

Práctica 0: Environment configuration

Morales Flores Víctor Leonel(2014081038)

Febrero 2019

1 Introducción

Servlets es un API de Java desarrollada con el objetivo de poder realizar aplicaciones web en este lenguaje de programación. En terminos muy simples un Servlet es una clase java que tiene la posibilidad para poder responder peticiones HTTP.

Para poder desarrollar aplicaciones web como los Servles es necesario tener un contenedor de Servlets para que puedan ser ejecutados. Existe una herramienta que surgió con ese propósito, funciona como un Servlet Container donde podemos almacenar nuestros proyectos para que puedan ser desplegados. Sin embarbo, Apache Tomcat es considerado un Web Container ya que también soporta otros archivos como son los JSP.

El desarrollo de aplicaciones web no requiere forzosamente de algún IDE para poder ser desarrolladas. Siempre que se cupla con la estructura definida para un proyecto Web los contenedores podrán interpretarlo y desplegarlo para que pueda comenzar a ofrecer el servicio.

Durante el contenido de la práctica se harán las configuraciones correspondientes para hacer la instalación de nuestro contenedor Web(Apache Tomcat), realizar configuraciones sobre Apache, instalar el kit de desarrollo de Java para poder compilar nuestras clases, generar una cuenta para Web App de Tomcat e instalaremos el IDE Eclipse para el desarrollo de nuestras aplicaciones web.

Aunque ya se comentó anteriormente que el uso de un IDE no es necesario, en el caso de aplicaciones que contengan una gran cantidad de Servlets, clases o recursos en general la complejidad para mantener un orden sin el uso de un IDE aumentará. Es por ello que se instaaará este IDE con el fin de facilitar el proceso de desarrollo y pruebas ya que con Eclipse también se instalará un servidor de desarrollo para realizar nuestras pruebas antes de desplegar nuestra aplicación web en nuestro servidor de producción.

2 Desarrollo

2.1 Instalación de Java 8:

El primer paso de la práctica consiste en instalar el kit de desarrollo de Java 8 que nos permitirá poder compilar y ejecutar códigos desarrollados en el lenguaje como lo son los Servlets.

Para ello accedemos a la página oficial de Oracle y descargaremos el archivo de instalación que aplique para nuestro sistema operativo. En nuestro caso el archivo que se descargará será el "jdk-8u201-linux-x64.tar.gz" que corresponde con Java Oracle 8.

Ya que se haya descargado el archivo procederemos a extraerlo y transferir los archivos resultantes de la extracción a la carpeta "/usr/local/java". Una vez transferidos debemos indicarle al sistema operativo dónde se encuentran los archivos necesarios para ejecutar y compilar aplicaciones Java. Esto lo haremos a través de la terminal con el uso de los siguientes comandos:

- `update-alternatives --install /usr/bin/java java /usr/local/java/jdk1.8.0_201/bin/java 1065`
- `update-alternatives --install /usr/bin/javac javac /usr/local/java/jdk1.8.0_201/bin/javac 1065`
- `update-alternatives --install /usr/bin/jar jar /usr/local/java/jdk1.8.0_201/bin/jar 1065`
- `update-alternatives --install /usr/bin/javaws javaws /usr/local/java/jdk1.8.0_201/bin/javaws 1065`

