

Sistemas Operativos – Joan Alexis Córdoba Narváez – A00232548

INFORME.

Para la realización de la actividad, se han instalado las debidas aplicaciones y servicios que ayuden a administrar las estructuras JSON, SQLite y Python, sobre el sistema operativo Centos 7 Minimal 16.11.

Se han creado tres Scripts de Python: llamado.py, obtenerRecursos.py y background.py.

En *obtenerRecursos.py* están definidas las funciones que obtienen la información de uso de memoria RAM, CPU, espacio libre en disco duro y el estado del servicio SSH.

```
192.168.182.132 192.168.182.132 192.168.182.132 192.168.182.132 192.168.182.132
#Retorna el porcentaje de memoria usada
def uso_mem():
    result1 = commands.getoutput('free|grep Mem:|tr -s " " " " |cut -d " " " " -f 2')
    result2 = commands.getoutput('free|grep Mem:|tr -s " " " " |cut -d " " " " -f 3')
    result3=int(result2)*100/int(result1)
    return result3

#Retorna uso de la CPU
def uso_cpu():
    grep_process = Popen(["mpstat", "", ""], stdout=PIPE, stderr=PIPE)
    return filter(None, grep_process)
    grep_process = Popen(["sar", "1", "1"], stdout=PIPE, stderr=PIPE)
    lista = Popen(["awk", '{print $5}'], stdin=grep_process.stdout, stdout=PIPE, stderr=PIPE).communicate()[0].split('\n')
    return filter(None, lista)

#Retorna espacio en HD
def uso_hdd():
    rep_process = Popen(["df", "/dev/mapper/cl-root", "-h"], stdout=PIPE, stderr=PIPE)
    lista= Popen(["awk",{print $4}'],stdin=grep_process.stdout, stdout=PIPE, stderr=PIPE).communicate()[0].split('\n')
    return filter(None, lista)

#Retornasi el servicio esta corriendo
def status_service():
    grep_process = Popen(["service", "httpd", "status"], stdout=PIPE, stderr=PIPE)
    lista = Popen(["awk", '-F', 'Active:', '{print $2}'], stdin=grep_process.stdout, stdout=PIPE, stderr=PIPE).communicate()
].split('\n')
    return filter(None, lista)
```

En *llamado.py* se llama a las funciones del anterior script para agregar los datos tanto a la base de datos *infra.db* (creada mediante SQLite) como al archivo *resultado.txt* que será utilizado para las respuestas JSON.

```
192.168.182.132 192.168.182.132
from obtenerRecursos import uso_mem, uso_cpu, uso_hdd, status_service
import json, time, sqlite3

file = open('resultados.txt','a+')
varDate= "Date: " + time.strftime("%c")
varRam = "Ram: " + uso_mem()
varCPU = "CPU Usage: " + uso_cpu()[2]
varHDD = "Hard Disks: " + uso_hdd()[1]
varServ = "Estado SSH: " + status_service()

miQuery = "INSERT INTO tblData (ram, cpu, hd, status) VALUES (" + varRam + ", " + varCPU + ", " + varHDD + ", " + varServ + ")"

con = sqlite3.connect(":infra:")
cursor = con.cursor()
print "Conexion abierta"

cursor.execute(miQuery)
con.commit()
print "Valores insertados"

con.close()
print "Conexión Cerrada"

file.write(varDate+"\n")
file.write(varRam+"\n")
file.write(varCPU+"\n")
file.write(varHDD+"\n")
file.write(varServ+"\n")
file.write("\n")
file.close()
```

En *background.py* están las líneas que ayudan a que la captura de datos se haga cada 60 segundos.

```
192.168.182.132 192.168.182.132 192.168.182.132
import commands
i=1
while i<=2
    result=commands.getoutput('llamado.py')
    time.sleep(60)
~
~
```

El archivo resultado.txt:

```
192.168.182.132 192.168.182.132
Date: Mon May 28 21:52:25 2017
Ram: 132 m used memory
CPU Usage: 0,00
HDD disponible: 15G
Estado SSH: inactive (dead)

Date: Mon May 28 21:53:27 2017
Ram: 132 m used memory
CPU Usage: 0,00
HDD disponible: 15G
Estado SSH: inactive (dead)
```

Para poner en marcha el script background.py, se ha ingresado la siguiente instrucción:

```
192.168.182.132 192.168.182.132 192.168.182.132 192.168.182.132
login as: root
root@192.168.182.132's password:
Last login: Mon May 29 21:21:59 2017 from 192.168.182.1
[root@localhost ~]# cd alexis-a00232548/
[root@localhost alexis-a00232548]# cd pythonSRC/
[root@localhost pythonSRC]# python background.py
[root@localhost pythonSRC]#
```

Luego se examinan las salidas en el navegador obteniendo los siguientes resultados:

DELETE	/v1.0/stats/	NO APLICA
Response Messages		
HTTP Status Code	Reason	Response Model
404	HTTP 404 NOT FOUND	Headers
Try it out!		

GET	/v1.0/stats/	Lista de recursos
Response Messages		
HTTP Status Code	Reason	Response Model
200	Recursos usados.	Headers
Try it out!		

POST
/v1.0/stats/
NO APLICA

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	HTTP 404 NOT FOUND		

Try it out!

PUT
/v1.0/stats/
NO APLICA

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	HTTP 404 NOT FOUND		

Try it out!

GET
/v1.0/stats/
Lista de recursos

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Recursos usados.		

Try it out!

[Hide Response](#)

Curl

```
curl -X GET -header 'Accept: application/json' 'http://192.168.182.131:8080/v1.0/stats/'
```

Request URL

```
http://192.168.182.132:8080/v1.0/stats
```

Response Body

```
{
  "HDD available": "\u0015G",
  "CPU Usage": "\u0000\u0000",
  "Estado SSH": ["inactive (dead)"],
}
```

Response Code

```
200
```

Response Headers

```
{
  "date": "Mon, 29 May 2017 21:58:16 GMT",
  "server": "Werkzeug/0.12.1 Python/2.7.13",
  "content-length": "154",
  "content-type": "application/json"
}
```

En la realizaci3n de esta actividad, hubo algunos problemas mostrando el estado del servicio y algunas incoherencias con las salidas. Estas fueron arregladas revisando con detalle las l3neas de c3digo.