SISTEMA DE RECOMENDACIONES PARA EL SERVICIO DE VOD, BASADO EN LA INFERENCIA DE EMOCIONES A PARTIR DE VARIABLES DEL CONTEXTO DE USUARIO

Mauricio Sánchez Barragán
Luis Alejandro Solarte Moncayo

Director: Mag. Gabriel Elías Chanchí Golondrino
Co-Director: PhD. José Luis Arciniegas Herrera

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de investigación en servicios avanzados de telecomunicaciones

Popayán, Agosto de 2016

SISTEMA DE RECOMENDACIONES PARA EL SERVICIO DE VOD, BASADO EN LA INFERENCIA DE EMOCIONES A PARTIR DE VARIABLES DEL CONTEXTO DE USUARIO

Trabajo de grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Mauricio Sánchez Barragán
Luis Alejandro Solarte Moncayo

Director: Mag. Gabriel Elías Chanchí Golondrino
Co-Director: PhD. José Luis Arciniegas Herrera

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de investigación en servicios avanzados de telecomunicaciones

Popayán, Agosto de 2016

Contenido

Lista de Figuras

Anexo A

Manual de usuario del S.R. propuesto

En este anexo es presentado el manual de usuario para el S.R. propuesto, el cual describe los pasos a seguir para el adecuado uso de este servicio. En este manual se van a mencionar los escenarios, interfaz y guías que se deben considerar en la conexión y ejecución del S.R. a través del servicio de VoD implementado. Se pretende, este documento sea de ayuda o soporte a trabajos futuros.

1. Inicialmente se deben conectar los sensores o dispositivos *wearables* implementados en el diseño del sistema hardware-software de la sección 3.7. Estos deben estar conectados entre el usuario y la tarjeta Arduino Yún de la siguiente manera, (ver figura A.1).

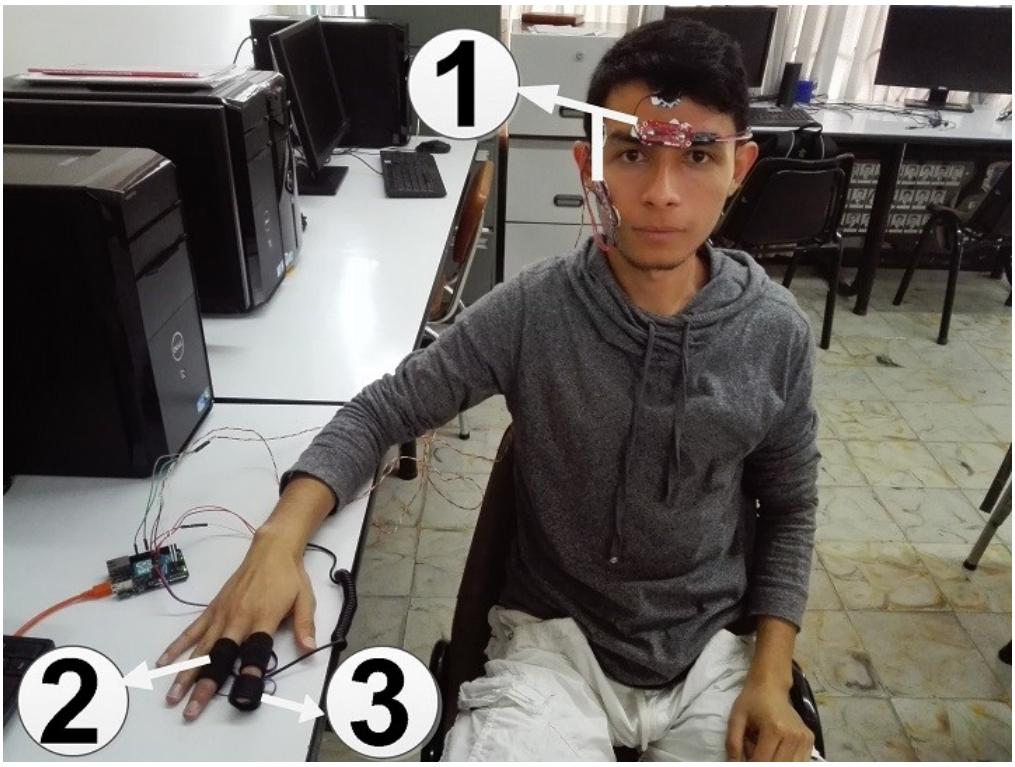


Figura A.1 Conexión sensores en usuario, fuente: propia

En la figura A.1 son mostrados cómo deben estar conectados los sensores en el cuerpo humano, encargados de la medición de variables fisiológicas. En “1” se puede observar cómo están conectados los 2 sensores Electromiográficos (EMG), encargados de

determinar la *valence* positiva a través del músculo *zygomaticus* y negativa a través del músculo *corrugator*. En "2" es mostrado como está conectado el sensor Galvánico RGP en los dedos índice y medio, encargado de capturar un estímulo emocional en el usuario. En "3" se observa cómo debe estar conectado el sensor de pulso cardiaco (*Heart Rate*), encargado de determinar un valor de *arousal* en el usuario, el cual se encuentra asociado a un valor de índice de estrés calculado a partir de la VFC.

2. Como segundo paso es arrancar los servidores de contexto y procesamiento, encargados de la captura y análisis de los datos fisiológicos del usuario (ver figura A.2 y figura A.3).

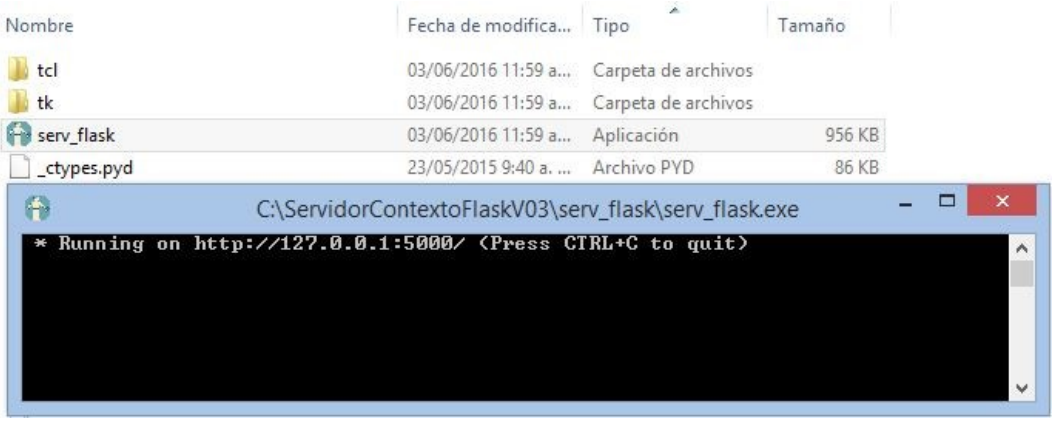


Figura A.2 Arrancar servidor de contexto, fuente: propia

En la figura A.2 es mostrado cómo se arranca el servidor de contexto desarrollado en Python por medio de la librería Flask a través del entorno de desarrollo Geany. Con el fin de facilitarle al usuario se creó una aplicación ejecutable en Windows, la cual permite al usuario de una manera más fácil arrancar el servidor.

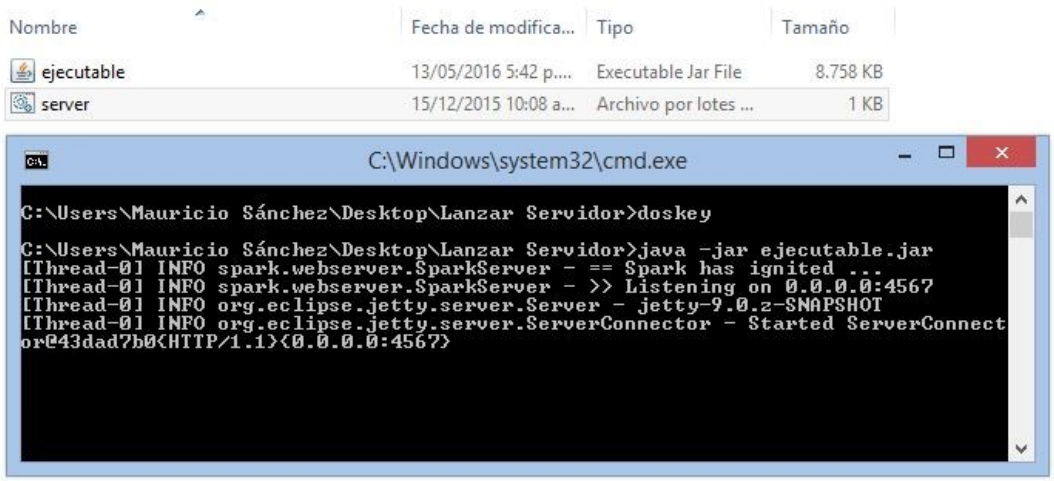


Figura A.3 Arrancar servidor de procesamiento, fuente: propia

En la figura A.3 es mostrado como se arranca el servidor de procesamiento desarrollado con el framework de Spark en Java, para ello se creó un ejecutable "jar" con el fin de facilitar el inicio del servidor al usuario.

3. En el tercer paso el usuario debe ingresar a la url del servicio de VoD: *http://www.unicauca.edu.co/SR,* página web mediante la cual va a poder usar el S.R. a través del servicio (ver figura A.4).

Figura A.4 Interfaz de login del servicio de VoD, fuente: propia

En la figura A.4 es observada la interfaz inicial del servicio de VoD al usuario, y con la cual va a poder validarse para iniciar sesión. En "1" es mostrado el menú de validación, donde el usuario debe ingresar sus credenciales (*login* y *password*) para poder acceder al servicio, en caso de no estar registrado se puede registrar a través del link Registrarse!. En "2" es presentado en tiempo real los datos asociados a las medidas fisiológicas (FC, VFC, RGP y EMG). En "3" es mostrado al usuario la gráfica de la frecuencia cardíaca vs tiempo durante un periodo de 60 segundos.