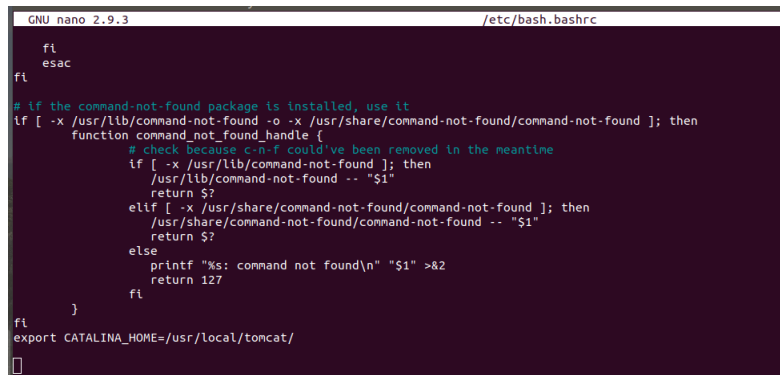
Con todo esto hecho solo nos queda indicarle al sistema de alternativas qué versión de Java responderá al nombre genérico Java. En nuestro caso sólo se encontraba una versión de Java por lo que el sistema indicó que sólo existía una alternativa en el grupo de enlaces java.

Una vez configurado Java en nuestra computadora el siguiente paso es la instalación y configuración de Apache Tomcat.

2.2 Instalación y configuración de Apache Tomcat 8

Para instalar tomcat se descargó en archivo "apache-tomcat-8.5.37.tar.gz" de la página oficial, se extrajo y el contenido de la carpeta resultante se movió a la ruta /usr/local/tomcat/.

Después editaremos nuestro archivo `bash.bashrc` y le agregaremos la variable de entorno `CATALINA_HOME` (`export CATALINA_HOME=/usr/local/tomcat/`). Con este último paso tomcat está instalado y lo siguiente a hacer es configurarlo.



```

GNU nano 2.9.3 /etc/bash.bashrc

fi
esac
fi

# if the command-not-found package is installed, use it
if [ -x /usr/lib/command-not-found -o -x /usr/share/command-not-found/command-not-found ]; then
    function command_not_found_handle {
        # check because c-n-f could've been removed in the meantime
        if [ -x /usr/lib/command-not-found ]; then
            /usr/lib/command-not-found -- "$1"
            return $?
        elif [ -x /usr/share/command-not-found/command-not-found ]; then
            /usr/share/command-not-found/command-not-found -- "$1"
            return $?
        else
            printf "%s: command not found\n" "$1" >&2
            return 127
        fi
    }
fi
export CATALINA_HOME=/usr/local/tomcat/

```

Figure 1: Archivo `bash.bashrc` con variable de entorno agregada.

Primero crearemos un nuevo archivo llamado `tomcat` en `/etc/init.d/` en el cual se almacenarán los comandos para que podamos iniciar, detener y reiniciar el servicio de tomcat a través de la consola con los comandos:

- `/etc/init.d/tomcat start`

- /etc/init.d/tomcat stop
- /etc/init.d/tomcat restart

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: tomcat
# Required-Start: $syslog
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: arranque automatico de tomcat
# Description:
### END INIT INFO
case $1 in
start)
sh /usr/local/tomcat/bin/startup.sh
;;
stop)
sh /usr/local/tomcat/bin/shutdown.sh
;;
restart)
sh /usr/local/tomcat/bin/shutdown.sh
sh /usr/local/tomcat/bin/startup.sh
;;
esac
exit 0
```

Figure 2: Archivo tomcat que se debe crear.

Para que esto funcione se le tienen que dar permisos de ejecución a través del comando "chmod 755 /etc/init.d/tomcat" y para que tomcat inicie con el sistema operativo se agregará un enlace simbólico al script de arranque de tomcat con el comando "update-rc.d tomcat defaults".

