

DESPLIEGUE DE APLICACIONES WEB

GUÍA DE RESOLUCIÓN DE LA TAREA 02

1.1.- Módulos de Tomcat.....	2
1.2.- Gestor de aplicaciones de Apache Tomcat.	4
Ejercicio 2.- Desplegando en Apache Tomcat.....	7
2.1. Genera un archivo war con el contenido de la aplicación web.....	11
2.2 Despliega el archivo war en el servidor Apache Tomcat.....	11
2.2.1 Copiando directamente la carpeta en la ruta adecuada:.....	11
2.2.2 Desplegando el archivo war desde el entorno web de Apache (página web del servidor de aplicaciones):	12
Ejercicio 3. Wildfly.....	13
3.1.- Instalación de Wildfly.	13
Actividad 3.2.- Acceso a la consola de Wildfly.....	16
3.3.- Desplegar un WAR en Wildfly.....	18
Ejercicio 4.- Apache ANT.	21
Actividad 4.1.- Instalación de Apache Ant.....	21
Actividad 4.2.- Elabora un fichero buildmain.xml que realice las siguientes acciones.	23
Cree un directorio llamado "compilado" y otro "distribucion/lib"	23
Debe compilar todos los archivos java del directorio "origen" y almacenarlos en el directorio "compilado".	24
Creación del fichero "jar" nombrado "daw22-23_nombre_apellidos_yyyyMMdd.jar", donde yyyyMMdd corresponde a la fecha del sistema.	24
Borre posteriormente el directorio "compilado" y todo su contenido.	24
Archivo buildmain finalizado:.....	24
Comprobando que funciona.	25

Ejercicio 1 Apache Tomcat.

En la unidad anterior hicimos una instalación básica de Apache Tomcat en la que debíamos únicamente acceder a la página principal del servidor (localhost:8080). Partiendo de este escenario en esta actividad vamos a realizar los siguientes apartados:

1.1.- Módulos de Tomcat.

Describe los módulos y enumera los módulos más importantes de Apache y de Tomcat. Y, además, describe los principales archivos de configuración de Apache. [Esto viene referenciado al principio del punto 3.]

La arquitectura del servidor Apache es muy modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web. Algunos de estos módulos son:

- `mod_ssl` - Comunicaciones Seguras vía TLS.
- `mod_rewrite` - reescritura de direcciones (generalmente utilizado para transformar páginas dinámicas como php en páginas estáticas html para así engañar a los navegantes o a los motores de búsqueda en cuanto a cómo fueron desarrolladas estas páginas).
- `mod_dav` - Soporte del protocolo WebDAV (RFC 2518).
- `mod_deflate` - Compresión transparente con el algoritmo deflate del contenido enviado al cliente.
- `mod_auth_ldap` - Permite autenticar usuarios contra un servidor LDAP.
- `mod_proxy_ajp` - Conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java (servlets y JSP).
- `mod_cfml` - Conector CFML usado por Railo.

Los principales archivos de configuración de Apache son:

- `httpd.conf`: Archivo principal que contiene la configuración global de Apache, incluyendo puertos, direcciones IP y módulos.
- `apache2.conf` o `httpd-vhosts.conf`: Archivos adicionales de configuración, donde `apache2.conf` puede contener configuraciones globales y `httpd-vhosts.conf` define la configuración de hosts virtuales.
- `ssl.conf` o `httpd-ssl.conf`: Archivo de configuración específico para SSL/TLS, incluyendo la configuración de certificados y claves privadas.
- `mods-available` y `mods-enabled`: Directorios que contienen archivos de configuración de módulos. `mods-available` tiene configuraciones disponibles, y `mods-enabled` contiene enlaces a

los módulos activos.

Cabe destacar que apache usa directivas para agregar o extender estas configuraciones y que los archivos `.htaccess` son los encargados de contener dichas directivas, localizados en una estructura típica de árbol de directorios.

Tomcat tiene una arquitectura jerárquica compuesta de diversos componentes, cada uno encargado de una función. Vamos a aproximarnos a los **elementos que componen** su estructura interna de la forma más sencilla posible:

- **Servidor**. Representa al contenedor de servlets en si mismo, es decir, a Tomcat en todo su conjunto. Dentro de él se ejecutan el resto de componentes.
- Dentro del **servidor** podemos tener uno o más **servicios** ejecutándose. La función de un **servicio** es enlazar el componente encargado de comunicarse con el cliente, con la parte que procesa la petición y genera una respuesta.
- Dentro de un servicio podremos tener:
 - Uno o más **conectores** (connectors). Es el componente encargado de recibir peticiones del cliente remoto y de pasárselas al motor (engine), una vez que el motor procesa la petición, el conector hará llegar la respuesta al cliente remoto.
 - **Motor** (engine). Será el encargado de procesar la petición y de devolver el contenido generado al conector para que lo haga llegar al cliente remoto. Para lograr su cometido tendrá que ejecutar, si se requiere, la aplicación web que hemos diseñado.
 - Dentro de un motor, podremos tener uno o varios **Hosts**. El propósito del **Host** es asociar un nombre de equipo al servidor Tomcat, de tal manera que nuestro servidor Tomcat pueda responder peticiones asociadas a un nombre de equipo concreto.

Los principales módulos de Tomcat son:

- Catalina: Actúa como el contenedor de servlets y JSP, gestionando la ejecución de aplicaciones Java en Tomcat.
- Jasper: Un compilador JSP que convierte archivos JSP en servlets para su ejecución en Tomcat.
- Connector (mod_jk/mod_proxy_ajp): Facilita la comunicación entre Apache y Tomcat para la ejecución de servlets y JSP.
- Realm: Gestiona la autenticación y autorización de usuarios para aplicaciones web en Tomcat.
- Cluster: Ofrece soporte para configuraciones de clúster, mejorando la escalabilidad y la disponibilidad.

No hay que olvidar que Tomcat es una aplicación Java y como tal podríamos consultar la API para hacerlos una idea de los módulos, clases y funcionalidades que tiene el aplicativo.

1.2.- Gestor de aplicaciones de Apache Tomcat.

Se pide:

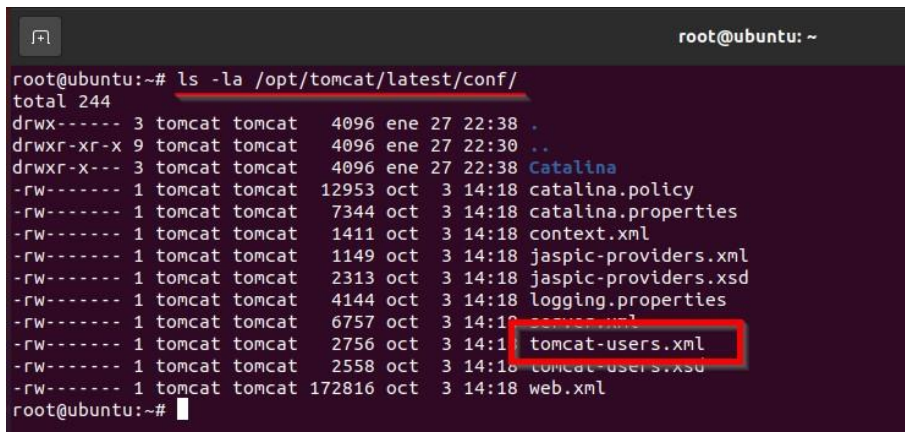
- Configura Tomcat, para que puedas acceder vía web con un usuario al gestor de aplicaciones de Tomcat
Crea un usuario que permita acceder a la interfaz html y permita el acceso a la interfaz de texto sin formato llamado usuarioXXX donde las X sean las 3 últimas cifras de tu nº de DNI, y password DAW.23.

Para poder acceder via web al gestor de aplicaciones de Tomcat debemos agregar un usuario con perfil de gestor (manager-gui) al fichero de usuarios.

Este fichero lo podemos encontrar en la carpeta de configuración de Tomcat. Dicha carpeta puede variar en función de si hemos hecho la instalación manual o automática de Tomcat.

En mi caso, como hice la instalación manual, el fichero está en:

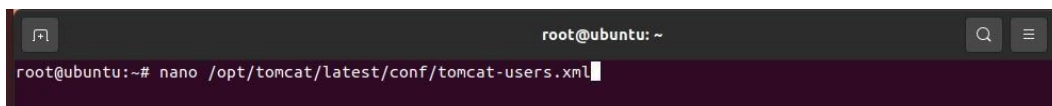
/opt/tomcat/latest/conf/tomcat-users.xml



```
root@ubuntu:~# ls -la /opt/tomcat/latest/conf/
total 244
drwx----- 3 tomcat tomcat 4096 ene 27 22:38 .
drwxr-xr-x 9 tomcat tomcat 4096 ene 27 22:30 ..
drwxr-x--- 3 tomcat tomcat 4096 ene 27 22:38 Catalina
-rw----- 1 tomcat tomcat 12953 oct 3 14:18 catalina.policy
-rw----- 1 tomcat tomcat 7344 oct 3 14:18 catalina.properties
-rw----- 1 tomcat tomcat 1411 oct 3 14:18 context.xml
-rw----- 1 tomcat tomcat 1149 oct 3 14:18 jaspic-providers.xml
-rw----- 1 tomcat tomcat 2313 oct 3 14:18 jaspic-providers.xsd
-rw----- 1 tomcat tomcat 4144 oct 3 14:18 logging.properties
-rw----- 1 tomcat tomcat 6757 oct 3 14:18 server.xml
-rw----- 1 tomcat tomcat 2756 oct 3 14:18 tomcat-users.xml
-rw----- 1 tomcat tomcat 2558 oct 3 14:18 tomcat-users.xsd
-rw----- 1 tomcat tomcat 172816 oct 3 14:18 web.xml
root@ubuntu:~#
```

Lo edito con nano y creo el usuario:

\$ nano /opt/tomcat/latest/conf/tomcat-users.xml



```
root@ubuntu:~# nano /opt/tomcat/latest/conf/tomcat-users.xml
```

Buscamos el bloque <tomcat-users...> </tomcat users> y añadimos:

```
<tomcat-users . . .> .
. .
<user username="dedaw" password="aguadulce" roles="manager-gui"/>
. . .
</tomcat-users>
```

```
root@ubuntu: ~
GNU nano 4.8 /opt/tomcat/latest/conf/tomcat-users.xml
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
  version="1.0">
<!--
By default, no user is included in the "manager-gui" role required
to operate the "/manager/html" web application.  If you wish to use this app,
you must define such a user - the username and password are arbitrary.

Built-in Tomcat manager roles:
- manager-gui - allows access to the HTML GUI and the status pages
- manager-script - allows access to the HTTP API and the status pages
- manager-jmx - allows access to the JMX proxy and the status pages
- manager-status - allows access to the status pages only

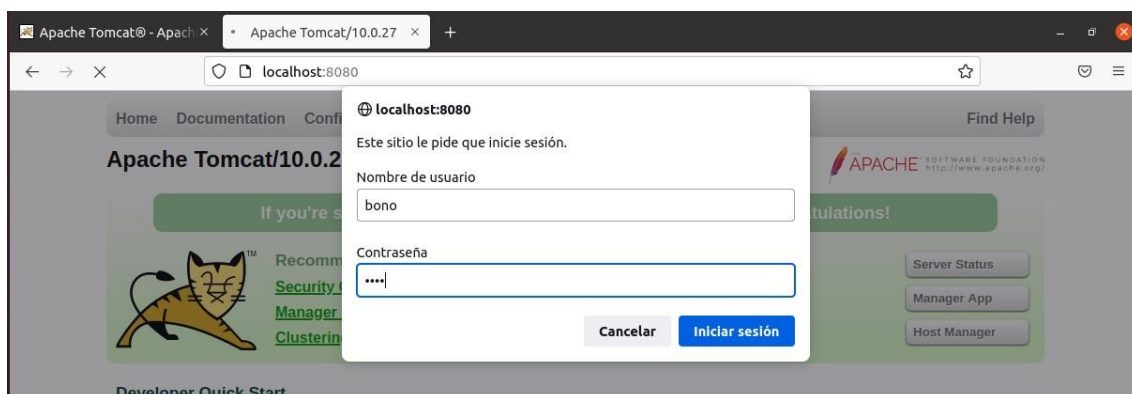
The users below are wrapped in a comment and are therefore ignored. If you
wish to configure one or more of these users for use with the manager web
application, do not forget to remove the <!-- ... --> that surrounds them. You
will also need to set the passwords to something appropriate.
-->
<!--
<user username="admin" password="<must-be-changed>" roles="manager-gui"/>
<user username="robot" password="<must-be-changed>" roles="manager-script"/>
-->
<!--
<user username="bono" password="bono" roles="manager-gui"/>
-->
The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!-- ... --> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
-->
</tomcat-users>
```

Para que los cambios tengan efecto debemos reiniciar el servidor:

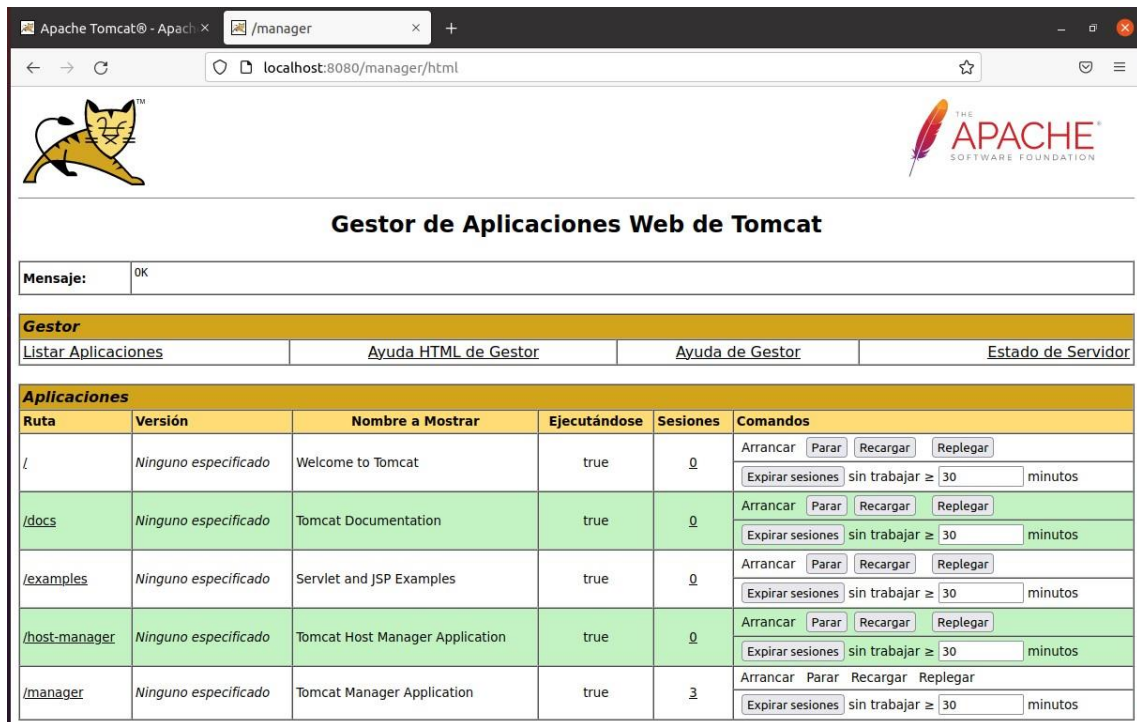
\$ *systemctl restart tomcat*

Ahora abrimos la página principal de Apache Tomcat en nuestro navegador e intentamos acceder al panel, bien al host-manager o al manager-app (este último es el que usaremos después).

Nos solicita el usuario y la contraseña que acabamos de crear:



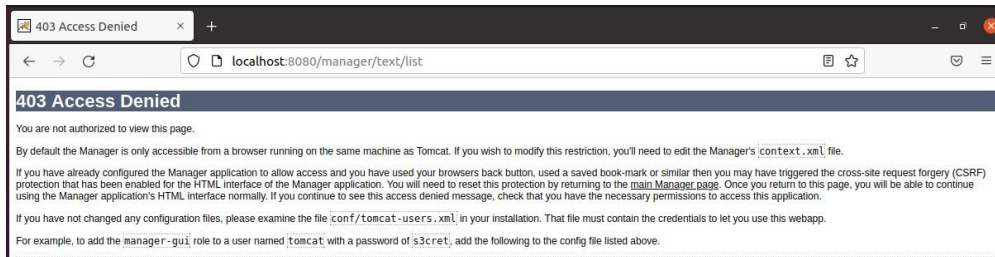
Y ya estamos dentro del panel:



The screenshot shows the Apache Tomcat Manager web interface in a browser. The title is "Gestor de Aplicaciones Web de Tomcat". Below the title is a message box that says "Mensaje: OK". There are four tabs: "Listar Aplicaciones", "Ayuda HTML de Gestor", "Ayuda de Gestor", and "Estado de Servidor". The "Listar Aplicaciones" tab is active, showing a table of applications.

Ruta	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 Expirar sesiones sin trabajar ≥ 30 minutos
/manager	Ninguno especificado	Tomcat Manager Application	true	3	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos

Sin embargo, si intentamos entrar en la interfaz de texto
(<http://localhost:8080/manager/html/list>) vemos que no nos lo permite:

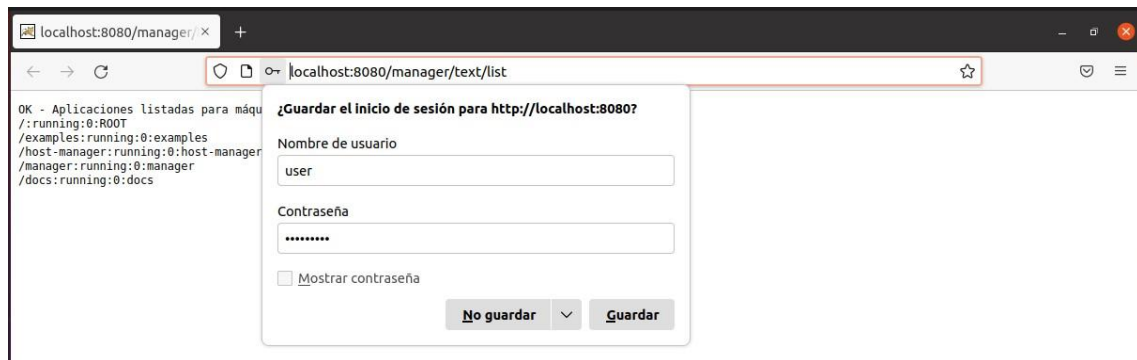


The screenshot shows a "403 Access Denied" error page. The text on the page says: "You are not authorized to view this page. By default the Manager is only accessible from a browser running on the same machine as Tomcat. If you wish to modify this restriction, you'll need to edit the Manager's context.xml file. If you have already configured the Manager application to allow access and you have used your browser's back button, used a saved book-mark or similar then you may have triggered the cross-site request forgery (CSRF) protection that has been enabled for the HTML interface of the Manager application. You will need to reset this protection by returning to the main Manager page. Once you return to this page, you will be able to continue using the Manager application's HTML interface normally. If you continue to see this access denied message, check that you have the necessary permissions to access this application. If you have not changed any configuration files, please examine the file conf/tomcat-users.xml in your installation. That file must contain the credentials to let you use this webapp. For example, to add the manager-gui role to a user named tomcat with a password of s3cret, add the following to the config file listed above."