4. Una vez desplegada la interfaz de *login*, como paso cuarto el usuario debe iniciar la toma de medidas fisiológicas a través del botón “Obtener Valores” durante un periodo de 60 segundos, tiempo necesario para determinar la emoción de entrada del usuario (ver figura A.5).

Figura A.5 Menú de la toma de medidas, fuente: propia

5. Ya obtenida la emoción de entrada de usuario, el usuario debe ingresar sus credenciales e iniciar sesión en el menú de validación (ver figura A.6). En caso de no estar registrado en el servicio de VoD, el usuario puede registrarse con sus datos para poder acceder (ver figura A.7).

Figura A.6 Menú de validación de usuario, fuente: propia

En la figura A.6 es presentado el menú de validación, donde el usuario debe ingresar el nombre de usuario o *login*, además del *password* para poder acceder al S.R.

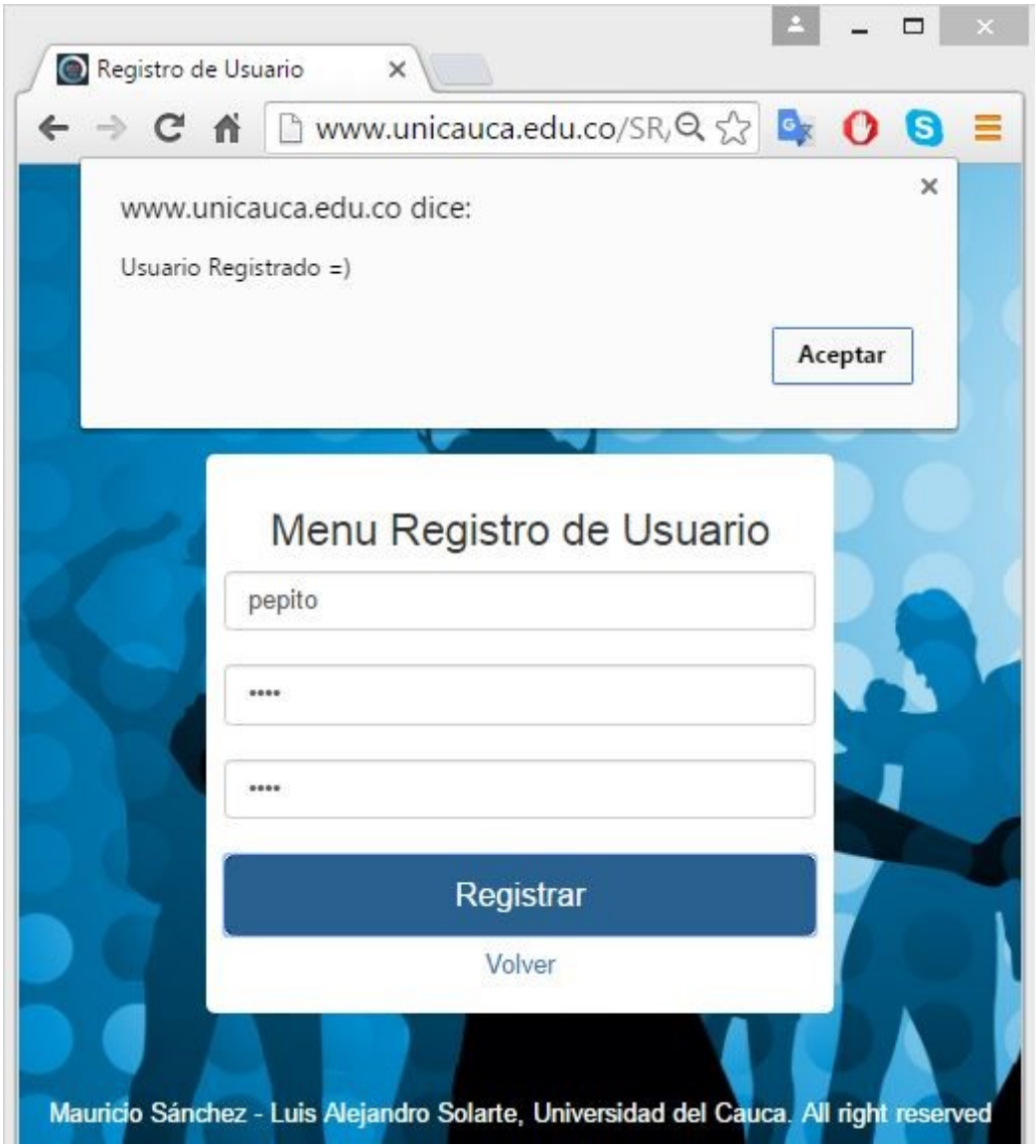


Figura A.7 Menú de registro de usuario, fuente: propia

En la figura A.7 es presentado el menú de registro de usuario, en este se debe registrar un nombre de usuario y una *password*, además pide verificar la *password* por seguridad y luego presionar en el botón de registrar, el sistema verifica si existe un usuario con ese mismo *login* y en caso de no haber algún registro, lo guarda en la base de datos MySQL del S.R.

6. Una vez iniciada la sesión de usuario, se ingresa al S.R. y es mostrada la interfaz de contenidos (ver figura A.8).

Figura A.8 Interfaz de contenidos del S.R., fuente: propia

En la figura A.8 es presentada la interfaz principal de contenidos del S.R, es allí donde el usuario puede hacer uso del S.R. a través del servicio de VoD y visualizar los contenidos de video asociado a las 5 emociones. En “1” es presentdo en tiempo real los datos asociados a las medidas fisiológicas (FC, VFC, RGP y EMG), además de la gráfica de frecuencia cardíaca vs tiempo. En “2” es mostrado el campo donde se reproduce el video recomendado por el S.R., además del menú de control del video donde puede realizar diferentes acciones (pausar video, detener el video, subir y bajar volumen, cambiar de video, ajustar a toda la pantalla el video, y sistema de valoración del mismo), esta última acción de valorar el contenido es muy importante para el sistema, ya que va a servir de retroalimentación en la recomendación de nuevos contenidos. En “3” es mostrado el catalogo o listado de contenidos de video recomendados por el S.R., en promedio unos 40 contenidos son proporcionados al usuario en cada recomendación.

Es importante mencionar lo siguiente, el S.R. va a estar actualizando cada 60 segundos el listado de recomendaciones de acuerdo al cambio emocional del usuario, y cada 90 segundos es reordenado el listado según el estímulo mental percibido de la acción de visualizar cada contenido. Donde el estímulo es determinado a partir de la medida galvánica de la piel, la cual está asociada a un cambio en la percepción del *arousal* del usuario. En la figura A.9 es posible observar como un usuario da una valoración a un contenido.

Figura A.9 (a)

Figura A.9 (b)

Figura A.9 Menú de valoración de contenidos, fuente: propia

En la figura A.9 es observado el menú para la valoración de los contenidos de video, en la figura A.9.a) es mostrado como el usuario decide valorar con 4 estrellas al contenido visualizado, donde esta valoración se da en un rango de 1 a 5 estrellas, siendo 5 estrellas el mejor resultado. Y en b) el usuario ya ha actualizado la información de la valoración y automáticamente está es actualizada en la base de datos MySQL del sistema. La valoración que el usuario proporcione sobre un contenido, es considerada por el clasificador bayesiano y la herramienta weka, para recomendar diferentes contenidos de acuerdo a si ha sido positiva o negativa la valoración del contenido.

7. Después de observar todos los pasos necesarios para que un usuario pueda utilizar el S.R. basado en contexto, y el cual le brinda recomendaciones adecuadas de contenidos de video de acuerdo a su estado de ánimo, es posible concluir lo siguiente, el S.R. es un sistema capaz de dotar de recomendaciones precisas, oportunas y de acuerdo a la necesidad emocional del usuario, la cual este presentando en un estado de tiempo. Lo anterior sirve para ver como un usuario x, está haciendo uso del S.R. a través del servicio de VoD (ver figura A.10).

Figura A.10 Usuario haciendo uso del S.R., fuente: propia

Anexo B

Pasos para la generación del dataset de contenidos musicales de video

En este anexo es presentado el proceso paso a paso de la generación del *dataset*, en los cuales se hace uso de la API de Echonest por medio de códigos y librerías adicionales en Python. El *dataset* permite generar el archivo descriptor de los contenidos musicales asociados al modelo de emociones presentado en la sección 2.6.1. Este aporte permitió obtener y descargar los contenidos de video utilizados en el servicio de VoD, a través del S.R basado en contexto.

