# Reto 3 – UD 4
# Despliegue de aplicaciones Java

**1.- Documenta la instalación, administración y despliegue de aplicaciones con un servidor de aplicaciones, bien sea GlassFish, WildFly, Payara u otro. En el caso de Glassfish (web profile)**

https://glassfish.org/docs/latest/quick-start-guide.html#deploying-and-undeploying-applications

Descargamos aplicación de muestra helloworld.war:

Descargamos glassFish:
https://howtoforge.es/como-instalar-el-servidor-de-aplicaciones-java-glassfish-en-rocky-linux/

Crear un nuevo usuario dedicado para GlassFish. En este ejemplo, ejecutarás el servidor de aplicaciones GLassFish a través del usuario no root '*glassfish*'.

```
arancha@daw2-01:~$ sudo useradd -m -d /opt/glassfish6 -U -s /bin/false glassfish
```

Una vez creado el usuario '*glassfish*', navega hasta el directorio *'/tmp* ' y descarga el paquete de distribución binaria de GlassFish mediante el comando wget que se indica a continuación.

```
arancha@daw2-01:/tmp$ ls
B402E497852071A55ECEA311E875854199C2FEA4
config-err-ICPfNX
CoreFxPipe_pipe_p7795#1
glassfish-web-7-0-11.zip
hsperfdata_arancha
lu333128ioau.tmp
mintUpdate
MozillaUpdateLock-4F96D1932A9F858E
OSL_PIPE_1004_SingleOfficeIPC_4b6284e737eb10c5d9646ce09dea56f8
```

Extraer el paquete GlassFish 'glassfish-6.2.5.zip' al directorio ' */opt* '.

```
arancha@daw2-01:/tmp$ sudo unzip /tmp/glassfish-web-7-0-11.zip -d /opt
[sudo] contraseña para arancha:
Archive:  /tmp/glassfish-web-7-0-11.zip
   creating: /opt/glassfish7/
   creating: /opt/glassfish7/META-INF/
   creating: /opt/glassfish7/bin/
   creating: /opt/glassfish7/glassfish/
```

```
arancha@daw2-01:/$ ls
bin   cdrom  etc   lib    lib64   lost+found  mnt  proc  run   snap  swapfile  tmp  var
boot  dev    home  lib32  libx32  media       opt  root  sbin  srv   sys       usr
arancha@daw2-01:/$ cd opt
arancha@daw2-01:/opt$ ls
containerd  glassfish6  glassfish7  google  pt  puppetlabs  vagrant
arancha@daw2-01:/opt$
```

Ahora que el paquete GlassFish está extraído en el directorio ' */opt/glassfish67*'.

Por último, ejecuta el siguiente comando para cambiar la propiedad del directorio de instalación de GlassFish *'/opt/glassfish7*' al usuario y grupo *'glassfish*'.

```
arancha@daw2-01:/opt$ sudo chown -R glassfish:glassfish /opt/glassfish7
arancha@daw2-01:/opt$ ls -l
total 28
drwx--x--x  4 root      root      4096 jun 22  2023 containerd
drwxr-x---  3 glassfish glassfish 4096 ene 19 13:25 glassfish6
drwxr-xr-x  6 glassfish glassfish 4096 dic  4 12:10 glassfish7
drwxr-xr-x  3 root      root      4096 jun 22  2023 google
drwxr-xr-x 14 root      root      4096 jun 22  2023 pt
drwxr-xr-x  6 root      root      4096 jun 22  2023 puppetlabs
drwxr-xr-x  4 root      root      4096 jun 22  2023 vagrant
arancha@daw2-01:/opt$
```

Ahora que ya has descargado el paquete GlassFish, puedes iniciar la aplicación GlassFish manualmente mediante el archivo binario *'/opt/glassfish6/bin/asadmin'*. Pero para hacerlo más fácil, vas a configurar y ejecutar GlassFish como un servicio systemd.

```
arancha@daw2-01:/opt$ sudo /opt/glassfish7/bin/asadmin
Use "exit" to exit and "help" for online help.
asadmin>
```

Crea un nuevo archivo de servicio systemd *'/lib/systemd/system/glassfish.service*' utilizando el siguiente editor nano.

```
  GNU nano 6.2                          /lib/systemd/system/glassfish.service *
[Unit]
Description = GlassFish Server v7
After = syslog.target network.target

[Service]
User=glassfish
ExecStart=/opt/glassfish7/bin/asadmin start-domain
ExecReload=/opt/glassfish7/bin/asadmin restart-domain
ExecStop=/opt/glassfish7/bin/asadmin stop-domain
Type = forking

[Install]
WantedBy = multi-user.target
```

A continuación, ejecuta el siguiente comando systemctl para recargar el gestor systemd y aplicar el nuevo archivo de servicio '*glassfish.service*'.

```
arancha@daw2-01:/opt$ sudo systemctl daemon-reload
```

Ahora que has recargado el gestor systemd, puedes iniciar y activar el servicio «*glassfish*» mediante el siguiente comando systemctl.

Sale error:

```
arancha@daw2-01:/opt$ sudo systemctl start glassfish
Job for glassfish.service failed because the control process exited with error code.
See "systemctl status glassfish.service" and "journalctl -xeu glassfish.service" for details.
arancha@daw2-01:/opt$ sudo systemctl start glassfish.service
Job for glassfish.service failed because the control process exited with error code.
See "systemctl status glassfish.service" and "journalctl -xeu glassfish.service" for details.
arancha@daw2-01:/opt$ sudo systemctl status glassfish
× glassfish.service - GlassFish Server v7
     Loaded: loaded (/lib/systemd/system/glassfish.service; disabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Fri 2024-01-19 13:57:30 CET; 17s ago
    Process: 36412 ExecStart=/opt/glassfish7/bin/asadmin start-domain (code=exited, status=1/FAILURE)
        CPU: 1min 9.223s
```

Por lo que pruebo con este otro tutorial:
https://comoinstalar.me/como-instalar-glassfish-en-ubuntu-20-04-lts/

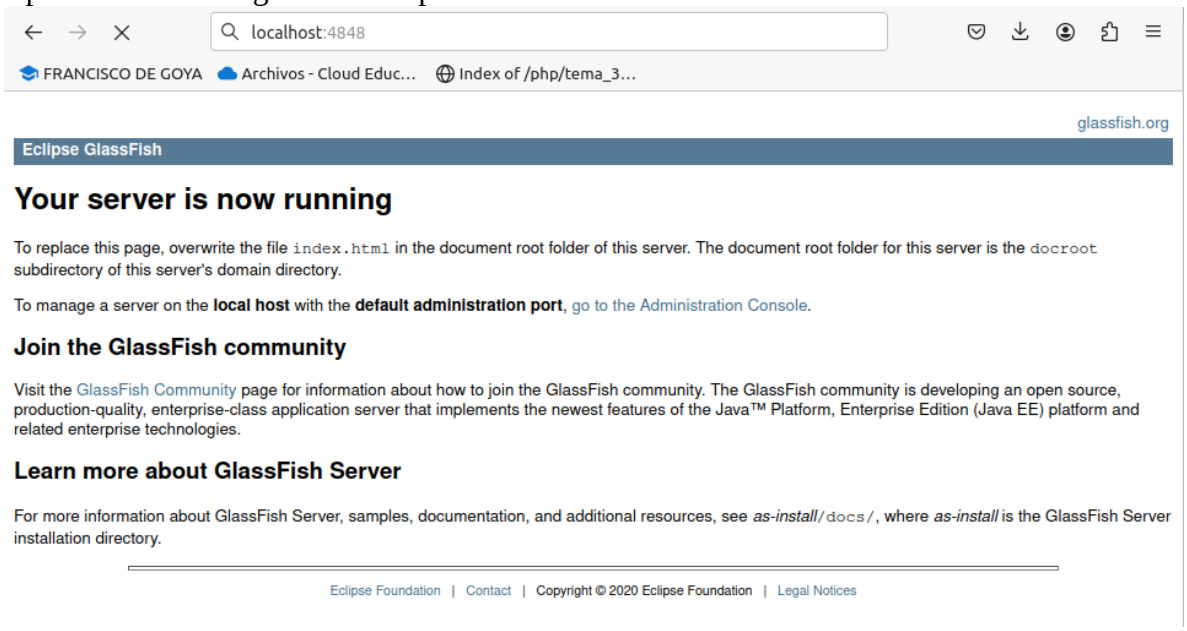
Si tienes activado el firewall UFW de Ubuntu 20.04 y quieres acceder a GlassFish desde red, será necesario añadir algunas reglas sobre los puertos principales.
Para la consola de administración, el puerto 4848 TCP:

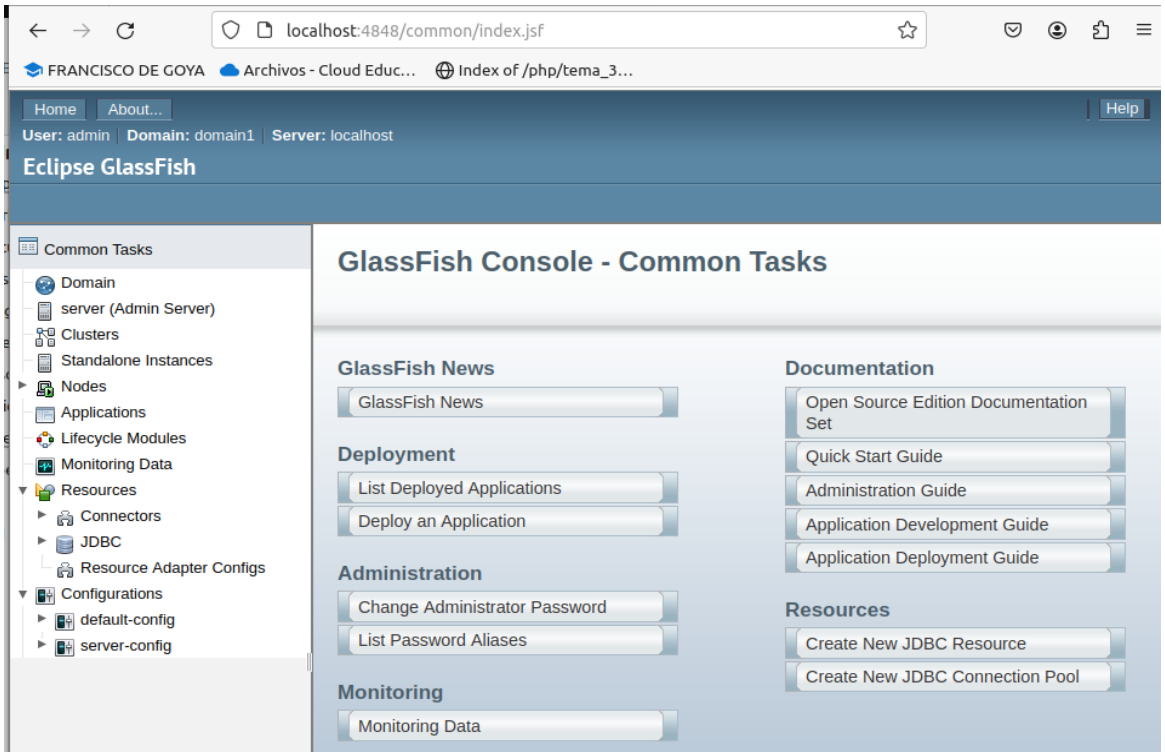
```
arancha@daw2-01:/opt/glassfish7/bin$ sudo ufw allow 4848/tcp
Reglas actualizadas
Reglas actualizadas (v6)
```

Y ejecuto glassfish7 de esta manera: me situo dentro de la carpeta glashfish/bin y ejecuto startserv:

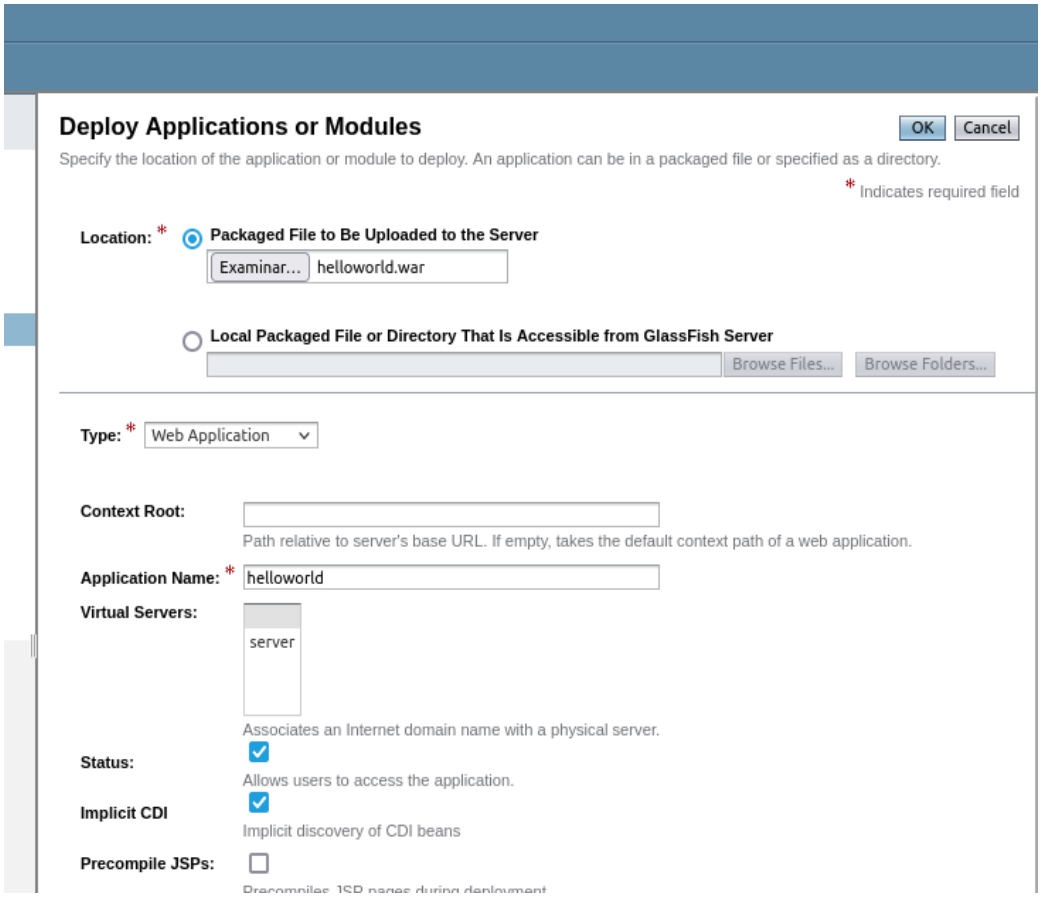
```
arancha@daw2-01:/opt/glassfish7/bin$ ls
asadmin  asadmin.bat  debug-asadmin  debug-asadmin.bat  startserv  startserv.bat  stopserv  stopserv.bat
arancha@daw2-01:/opt/glassfish7/bin$ . /startserv
bash: /startserv: No existe el archivo o el directorio
arancha@daw2-01:/opt/glassfish7/bin$ ./startserv
bash: ./startserv: Permiso denegado
arancha@daw2-01:/opt/glassfish7/bin$ sudo ./startserv
Launching GlassFish on Felix platform
OSGI framework packages:
null, org.glassfish.main.jul;uses:="org.glassfish.main.jul.cfg,org.glassfish.main.jul.handler";version="7.0.11
",org.glassfish.main.jul.cfg;version="7.0.11",org.glassfish.main.jul.env;version="7.0.11",org.glassfish.main.j
ul.formatter;uses:="org.glassfish.main.jul.cfg,org.glassfish.main.jul.record";version="7.0.11",org.glassfish.m
ain.jul.handler;uses:="org.glassfish.main.jul.cfg,org.glassfish.main.jul.formatter,org.glassfish.main.jul.reco
rd";version="7.0.11",org.glassfish.main.jul.record;version="7.0.11",org.glassfish.main.jul.rotation;version="7
```

