APUNTES PARA EXAMEN UD3

Contenido

[1. Instalación Tomcat 2](#_Toc126697984)

[1.1. Windows 2](#_Toc126697985)

[1.2. Debian 3](#_Toc126697986)

[2. Desplegar app nun host 5](#_Toc126697987)

[2.1. Windows 5](#_Toc126697988)

[2.1.1. Autenticación Basic 5](#_Toc126697989)

[2.1.2. Listar archivos 6](#_Toc126697990)

[2.1.3. Creación de log específico 7](#_Toc126697991)

[2.1.4. Definir Host 9](#_Toc126697992)

[2.1.5. Cambio de puerto 10](#_Toc126697993)

[2.1.6. Funcionamiento 11](#_Toc126697994)

[2.1.7. Configuración SSL 13](#_Toc126697995)

[2.2. Debian 14](#_Toc126697996)

[2.2.1. Creación de un certificado 14](#_Toc126697997)

[2.2.2. Completo sin explicación en Debian (Solo capturas) 14](#_Toc126697998)

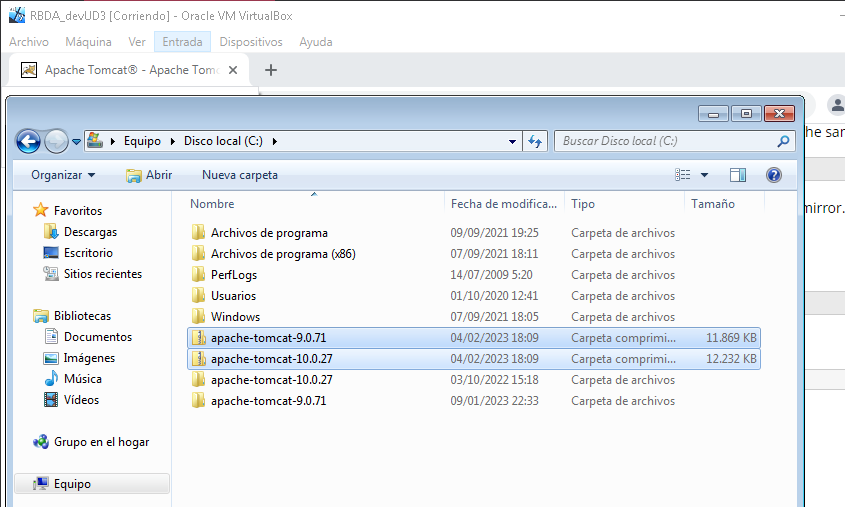
[3. Datasource 18](#_Toc126697999)

# Instalación Tomcat

## Windows

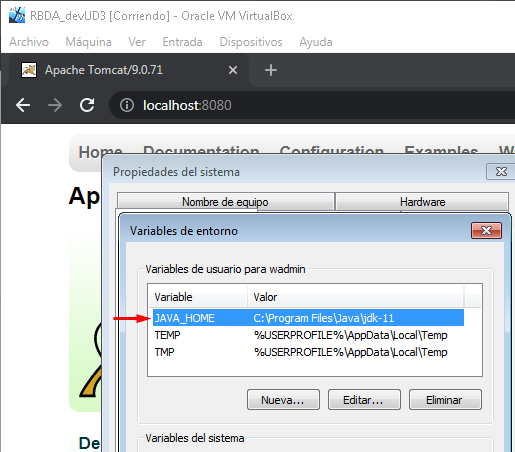
Descargarmos os arquivos comprimidos respectivos de tomcat 10 e 9.

Unha vez descargados ubicamos os arquivos comprimidos na ubicación desexada e descomprimimolos.



Instale Java™ Runtime Environment (JRE) o Java developer kit (JDK).

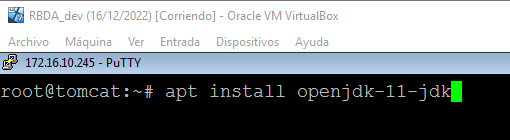
Establezca una variable de entorno con la vía de acceso al JRE o JDK.



## Debian

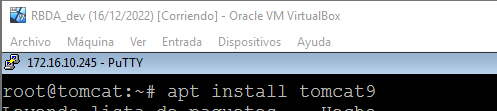
Primeiro instalar java.

* apt install openjdk-11-jdk

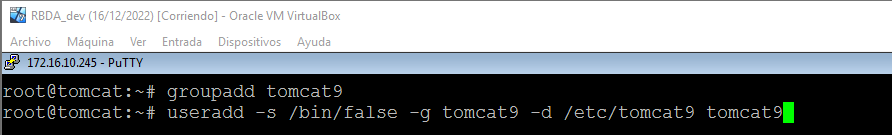


Instalar tomcat9.

* apt install tomcat9



Añadir un usuario y grupo de trabajo para tomcat

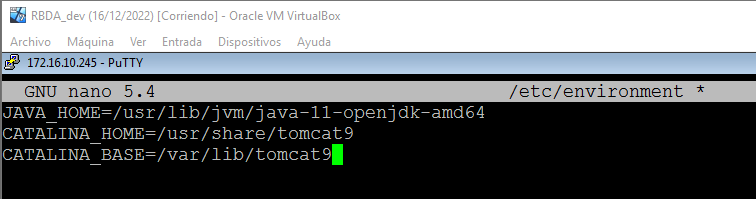


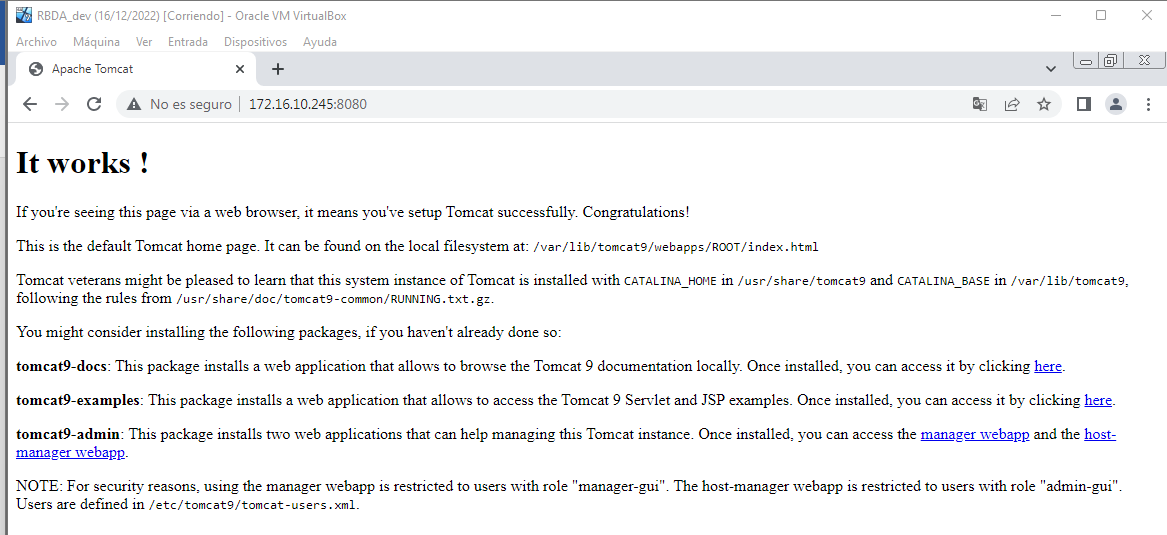
Crear variables de entorno:

JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

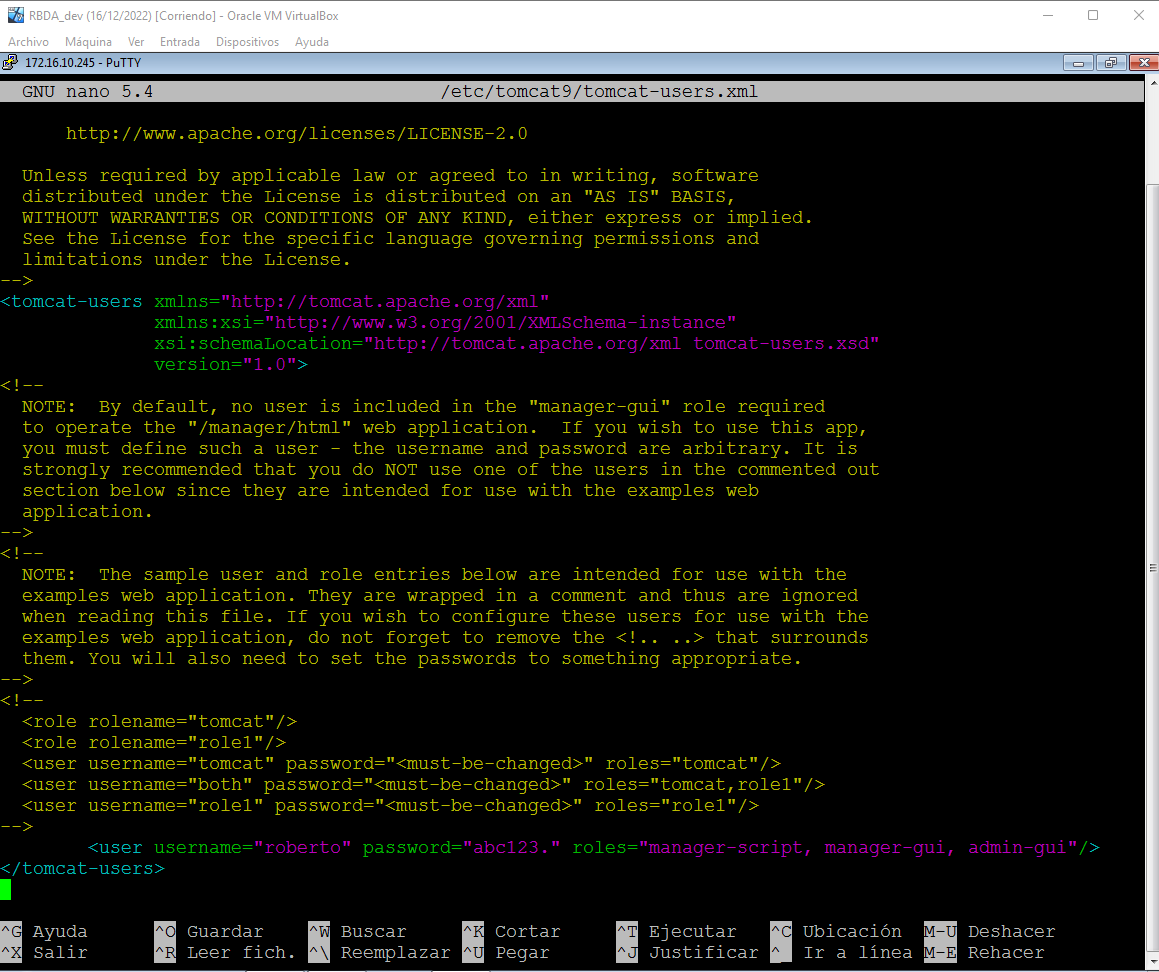
CATALINA\_HOME=/usr/share/tomcat9

CATALINA\_BASE=/var/lib/tomcat9





Creamos usuario en /etc/tomcat9/tomcat-users.xml

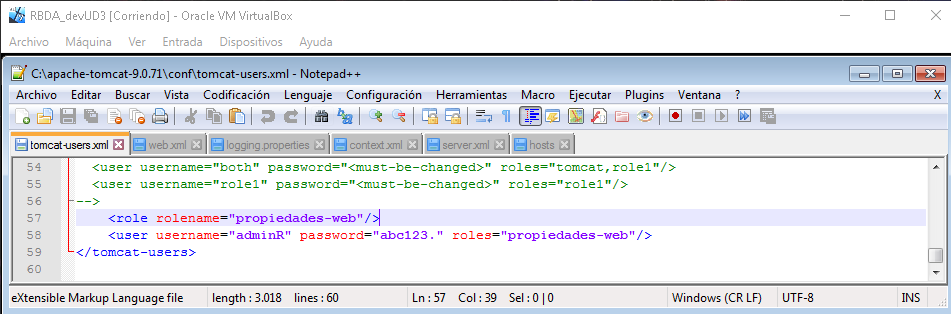


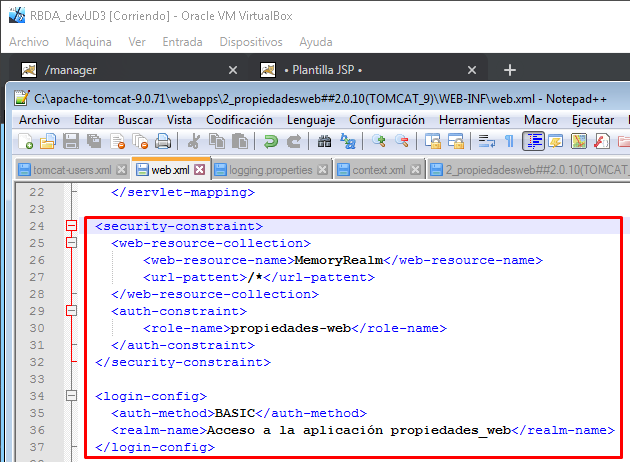
# Desplegar app nun host

## Windows

### Autenticación Basic

Añadimos el usuario para acceder a la aplicación web.