```

root@victor-Inspiron-3443:/# /etc/init.d/tomcat stop
Using CATALINA_BASE:   /usr/local/tomcat/
Using CATALINA_HOME:   /usr/local/tomcat/
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
root@victor-Inspiron-3443:/#
root@victor-Inspiron-3443:/#
root@victor-Inspiron-3443:/# /etc/init.d/tomcat start
Using CATALINA_BASE:   /usr/local/tomcat/
Using CATALINA_HOME:   /usr/local/tomcat/
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Tomcat started.
root@victor-Inspiron-3443:/#
root@victor-Inspiron-3443:/#
root@victor-Inspiron-3443:/#
root@victor-Inspiron-3443:/# /etc/init.d/tomcat stop
Using CATALINA_BASE:   /usr/local/tomcat/
Using CATALINA_HOME:   /usr/local/tomcat/
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
root@victor-Inspiron-3443:/#

```

Figure 3: Resultados esperados en consola tras la creación del archivo.

Para verificar que se haya realizado de forma correcta la instalación y configuración de tomcat usaremos un navegador web y nos dirigiremos a la ruta "http://127.0.0.1:8080". Con esta dirección se debería mostrar la página de inicio de Tomcat.

El último paso con tomcat consiste en crear un usuario para el gestor de tomcat. Esto se hará agregando al bloque "tomcat-users" del archivo tomcat-users.xml(/usr/local/tomcat/conf/tomcat-users.xml) las siguientes líneas:

- <role rolename="manager" />
- <role rolename="manager-gui" />
- <role rolename="admin" />
- <user username="admin" password="admin" roles="manager,manager-gui,admin" />

```

<role rolename="manager" />
<role rolename="manager-gui" />
<role rolename="admin" />
<user username="admin" password="admin" roles="manager,manager-gui,admin" />
</tomcat-users>

```

Figure 4: Bloque de código agregado al archivo tomcat-users.xml.

Reiniciaremos tomcat y al ingresar al Manager App deberá reconocer a nuestro usuario creado con su respectiva contraseña.

Una última modificación que hay que hacer consiste en cerrar el puerto 8080. Con esto las peticiones las controlará Apache y en prácticas posteriores se podrá colocar un 'balanceador de carga' que vuelva más segura la aplicación contra ataques por exceso de peticiones. Para ello hay que modificar el archivo `server.xml` que se encuentra en `'/usr/local/tomcat/conf'`.

```
Documentation at /docs/config/service.html
-->
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
    maxThreads="150" minSpareThreads="4"/>
  -->

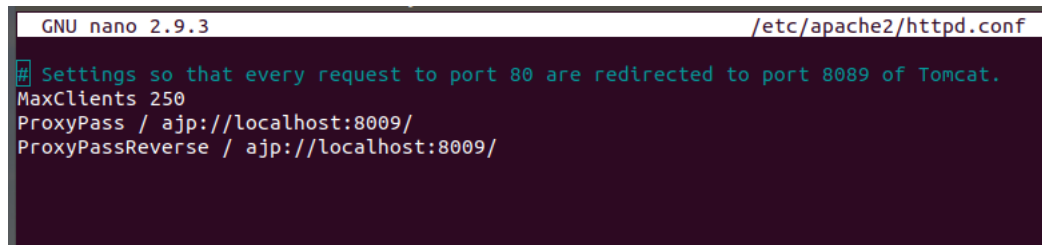
  <!-- A "Connector" represents an endpoint by which requests are received
  and responses are returned. Documentation at :
  Java HTTP Connector: /docs/config/http.html
  Java AJP Connector: /docs/config/ajp.html
  APR (HTTP/AJP) Connector: /docs/apr.html
  Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
  -->
  <!--<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
  -->
  <!-- A "Connector" using the shared thread pool-->
  <!--
  <Connector executor="tomcatThreadPool"
    port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
  -->
  <!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443
  This connector uses the NIO implementation. The default
  SSLImplementation will depend on the presence of the APR/native
  library and the useOpenSSL attribute of the
```

Figure 5: Archivo `server.xml` con líneas del puerto 8080 comentadas para deshabilitar el puerto.

2.3 Configuración de Apache

Para este proceso abriremos el archivo `"httpd.conf"` que se encuentra en la ruta `"/etc/apache2/"` y al final del documento agregaremos las siguientes líneas:

- `MaxClients 250`
- `ProxyPass / ajp://localhost:8009/`
- `ProxyPassReverse / ajp://localhost:8009/`