Y pruebo en el navegador con el puerto 4848:

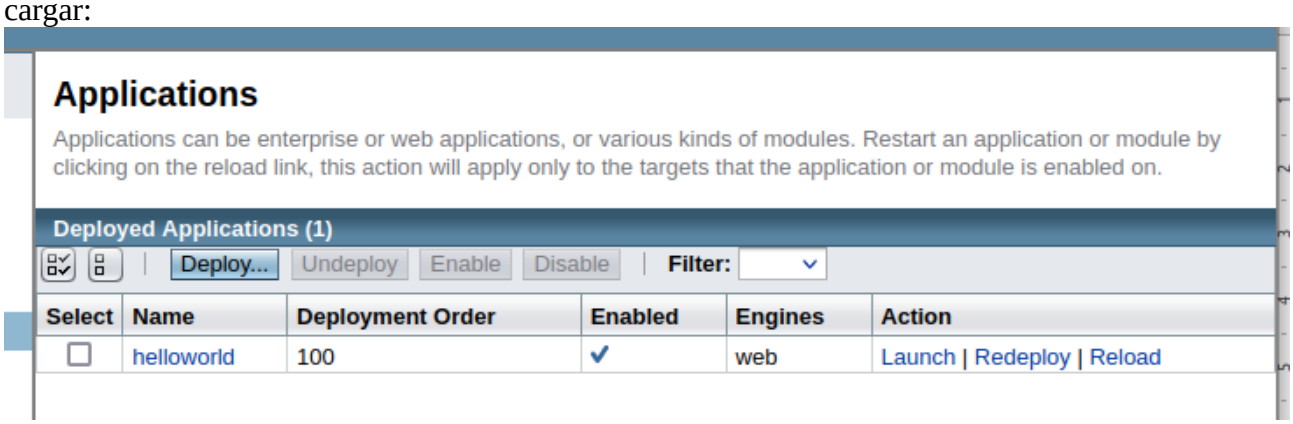




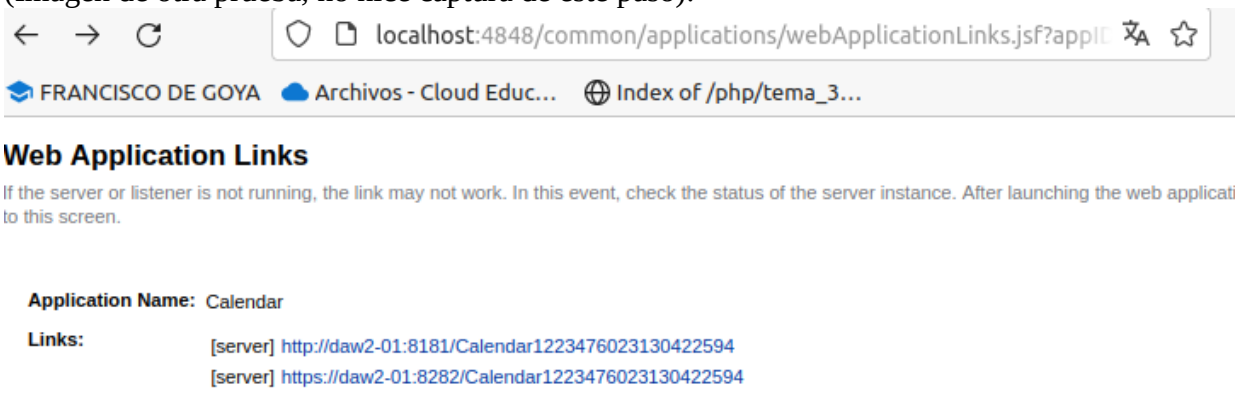

Ahora voy a desplegar el archivo hellowordl.war.
Desde la página*Common Task* seleccionamos*Deploy an Application.*
Y ahí, cargamos el archivo .war que queremos desplegar.
Dejamos todas las opciones como están y confirmamos:



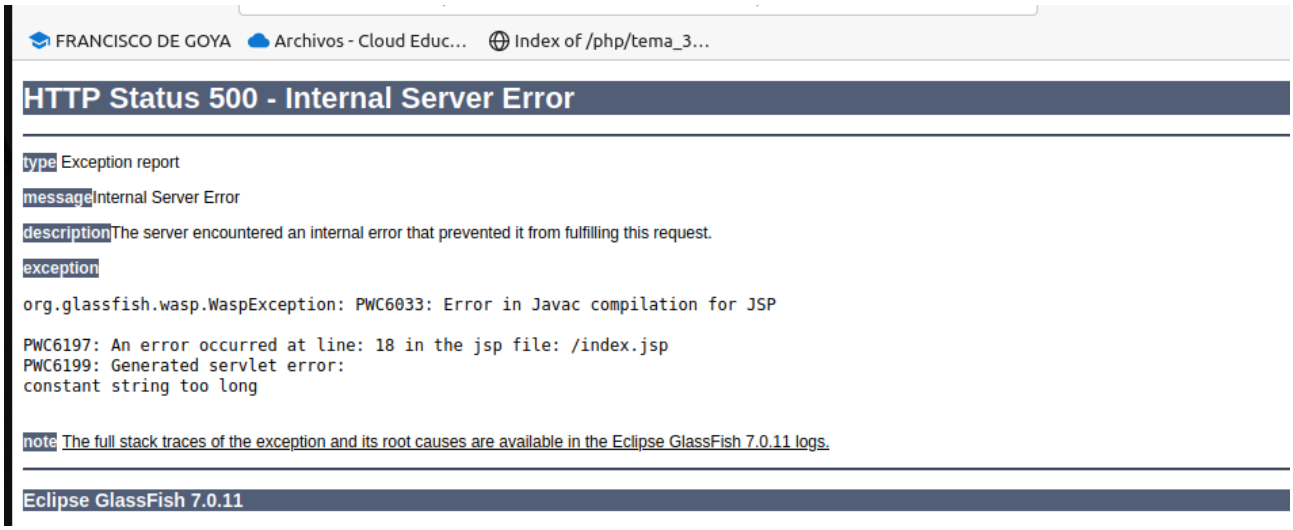
Y aparece la página de aplicaciones, donde vemos en el listado la aplicación que acabamos de cargar:




Si queremos probar la aplicación usaremos el enlace *Launch* de la columna *Action* que abrirá una nueva página que mostrará los enlaces a la aplicación, tanto la versión HTTP estándar como la conexión segura SSL:
(Imagen de otra prueba, no hice captura de este paso):