1. Se declaran las librerías necesarias en Python, y se establece la llave para conexión con la API de Echonest (ver figura B.1).

```
56  #*************** Librerias y Llaves ***************#
57  import math,sys,youtube_dl,time,cookielib,urllib,urllib2,re,logging
58  from difflib import SequenceMatcher
59  from tinydb import TinyDB, where
60  from pyechonest import config
61  from pyechonest import artist
62  from pyechonest import track
63  from pyechonest import song
64
65  config.ECHO_NEST_API_KEY="EPX09QF26IUQIBUCP"
66  #************************************************#
```

Figura B.1 Librerías utilizadas y llave de Echonest, fuente: propia

2. Se genera la petición de canciones más populares en Echonest (ver figura B.2).

```
190  #************* Metodos Pyechonest ***************#
191  def generar_script(ini):
192    while True:
193      res = song.search(sort='song_hotttnesss-desc', buckets=['audio_summary'], results=100, start=ini)
194
195      tam=len(res)
196      tiempos=[18,36,54,72,90]
197      lista_repetidos=list()
198
199      for i in range(tam):
200        if(i in tiempos):
201          time.sleep(60)
202
203        cancion=res[i]
204        cur=cancion.song_currency
```

Figura B.2 Búsqueda de canciones más populares EchoNest, fuente: propia

3. Se obtienen de las canciones más populares, sus características musicales mencionadas en la sección 2.3 (ver figura B.3).

```
206         if(cur not in lista_repetidos):
207           title=cancion.title
208           artist=cancion.artist_name
209           energy=cancion.audio_summary['energy']
210           valence=cancion.audio_summary['valence']
211           liveness=cancion.audio_summary['liveness']
212           tempo=cancion.audio_summary['tempo']
213           speechiness=cancion.audio_summary['speechiness']
214           acousticness=cancion.audio_summary['acousticness']
215           instrumentalness=cancion.audio_summary['instrumentalness']
216           mode=cancion.audio_summary['mode']
217           time_signature=cancion.audio_summary['time_signature']
218           duration=cancion.audio_summary['duration']
219           loudness=cancion.audio_summary['loudness']
220           danceability=cancion.audio_summary['danceability']
221           url=''
222           visua=''
```

Figura B.3 Obtención de las características del contenido, fuente: propia

4. Se asocian las características de *arousal* y *valence* con un estado de ánimo, de acuerdo al modelo propuesto en la sección 2.6.1 (ver figura B.4).

```
224           e_n=2*(energy-0.5)
225           v_n=2*(valence-0.5)
226           ang=math.degrees(math.atan2(e_n,v_n))
227           if ang<0:
228             ang=ang+360
229           else:
230             ang=ang
231
232           # Estados de animo : 5 estados
233           if (ang>0 and ang<=54) or (ang>342):
234             ani2='Happy'
235           elif ang>54 and ang<=126:
236             ani2='Excited'
237           elif ang>126 and ang<=198:
238             ani2='Angry'
239           elif ang>198 and ang<=270:
240             ani2='Sad'
241           elif ang>270 and ang<=342:
242             ani2='Relaxing'
```

Figura B.4 Obtención de la emoción del contenido, fuente: propia

5. Se obtiene la URL de cada contenido musical, a través de la API de Youtube; con el fin de obtener el video asociado a cada contenido musical (ver figura B.5).

```python
def obtener_url(artist,title):
  hdr = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11',
       'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
       'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
       'Accept-Encoding': 'none',
       'Accept-Language': 'en-US,en;q=0.8',
       'Connection': 'keep-alive'}

  consulta=artist+' - '+title+' Official Video'
  print consulta
  query_string = urllib.urlencode({'search_query' : consulta})
  html_content = urllib2.Request('https://www.youtube.com/results?' + query_string, headers=hdr)
  try:
    response = urllib2.urlopen(html_content)
    search_results = re.findall(r'href=\"\/watch\?v=(.{11})', response.read())
  except urllib2.HTTPError, e:
    print e.fd.read()
  lista=list()
  visitas=list()
  datos = []
  url_elegida=''
  for i in range(0,len(search_results)):
    url='http://www.youtube.com/watch?v='+search_results[i]
    logging.getLogger().setLevel(logging.ERROR)
    if(url not in lista):
      try:
        video=obtener_info_video(url)
        time.sleep(2)
        titulo=video['title']
        visual=video['view_count']
        catego=video['categories']
      except (EOFError, IOError, AttributeError, TypeError, ValueError, UnicodeError, UnicodeDecodeError, UnicodeEncodeError):
        print 'Oops! ERROR'
        pass
      proba=similar(consulta,titulo)
      print proba
      lista.append(url)
      if catego[0].encode('utf-8') == 'Music':
        if proba >= 0.6:
          visitas.append(visual)
          datos.append({'visualizaciones':visual,'url':url})
          print 'POSIBLE URL:',url
      logging.getLogger().setLevel(logging.ERROR)

  maxvis=video_mvisitado(visitas)
  for i in range(0,len(datos)):
    if(maxvis == datos[i]['visualizaciones']):
      url_elegida=datos[i]['url']
  print '**********************************'
  print 'URL ELEGIDA: ',url_elegida
  print '**********************************'
  datos=[url_elegida,maxvis]
  return datos
```

Figura B.5 Método para la obtención de url asociada a cada contenido, fuente: propia

6. Se genera un archivo JSON con la información de los contenidos, cabe mencionar que en este proyecto se seleccionaron 40 contenidos por cada uno de los cinco estados de ánimo, para un total de 200 contenidos (ver figura B.6).

```python
#*************** Metodos TinyDB *****************#
def agregar_cancion(animo,title,artist,energy,valence,liveness,tempo,speechiness,acousticness,
                    instrumentalness,mode,time_signature,duration,loudness,danceability,cur,url,visua):

  db = TinyDB('contents.json')
  lista=db.search((where('title')==title) & (where('artist')==artist))
  if(len(lista)==0):
    db.insert({'mood':animo,'title':title,'artist':artist,'energy':energy,'valence':valence,
               'liveness':liveness,'tempo':tempo,'speechiness':speechiness,'acousticness':acousticness,
               'instrumentalness':instrumentalness,'mode':mode,'time_signature':time_signature,
               'duration':duration,'loudness':loudness,'danceability':danceability,'currency':cur,'url':url,'view_count':visua})
  print '**********************'
  print 'CONTENIDO ALMACENADO'
  print '**********************'

def actualizar_url_view(artista,titulo,url,visi):
  db = TinyDB('contents.json')
  db.update(({'url':url}),((where('artist')==artista) & (where('title')==titulo)))
  db.update(({'view_count':visi}),((where('artist')==artista) & (where('title')==titulo)))
  print '**********************'
  print 'CONTENIDO ACTUALIZADO'
  print '**********************'

def listar_todo():
  db = TinyDB('contents.json')
  lista=db.all()
  return lista
#************************************************#
```

Figura B.6 Método TinyDB y generación de archivo JSON, fuente: propia

7. Finalmente, son descargados los contenidos de video, a través de la librería YoutubeDL para Python (ver figura B.7).

```
175  def download_video(url,art,tit,ani):
176    nombre=ani+'/'+art+' - '+tit+'.mp4'
177    ydl_opts = {
178      'format': '22,18',
179      'outtmpl': nombre,
180      'forcethumbnail': True,
181      'writethumbnail': True
182    }
183    with youtube_dl.YoutubeDL(ydl_opts) as ydl:
184      ydl.download([url])
185    time.sleep(2)
186    return 'Descargado: 100%'
187  #************************************************#
```

Figura B.7 Descarga de contenidos multimedia, fuente: propia

Anexo C

Instalación y manejo de servidor Spark

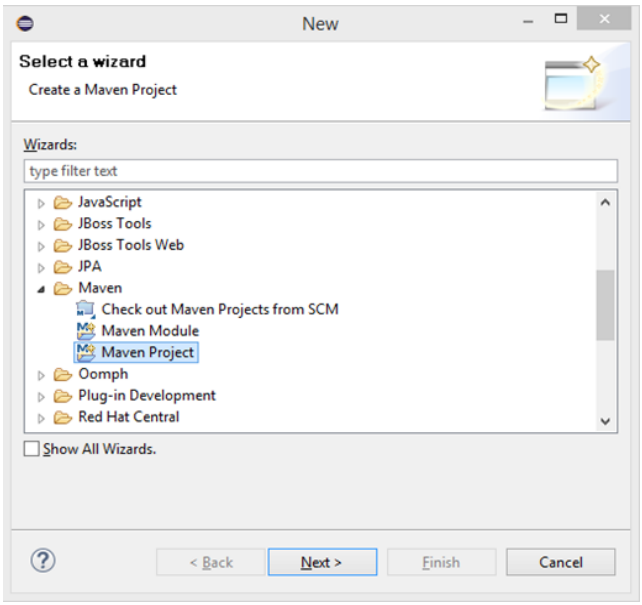
En este anexo es presentada la instalación y creación de un servicio web básico como un HelloWorld a través del framework Spark de Java, el cual permite crear aplicaciones o servicios web sencillos y livianos en el lenguaje de programación Java. Esta herramienta es usada en el presente trabajo de grado con el fin de conformar el servidor de procesamiento junto a la herramienta de minería de datos Weka y el gestor de base de datos MySQL, encargados del módulo clasificador bayesiano el cual provee de recomendaciones al S.R. Para el presente ejemplo se hizo uso de la plataforma Eclipse Mars IDE y el navegador Google Chrome.