```
GNU nano 2.9.3 /etc/apache2/httpd.conf
# Settings so that every request to port 80 are redirected to port 8089 of Tomcat.
MaxClients 250
ProxyPass / ajp://localhost:8009/
ProxyPassReverse / ajp://localhost:8009/
```

Figure 6: Archivo httpd.conf editado.

Con esto estableceremos el número máximo de clientes a los que Apache contestará de forma recurrente, redijiremos el tráfico y configuraremos un Proxy inverso para que Apache pueda hacer cambios en los headers de las respuestas que se envían a Apache Tomcat. De esta forma los usuarios que son externos verán la respuesta por medio de Apache y no de Apache Tomcat.

Posteriormente instalamos mod_proxy con los siguientes comandos y reiniciamos Apache.

- # a2enmod proxy_ajp
- # a2enmod proxy

Configure Apache's AJP module to redirect requests to port 80 to Apache Tomcat.

En caso de que Tomcat no esté conectado con Apache agregaremos las siguientes líneas de código al archivo apache2.conf(/etc/apache2/apache2.conf):

```
GNU nano 2.9.3 /etc/apache2/apache2.conf

# (the actual bytes sent including headers) instead of %b (the size of the
# requested file), because the latter makes it impossible to detect partial
# requests.
#
# Note that the use of %[X-Forwarded-For]i instead of %h is not recommended.
# Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%[Referer]i\" \"%[User-Agent]i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%[Referer]i\" \"%[User-Agent]i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%[Referer]i -> %U" referer
LogFormat "%[User-agent]i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
# Include all the user configurations:
Include httpd.conf

```

Figure 7: Captura del archivo apache2.conf con las últimas 3 líneas añadidas.

2.4 Instalación de Eclipse y Jetty

Para la instalación de Eclipse se buscó el archivo para el sistema operativo de la computadora en la página oficial (<https://www.eclipse.org/downloads/packages/>). Una vez descargado se descomprimió el archivo y, de igual forma que con los programas anteriores, se trasladó a la carpeta `/usr/local/eclipse`.

Para crear el ícono y poder ejecutar la aplicación de forma sencilla se creó el archivo `eclipse.desktop` en `/usr/share/applications/` con las siguientes líneas:

- Desktop Entry
- Name=Eclipse
- Comment= IDE Eclipse EE