Vayamos a la segunda parte en la que crearemos el usuario llamado user con el permiso para interfaz de texto. Para ello le añadiremos el rol de "manager-script":

```
<!--  
<user username="admin" password="<must-be-changed>" roles="manager-gui"/>  
<user username="robot" password="<must-be-changed>" roles="manager-script"/>  
-->  
<user username="bono" password="bono" roles="manager-gui"/>  
<user username="user" password="J.ikop,77" roles="manager-script"/>  
!--  
The sample user and role entries below are intended for use with the
```

Y al intentar acceder a la interface de texto, nos identificamos y con este nuevo usuario vemos que funciona:

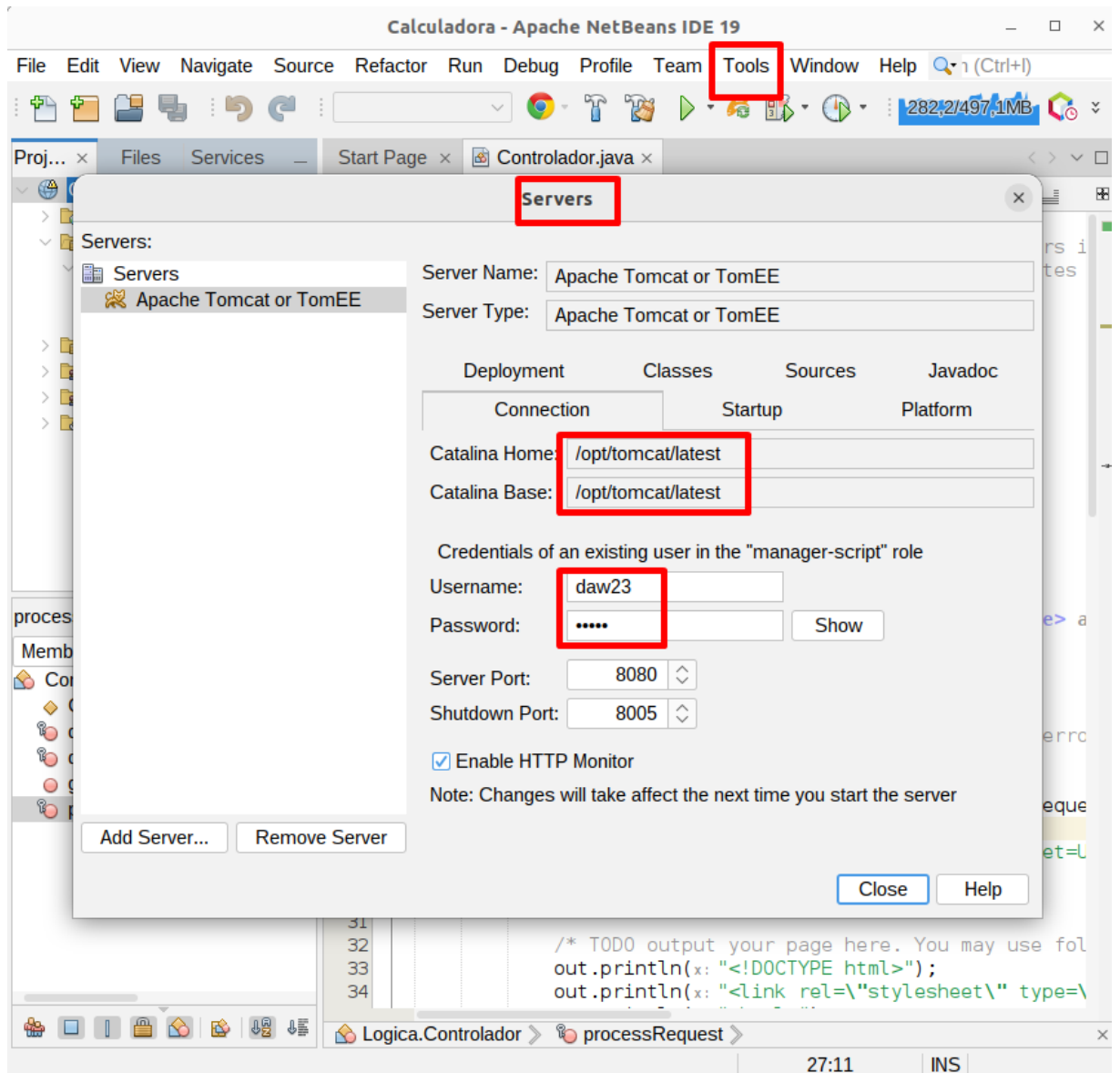


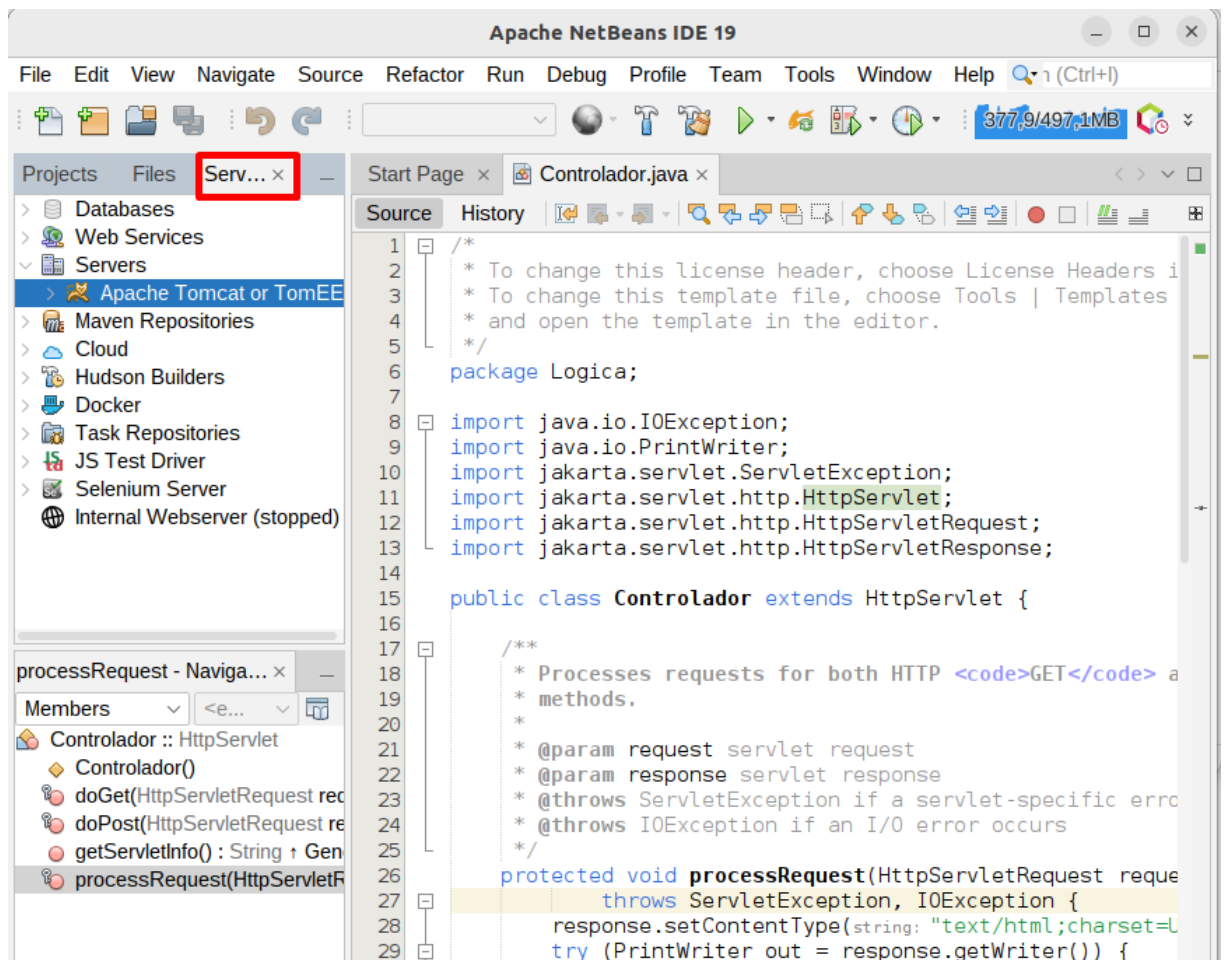
Ejercicio 2.- Desplegando en Apache Tomcat.

A partir de un programa de calculadora pruébalo NetBeans y si todo está correcto desplégalo en el servidor Tomcat que instalamos en la tarea online 1. Realiza los subapartado:

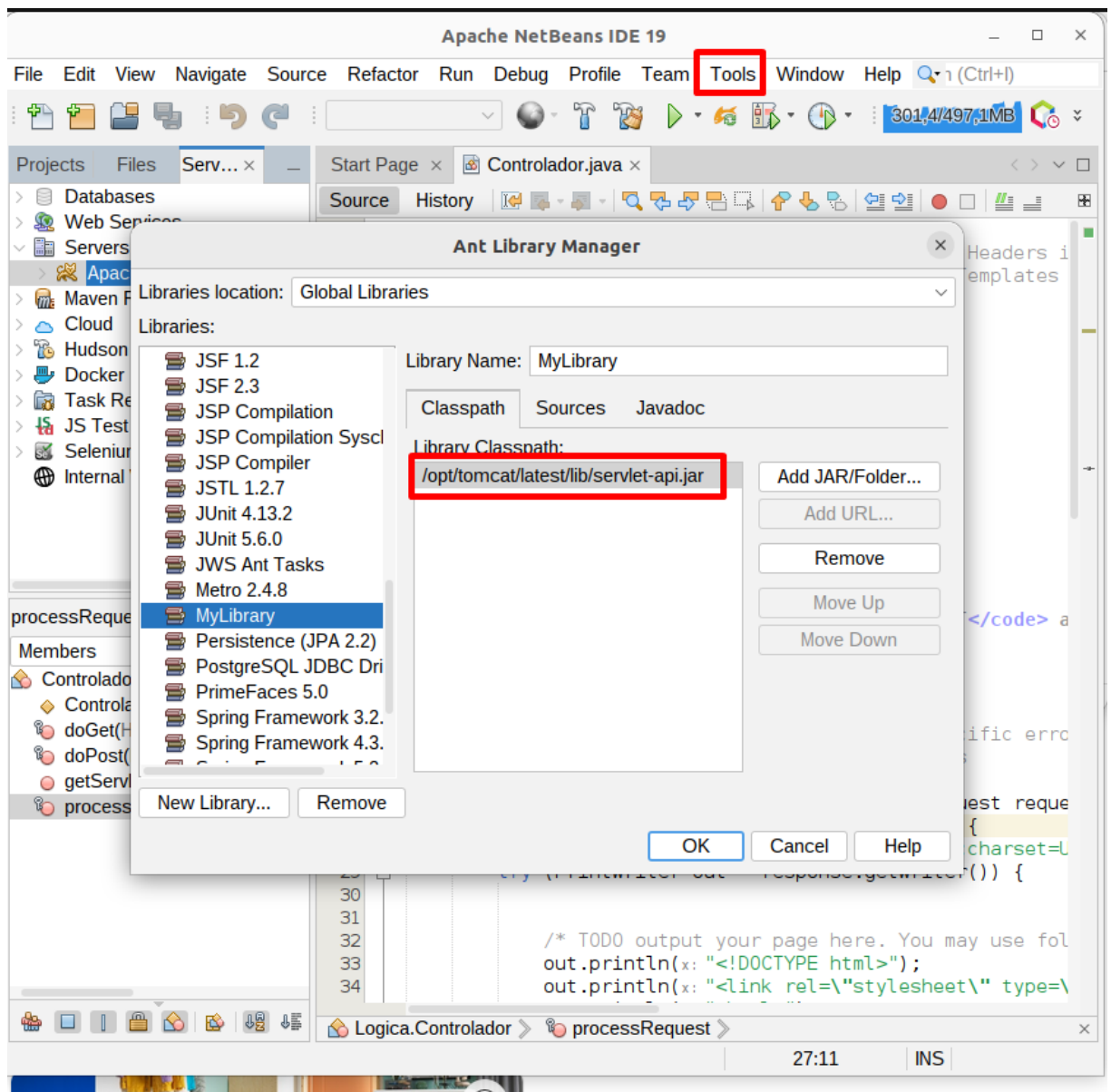
Lo primero que haremos será verificar la aplicación en Netbeans. Si no estuviese todo correcto nos daría error el intentar genera el archivo war en el proyecto.

1. Descargo el proyecto calculadora.zip en la carpeta Descargas y descomprimo la carpeta Calculadora.
2. Instalo Apache Netbeans (En mi caso utilicé Ubuntu Software)
3. Abro el proyecto calculadora y me da el error de que no hay ningún servidor configurado. Me voy a Services y añado el servidor Tomcat indicándole el path de donde lo tengo instalado, en mi caso /opt/tomcat/latest. No olvides darle acceso si no tiene permisos de acceso.

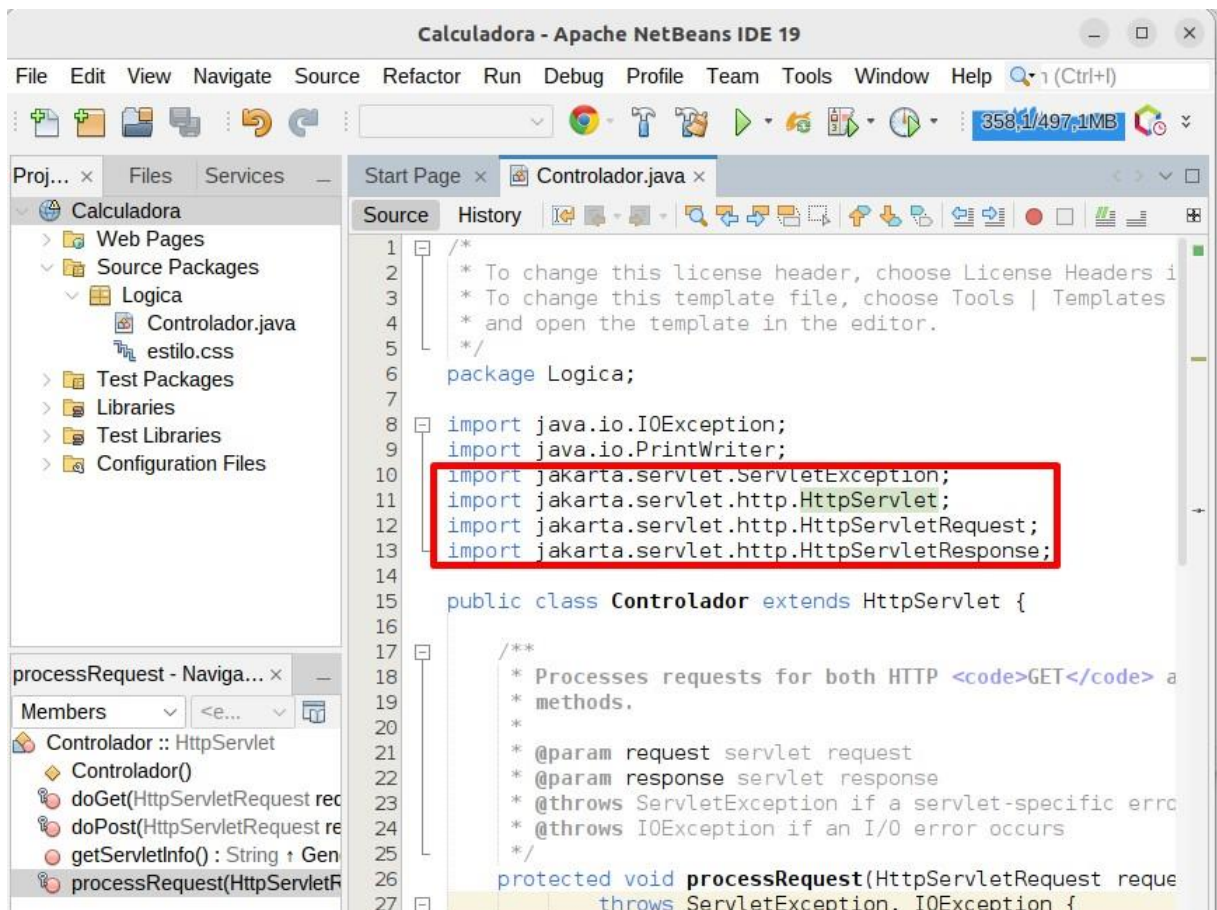




4. Ahora al proyecto debemos arreglarle las referencias de los servlet que estarán en tomcat. Para ello propiedades del proyecto, y en Libraries añadiremos la librería que falta: `/opt/tomcat/latest/lib/servlet-api.jar`



5. En tomcat10 en la librería han cambiado el nombre y he de cambiar javax por jakarta.



En este momento ya no hay errores. Es el momento de lanzar la aplicación y ver que todo funciona.

Llegado este momento, en la carpeta dist del proyecto Netbeans tenemos el fichero calculadora.war ya generado. En cualquier caso, como la tarea nos pide generarlo nosotros, debemos ejecutar:

2.1. Genera un archivo war con el contenido de la aplicación web.

```
$ jar cf calculadora.war -C ./Calculadora/web/ . (¡Ojo que sin el punto al final no funciona!)
```

2.2 Despliega el archivo war en el servidor Apache Tomcat

2.2.1 Copiando directamente la carpeta en la ruta adecuada:

Creamos una carpeta nueva que llamaremos calculadora.

```
$ mkdir /opt/tomcat/latest/webapps/calculadora
```

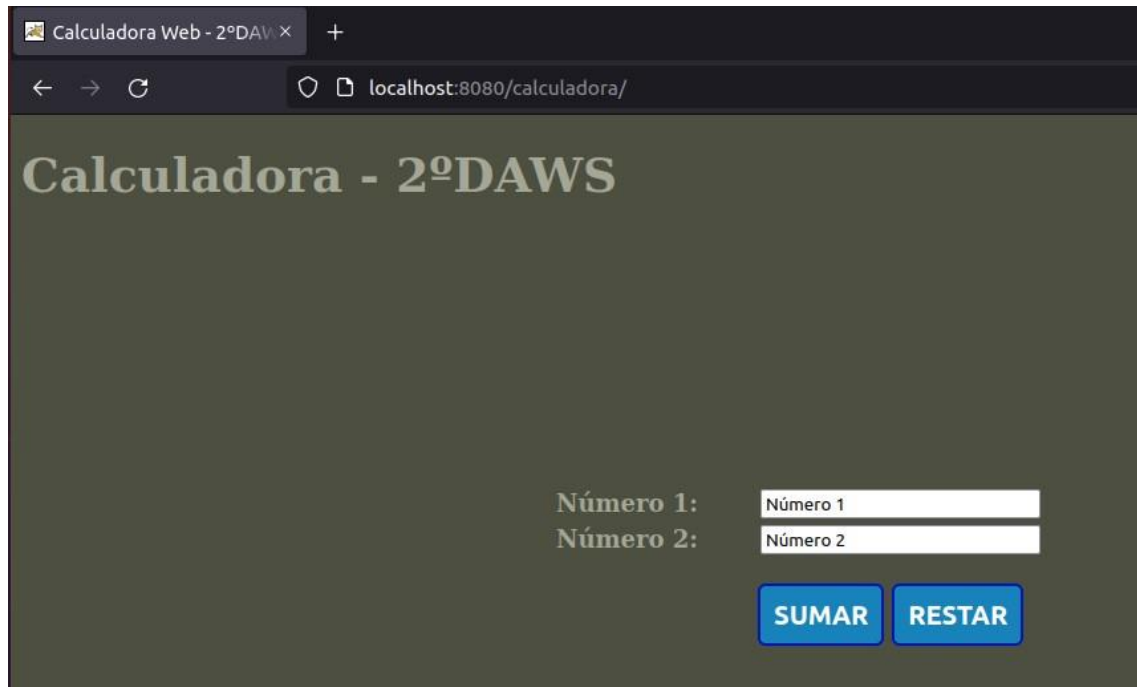
Copiaremos ahí la carpeta web del proyecto calculador con la estructura debida.

```
$ cp -r ./Calculadora/web/* /opt/tomcat/latest/webapps/calculadora/
```