**<security-constraint>**

**<web-resource-collection>**

**<web-resource-name>MemoryRealm</web-resource-name>**

**<url-pattern>/\*</url-pattern>**

**</web-resource-collection>**

**<auth-constraint>**

**<role-name>propiedades-web</role-name>**

**</auth-constraint>**

**</security-constraint>**

**<login-config>**

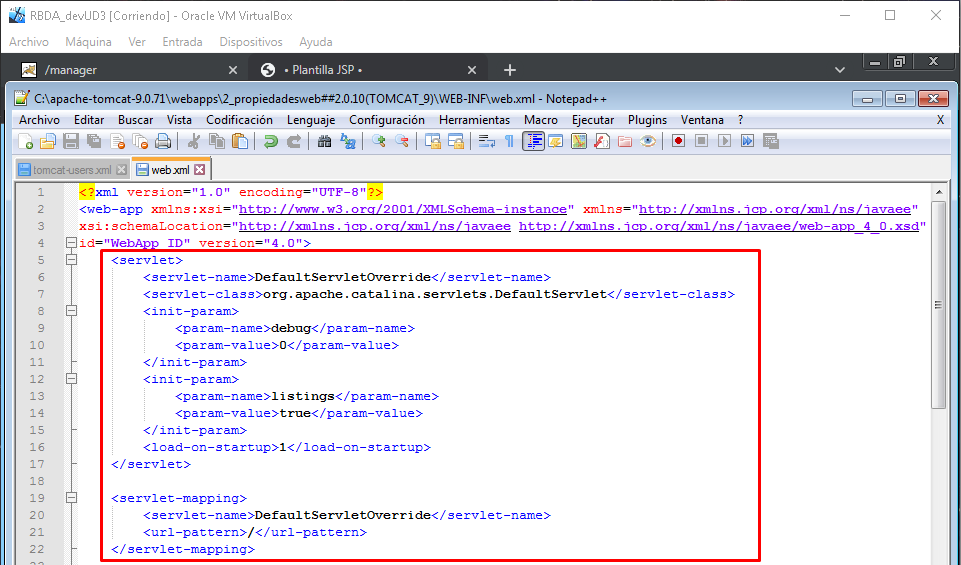
**<auth-method>BASIC</auth-method>**

**<realm-name>Acceso a la aplicación propiedades\_web</realm-name>**

**</login-config>**

### Listar archivos

Para listar archivos:



**<servlet>**

**<servlet-name>DefaultServletOverride</servlet-name>**

**<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>**

**<init-param>**

**<param-name>debug</param-name>**

**<param-value>0</param-value>**

**</init-param>**

**<init-param>**

**<param-name>listings</param-name>**

**<param-value>true</param-value>**

**</init-param>**

**<load-on-startup>1</load-on-startup>**

**</servlet>**

**<servlet-mapping>**

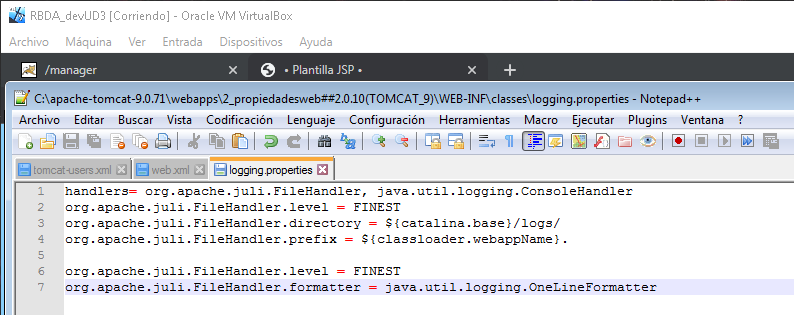
**<servlet-name>DefaultServletOverride</servlet-name>**

**<url-pattern>/</url-pattern>**

**</servlet-mapping>**

### Creación de log específico

Creamos el archivo logging.properties en **C:\apache-tomcat-9.0.71\webapps\2\_propiedadesweb##2.0.10(TOMCAT\_9)\WEB-INF\classes**. Con esto se creará el archivo .log.



**handlers= org.apache.juli.FileHandler, java.util.logging.ConsoleHandler**

**org.apache.juli.FileHandler.level = FINEST**

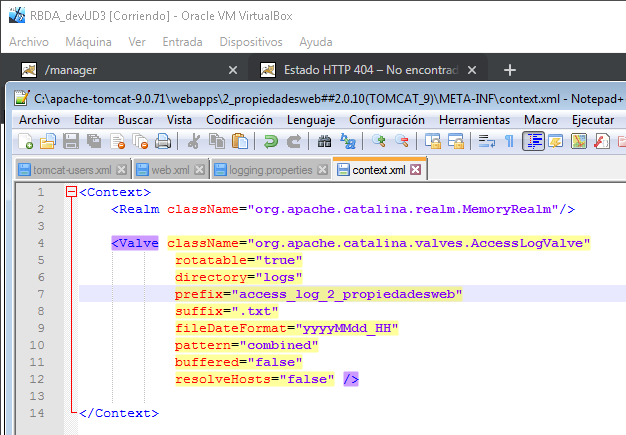
**org.apache.juli.FileHandler.directory = ${catalina.base}/logs/**

**org.apache.juli.FileHandler.prefix = ${classloader.webappName}.**

**org.apache.juli.FileHandler.level = FINEST**

**org.apache.juli.FileHandler.formatter = java.util.logging.OneLineFormatter**

Creamos el archivo context.xml en **C:\apache-tomcat-9.0.71\webapps\2\_propiedadesweb##2.0.10(TOMCAT\_9)\META-INF**. Con esto se creará el archivo acces\_log.txt.