Al abrir cualquiera de los enlaces se debería mostrar la aplicación que acabamos de instalar, pero sale este error:

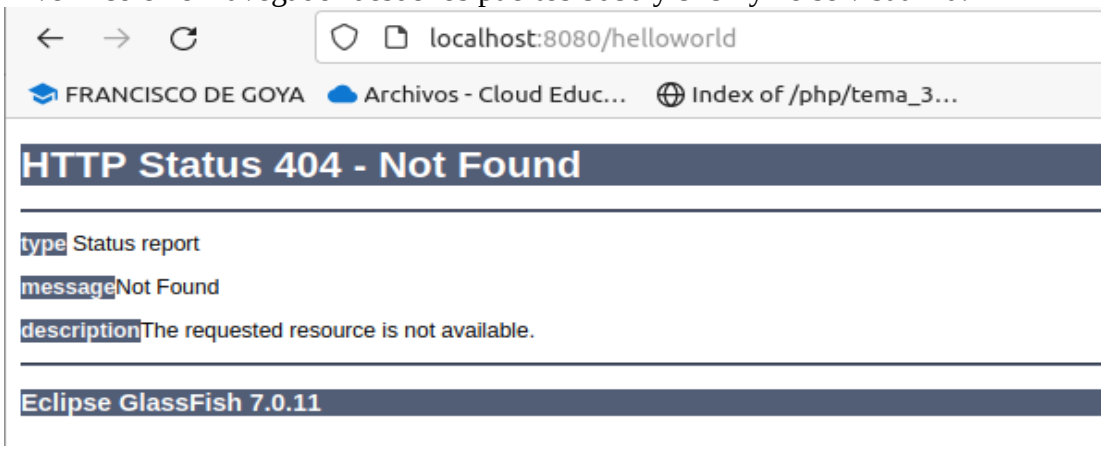


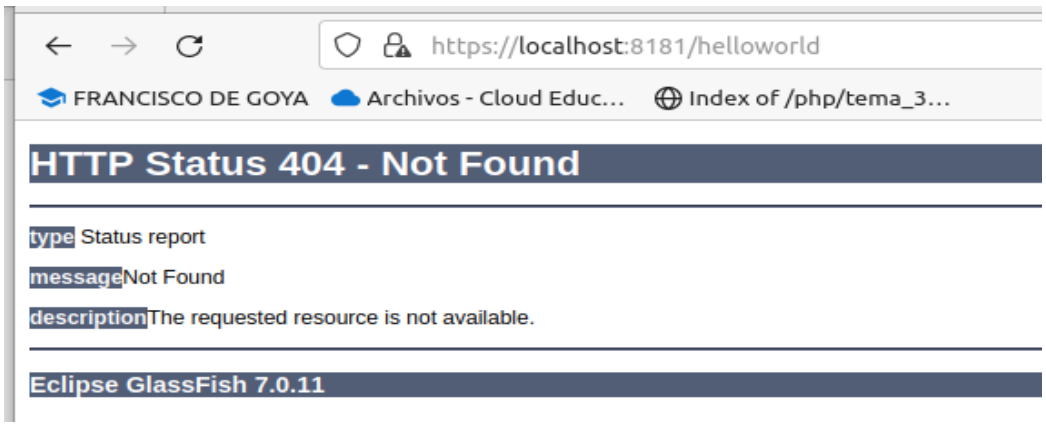

Reviso en el archivo de configuración domains.xml de glassfish para asegurarme del puerto de escucha:

domain.xml

```
            <file-cache></file-cache>
          </http>
          <ssl classname="com.sun.enterprise.security.ssl.GlassfishSSLImpl" cert-nickname="slas"></ssl>
        </protocol>
        <protocol name="admin-listener">
          <http encoded-slash-enabled="true" max-connections="250" default-virtual-server="__asadmin">
            <file-cache></file-cache>
          </http>
        </protocol>
      </protocols>
      <network-listeners>
        <network-listener protocol="http-listener-1" port="8080" name="http-listener-1" thread-pool="http-thread-pool"
transport="tcp"></network-listener>
        <network-listener protocol="http-listener-2" port="8181" name="http-listener-2" thread-pool="http-thread-pool"
transport="tcp"></network-listener>
        <network-listener protocol="admin-listener" port="4848" name="admin-listener" thread-pool="admin-thread-pool"
transport="tcp"></network-listener>
      </network-listeners>
```

Y verifico en el navegador desde los puertos 8080 y 8181 y no se visualiza:



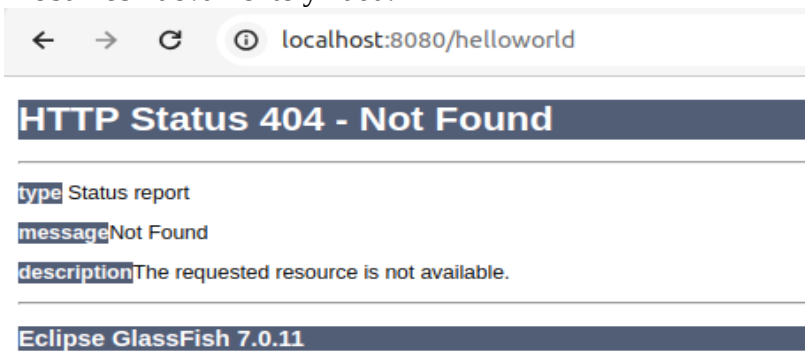

Pruebo a iniciar glassfish desde otro comando:

```
arancha@daw2-01:/opt/glassfish7/bin$ sudo ./asadmin start-domain
Waiting for domain1 to start ...
Waiting finished after 2.681 ms.
Successfully started the domain : domain1
domain  Location: /opt/glassfish7/glassfish/domains/domain1
Log File: /opt/glassfish7/glassfish/domains/domain1/logs/server.log
Admin Port: 4.848
Command start-domain executed successfully.
arancha@daw2-01:/opt/glassfish7/bin$
```

Y tampoco se visualiza la app en el navegador.
Verifico desde línea de comandos que la aplicación está desplegada y vemos que sí lo está:

```
arancha@daw2-01:/opt/glassfish7/bin$ sudo ./asadmin list-applications
sample      <web>
helloworld  <web>
Command list-applications executed successfully.
arancha@daw2-01:/opt/glassfish7/bin$
```