Por último reiniciaremos el servicio de tomcat.

```
$ systemctl restart apache2.service
```

Para comprobar su funcionamiento abrimos un navegador y nos vamos a `http://localhost:8080/calculadora/`, ahí ha de estar funcionando la aplicación:

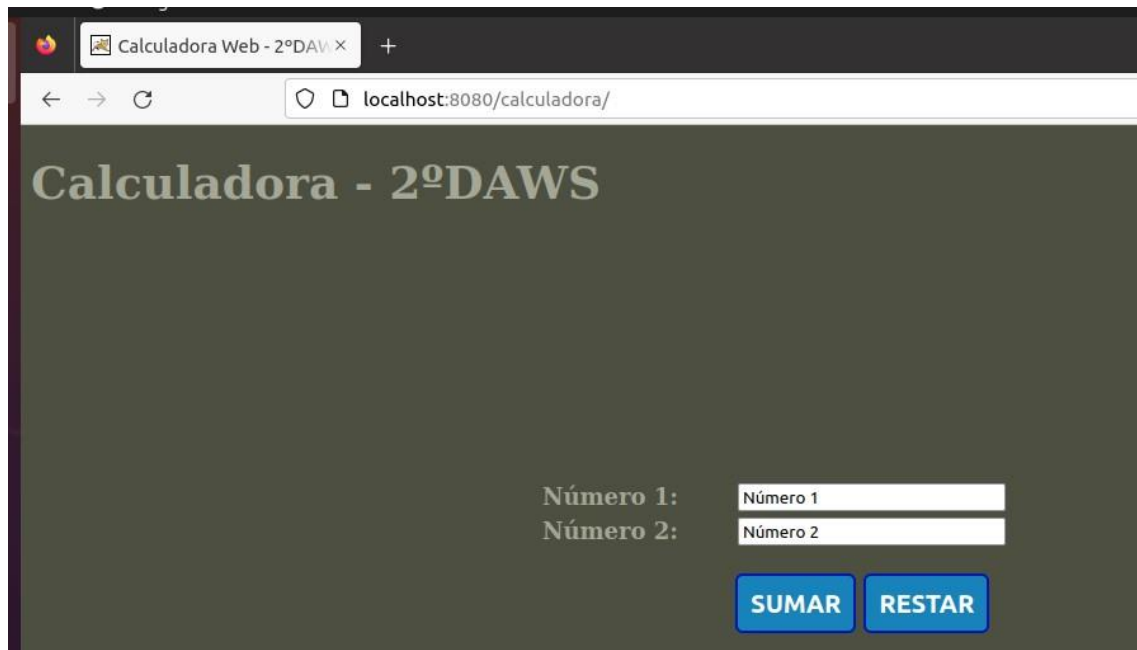


2.2.2 Desplegando el archivo war desde el entorno web de Apache (página web del servidor de aplicaciones):

Verlo a través de web: Nos logeamos con el usuario con rol manager-gui, y vamos al apartado “Archivo WAR a desplegar”, seleccionamos el archivo war generado y pinchamos en el botón Desplegar:



Se recarga la página, y seleccionamos la aplicación calculadora y se lanza:



Ejercicio 3. Wildfly.

Ya hemos visto cómo funciona Apache Tomcat y cómo se despliegan aplicaciones en él de forma manual y mediante el archivo WAR. En esta actividad vamos a realizar la instalación y configuración de Wildfly y a continuación desplegaremos una aplicación Web.

3.1.- Instalación de Wildfly.

Instala la aplicación Wildfly en el directorio **/opt** y cambia el puerto de escucha por defecto (el mismo que el de Tomcat, el 8080) por uno diferente, para que no entren en conflicto.

- Enlace de descarga de [Wildfly](https://github.com/wildfly/wildfly/releases/download/31.0.0.Beta1/wildfly-31.0.0.Beta1.tar.gz)

Para la instalación de Wildfly procedemos como en las anteriores, visitando la página oficial para localizar el enlace de descarga que necesitamos. Yo voy a descargar la versión 30.0.1: <https://github.com/wildfly/wildfly/releases/download/31.0.0.Beta1/wildfly-31.0.0.Beta1.tar.gz>



WildFly 31 Beta1 is now available

Que se corresponde con:

30.0.1.Final	FINAL	Dec 05, 2023
	WildFly Distribution	zip SHA-1
		tgz SHA-1
	WildFly Preview Distribution	zip SHA-1
		tgz SHA-1
	Application Server Source Code	zip SHA-1
		tgz SHA-1
	Quick Start Source Code	Source
	Release Notes	Notes

Descargo el tarball:

```
$ wget https://github.com/wildfly/wildfly/releases/download/30.0.1.Final/wildfly-30.0.1.Final.tar.gz
```

Lo descomprimo y lo movemos la carpeta que nos genera al directorio /opt con el nombre /wildfly30, que es más cómodo:

```
$ tar xvfz wildfly-30.0.1.Final.tar.gz
```

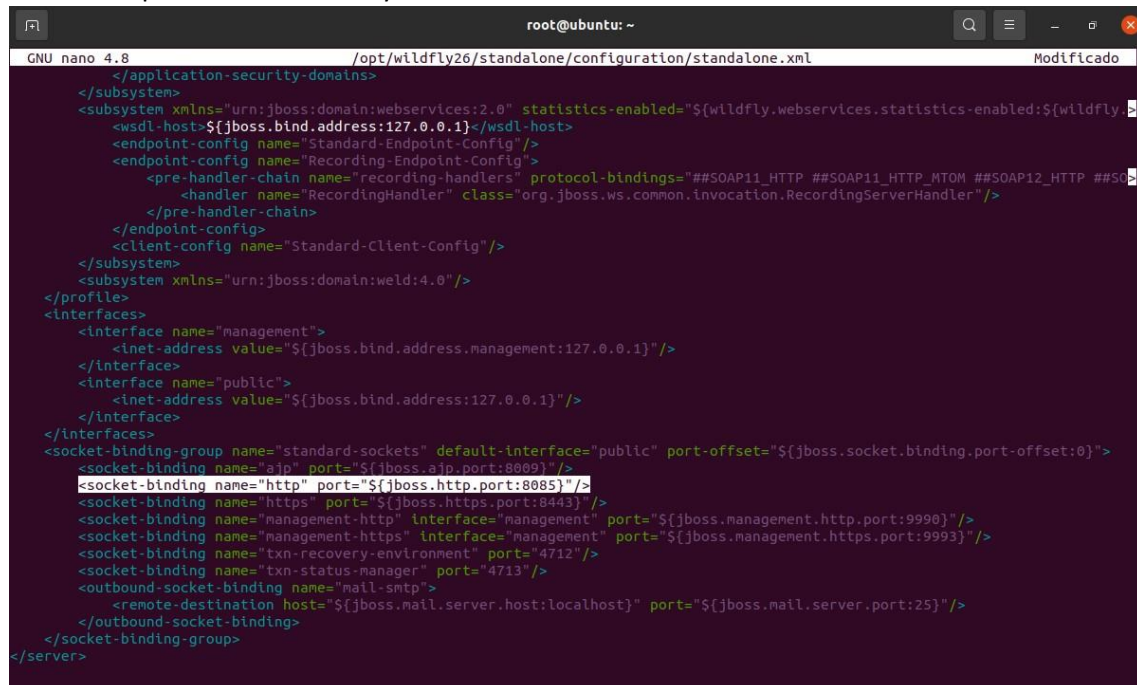
```
$ mv ./wildfly-30.0.1.Final /opt/wildfly36
```

Ahora vamos a configurar el puerto http de Wildfly, cambiándolo de 8080, que interfiere con Tomcat, por el 8085 (cada uno el que le corresponda). No es necesario tocar el offset, basta con cambiar el valor del puerto.

Editamos el fichero de configuración del modo standalone, que es el que nos interesa:

```
$ nano /opt/wildfly30/standalone/configuration/standalone.xml
```

Casi al final del fichero tenemos un bloque <socket-binding-group> y dentro de él la directiva <socket-binding name="http" port="\${jboss.http.port:8080}"/>, donde cambiaremos el 8080 por nuestro puerto. Guardamos y salimos.



```
GNU nano 4.8 /opt/wildfly26/standalone/configuration/standalone.xml Modificado
</application-security-domains>
</subsystem>
<subsystem xmlns="urn:jboss:domain:webservices:2.0" statistics-enabled="${wildfly.webservices.statistics-enabled:${wildfly.
<wsdl-host-${jboss.bind.address:127.0.0.1}</wsdl-host>
<endpoint-config name="Standard-Endpoint-Config"/>
<endpoint-config name="Recording-Endpoint-Config">
  <pre-handler-chain name="recording-handlers" protocol-bindings="##SOAP11_HTTP ##SOAP11_HTTP_MTOM ##SOAP12_HTTP ##St
  <handler name="RecordingHandler" class="org.jboss.ws.common.invocation.RecordingServerHandler"/>
</pre-handler-chain>
</endpoint-config>
<client-config name="Standard-Client-Config"/>
</subsystem>
<subsystem xmlns="urn:jboss:domain:weld:4.0"/>
</profile>
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
</interfaces>
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
  <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
  <socket-binding name="http" port="${jboss.http.port:8085}"/>
  <socket-binding name="https" port="${jboss.https.port:8443}"/>
  <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"/>
  <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9993}"/>
  <socket-binding name="txn-recovery-environment" port="4712"/>
  <socket-binding name="txn-status-manager" port="4713"/>
  <outbound-socket-binding name="mail-smtp">
    <remote-destination host="${jboss.mail.server.host:localhost}" port="${jboss.mail.server.port:25}"/>
  </outbound-socket-binding>
</socket-binding-group>
</server>
```

