidor
- Response.html: vista, la cual va a ser la interface con el usuario ya sea para mostrar o ingresar datos.

Una vez creados los archivos, fue necesario disponer de herramientas, una de ellas es flask:

```
$ pip instal flask
```

Pasando al código de app.py:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return render_template('response.html');
if __name__ == "__main__":
    app.run()
```

Nuestra vista:

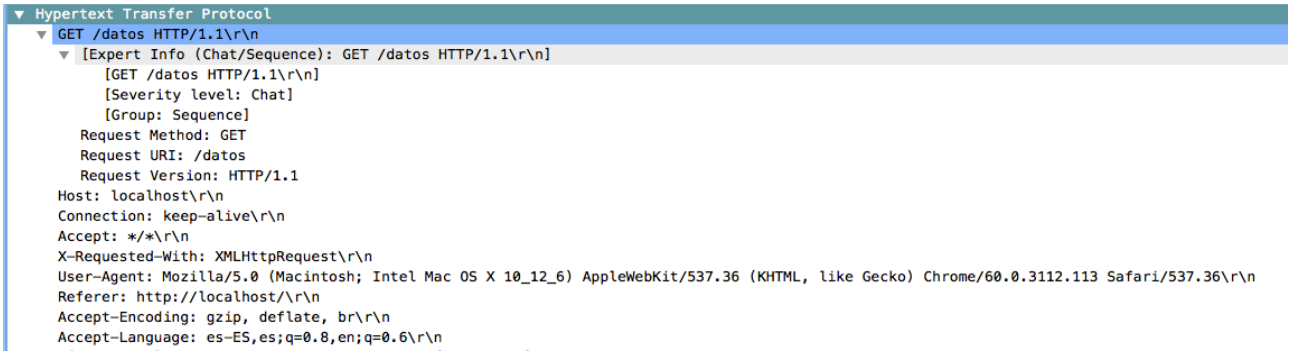
```
<h1> Hola Mundo </h1>
```

Como se puede ver se llamo al comando de instalación, sin utilizar el archivo requeriments.txt. Si quisiéramos realizar un archivo llamado requirements.txt donde se guarden las dependencias necesarias a ser instaladas, el contenido del archivo contendría solo la cadena "Flask", resultando el comando de instalación de la siguiente manera:

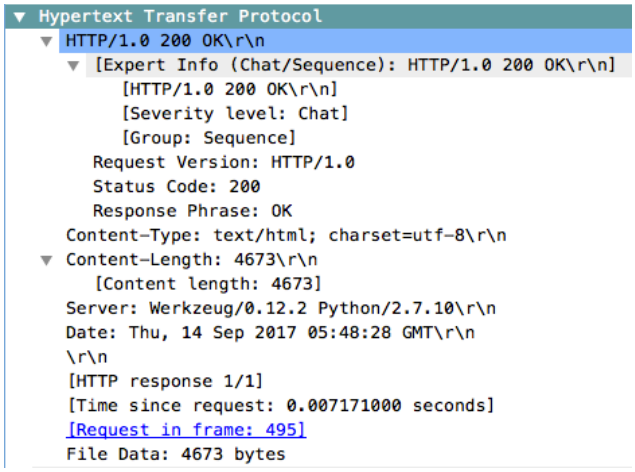
```
$ pip install -r requirements.txt
```

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 este experimento se hizo uso de la herramienta de sniffing de paquetes llamada Wireshark. Se tomó como caso de prueba la aplicación web creada en el ejercicio 3. A continuación se pueden observar en las imágenes de captura del Wireshark que el servidor no solo envía el documento HTML sino que también agrega distintos datos en las respuestas que envía de acuerdo a los pedidos HTTP que recibe. Entre esos datos se encuentran los diversos campos del encabezado del pedido HTTP, estos ayudan al navegador a entender cómo debe mostrarse la respuesta de ese pedido y también en algunos casos cómo debe comportarse el mismo navegador.