**<Context>**

**<Realm className="org.apache.catalina.realm.MemoryRealm"/>**

**<Valve className="org.apache.catalina.valves.AccessLogValve"**

**rotatable="true"**

**directory="logs"**

**prefix="access\_log\_2\_propiedadesweb"**

**suffix=".txt"**

**fileDateFormat="yyyyMMdd\_HH"**

**pattern="combined"**

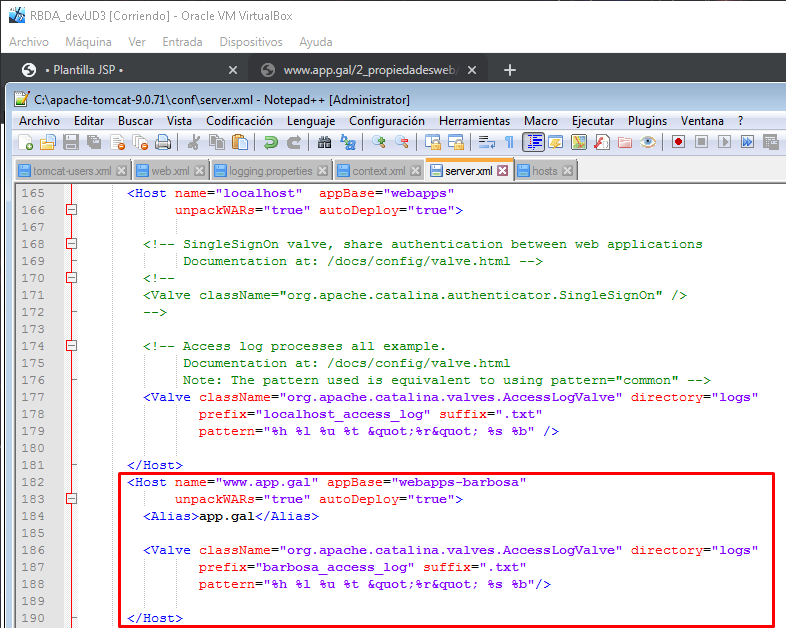
**buffered="false"**

**resolveHosts="false" />**

**</Context>**

### Definir Host

Definimos un host para a app e crearemos a carpeta que a conterá, neste caso webapps-barbosa.



**<Host name="www.app.gal" appBase="webapps-barbosa"**

**unpackWARs="true" autoDeploy="true">**

**<Alias>app.gal</Alias>**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

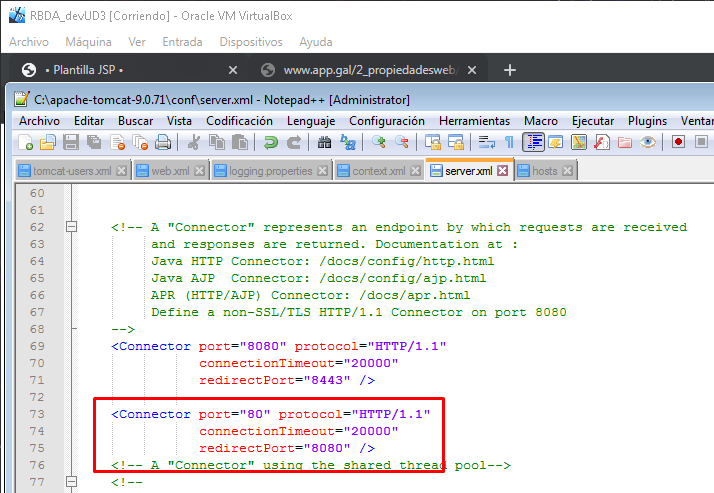
**prefix="barbosa\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b"/>**

**</Host>**

### Cambio de puerto

Para que pueda servir en el puerto 80.



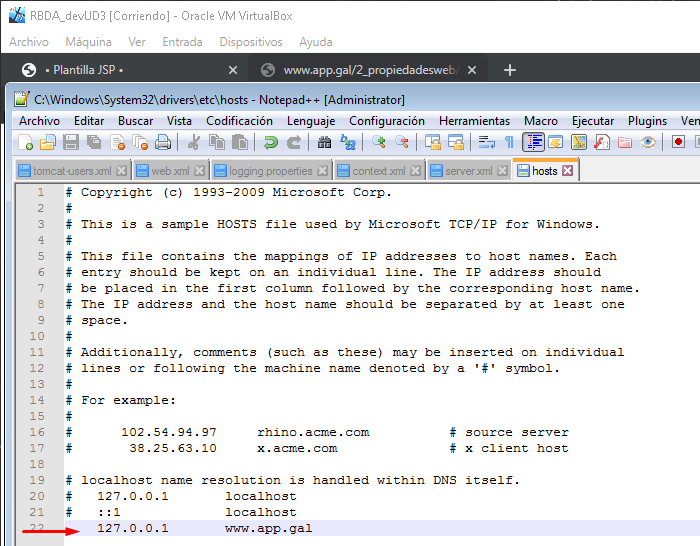
**<Connector port="80" protocol="HTTP/1.1"**

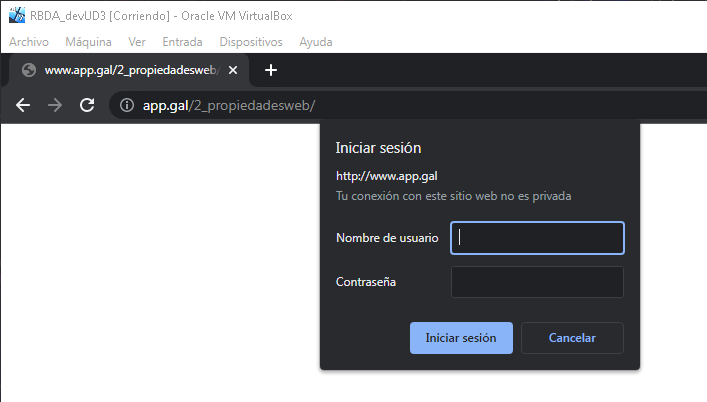
**connectionTimeout="20000"**

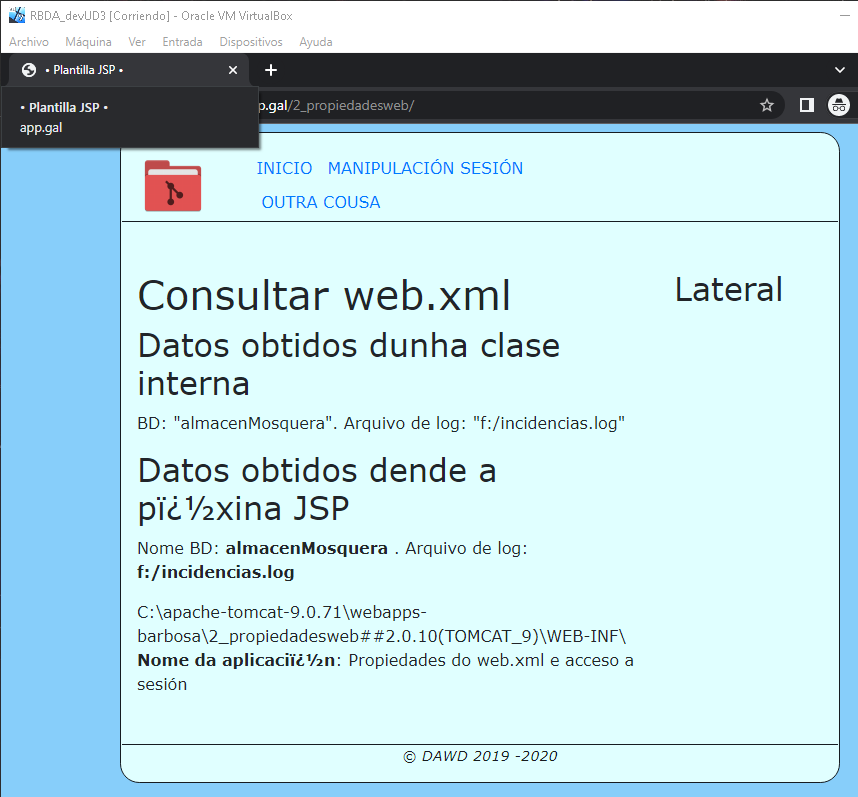
**redirectPort="8080" />**

### Funcionamiento

Para que funcione editar el fichero hosts.