Ahora ya podemos ejecutar Wildfly. Lo ideal sería hacer como en Tomcat y crear un servicio que se ejecutase en segundo plano. Pero para no entretenernos más de la cuenta, basta con ejecutar el script que lanza la aplicación en modo standalone:

```
$ cd /opt/wildfly30/bin $ ./standalone.sh
```



```
root@ubuntu: /opt/wildfly26/bin
root@ubuntu: /opt/wildfly26/bin# ./standalone.sh
=====
JBoss Bootstrap Environment

JBOSS_HOME: /opt/wildfly26

JAVA: java

JAVA_OPTS: -server -Xms64m -Xmx512m -XX:MetaspaceSize=96m -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true --add-exports=java.desktop/sun.awt=ALL-UNNAMED --add-exports=java.naming/com.sun.jndi.ldap=ALL-UNNAMED --add-exports=java.naming/com.sun.jndi.url.ldap=ALL-UNNAMED --add-exports=java.naming/com.sun.jndi.url.ldap=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.security=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.management/javax.management=ALL-UNNAMED --add-opens=java.naming/javax.naming=ALL-UNNAMED
=====

21:48:54,081 INFO [org.jboss.modules] (main) JBoss Modules version 2.0.2.Final
21:48:54,669 INFO [org.jboss.msc] (main) JBoss MSC version 1.4.13.Final
21:48:54,675 INFO [org.jboss.threads] (main) JBoss Threads version 2.4.0.Final
21:48:54,765 INFO [org.jboss.as] (MSC service thread 1-2) WFLYSRV0049: WildFly Full 26.1.3.Final (WildFly Core 18.1.2.Final) starting
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.wildfly.extension.elytron.SSLDefinitions (jar:file:/opt/wildfly26/modules/system/layers/base/org.wildfly.extension.elytron/main/wildfly-elytron-integration-18.1.2.Final.jar!) to method com.sun.net.ssl.internal.ssl.Provider.isFIPS()
=====
```

Ya podemos acceder a la página principal de Wildfly en el puerto configurado:



Actividad 3.2.- Acceso a la consola de Wildfly

Una vez instalado Wildfly habilita el acceso a la consola de administración. Crea un usuario para poder acceder vía web al panel de administración de aplicaciones, con el script `add-user.sh`. ☐ Documentación de [Wildfly 10](#) (añadir usuario)

Para añadir un usuario que nos permita gestionar la consola de Wildfly ejecutamos el script adduser.sh que se encuentra en la carpeta /opt/wildfly30/bin .

```
$ cd /opt/wildfly26/bin
```

```
$ ./add-user.sh
```

```
root@ubuntu: /opt/wildfly26/bin
root@ubuntu: /opt/wildfly26/bin# ./add-user.sh

What type of user do you wish to add?
  a) Management User (mgmt-users.properties)
  b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : bono
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
- The password should be different from the username
- The password should not be one of the following restricted values {root, admin, administrator}
- The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[ ]:
About to add user 'bono' for realm 'ManagementRealm'
Is this correct yes/no? y
Added user 'bono' to file '/opt/wildfly26/standalone/configuration/mgmt-users.properties'
Added user 'bono' to file '/opt/wildfly26/domain/configuration/mgmt-users.properties'
Added user 'bono' with groups to file '/opt/wildfly26/standalone/configuration/mgmt-groups.properties'
Added user 'bono' with groups to file '/opt/wildfly26/domain/configuration/mgmt-groups.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server Jakarta Enterprise Beans
calls.
yes/no? n
root@ubuntu: /opt/wildfly26/bin#
```

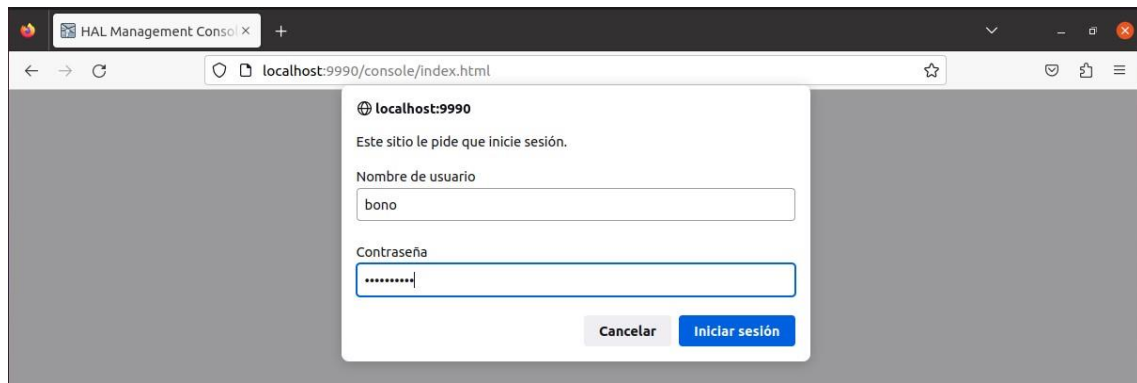
Arrancamos de nuevo Wildfly y accedemos al panel con el nuevo usuario:

```
$ ./standalone.sh
```

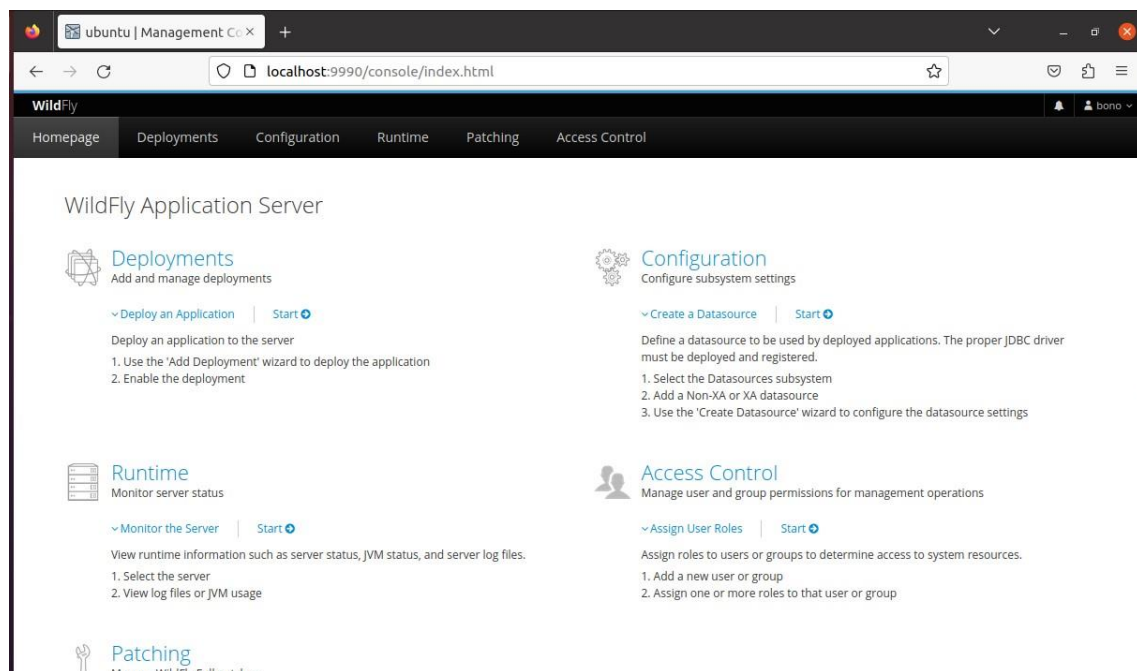
Abrimos nuestro navegador y vamos a la página principal de Wildfly:



Ahora pinchamos en el enlace “Administration Console”. Nos aparecerá una ventana emergente pidiéndonos el usuario y la contraseña que acabamos de crear:



Una vez validada, nos mostrará la consola, desde la que podemos manejar Wildfly.

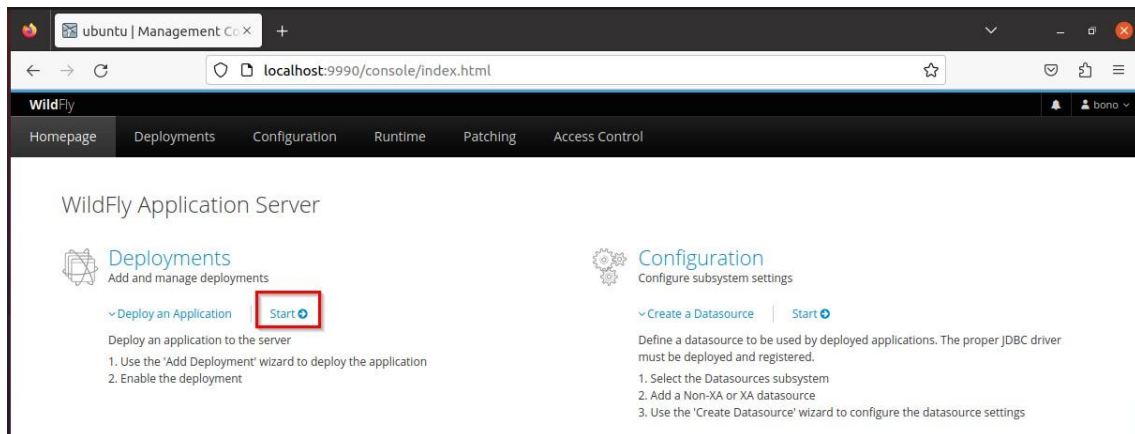


3.3.- Desplegar un WAR en Wildfly.