- Exec=/usr/local/eclipse/eclipse
- Icon=/usr/local/eclipse/icon.xpm
- Terminal=false
- Type=Application

Posteriormente, para la instalación de Jetty se abrió el IDE, nos dirigimos a "Help-¿Eclipse Marketplace..." y se buscó Jetty en la ventana emergente.

La instalación de Jetty sólo consistió en escoger la opción "Run-Jetty-Run 1.3.5-nightly" y darle clic en "Install". Tras terminar la instalación se reinició el IDE y Jetty ya se podía utilizar como servidor de desarrollo.

3 Pruebas

3.1 Versión de Java instalada

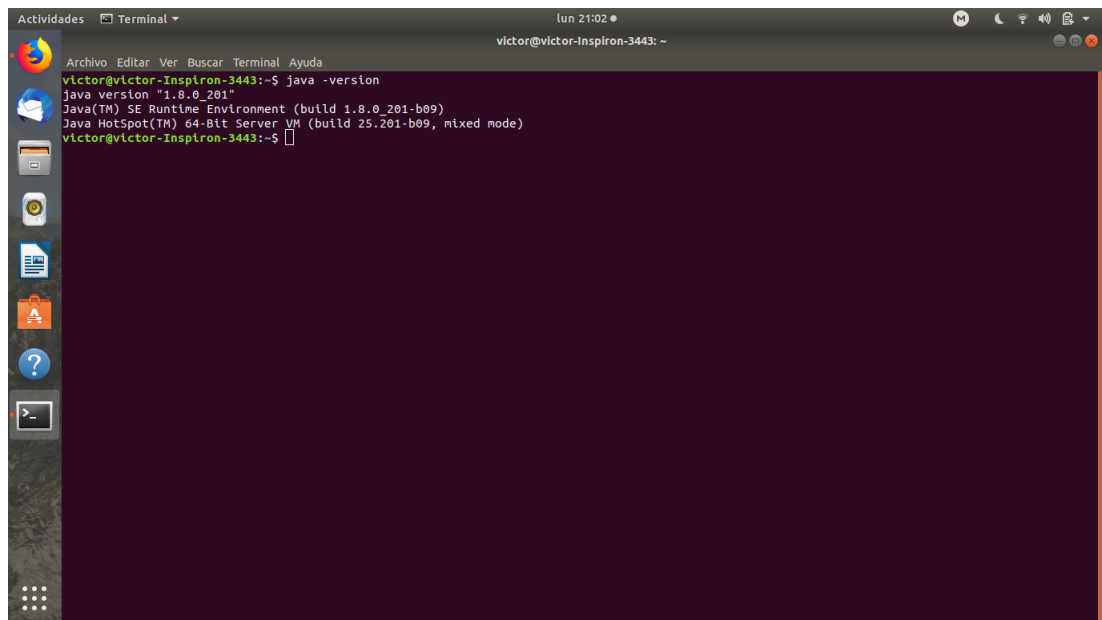
A screenshot of a terminal window titled "Terminal" with a menu bar containing "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal shows the command `java -version` and its output: `java version "1.8.0_201"`, `Java(TM) SE Runtime Environment (build 1.8.0_201-b09)`, and `Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)`. The prompt is `victor@victor-Inspiron-3443:~$`. The terminal window is part of a desktop environment with a sidebar on the left showing various application icons and a top status bar with system icons and the time "lun 21:02".

Figure 8: Captura de Java 8 instalado.