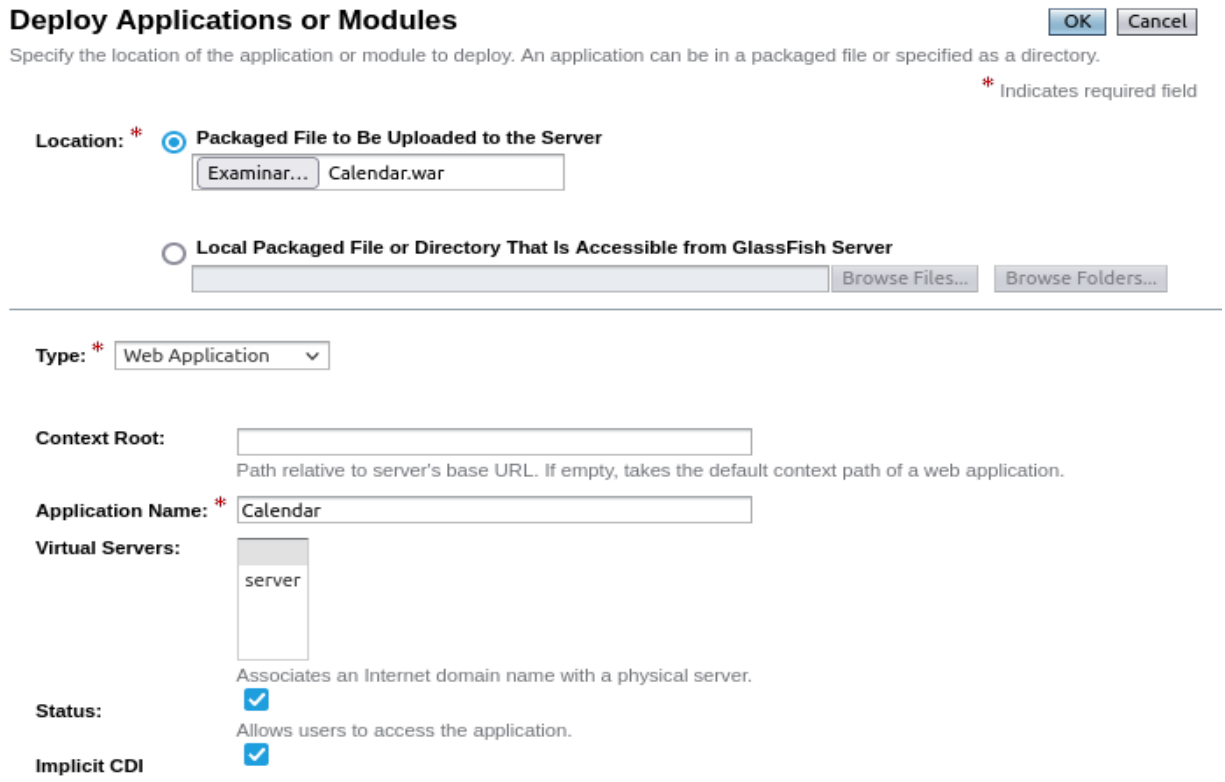
Probamos nuevamente y nada:



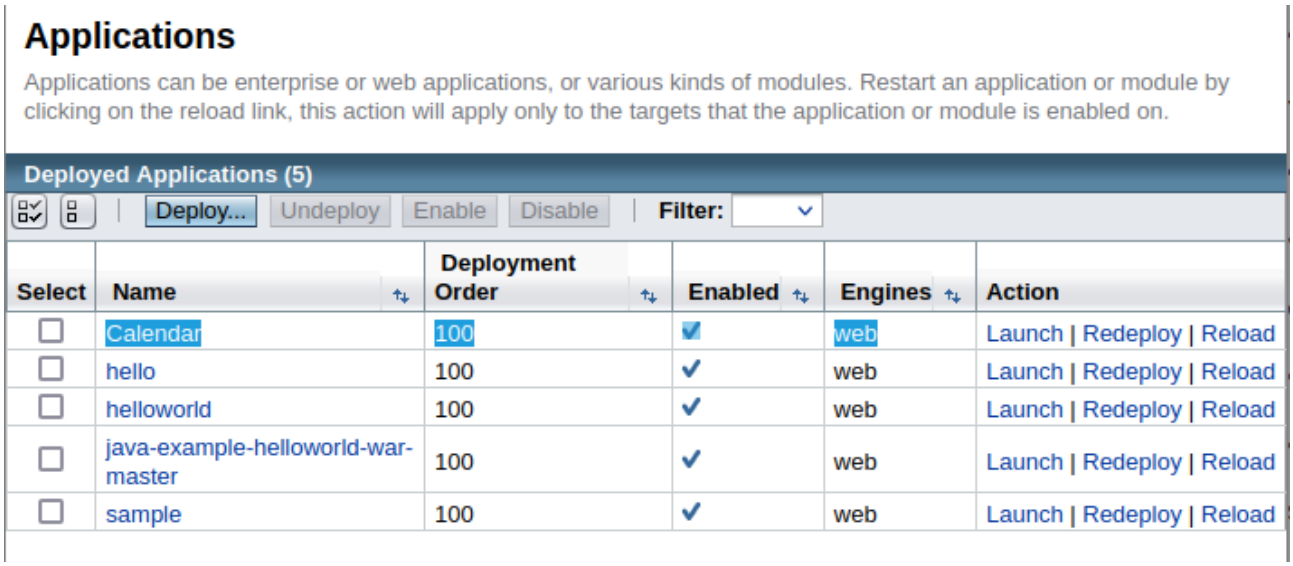

Comprobamos también que el dominio esté running:

```
arancha@daw2-01:/opt/glassfish7/bin$ sudo ./asadmin list-domains
domain1 running
Command list-domains executed successfully.
```

Pruebo con otro archivo .war, esta vez será calendar.war:




Vemos en el listado de aplicaciones, varias aplicaciones que he desplegado y con las que he hecho pruebas:



## Applications

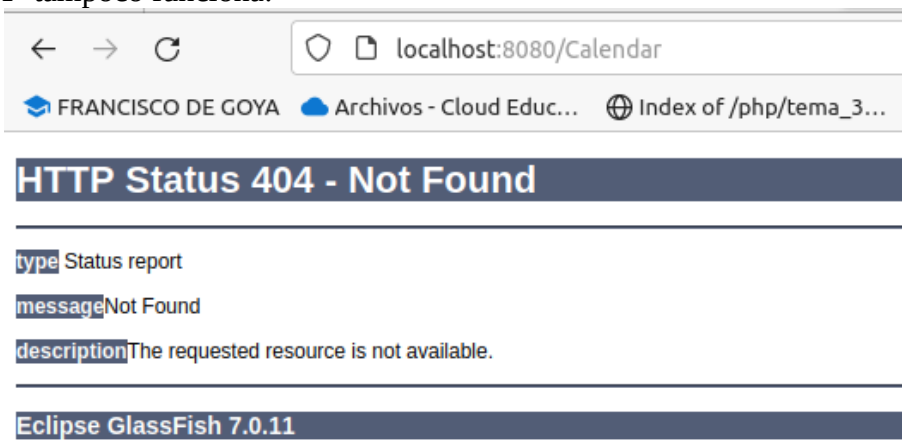
Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.

Deployed Applications (5)

Deploy... Undeploy Enable Disable Filter:

| Select | Name | Deployment Order | Enabled | Engines | Action |
|---|---|---|---|---|---|
| ☐ | Calendar | 100 | ☑ | web | Launch \| Redeploy \| Reload |
| ☐ | hello | 100 | ✓ | web | Launch \| Redeploy \| Reload |
| ☐ | helloworld | 100 | ✓ | web | Launch \| Redeploy \| Reload |
| ☐ | java-example-helloworld-war-master | 100 | ✓ | web | Launch \| Redeploy \| Reload |
| ☐ | sample | 100 | ✓ | web | Launch \| Redeploy \| Reload |

Y tampoco funciona:

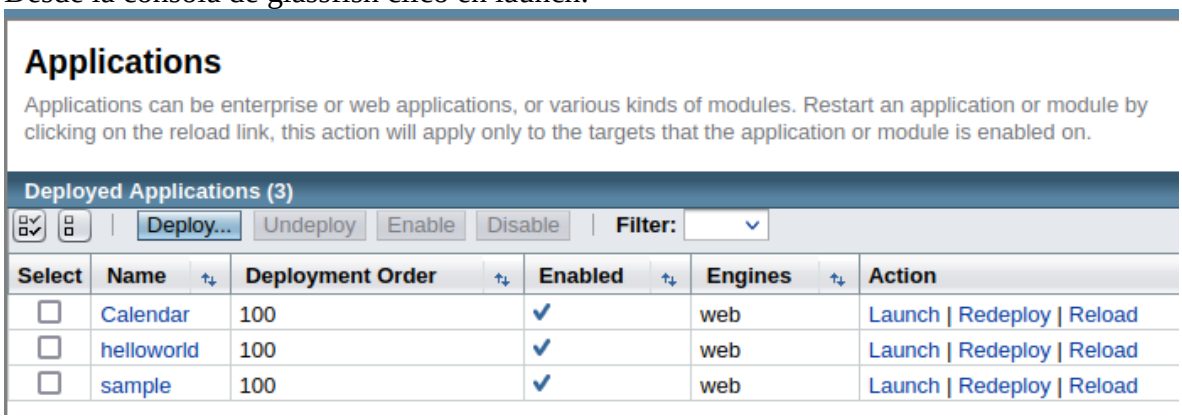



Por si hubiera algún choque con el puerto 8080 con apache, modifico el puerto de escucha para glassfish. **Consejo de mi compañero Rodri.**
Modifico el puerto de escucha al 8181, en el archivo domains.xml en la carpeta de glassfish:

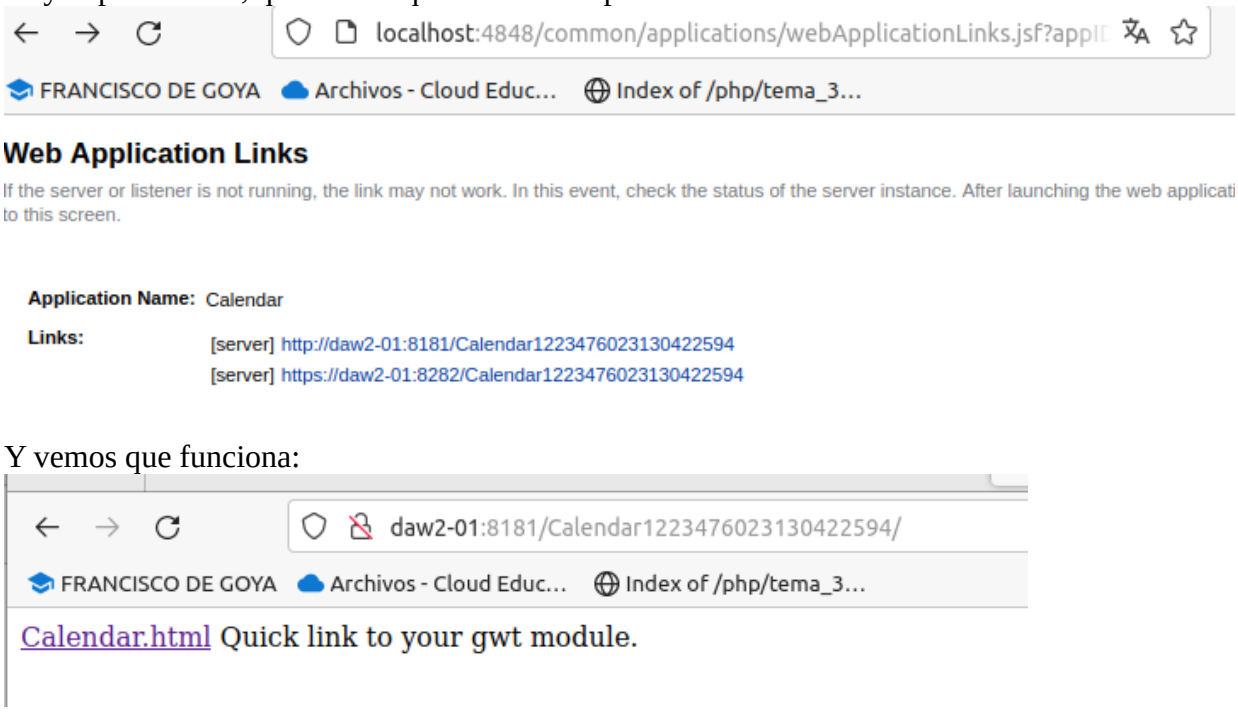

Reinicio glassfish:

```
arancha@daw2-01:/opt/glassfish7/bin$ sudo ./asadmin stop-domain
Waiting for the domain to stop .
Waiting finished after 118 ms.
Command stop-domain executed successfully.
arancha@daw2-01:/opt/glassfish7/bin$ sudo ./asadmin start-domain
Waiting for domain1 to start ...
Waiting finished after 2.515 ms.
Successfully started the domain : domain1
domain  Location: /opt/glassfish7/glassfish/domains/domain1
Log File: /opt/glassfish7/glassfish/domains/domain1/logs/server.log
Admin Port: 4.848
Command start-domain executed successfully.
arancha@daw2-01:/opt/glassfish7/bin$
```

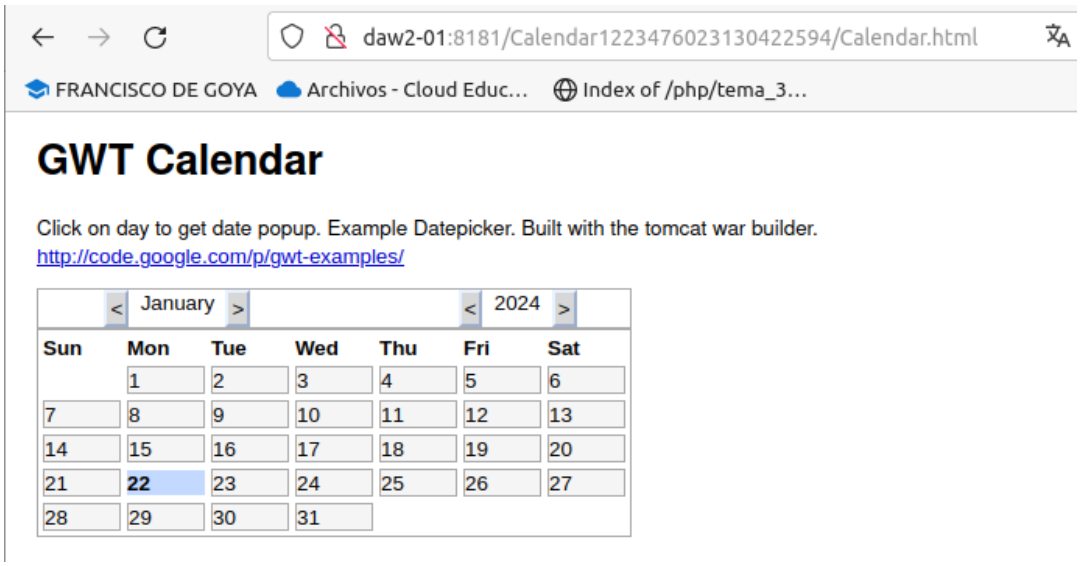
Desde la consola de glassfish clico en launch:




Doy al primer link, que vemos que es desde el puerto 8181:

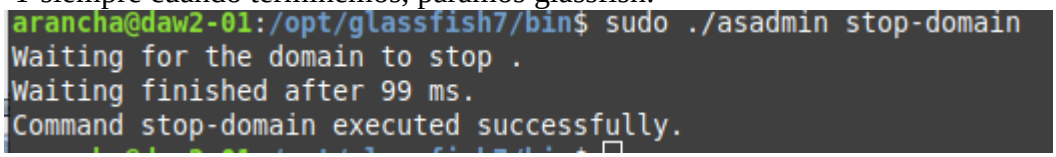



Y vemos que funciona:




Probamos ahora con el helloworld:
Y ese no se visualiza, por error en el archivo index.jsp. Ahora al menos sale el error.