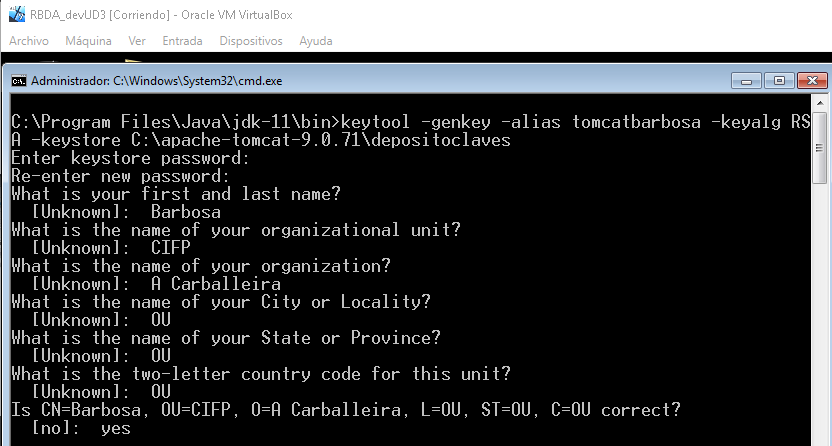




### Configuración SSL

Crear certificado  autofirmado en Windows.

**keytool –genkey –alias tomcatbarbosa –keyalg RSA –keystore C:\apache-tomcat-9.0.71\depositoclaves**



Configuramos https en **C:\apache-tomcat-9.0.71\conf\server.xml.**

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

               maxThreads="150" SSLEnabled="true"

   scheme="https"

   secure="true"

   clientAuth="false"

   sslProtocol="TLS"

   keystoreFile="c:\apache-tomcat-9.0.71\depositoclaves"

   keystorePass="abc123."

   keyAlias="tomcatbarbosa">

    </Connector>



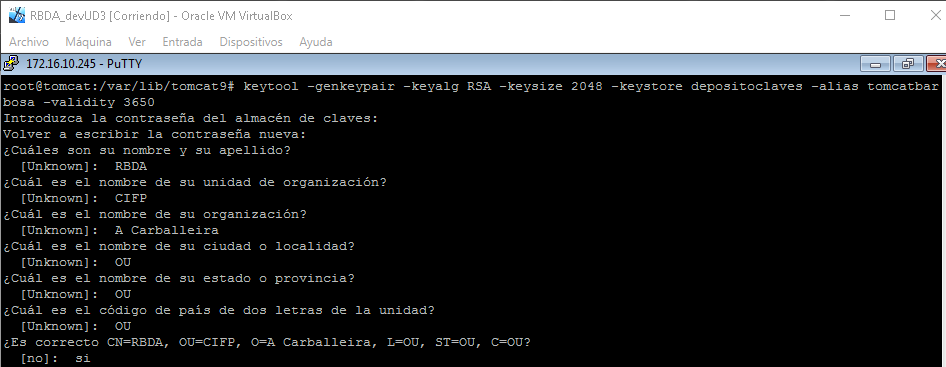
## Debian

En Debian é igual que en Windows solo mostraremos as diferencias.

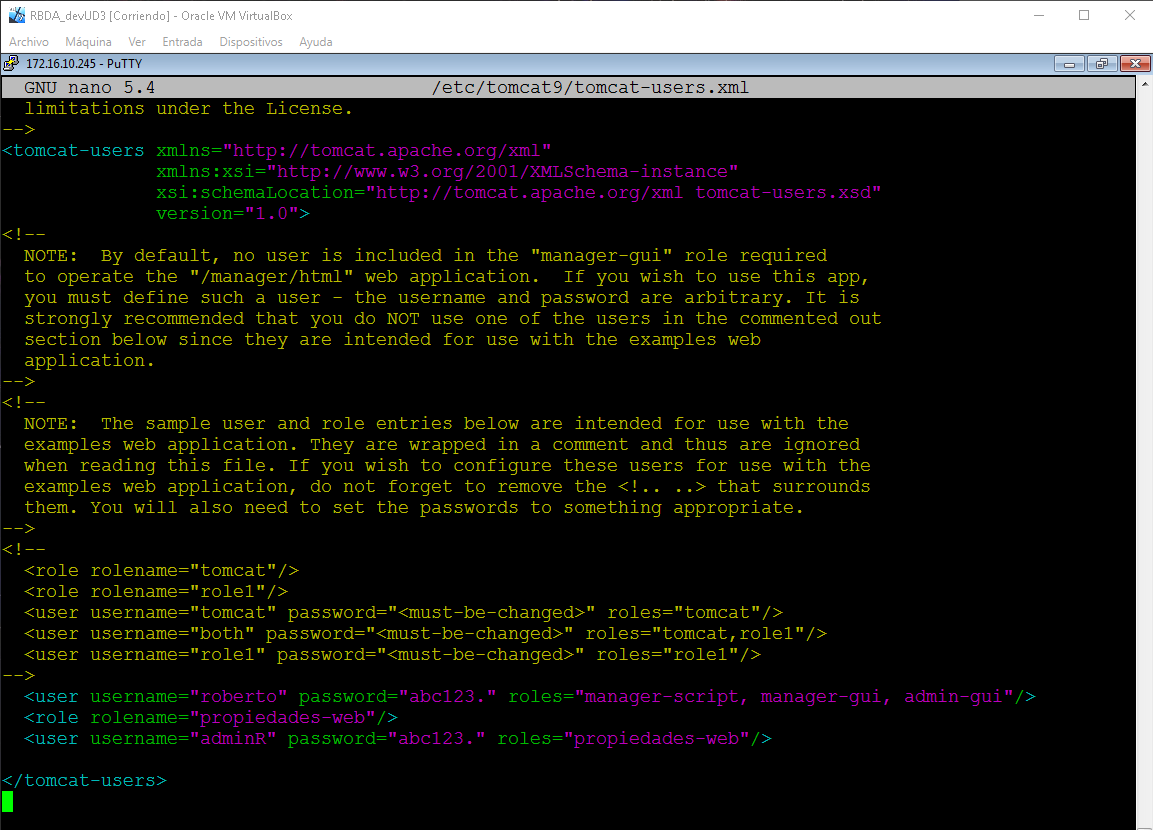
Cabe mencionar que hay que dar permisos a las carpetas para poder visualizar las apps.