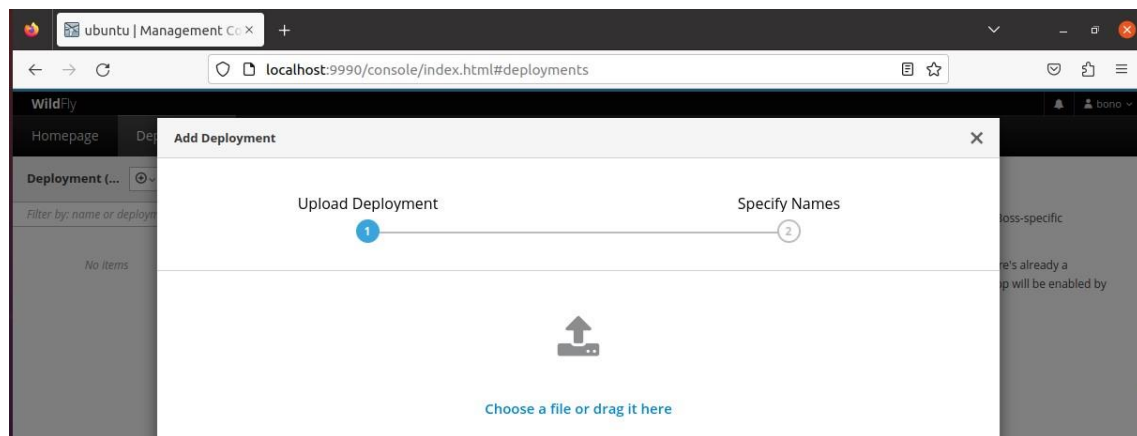
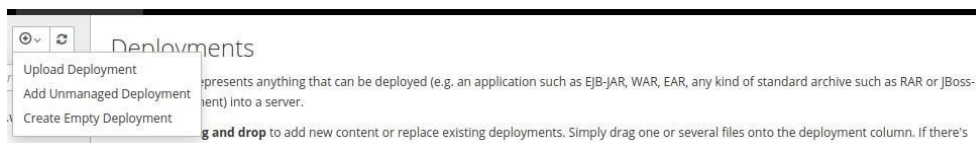
Accede al panel de administración y despliega en el servidor el war generado en el ejercicio 2.

Ahora que tenemos acceso a la consola de administración, desplegaremos la aplicación web con el fichero WAR que generamos en el apartado 2.1.

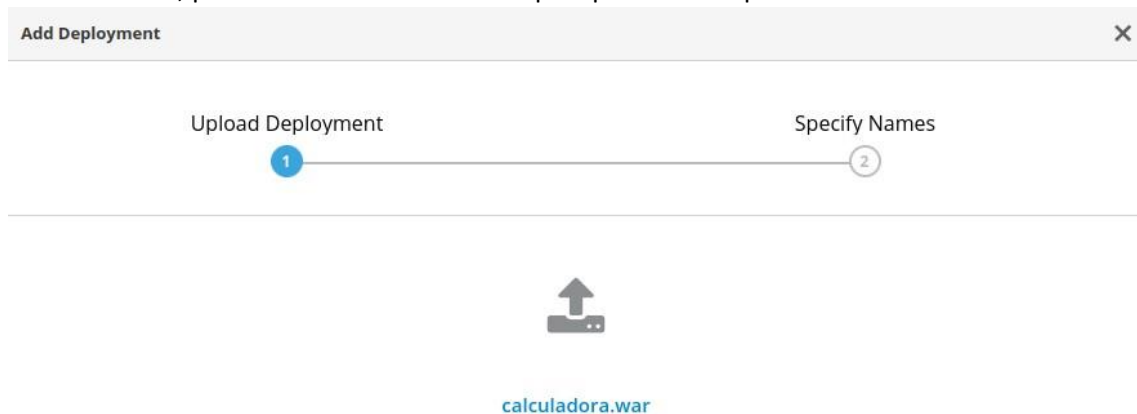
Partiendo de la actividad anterior, logeados ya en el panel de administración, vamos al bloque de opciones “Deployments” y seleccionamos “Deploy an Application >> Start”.



En la pantalla siguiente pinchamos en el símbolo (+) que hay en la parte superior izquierda y en el desplegable que nos muestra seleccionamos “Upload Deployment”



Seleccionamos el fichero que queremos cargar en Wildfly y clicamos en el botón verde “Abrir”. A continuación, pinchamos en el botón azul que aparece en la parte inferior derecha “Next”



Una vez cargado nos pedirá que especifiquemos un nombre para nuestra aplicación. Podemos dejarlo como está y clicar en el botón “Finish”

Add Deployment

X

Upload Deployment

Specify Names

1

2

Help

Name

calculadora

Runtime Name

calculadora

Enabled

ON

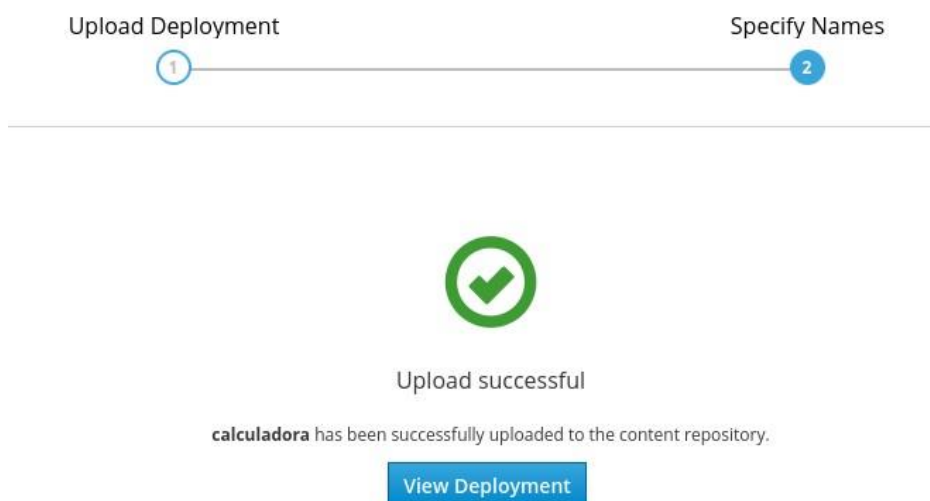
Required fields are marked with *

Cancel

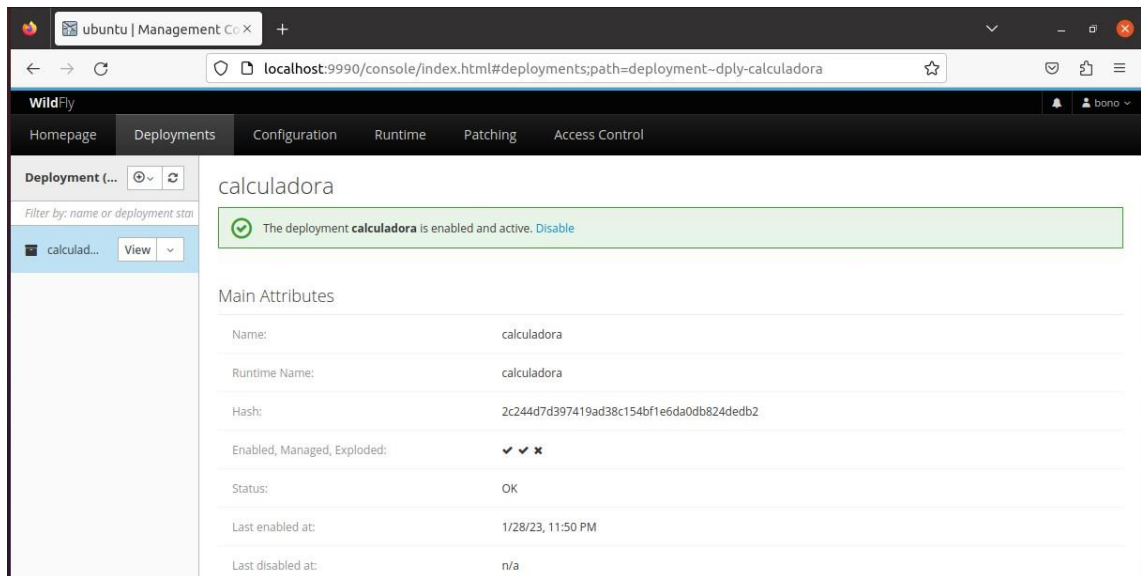
< Back

Finish

Si todo ha ido bien en el despliegue nos mostrará un mensaje de éxito “Upload successful”:



Podemos clicar en “Close” para terminar o en “View Deployment” para ver la ficha de despliegue de nuestra app. Elegimos esta última opción:



Ejercicio 4.- Apache ANT.

Ahora que ya has desplegado en Tomcat tu aplicación de forma manual, vamos a hacerlo generando un fichero de despliegue tipo WAR con la herramienta Apache ANT. Para ello deberás realizar las siguientes tareas:

Actividad 4.1.- Instalación de Apache Ant.

Instala la última versión de Apache ANT disponible en la web de Apache. Es recomendable que realices la instalación en el directorio `/opt`. Deberás configurar las variables de entorno 'ANT_HOME' y PATH (ANT_HOME apuntará al directorio de instalación de Ant; PATH permitirá ejecutar comandos dentro del directorio bin de la instalación de Ant). En caso de que necesites información adicional sobre la instalación de Ant puedes consultar el siguiente enlace:

- [Instalación de Apache Ant.](#)

En primer lugar, tenemos que descargar la aplicación de la página oficial de Apache, usando el mismo método que usábamos para descargar Tomcat, localizando el enlace de descarga y usándolo con el comando `wget`.