A continuación, son descritos los pasos para crear un servicio web básico a través de Spark:

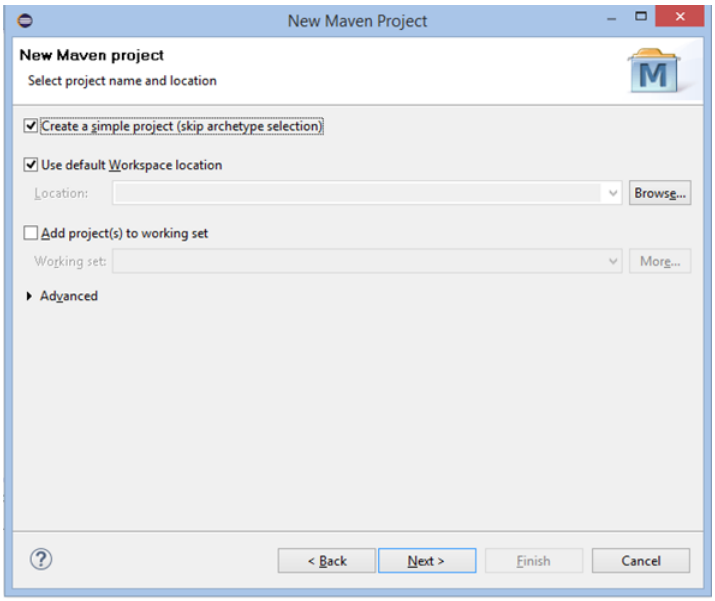
1. Se ejecuta Eclipse Mars y se crea un nuevo proyecto Maven, para ello deben ir a:
   File -> New -> Other…-> Maven -> Maven Project
   o
   Archivo -> Nuevo -> Otros… -> Maven -> Proyecto Maven

   Para ello es necesario seguir los tres siguientes pasos:

   Primer paso



Segundo paso



Tercer paso

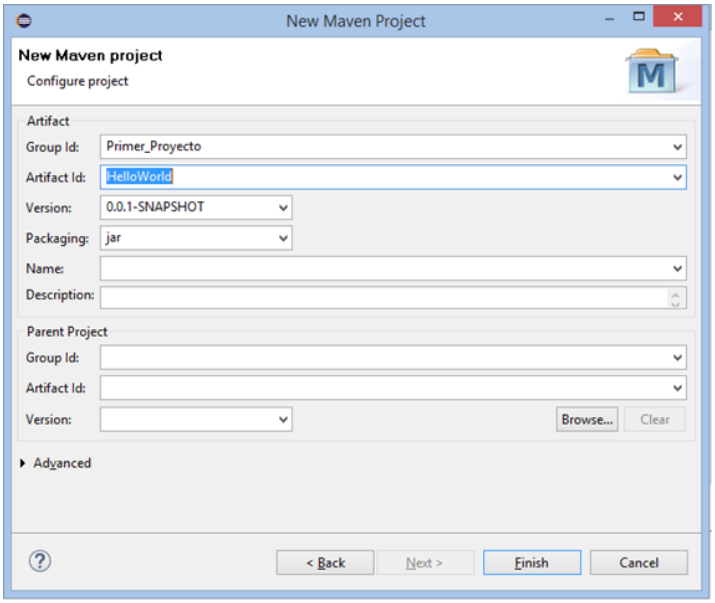
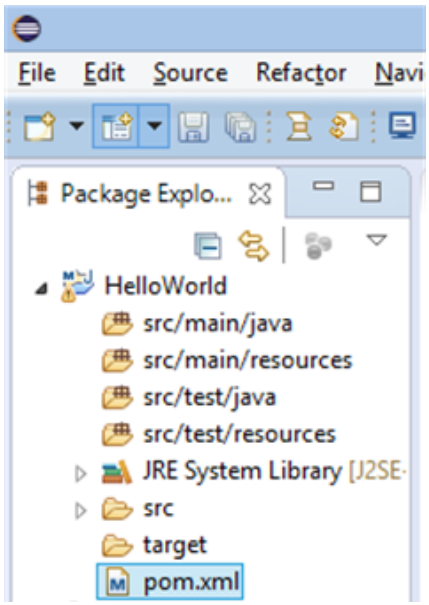


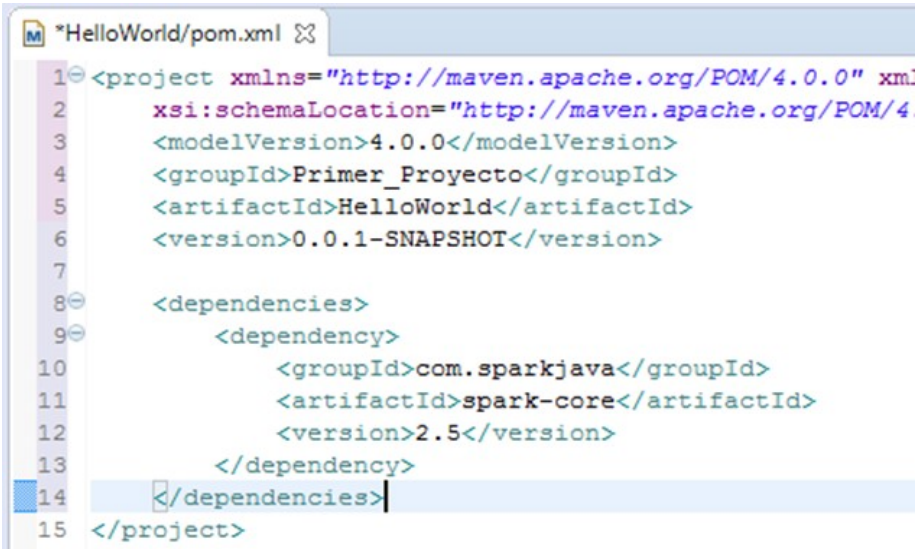
Figura C.1 Creación proyecto maven, fuente: propia

2. Se agrega la dependencia de Spark en el archivo descriptor del proyecto, seguir los tres pasos siguientes:

Primer paso



Segundo paso



Tercer paso

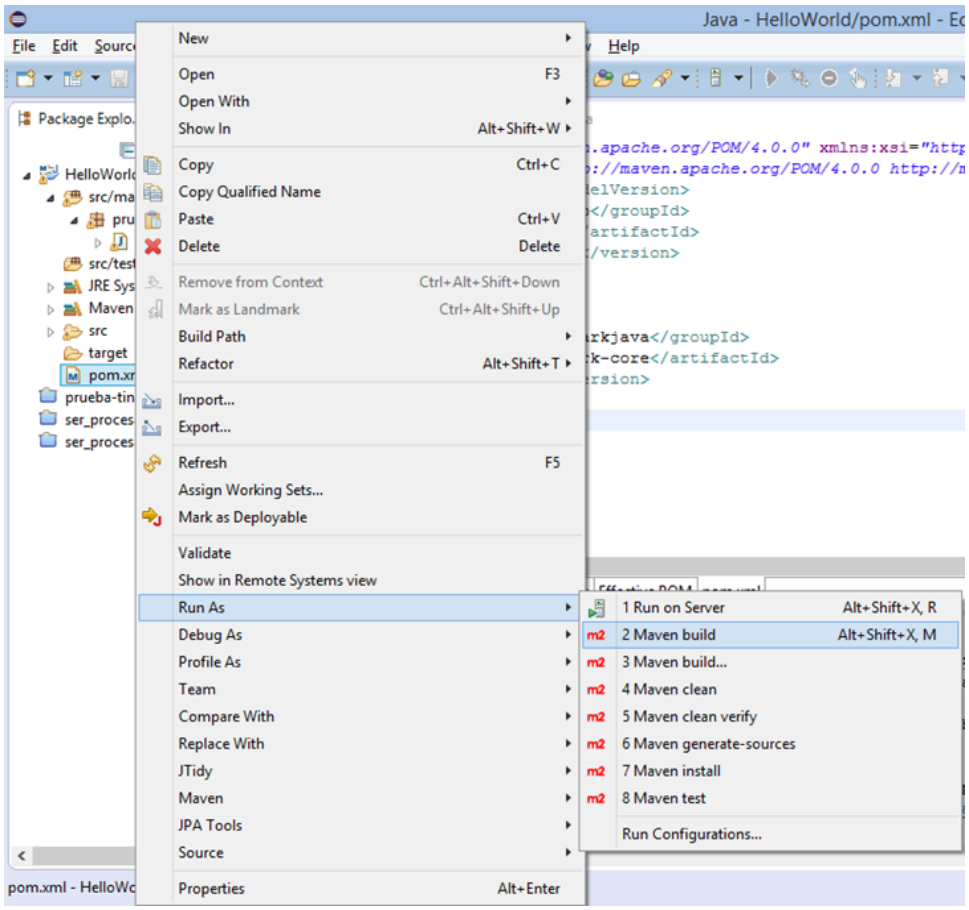
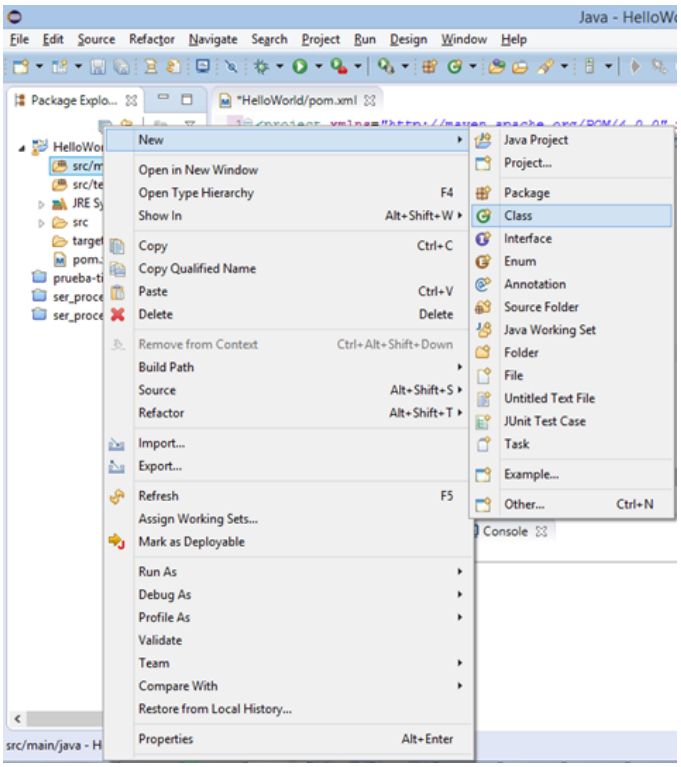