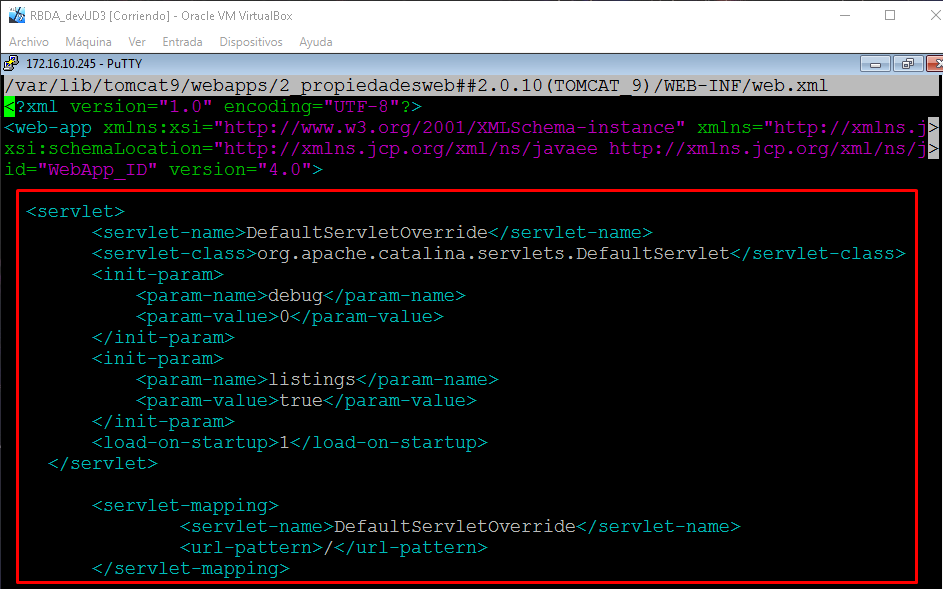
### Creación de un certificado

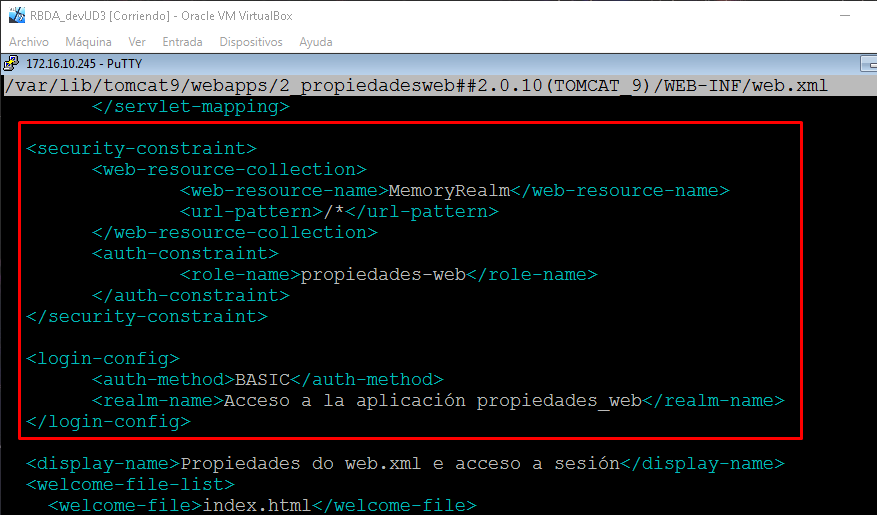
**keytool -genkeypair -keyalg RSA -keysize 2048 -keystore depositoclaves -alias tomcatbarbosa -validity 3650**

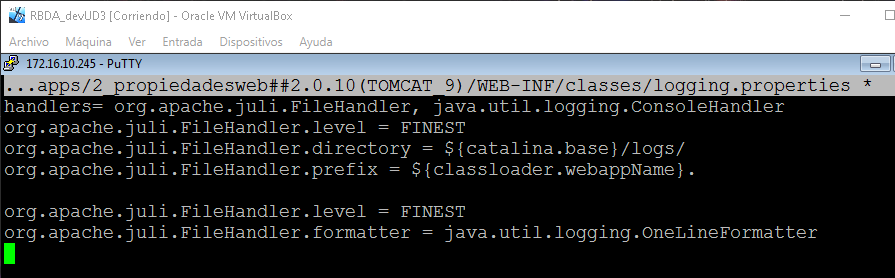


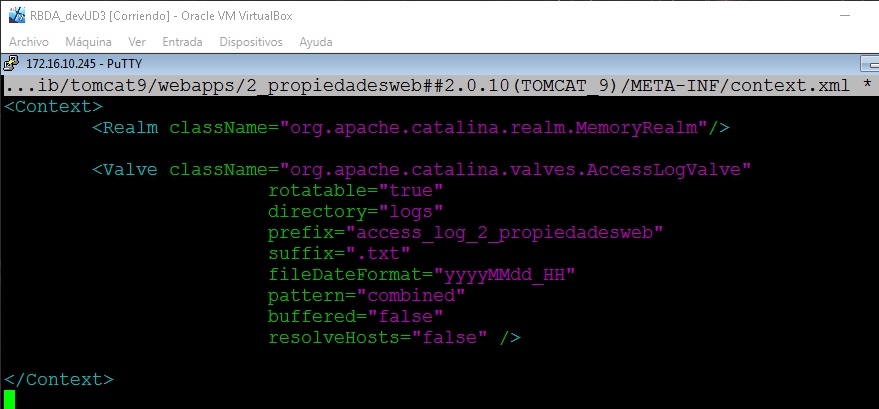
### Completo sin explicación en Debian (Solo capturas)