Current Release of Ant

The Apache Ant team currently maintains two lines of development. The 1.9.x releases require Java5 at runtime and 1.10.x requires Java8 at runtime. Both lines are based off of Ant 1.9.7 and the 1.9.x releases are mostly bug fix releases while additional new features are developed for 1.10.x. We recommend using 1.10.x unless you are required to use versions of Java prior to Java8 during the build process.

Currently, Apache Ant 1.9.16 and 1.10.13 are the best available versions, see the [release notes](#).

Note

Ant 1.10.13 has been released on 10th January 2023 and may not be available on all mirrors for a few days.

Tar files may require gnu tar to extract

Tar files in the distribution contain long file names, and may require gnu tar to do the extraction.

1.9.16 release - requires minimum of Java 5 at runtime

- 1.9.16 .zip archive: [apache-ant-1.9.16-bin.zip](#) [PGP] [SHA512]
- 1.9.16 .tar.gz archive: [apache-ant-1.9.16-bin.tar.gz](#) [PGP] [SHA512]
- 1.9.16 .tar.bz2 archive: [apache-ant-1.9.16-bin.tar.bz2](#) [PGP] [SHA512]

1.10.13 release - requires minimum of Java 8 at runtime

- 1.10.13 .zip archive: [apache-ant-1.10.13-bin.zip](#) [PGP] [SHA512]
- 1.10.13 .tar.gz archive: [apache-ant-1.10.13-bin.tar.gz](#) [PGP] [SHA512]
- 1.10.13 .tar.bz2 archive: [apache-ant-1.10.13-bin.tar.bz2](#) [PGP] [SHA512]
- 1.10.13 .tar.xz archive: [apache-ant-1.10.13-bin.tar.xz](#) [PGP] [SHA512]

Yo he seleccionado la versión 1.9.16 cuyo enlace de descarga es:

<https://dlcdn.apache.org//ant/binaries/apache-ant-1.9.16-bin.tar.gz>

```
root@ubuntu: /tmp
root@ubuntu:/tmp# wget https://dlcdn.apache.org//ant/binaries/apache-ant-1.9.16-bin.tar.gz
--2022-02-04 10:24:07-- https://dlcdn.apache.org//ant/binaries/apache-ant-1.9.16-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5839799 (5.6M) [application/x-gzip]
Saving to: 'apache-ant-1.9.16-bin.tar.gz'
```

Ahora descomprimo el tarball.

\$ tar -zxvf apache-ant-1.9.16-bin.tar.gz

```
root@ubuntu: /tmp
root@ubuntu:/tmp# tar -zxvf apache-ant-1.9.16-bin.tar.gz
```

```
root@ubuntu: /tmp# ls
apache-ant-1.9.16
apache-ant-1.9.16-bin.tar.gz
```

Muevo los archivos al directorio /opt :

\$ mv ./apache-ant-1.9.16 /opt/

```
root@ubuntu: /opt# ls -l
total 12
drwxr-xr-x  6 root  root  4096 Jul 10  2021 apache-ant-1.9.16
drwxr-xr-x  3 tomcat tomcat 4096 Dec 19 11:20 tomcat
```

El siguiente paso es exportar las variables de entorno de ANT, creando un script Shell en el directorio /etc/profile.d. Yo lo he llamado ant.sh pero lo podéis llamar como queráis. Las

variables de entorno que exportamos son ANT_HOME y PATH, las de JAVA no son imprescindibles para esta tarea. Las rutas variarán en función de cómo y dónde tengáis ANT y JAVA

\$ nano /etc/profile.d/ant.sh

```
root@ubuntu: /etc/profile.d
GNU nano 4.8 ant.sh
export ANT_HOME=/opt/apache-ant-1.9.16
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH:$ANT_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
```

Ahora ejecutamos el script y después invocamos la aplicación para ver si todo ha ido bien.

\$ source /etc/profile.d/ant.sh \$ ant -v

```
root@ubuntu: /etc/profile.d
root@ubuntu:/etc/profile.d# nano ant.sh
root@ubuntu:/etc/profile.d# source /etc/profile.d/ant.sh
root@ubuntu:/etc/profile.d# ant
Buildfile: build.xml does not exist!
Build failed
root@ubuntu:/etc/profile.d# ant -v
Apache Ant(TM) version 1.9.16 compiled on July 10 2021
Trying the default build file: build.xml
Buildfile: build.xml does not exist!
Build failed
root@ubuntu:/etc/profile.d#
```

Actividad 4.2.- Elabora un fichero buildmain.xml que realice las siguientes acciones.

Situándonos dentro de la carpeta del proyecto Calculadora, crearemos un fichero build de Ant vacío, y lo llamaremos buildmain.xml para no interferir con build.xml del proyecto Calculadora.

\$ touch buildmain.xml

Que realice las siguientes acciones:

- Cree un directorio llamado "compilado"
- Debe compilar todos los archivos java del directorio "origen" y almacenarlos en el directorio "compilado".
- Cree un directorio en "distribucion/lib"
- En este directorio se debe crear un fichero "jar" de forma que contenga todas las clases compiladas del directorio "compilado".
- El archivo jar se nombrará "daw22-23_nombre_apellidos_yyyyMMdd.jar", donde yyyyMMdd corresponde a la fecha del sistema.
- Borre posteriormente el directorio "compilado" y todo su contenido.

Veamos como traducir lo que se pide usando la herramienta ANT por partes y finalmente lo uniremos y probaremos:

Cree un directorio llamado "compilado" y otro "distribucion/lib"

```
<target name="init">
    <mkdir dir="compilado" />
    <mkdir dir="distribucion/lib" />
</target>
```

Debe compilar todos los archivos java del directorio "origen" y almacenarlos en el directorio "compilado".

Primero indicaremos donde está la librería externa que necesitamos para la compilación, para posteriormente compilar el proyecto y guardarlo en la carpeta "compilado":

```
<path id="dependency.path">
    <fileset dir="/opt/toncat/latest/lib" includes="**/*.jar" /> <!-- Ruta donde agarrar la librería externa -->
</path>

<target name="compile" depends="init"> <!-- Compilaremos el proyecto indicando ruta de librería externa -->
    <javac srcdir="${basedir}" destdir="compilado" classpathref="dependency.path" />
</target>
```

Creación del fichero "jar" nombrado "daw22-23_nombre_apellidos_yyyyMMdd.jar", donde yyyyMMdd corresponde a la fecha del sistema.

```
<target name="build" depends="compile"> <!-- Aqui montaremos el archivo jar. -->
    <tstamp>
        <format property="CURRENT_DATE"
            pattern="yyyyMMdd"
            locale="es,ES" />
    </tstamp>
    <jar destfile="distribucion/lib/daw22-23_Jesus_Bono_${CURRENT_DATE}.jar"
        basedir="${basedir}/compilado" />
</target>
```

Borre posteriormente el directorio "compilado" y todo su contenido.

```
<target name="clean" depends="build">
    <delete dir="compilado"/>
</target>
```

Archivo buildmain finalizado:


```
root@ubuntu: /home/despliegue/Descargas/Calculadora
GNU nano 4.8 buildmain.xml
<project name="Tarea04" default="run" basedir=".">

    <target name="init">
        <mkdir dir="compilado" />
        <mkdir dir="distribucion/lib" />
    </target>

    <path id="dependency.path">
        <fileset dir="/opt/tomcat/latest/lib" includes="**/*.jar" /> <!-- Ruta donde agarrar la libreria externa -->
    </path>

    <target name="compile" depends="init"> <!-- Compilaremos el proyecto indicando ruta de libreria externa -->
        <javac srcdir="${basedir}" destdir="compilado" classpathref="dependency.path" />
    </target>

    <target name="build" depends="compile"> <!-- Aqui montaremos el archivo jar. -->
        <tstamp>
            <format property="CURRENT_DATE"
                pattern="yyyyMMdd"
                locale="es,ES"/>
        </tstamp>
        <jar destfile="distribucion/lib/daw22-23_Jesus_Bono_${CURRENT_DATE}.jar"
            basedir="${basedir}/compilado"/>
    </target>

    <target name="clean" depends="build">
        <delete dir="compilado"/>
    </target>

    <target name="run" depends="clean"/>
</project>
```

Por defecto comenzará buscando el target llamado “run”, y a través de las dependencias (depends) irá creando un árbol de tareas en las que comenzará a ejecutando la tarea “init”, pasando por todas hasta finalizar.

Comprobando que funciona.

Vamos a lanzar la herramienta ant, usando la opción `-file` para definirle el nombre del archivo que debe ejecutar, pues si no por defecto buscaría `build.xml`, que en este caso pertenece al proyecto Calculadora, proyecto que usamos de ruta base para crearlo todo. **\$ ant -file buildmain.xml**

```
root@ubuntu: /home/despliegue/Descargas/Calculadora
root@ubuntu: /home/despliegue/Descargas/Calculadora# ant -file buildmain.xml
Buildfile: /home/despliegue/Descargas/Calculadora/buildmain.xml

init:
[mkdir] Created dir: /home/despliegue/Descargas/Calculadora/compilado

compile:
[javac] /home/despliegue/Descargas/Calculadora/buildmain.xml:13: warning: 'includeantruntime' was not set, defaulting to build.s
ysclasspath=last; set to false for repeatable builds
[javac] Compiling 1 source file to /home/despliegue/Descargas/Calculadora/compilado

build:
[jar] Building jar: /home/despliegue/Descargas/Calculadora/distribucion/lib/daw22-23_Jesus_Bono_20230129.jar

clean:
[delete] Deleting directory /home/despliegue/Descargas/Calculadora/compilado

run:

BUILD SUCCESSFUL
Total time: 1 second
root@ubuntu: /home/despliegue/Descargas/Calculadora#
```

Y verificamos que se ha generado el fichero .jar correctamente:



root@ubuntu: /home/despliegue/Descargas/Calculadora/distribucion/lib

```
root@ubuntu: /home/despliegue/Descargas/Calculadora/distribucion/lib# ls  
daw22-23_Jesus_Bono_20230129.jar
```