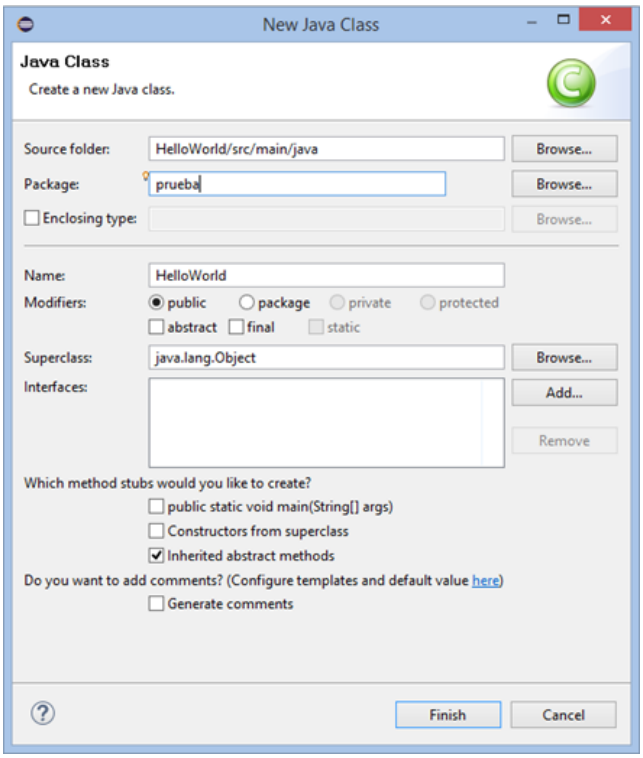
Figura C.2 Agregar dependencia de Spark al proyecto, fuente: propia

3. Se crea la clase con el ejemplo de HelloWorld para el servidor Spark, ver los cuatro pasos siguientes:

Primer paso



Segundo paso



Tercer paso

```
1  package prueba;
2
3  import static spark.Spark.*;
4
5  public class HelloWorld {
6      public static void main(String[] args){
7          get("/hello",(req,res)->"<h1>HELLO WORLD</h1>");
8      }
9  }
```

Cuarto paso

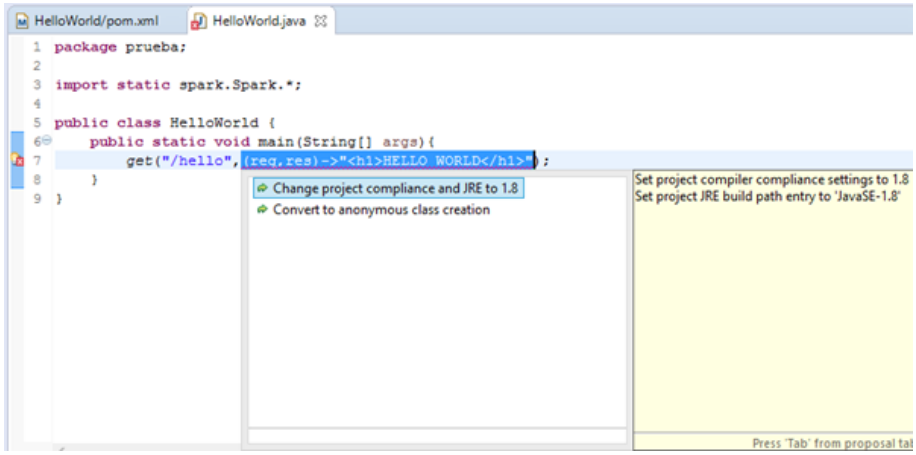
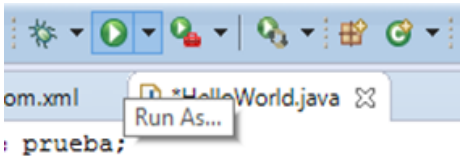


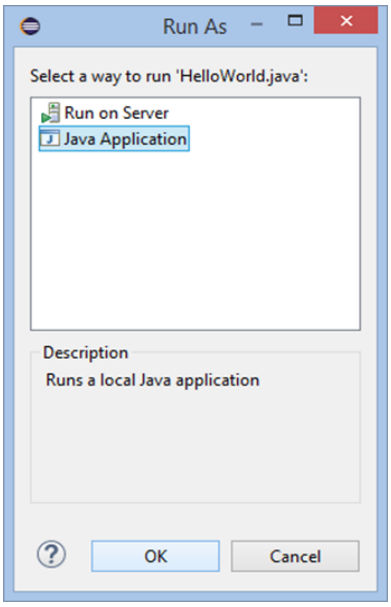
Figura C.3 Ejemplo HelloWorld en Spark, fuente: propia

4. Desplegar ejemplo de HelloWorld en navegador, a través de los tres siguientes pasos:

Primer paso



Segundo paso



Tercer paso

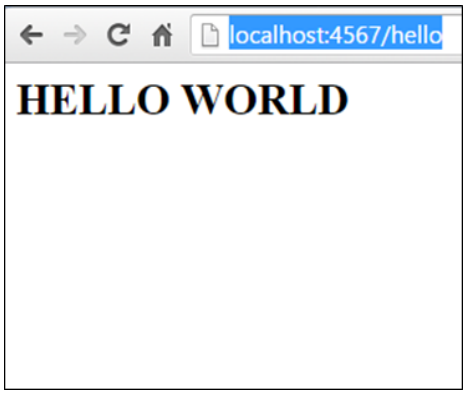


Figura C.4 Desplegar ejemplo HelloWorld, fuente: propia

Anexo D

Instalación y manejo de servidor Flask

En este anexo es presentada la instalación y creación de un servicio web básico a través de la librería Flask de Python, el cual permite crear aplicaciones o servicios web sencillos. Esta herramienta es usada en el presente trabajo de grado para implementar el servidor de contexto, encargado del procesamiento de las variables fisiológicas y emoción de usuario, junto al módulo hardware Arduino Yún. Para el presente ejemplo se hizo uso de la herramienta Open Source Geany y el navegador Mozilla Firefox. A continuación, son descritos los pasos para la instalación del framework Flask en el sistema operativo Linux:

1. Se instala el lenguaje de programación Python, en caso de que su versión de Linux ya brinde soporte de este lenguaje saltar este paso (ver figura D.1).

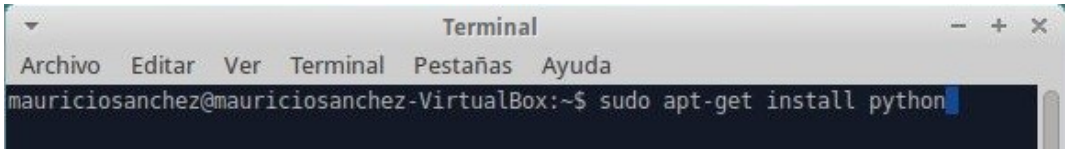


Figura D.1 Instalación de Python en Linux, fuente: propia

2. Se instala la librería de Flask para Python (ver figura D.2).

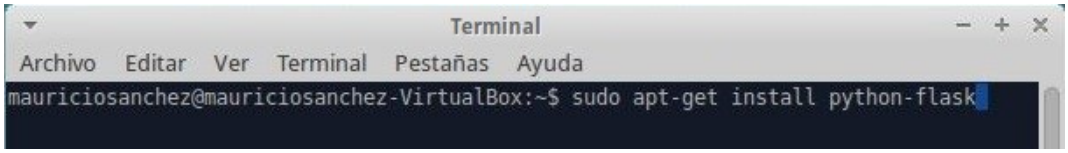


Figura D.2 Instalación de librería Flask, fuente: propia

3. Se instala el entorno de desarrollo Geany, para la crear proyecto HelloWorld en Flask (ver figura D.3).

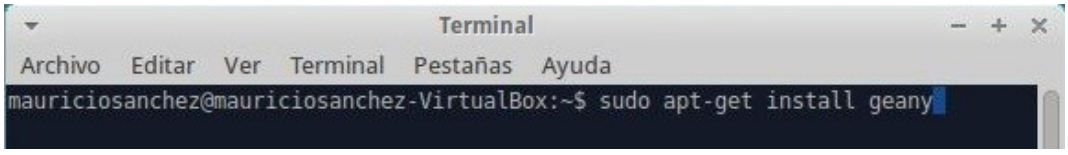


Figura D.3 Instalación del entorno de desarrollo Geany, fuente: propia

4. Se abre el entorno de desarrollo Geany, desde la consola terminal con el comando "geany" y se crea el script para el helloworld.py de Python (ver figura D.4).

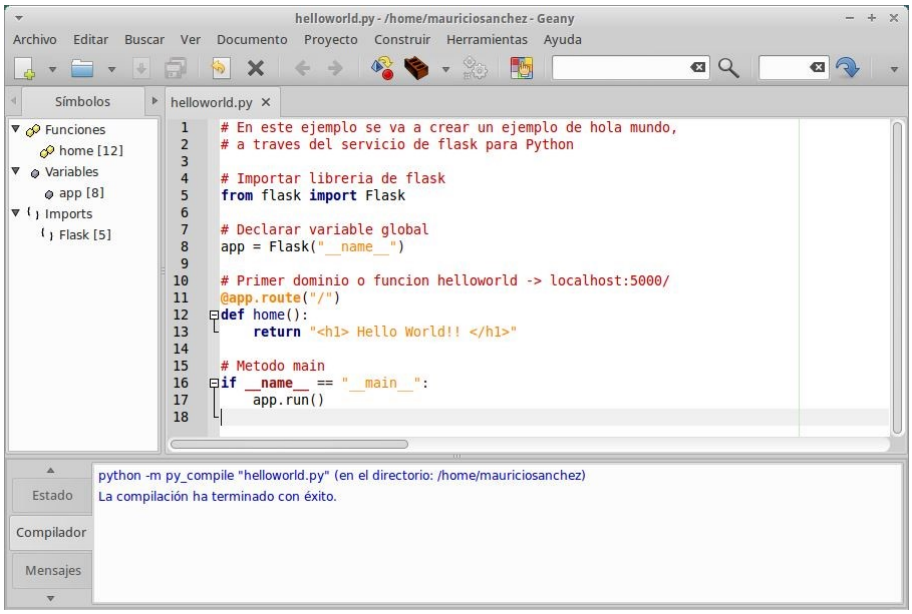


Figura D.4 Proyecto HelloWorld en Flask, fuente: propia

Como es observado en la figura D.4, primero es importada la librería de flask para Python, luego es necesario crear una variable global estática con la cual se va arrancar el proyecto. Luego es creado el dominio "/" con la función *home()*, la cual va a retornar el mensaje "Hello World!!" cada vez que es invocada la petición: localhost:5000/. Y finalmente es declarado el método *main*, el cual se encarga de arrancar el proyecto.