3.2 Apache Tomcat(Servidor de producción)

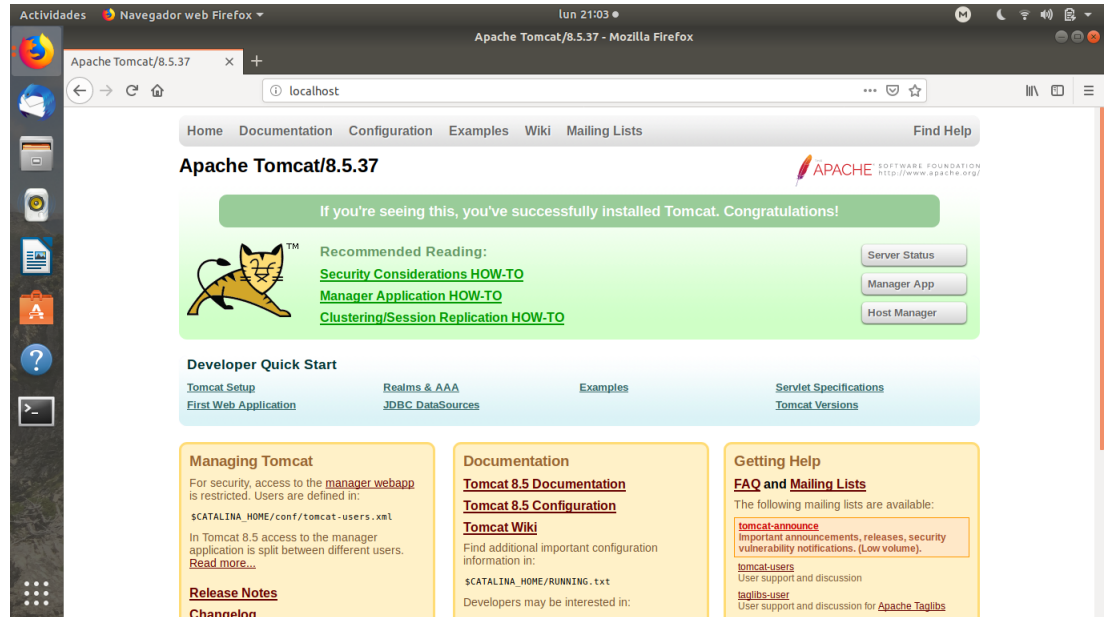
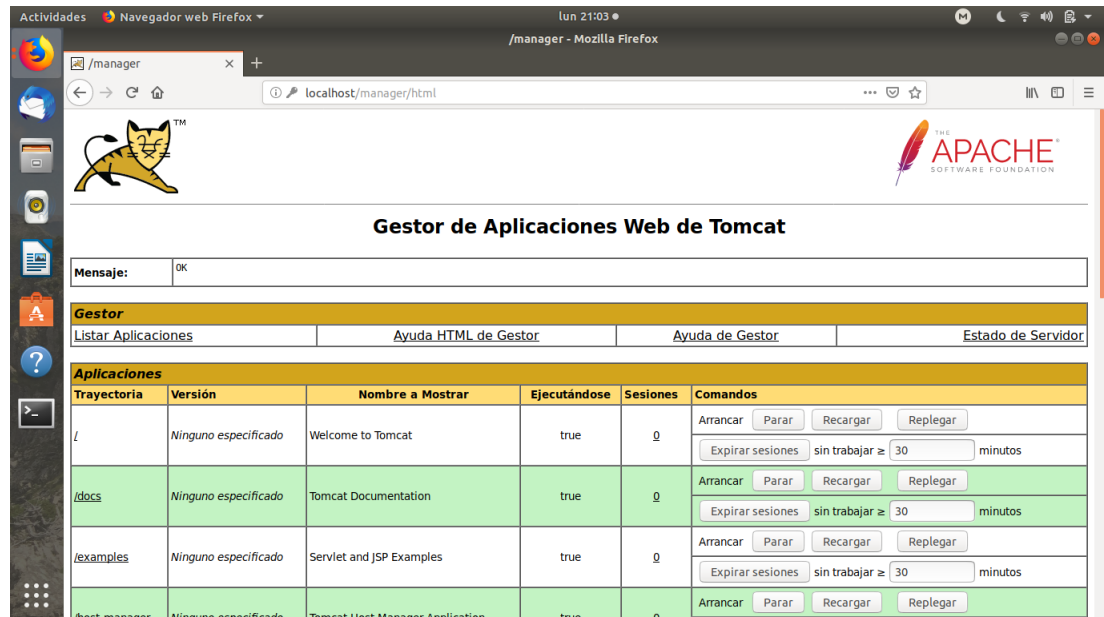




Figure 9: Captura de Tomcat en ejecución.

3.3 WebManager de Tomcat



Actividades Navegador web Firefox lun 21:03 /manager - Mozilla Firefox

/manager localhost/manager/html

Gestor de Aplicaciones Web de Tomcat

Mensaje: OK

Gestor

Listar Aplicaciones Ayuda HTML de Gestor Ayuda de Gestor Estado de Servidor

Aplicaciones					
Trayectoria	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado	Welcome to Tomcat	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/docs	Ninguno especificado	Tomcat Documentation	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/examples	Ninguno especificado	Servlet and JSP Examples	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	0	Arrancar Parar Recargar Replegar

Figure 10: Captura de Web App Manager.

3.4 Puerto 8080 cerrado

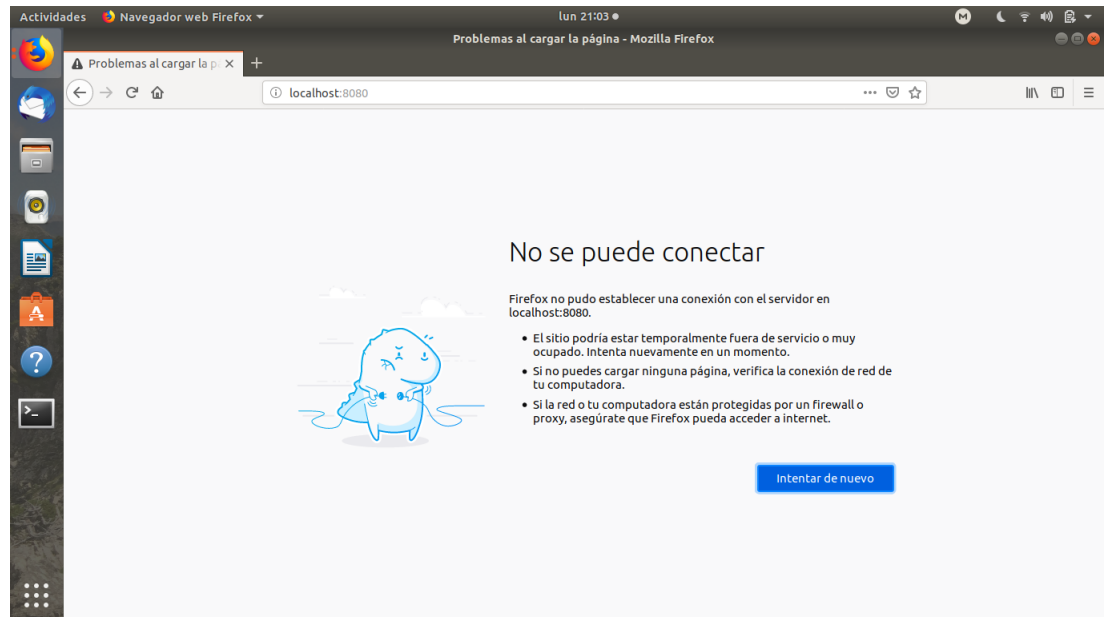


Figure 11: Captura del puerto 8080 cerrado.

3.5 IDE Eclipse instalado

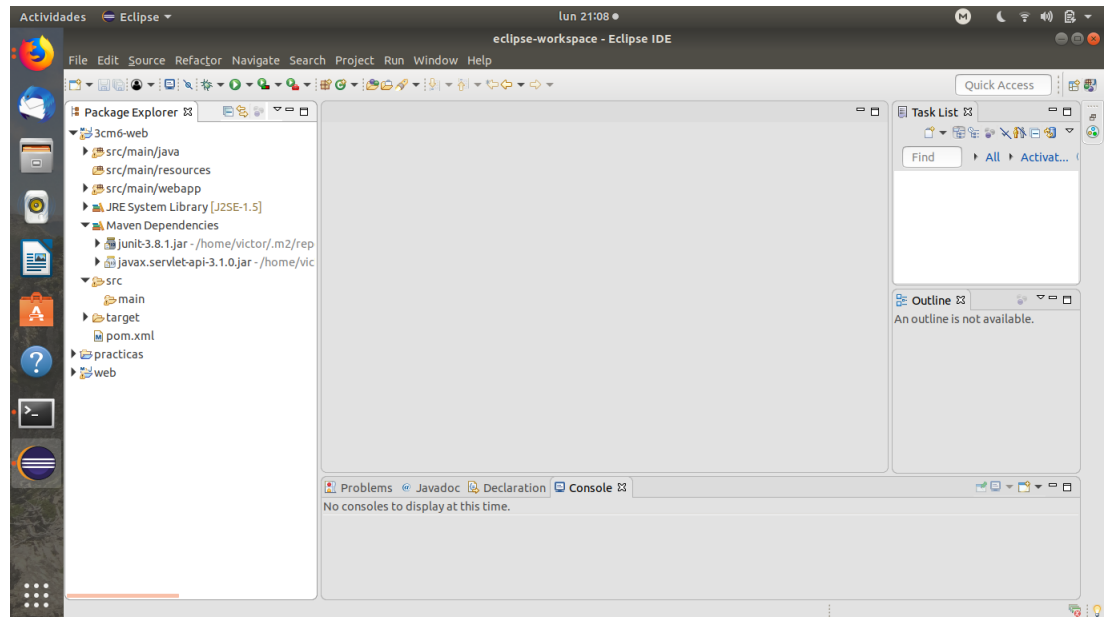


Figure 12: Captura de IDE Eclipse en ejecución.

3.6 Uso de Jetty como servidor de desarrollo