Y siempre cuando terminemos, paramos glassfish:



**2.-Para el bien intenta el despliegue de una aplicación más grande, como este CMS:**
**https://www.openwga.com/home/downloads/openwga-cms-server/war-file-distribution.38.en.html**

**u otros**
**https://www.jenkins.io/doc/book/installing/war-file/**
**o compilando alguno de estos otros ejemplos:**
**https://github.com/eclipse-ee4j/glassfish-samples**
**https://github.com/javaee/tutorial-examples**

Como dicho en clase, al dar fallos no he realizado este apartado. He visto tres de los enlaces, y como o había que hacer varias configuraciones en Ubuntu (yo tengo Windows y con la máquina virtural de Ubuntu tengo problemas y no trabajo bien) o se necesitaba mucha memoria, como en el caso de Jenkins, he preferido no empezarlo pues se sabía de antemano los errores.

**3.- Para el excelente lee y resume lo aprendido de los siguientes documentos:**
**https://raul-profesor.github.io/DEAW/ServAplic/**
**https://www.middlewareinventory.com/blog/sample-web-application-war-file-download/**

**https://docs.spring.io/spring-boot/docs/current/reference/html/deployment.html**
**https://docs.spring.io/spring-boot/docs/current/reference/html/howto.html#howto.traditional-deployment**

# SERVIDOR DE APLICACIONES

Un servidor de aplicaciones es un marco mixto de software que permite tanto la creación de aplicaciones web como un entorno de servidor para ejecutarlas.

Situado entre el servidor web y el nivel de backend del servidor de bases de datos, el servidor de aplicaciones es un intermediario para el servidor de bases de datos y los usuarios que soporta mediante el uso de protocolos e interfaces de programación de aplicaciones (API).

Es habitual que se utilice junto con un servidor web o que contenga un servidor web.

Cuando los usuarios de las aplicaciones solicitan acceso a una aplicación, el servidor de aplicaciones suele hacer el trabajo en el backend para almacenar y procesar las solicitudes dinámicas de las aplicaciones.

**¿Por qué necesitamos servidores de aplicaciones?**
Cada aplicación en uso, está siendo consultada en un servidor de aplicaciones y devuelta a través de un servidor web.
Los servidores web se encargan de servir a los clientes web peticiones HTTP con respuestas HTTP.

Los servidores de aplicaciones optimizan el tráfico y añaden seguridad.

**Servidores de aplicaciones: El mejor amigo de un servidor web**
Los servidores de aplicaciones hacen de conector de los servidores web. Cuando los servidores web tienen una petición del cliente que no pueden soportar, los servidores de aplicaciones hacen posible mantener la comunicación con el contenido web dinámico.

**¿Qué es el despliegue de aplicaciones web?**
Significa pasar los cambios o actualizaciones de un entorno de funcionamiento a otro. Al configurar un sitio web, siempre se tendrá el entorno en vivo o entorno de producción.
Para hacer cambios sin afectar a un sitio web en producción, hay que añadir entornos adicionales que suelen ser local, de desarrollo y de preparación o preproducción. Estos entornos se llaman entornos de desarrollo o entornos de despliegue.
El proceso de despliegue consta de 5 pasos: Planificación, desarrollo, pruebas, despliegue y supervisión. Y una vez completado este proceso de despliegue, los nuevos cambios serán visibles en el entorno activo.

**Tipos de despliegue**
- Metadatos: incluyen los cambios en el código, las plantillas, las hojas de estilo, los archivos, etc. Estos cambios a menudo requerirán una comprobación de validación entre entornos para ver si tiene algún conflicto.
- Contenidos: El contenido se maneja de forma diferente durante el despliegue. A menudo las herramientas de despliegue hacen que el despliegue de contenido sea accesible para los editores de contenido. De esta manera, un editor de contenidos no depende de un desarrollador.

**Mejores prácticas de despligue**
- Utilizar GIT: sistema de control de versiones para el flujo de trabajo de despliegue.

- Usar ramas: permitirá trabajar en varias cosas al mismo tiempo.
- Usar entorno local como entorno de desarrollo: Al instalar el sitio web o el software de forma local, se podrá trabajar de forma más eficiente y acelerar las pruebas y la verificación del código.
- Grupos de usuarios con diferentes permisos: para restringir quién puede desplegarlos en vivo.

## Qué es una aplicación web de Java

J2EE tiene varios componentes y servicios y los componentes J2EE se pueden clasificar en dos grupos principales:

1. Componentes Web (HTML, JSF, Servlet, JSP)
2. Componentes EJB (EJB, EDB, SDB, MDB)

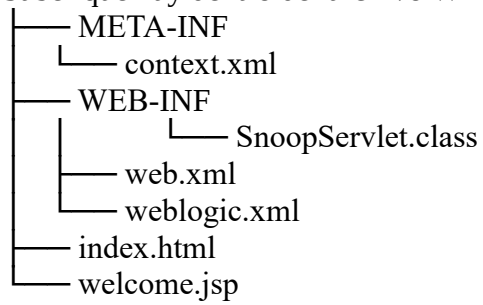
Los componentes dinámicos que recibirán las peticiones HTTP en el servidor serán los servlets y JSPs y podrán analizar esta petición y utilizar otros componentes Java para realizar las acciones necesarias (beans, EJBs, etc)

La aplicación web o una WAR es un Colectivo de estos Componentes Web definidos por J2EE con una estructura y formatos de Directorio Estándar.

**Empaquetamiento**
Una forma de distribuir aplicaciones Web es empaquetar la aplicación de un fichero WAR, que podremos utilizarlo en los diferentes servidores de aplicaciones JavaEE existentes.

**¿Qué es archivo WAR y qué significa?**
**WAR** es un acrónimo de **W** eb **Ar** chive
Cuando agrupas los componentes web en una arquitectura específica y definida, se convierte en una aplicación web completa, y en su mayoría tienen *.war como extensión.
Son un tipo especial de JAR utilizado para distribuir los artefactos o contenido de las aplicaciones Web en tecnología JEE

Saber qué hay dentro del archivo WAR y la estructura de directorios del archivo WAR.




**Dónde desplegar o alojar aplicación web Java**
Basado en el Tipo de Archivo o tipo de Aplicación Web Java, Puede ser implementado o alojado en uno de los siguientes servidores de aplicaciones J2EE:

- Apache Tomcat - Soporta sólo archivo web / WAR
- JBoss - Soporta WAR y Enterprise
- Oracle Weblogic - Soporta WAR y Enterprise
- IBM Websphere - Soporta tanto WAR como Archivo Empresarial
- GlassFish - Soporta la WAR y EAR

**Apache Tomcat vs Todos los demás servidores de aplicaciones J2EE**
La principal diferencia entre Tomcat y otro servidor de aplicaciones J2EE respaldado por la corporativa es el soporte de componentes y servicios J2EE
Puede construir su Java WebApp con Spring y desplegarlo en tomcat.

WEB-INF es obligatorio para toda la aplicación web y META-INF es opcional para una aplicación web.

**¿Cómo funciona un servidor de aplicaciones Java?**
Servlets: programa Java que se ejecuta en un servidor Web y construye o sirve páginas web dinámicas, basadas en diferentes fuentes variables. Es más sencillo, eficiciente, potente y portable que un CGI y con ellos podremos, procesar, sincronizar y coordinar múltiples peticiones de clientes

1. El cliente abre un navegador y solicita acceso a un sitio web.
2. El servidor web recibe la petición HTTP y responde con la página web deseada.
3. El servidor web gestiona las peticiones de datos estáticos, pero el cliente quiere utilizar una herramienta interactiva
4. Al tratarse de una petición de datos dinámicos, el servidor web transfiere la petición a un servidor de aplicaciones
5. El servidor de aplicaciones recibe la petición HTTP y la convierte en una petición de servlet.
6. El servlet llega al servidor de la base de datos, y el servidor de aplicaciones recibe una respuesta del servlet.
7. El servidor de aplicaciones traduce la respuesta del servlet al formato HTTP para el acceso del cliente.