5. Se lanza el servidor Flask a través del botón compilar y es desplegado en el navegador Mozilla Firefox con la petición *http://localhost:5000/*.

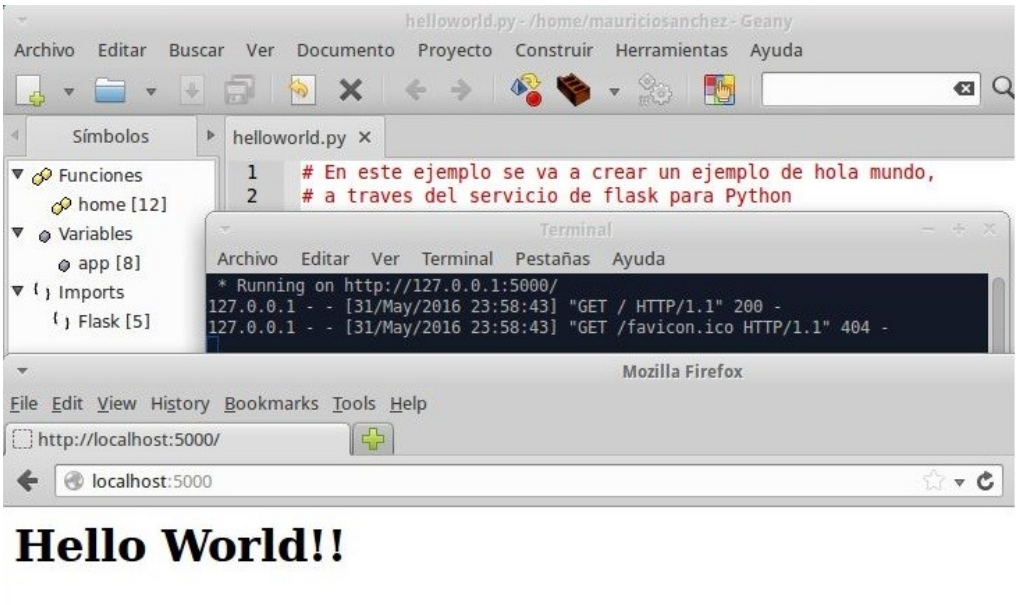


Figura D.5 Ejecución del proyecto HelloWorld en Flask, fuente: propia

Anexo E

Configuración y creación de servicio web RESTFul con la plataforma de Arduino Yún

En este anexo es presentada la configuración, creación y despliegue de un servicio web por medio de la plataforma Arduino Yún, para ello en primera instancia es necesario para iniciar la configuración del dispositivo, ingresar a su interfaz de red por medio del módulo wifi o ethernet, para motivos del ejemplo se hace uso de la interfaz de red wifi, como es apreciado en la figura E.1.

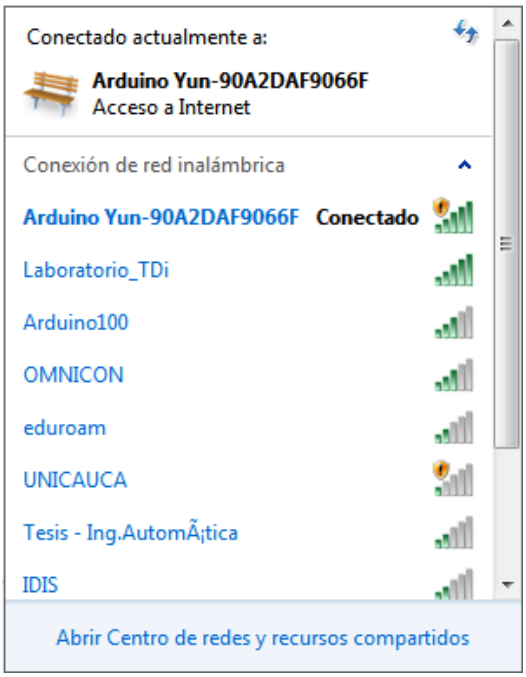


Figura E.1 Conexión a la red arduino yún, fuente: propia

Luego de estar conectado al Arduino Yún, en el navegador web del computador se ingresa a la puerta de enlace de red con la ip 192.168.240.1, esta pide una contraseña para permitir el acceso a la configuración que por defecto será “arduino”, una vez se realiza la validación es presentada la siguiente interfaz (ver figura E.2).

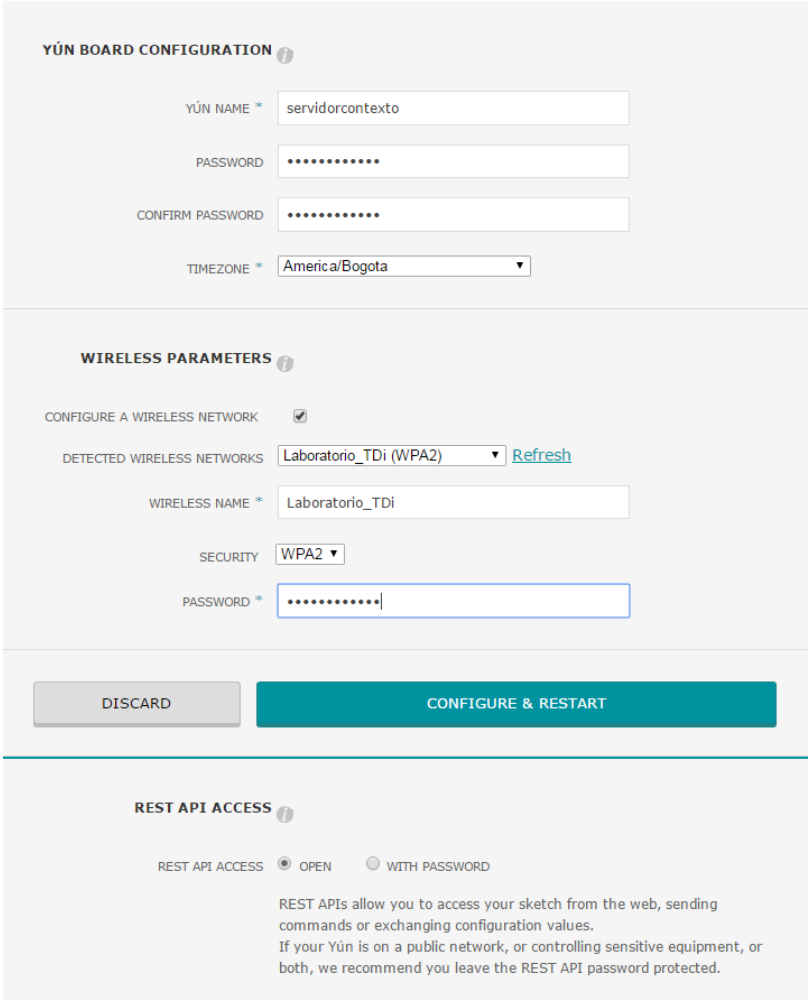


Figura E.2 Interfaz de configuración arduino yún, fuente: propia

En la interfaz de la figura E.2 se tienen 3 aspectos a configurar, como son la tarjeta yún y los parámetros de la red, y el acceso a la API rest. En la primera es ingresada la información referente a la validación para poder realizar modificación en la tarjeta y la carga de códigos. La segunda hace parte de la configuración de la red con la cual se piensa conectar la tarjeta Arduino Yún. Por último, es posible seleccionar si para el acceso al servicio es necesario ingresar una contraseña o por lo contrario cualquiera puede tener acceso y hacer uso del servicio.

Una vez es presionado el botón configurar y resetear, es guardada toda la información y es posible hacer uso de la tarjeta. Para ello en el IDE de Arduino en herramientas es escogida la placa y el puerto que se reconocerá con el nombre configurado en la yún seguido de la ip, luego de esto se procede con la carga del código en el dispositivo (ver figura E.3)

holamundo §

```
// se declaran la librerias para el montaje del servidor y el cliente HTTP
#include <Bridge.h>
#include <BridgeServer.h>
#include <BridgeClient.h>
BridgeServer server;

void setup() {
  Serial.begin(115200);
  Bridge.begin();
  // se indica al servidor para empezar a escuchar las conexiones entrantes puerto:5555
  server.listenOnLocalhost();
  server.begin();
}

void loop() {
  // se crea el objeto que hace uso de la clase BridgeClient
  BridgeClient cliente = server.accept();
  if(cliente){
    String comando = cliente.readString();
    comando.trim();
    Serial.println(comando);
    if(comando == "helloworld"){
      cliente.print("Hello World!!");
    }
    cliente.stop();
  }
  delay(50);
}
```

Figura E.3 Código prueba de arduino yún, fuente: propia

Para acceder a la API rest de Arduino, se ingresa en el navegador la url con la petición al servidor, esta contiene: los datos de ip de la Arduino, una marca con la representación “arduino” y el comando para lanzar la aplicación, en este caso es “helloworld” (ver figura E.4).