Por ejemplo en la imagen siguiente se observa un campo llamado Content-Type cuyo valor está especificado en text/html; charset=utf-8, y como bien parece esto le indica al navegador que el contenido recibido se trata de un documento html con codificación en utf-8, así como este campo es de gran utilidad también el Status Code lo es ya que este especificará a ciencia cierta si el pedido fue realizado y respondido de manera correcta o no.



3. **Generar un proyecto de simulación de acceso a valores de temperatura, humedad, presión atmosférica y velocidad del viento.**
   A. **Un proceso simulará una placa con microcontrolador 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).**
   B. **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.**

Para generar datos se creo una aplicación la cual cada t segundos genera datos de forma aleatoria y los ingresa en una base de datos. El programa consiste de un loop infinito en el cual se duerme al proceso por un determinado tiempo simulando así un período de muestreo de los datos por parte de la estación meteorológica. El código se encuentra en el anexo o en nuestro repositorio en github.

El proceso que genera el HTML se conecta a la base de datos y pide de la misma los valores a mostrar. El cálculo de los promedios se realizan directamente en la base de datos. Para hacer de la página responsiva se hizo uso de Bootstrap.


El usuario solo debe ingresar a la dirección raíz de la página y podrá ver todos los datos de la estación meteorológica.

Para correr ambos procesos se deben ejecutar con python los dos archivos app.py contenidos en carpetas distintas siendo una para la aplicación web y otra para simulador de la estación. Una vez ambos están corriendo, cuando se carga la aplicación web en un navegador éste le hace un pedido GET via ajax al proceso simulador para avisarle que debería empezar a generar los datos aleatorios (usamos ajax para generar pedidos HTTP sin necesidad de recargar el navegador, en nuestro caso usamos la librería jQuery para realizar el pedido)

El código se puede igualmente obtener desde nuestro repositorio, en la carpeta /py se encuentra la aplicación web y en /py_timer el simulador de la estación.

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 realizar esto se creó una ruta /configurar en la aplicación web que reciba solamente el método POST y un parámetro llamado periodo. Esta ruta se encarga de guardar en la base de datos el período según el valor pasado por parámetro. Entonces la aplicación simuladora se encarga de leer ese periodo para actualizar su frecuencia de muestreo.

En la vista se tiene un elemento select el cual tiene las opciones 1, 2, 5, 10, 30 y 60 segundos para que el usuario seleccione y cambie automáticamente el período de muestreo. Para esto, nuevamente se hace uso de ajax para realizar el POST mencionado recientemente y a su vez se agrega un event listener al elemento select, para que cada vez que cambie su valor seleccionado se ejecute el POST para configurar el período.

El código se puede ver desde el repositorio en la carpeta llamada /py.

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.**

Los problemas de concurrencia derivan, básicamente, del acceso a la base de datos ya que a la base de datos acceden dos usuarios a la vez (el que alimenta la base de datos y el que consulta la base de datos). Por lo tanto, los datos reflejados pueden no estar correctos, ejemplo, por la actualización posterior a la muestra (datos viejos).

Un sistema de tiempo real debe cumplir los requisitos de cumplir una determinada tarea en un determinado tiempo, con lo cual nuestro sistema debe actuar/reflejar los datos según los tiempos acordados. Como consecuencia del incumplimiento de los tiempos, se obtienen datos erróneos o antiguos, provocando el mal funcionamiento de la aplicación. Por lo tanto, el tiempo real y la concurrencia son aspectos muy importantes a tener en cuenta.

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.**

Como primer aspecto a tener en cuenta son los tipos de datos, por un lado los datos ingresados en la simulación son aleatorios y no son reales, es decir no provienen de ninguna fuente que interactúe con el mundo exterior, ya sean sensores, actuadores, etc. Los datos en un sistema de tiempo real se deben reflejar en la base de datos en una cota de tiempo, que a diferencia de la simulación se reflejan instantáneamente.

Los datos pueden provenir de un microcontrolador, otra computadora o un servidor los cuales poseen acceso a la base de datos.