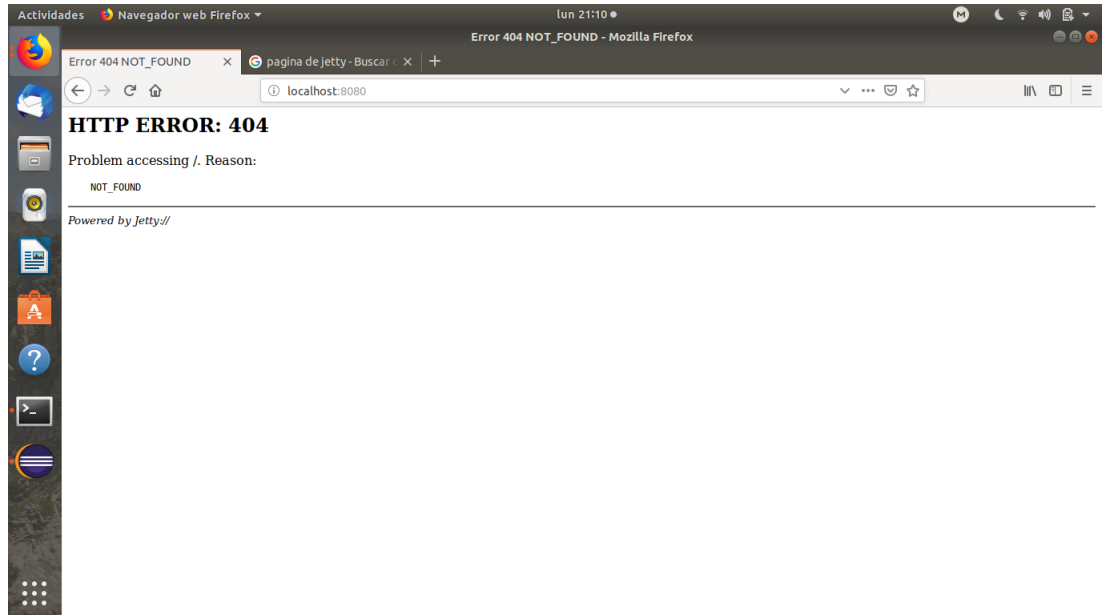


Figure 13: Captura de servidor de desarrollo en ejecución(Jetty)

4 Conclusiones

Para poder desarrollar aplicaciones web con el uso de Servlets se deben tener instaladas algunas herramientas que nos permita desplegar la aplicación para que puedan atender las peticiones de los clientes.

Hay detalles que usualmente no se contemplan como lo es el dejar el puerto 8080 abierto. Aunque al inicio de las prácticas tal vez esto no cobre relevancia, el hacerlo nos permitirá que posteriormente se le pueda agregar "balanceadores de carga" a nuestras aplicaciones y volverlas más confiables y menos susceptibles a un ataque.

Otro punto importante es el tener disponibles herramientas como Jetty que nos proporcionan un servidor de desarrollo para que llevemos a cabo todas nuestras pruebas antes de que la aplicación sea desplegada en un servidor de producción.

El contar con un IDE para desarrollar las aplicaciones web nos será de mucha ayuda, ya que éste se encargará de crear la estructura de directorios como se debe, mantener un orden a lo largo de todo el desarrollo y así evitar complicaciones en el futuro.