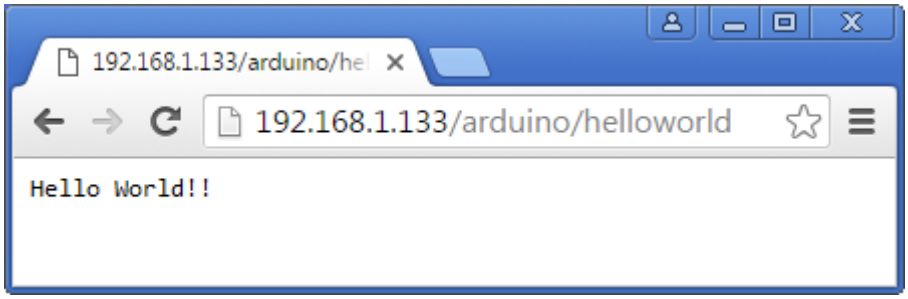


Figura E.4 Ejemplo de Hola mundo a través de la tarjeta arduino yún, fuente: propia

Anexo F

Manejo de la Herramienta Weka 3.6

En este anexo es mostrado el uso básico de la herramienta Weka 3.6[1], la cual contiene una serie de algoritmos de aprendizaje automático para tareas de minería de datos, de dónde se implementa el algoritmo de Naive Bayes para el desarrollo del clasificador del presente trabajo de grado. Para ello se hará uso de la interfaz gráfica con la cual esta cuenta, además, este proceso puede ser llevado acabo por medio de la consola de comandos o través de la API provista por la herramienta. La herramienta Weka trabaja con distintas fuentes de datos como son documentos, URL y base de datos de distintos formatos.

Para ilustrar este proceso la fuente de entrada de los datos en este caso es un documento de formato arff (*Attribute-Relation file format*) y es con el cual trabaja Weka, el cual presenta la siguiente estructura (ver figura F.1).

```
@relation peliculas

@attribute genero {drama,comedia, accion}
@attribute ano {2013,2014,2015}
@attribute class {buena,regular,mala}

@data
drama,2013,buena
drama,2013,regular
comedia,2013,mala
comedia,2014,regular
accion,2013,mala
comedia,2013,mala
comedia,2014,regular
drama,2013,buena
drama,2015,regular
drama,2015,mala
drama,2014,?
```

Figura F.1 Archivo arff weka, fuente: propia

El documento arff es cargado en el explorador de Weka de la figura F.2, y el cual contiene la información de un conjunto de películas, esto con el fin de solicitar la predicción del atributo *class* que tendría una película de género drama del 2014.

[1] Plataforma software para minería de datos. Página web oficial de weka: http://cs.waikato.ac.nz/~ml/weka.

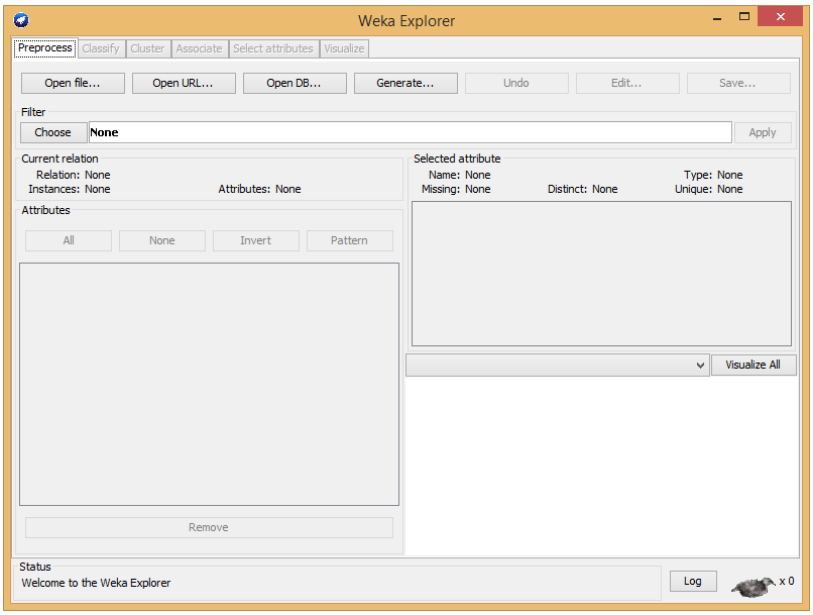


Figura F.2 Explorador herramienta weka, fuente: propia

En el explorador es cargado el archivo, luego la herramienta permite visualizar de forma gráfica los atributos del documento como es observado en la figura F.3.

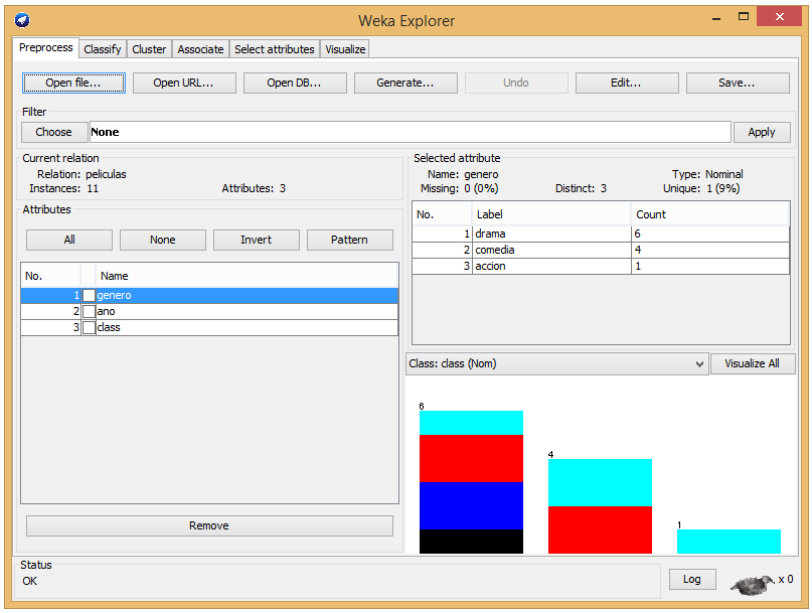


Figura F.3 Interfaz de atributos weka, fuente: propia

La herramienta Weka permite seleccionar entre distintos tipos de filtros con el fin de realizar trasformaciones sobre los atributos del documento (ver figura F.4), en caso de querer hacer un tratamiento especial para los datos de estudio, para lo cual solo se debe pulsar el botón de *choose* y elegir uno. Si no es el caso bien es posible empezar la clasificación de estos.

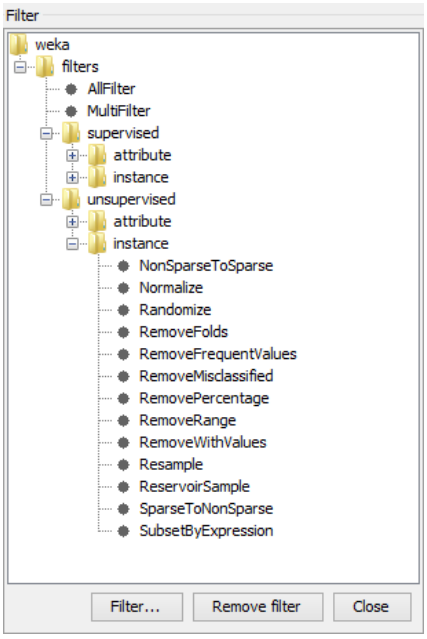


Figura F.4 Filtros de weka, fuente: propia

Una vez cargados los datos en la pestaña de *Classify* o clasificar se encuentran los distintos métodos que se pueden aplicar para el estudio de los datos como es mostrado en la figura F.5. Además, al pulsar sobre el clasificador escogido es posible configurar las propiedades de este según la necesidad. En este caso se hará uso del clasificador basado en Naive Bayes para el análisis de los datos del archivo arff.

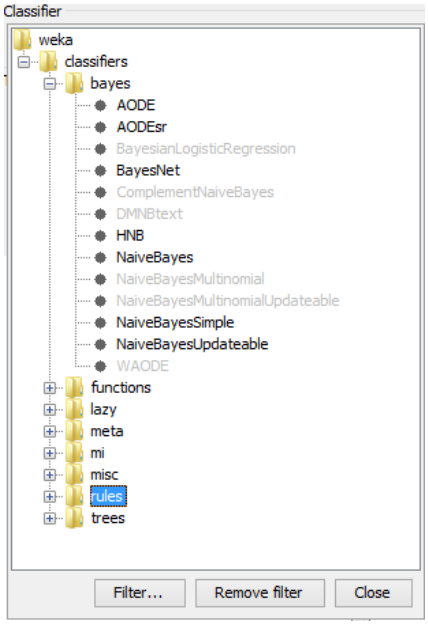


Figura F.5 Algoritmos de clasificación weka, fuente: propia

Al aplicar el tipo de clasificador y escoger las opciones del test en este caso fue el (*use training set*), es decir, la herramienta entrenara el método elegido con los datos proporcionados y luego lo aplicará otra vez sobre los mismos. Para ello se da inicio a la herramienta y en la pantalla del explorador de Weka es desplegada toda la información de análisis que provee la misma con relación a los datos proporcionados, como es apreciado en la figura F.7. Una vez se haya habilitado la pestaña (*Output predictions*) en las opciones de evaluación del clasificador (ver figura F.6) la cual permite el despliegue en pantalla del análisis, donde la herramienta proporciona las predicciones de los atributos.