**Maven**
Es una herramienta open-source, que se creó con el objetivo de simplificar los procesos de build, para compilar y generar ejecutables a partir del código fuente.
Es una herramienta capaz de gestionar un proyecto software completo.

**Node.js**
Es un entorno de ejecución de JavaScript rápido que utilizamos para construir aplicaciones del lado del servidor.

**Npm**
Manejador de paquetes de node, es la herramienta por defecto de JavaScript para la tarea de compartir e instalar paquetes.

## Implementación de aplicaciones Spring Boot en la nube

Se puede implementar aplicaciones Spring Boot en una variedad de plataformas en la nube, en máquinas virtuales/reales, o hacerlas completamente ejecutables para sistemas Unix.
Los archivos ejecutables de Spring Boot están preparados para los proveedores de PaaS (plataforma como servicio) en la nube más populares.
Heroku y Cloud Foundry, emplean un enfoque de "paquete de construcción". El paquete de compilación envuelve su código implementado en lo que sea necesario para *iniciar* su aplicación.

**Cloud Foundry**
Proporciona paquetes de compilación predeterminados. Tiene un excelente soporte para aplicaciones Spring, incluido Spring Boot. Puede implementar aplicaciones jar ejecutables independientes, así como .waraplicaciones empaquetadas tradicionales.
Los metadatos sobre la aplicación en ejecución se exponen a la aplicación como variables de entorno.

**Heroku**
Para personalizar las compilaciones de Heroku, proporciona un Procfile, que proporciona el encantamiento necesario para implementar una aplicación. Heroku asigna un portpara que lo use la aplicación Java y luego se asegura de que el enrutamiento al URI externo funcione.

**Servicios web de Amazon (AWS)**
Ofrece múltiples formas de instalar aplicaciones basadas en Spring Boot, ya sea como aplicaciones web tradicionales (guerra) o como archivos jar ejecutables con un servidor web integrado. Las opciones incluyen:

- AWS Elástico Beanstalk
- Implementación de código de AWS
- Funciona con AWS OPS
- Formación de la nube de AWS
- Registro de contenedores de AWS

**Kubernetes**
Spring Boot detecta automáticamente los entornos de implementación de Kubernetes comprobando las variables y el. Puede anular esta detección con la propiedad de configuración. Cuando Kubernetes elimina una instancia de aplicación, el proceso de cierre involucra varios subsistemas simultáneamente, y mientras tanto Cuando Kubernetes elimina una instancia de aplicación, el proceso de cierre involucra varios subsistemas simultáneamente

**OpenShift**
Tiene muchos recursos que describen cómo implementar aplicaciones Spring Boot, que incluyen:

- Usando el constructor S2I
- guía de arquitectura
- Ejecutándose como una aplicación web tradicional en Wildfly
- Informe de OpenShift Commons

**CloudCaptain y servicios web de Amazon**
Convierte su jar o war ejecutable de Spring Boot en una imagen de VM mínima que se puede implementar sin cambios en VirtualBox o en AWS. CloudCaptain viene con una integración profunda para Spring Boot y configura automáticamente puertos y URL de verificación de estado.

**Google Cloud**
Tiene varias opciones que se pueden utilizar para iniciar aplicaciones Spring Boot. El método más fácil sea App Engine y en el contenedor con Container Engine o en una máquina virtual con Compute Engine.

# Instalación de aplicaciones Spring Boot

Las aplicaciones Spring Boot también se pueden ejecutar como systemd, init.d o Windows services.

**Systemd Service**
Estas aplicaciones se pueden lanzar usando scripts de systemd service. Si hay una aplicación spring bot en un jar, para instatlar como un systemd service, se crea un script llamado _.service y se sitúa en /etc/systemd/system

**Microsoft Windows Service**
Una aplicación Spring Boot se puede iniciar como un servicio de Windows usando winsw.

**Init.d Service**
Para utilizar su aplicación como init.dservicio, configure su compilación para producir un jar completamente ejecutable , pudiéndose hacer con Maven.

# Implementaciones eficientes

**Descomprimir el Jar ejecutable**

**Usar procesamiento anticipado con JVM**
Utilizando el código de inicialización generado por AOT.

**Checkpoint y restauración con la JVM**
La restauración coordinada en Checkpoint (CRaC) es un proyecto OpenJDK que define una nueva API de Java para permitirle controlar y restaurar una aplicación en HotSpot JVM.

# Implementación tradicional

**Crear archivo war implementable**
- Proporcionar una SpringBootServletInitializersubclase en la clase principal y anular su configuremétodo.
- Actualizar la configuración de compilación para que el proyecto produzca un archivo war en lugar de un archivo jar. En Maven el archivo pom.xml y en Gradle el build.gradle.
- Marcar la dependencia del contenedor de servlet integrado como proporcionada.

**Convertir una aplicación existente a Spring Boot**
Reemplace el código que crea su ApplicationContexty reemplácelo con llamadas a SpringApplicationo SpringApplicationBuilder.
Las aplicaciones web Spring MVC generalmente se pueden crear primero una aplicación war implementable y luego migrarla más tarde a una guerra o jar ejecutable.
Las aplicaciones pueden pertenecer a más de una categoría:

- Aplicaciones Servlet 3.0+ sin web.xml.
- Aplicaciones con web.xml.
- Aplicaciones con jerarquía de contexto.
- Aplicaciones sin jerarquía de contexto.

**Implementación de un WAR en WebLogic**
El inicializacor de sevlet debe implementar directamente WebApplicationInitialize.
Indicar a WebLogic que prefiera la versión empaquetada en lugar de la versión preinstalada con el servidor.

# CONCLUSIONES

Cómo he ido indicando en el documente, he ido teniendo problemas a la hora de instalar glassfish y luego para probar las aplicaciones, finalmente solo pude probar la de calendar.
Una vez probada, me parece interesante ver en el navegador la aplicación desplegada, y hacerlo a través de un servidor de aplicaciones.

He usado el comando ufw para la configuración del firewall para permitir el acceso a GlassFish desde la red en el puerto 4848 TCP, y nunca lo había usado.

A pesar de seguir los pasos de un tutorial específico, se encontraron errores al intentar iniciar GlassFish, lo que me llevó a buscar y probar otro tutorial para resolver el problema.

Durante el despliegue de la aplicación helloworld.war, se encontraron problemas de visualización en el navegador, incluso después de aseguranos de que la aplicación estuviera desplegada y el dominio estuviera en ejecución. Lo que me llevó a la revisión de archivos de configuración y ajustes en los puertos de escucha.
El cambio en el puerto de escucha de GlassFish de 8080 a 8181 solucionó los problemas de visualización de la aplicación, con lo que podemos destacar la importancia de la configuración adecuada.

**Cosas aprendidas a destacar**

- La importancia de configurar el firewall adecuadamente para permitir el acceso a GlassFish desde la red.
- La necesidad de verificar y ajustar los puertos de escucha en GlassFish para evitar conflictos y asegurar la visualización de las aplicaciones desplegadas.
- Conocer la estructura, en este caso de Glassfish, sobre todo para tener localizado el archivo de configuración por si necesitamos hacer algún ajuste.