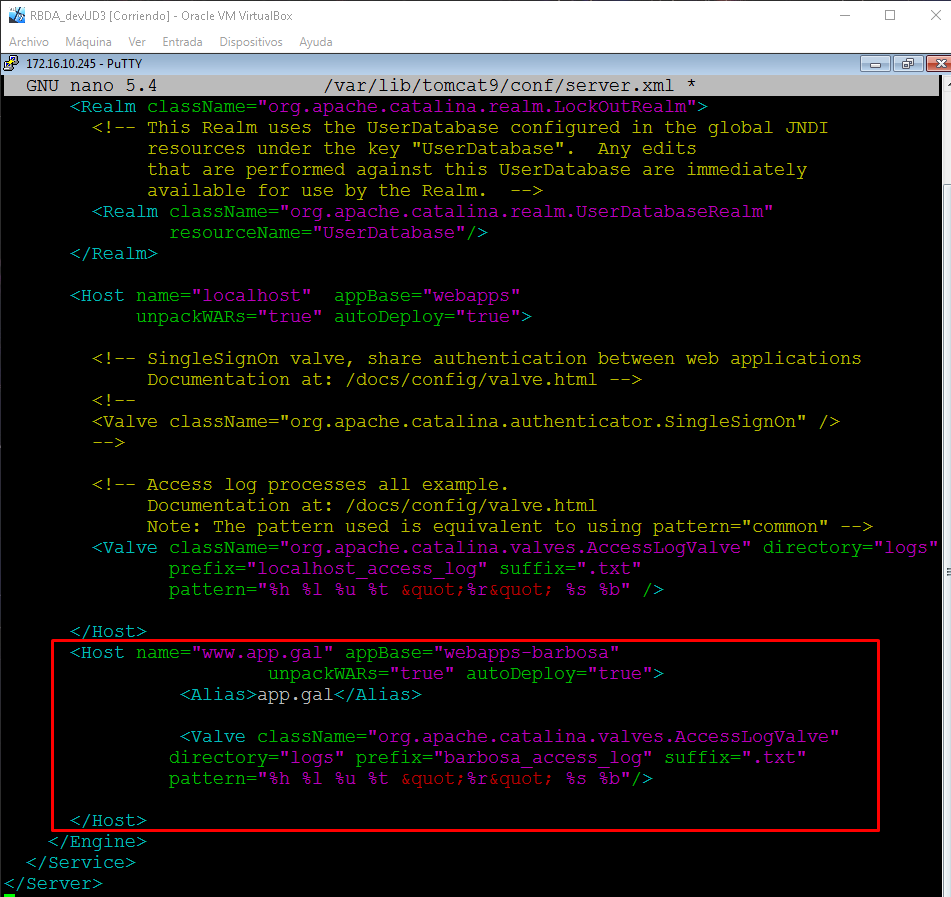


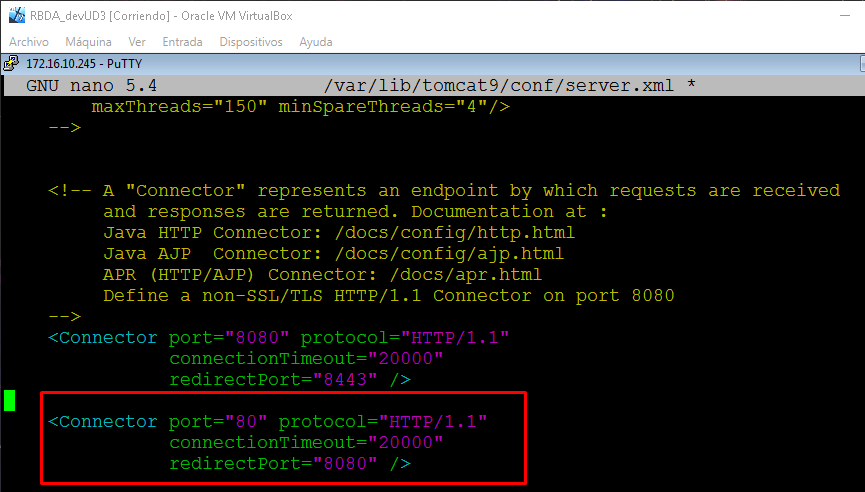




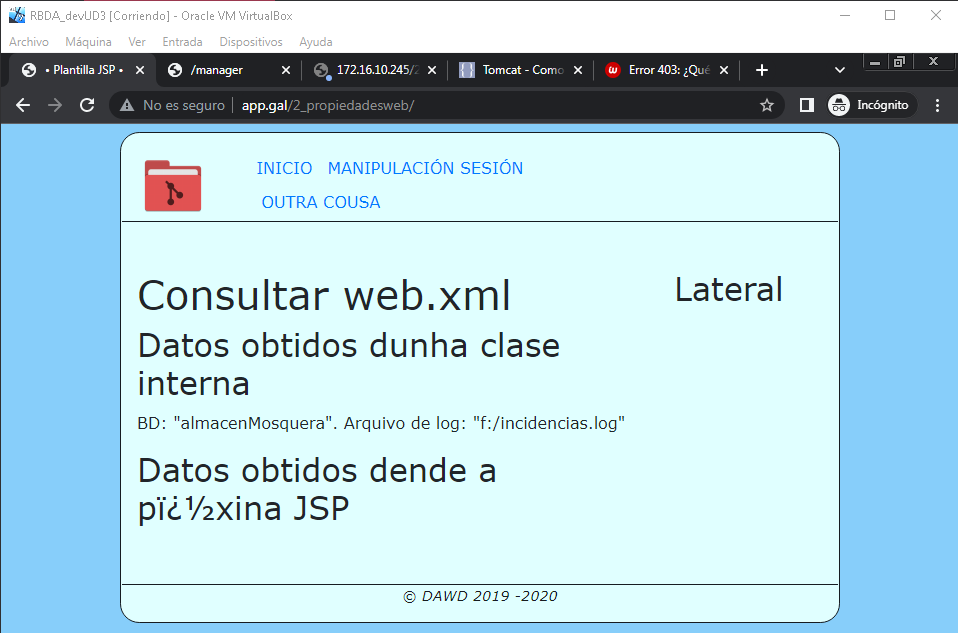




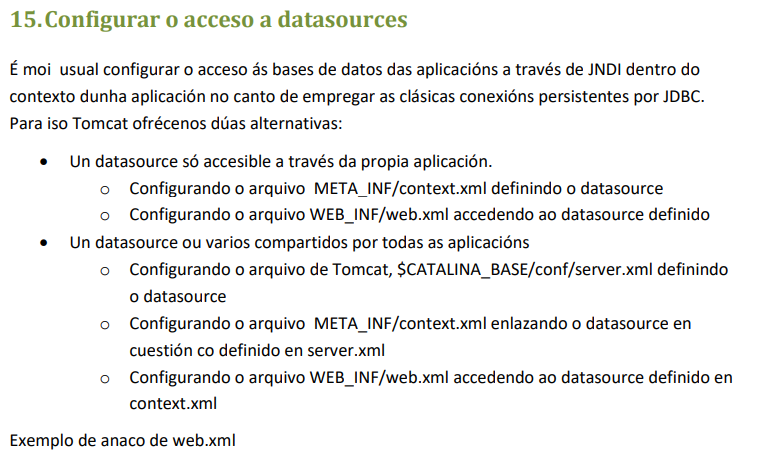




A parte de



# Datasource



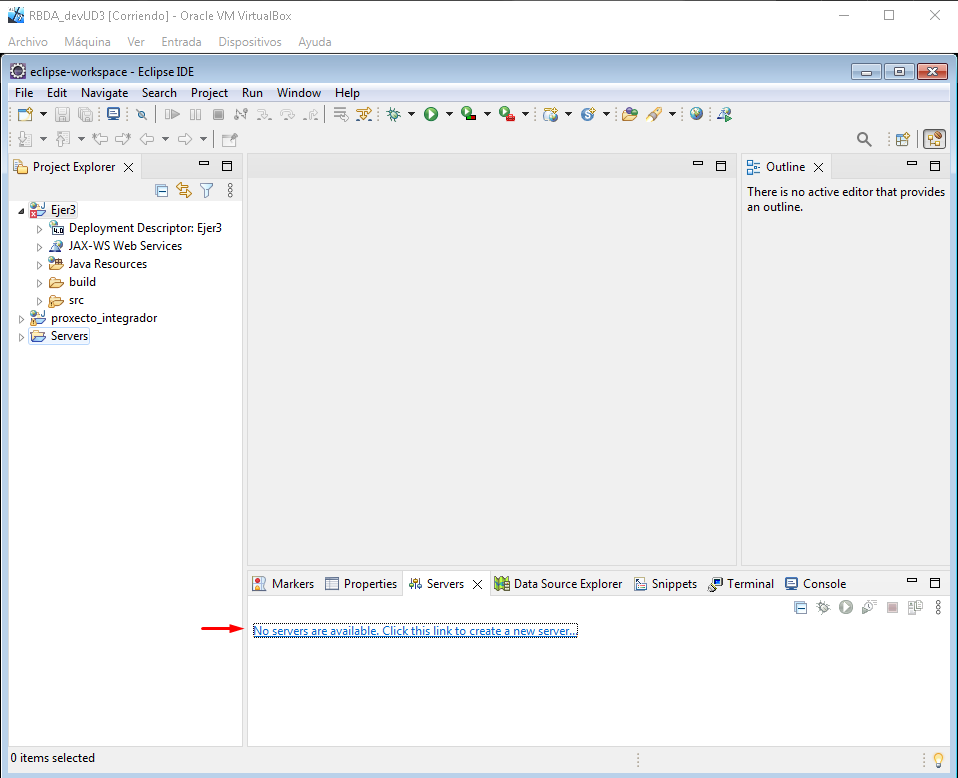


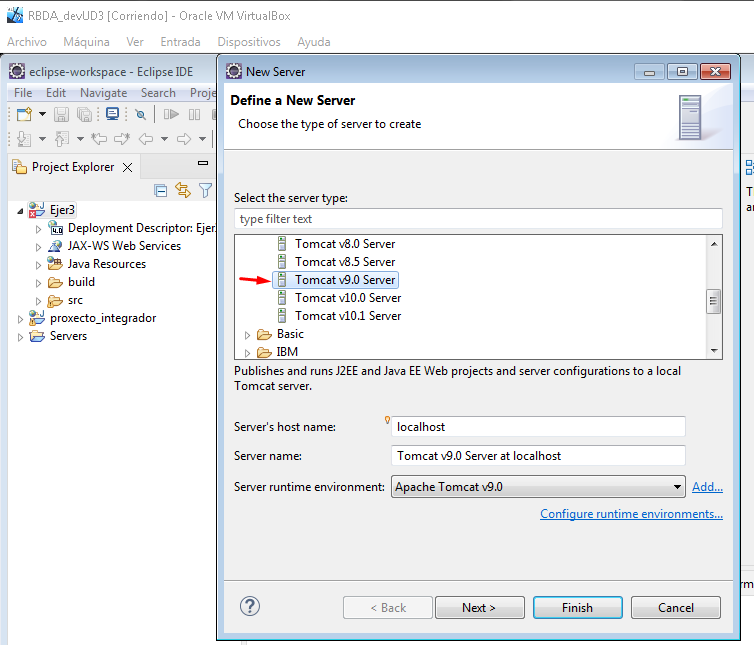


# Configuración Eclipse

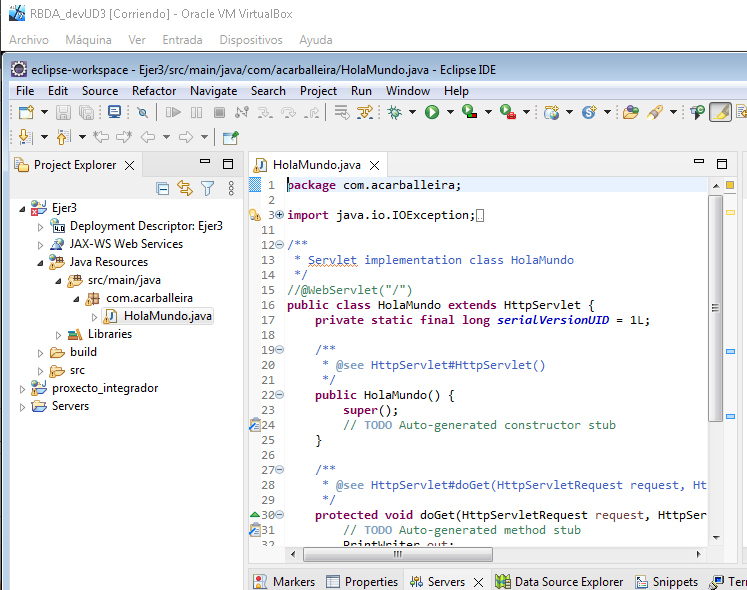
Descargamos Eclipse: <https://www.eclipse.org/downloads/packages/release/2022-12/r/eclipse-ide-enterprise-java-and-web-developers>

Lo descomprimimos y listos para empezar a trabajar.





Creamos nuestro proyecto de web dinamico que estará en formato .war con la entrega y lo corremos en nuestro servidor Tomcat.



Creando dentro un servlet con el siguiente contenido:

package com.acarballeira;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class HolaMundo

\*/

//@WebServlet("/")

public class HolaMundo extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public HolaMundo() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out;

out = response.getWriter();

response.setContentType("text/html");

out.println("<html>");

out.println("<head><title>Ejercicio3 UD3</title></head>");

out.println("<body>");

out.println("<h1>Hola Mundo UD3 DAW</h1>");

out.println("</body></html>");

}

}

Para exportarlos, click derecho sobre nuestro proyexto **Exportar->Archivo WAR,** optimizado para tomcat 9 y elegimos la ubicación del archivo.