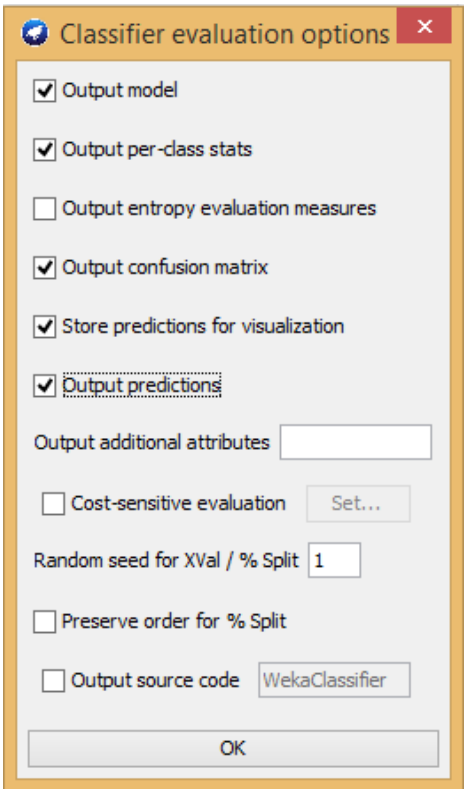


Figura F.6 Opciones de salida, fuente: propia

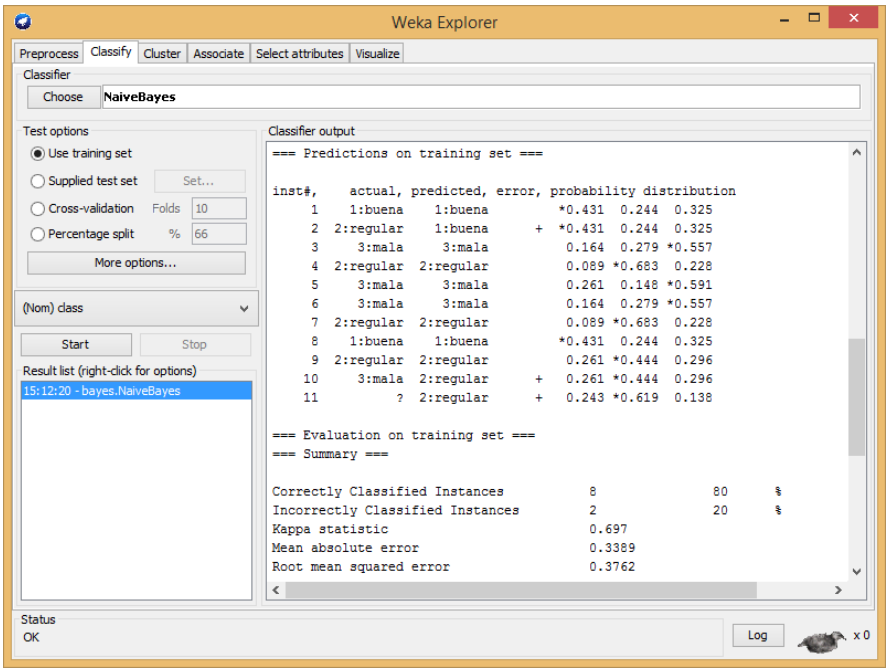


Figura F.7 Análisis de la herramienta weka sobre los datos iniciales, fuente: propia

La herramienta Weka proporciona información sobre las distintas predicciones realizadas, resumen de métricas estadísticas en cuanto el error como lo son: Root mean squared error, Mean absolute error, Relative absolute error, entre otros, además proporciona información relevante en cuenta el error del clasificador conocida como matriz de confusión, en este caso la predicción del atributo *class* realizada por la aplicación para la película drama del año 2014 es “regular”, con una probabilidad de 0.619.

Anexo G

Manejo de la herramienta apache benchmark

A continuación, es descrito el uso de la herramienta apache benchmark, la cual fue utilizada en el presente trabajo para la realización de pruebas de carga y estrés sobre el servidor de procesamiento y el servicio web proporcionado por módulo Arduino para la obtención de las variables fisiológicas. Para este ejemplo se hará uso del paquete software XAMP, el cual contiene en la carpeta de instalación el servidor apache con la herramienta ab.exe la cual es ejecutada desde CMD.

Primero antes de usar la herramienta se debe verificar si el servidor que aloja la aplicación está activo, para motivos de este ejemplo se ha puesto en funcionamiento el servidor de procesamiento trabajado en el proyecto como es apreciado en la figura G.1.

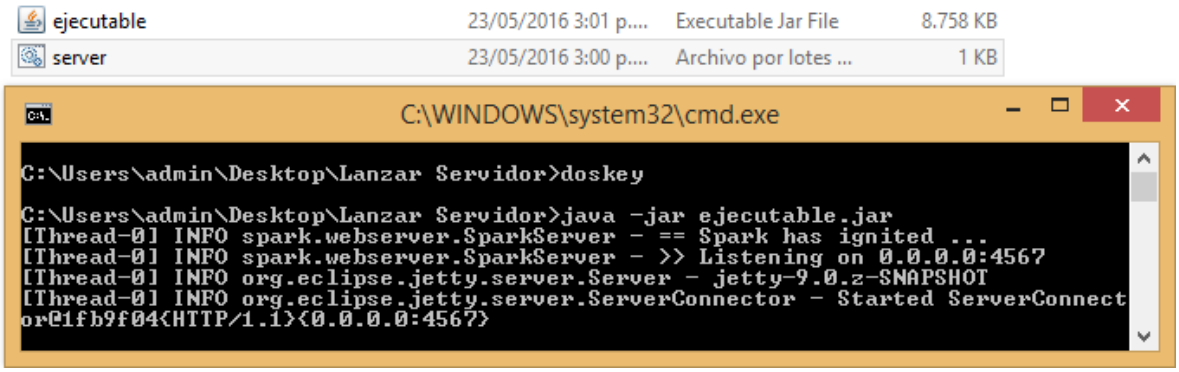


Figura G.1 Lanzado el servidor, fuente: propia

Luego es ejecutado el comando para ello si la carpeta del programa no se encuentra en el path del sistema, es necesario ingresar, en la carpeta bin de apache e ingresar el comando de la figura G.2.

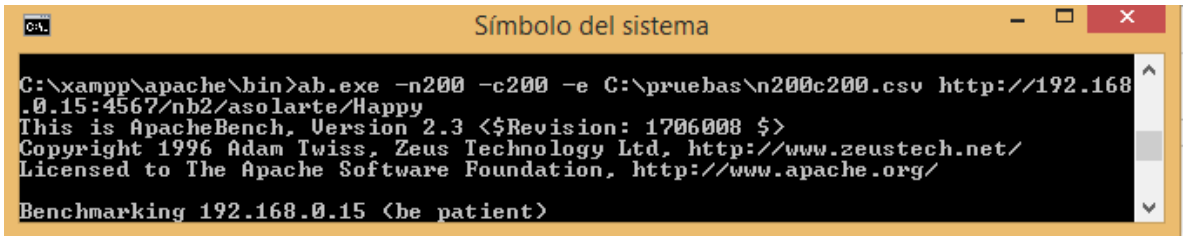


Figura G.2 Comando apache-benchmark, fuente: propia

Donde ab hace referencia al apache benchmark, para el número de conexiones a realizar es empleado el comando –n, para determinar el número de conexiones concurrentes que se deseen hacer se utiliza –c, y si se quiere guardar un archivo con el tiempo máximo en dar respuesta el servidor se hace uso del comando –e, lo cual genera un documento “csv”. Después de haber establecido los parámetros se ingresa la url y se ejecuta y en la misma consola es obtenido un resumen con los resultados como es apreciado en la figura G.3.

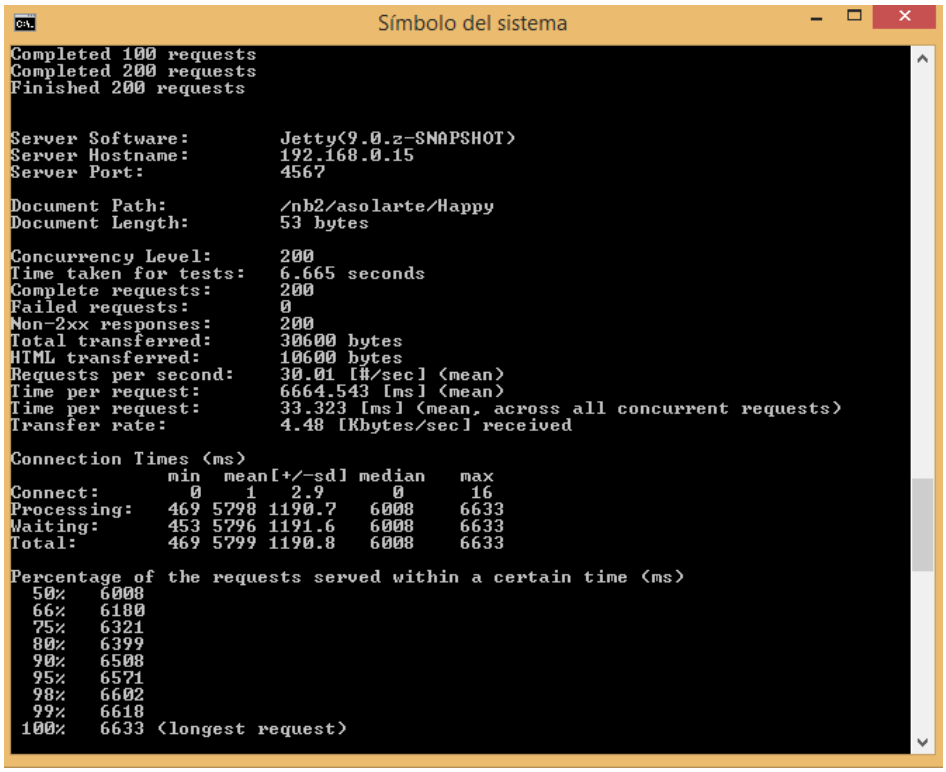


Figura G.3 Despliegue de resultados, fuente: propia

Donde la información relevante de los resultados es: Requests per second o conexiones atendidas por segundo durante la prueba, time per request (mean) o tiempo medio donde el servidor ha tardado en atender un grupo de conexiones concurrentes y time per request (mean, across all concurrent request) o tiempo medio donde el servidor tardó en atender una petición de forma individual.