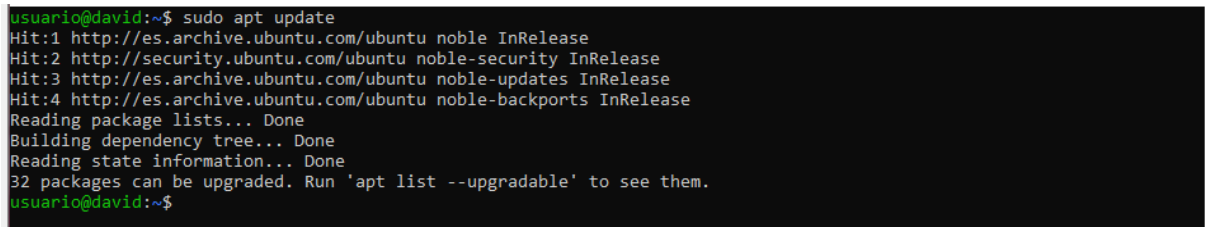
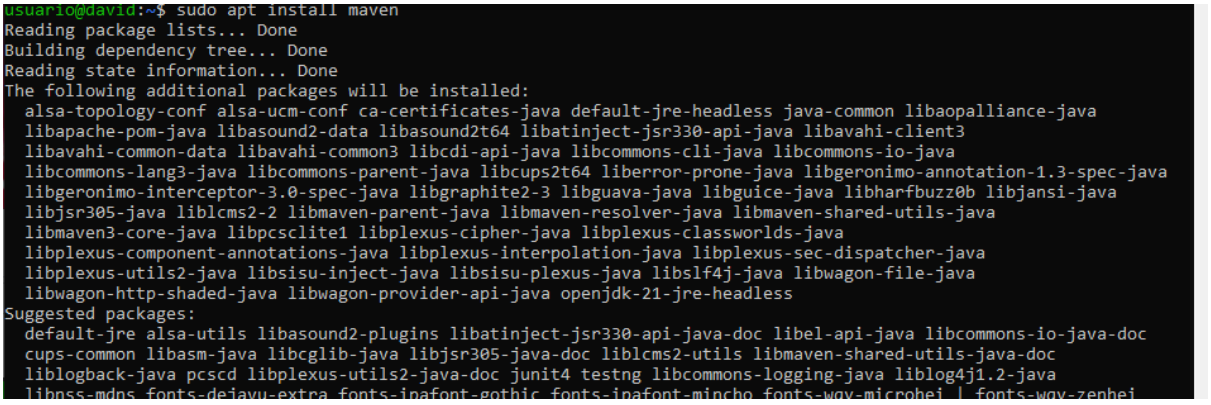
Instalación de Maven

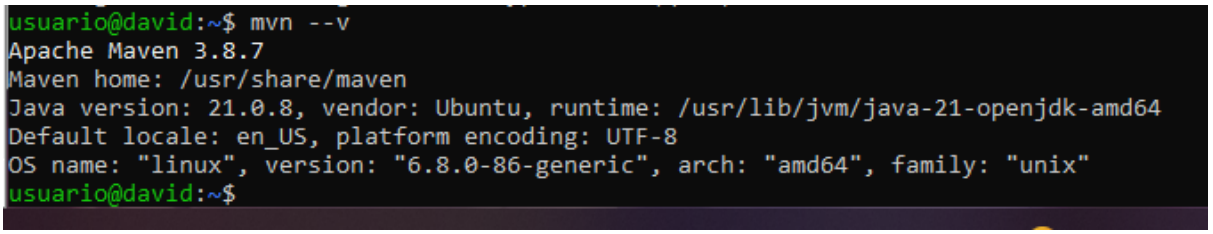
Comenzamos actualizando los repositorios:
sudo apt update



E instalamos Maven:
sudo apt install maven



Para comprobar que todo ha ido correctamente, podemos ver la versión instalada de Maven: mvn --v

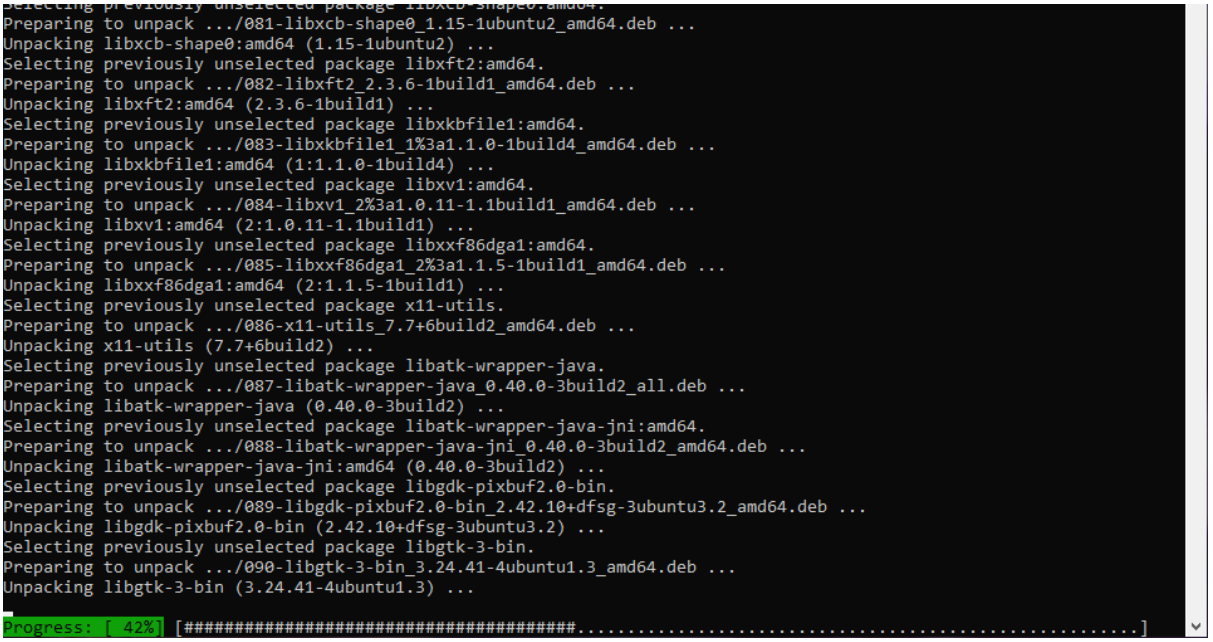


Instalación del servidor de aplicaciones WildFly

Instalar Java Development KIt (JDK)

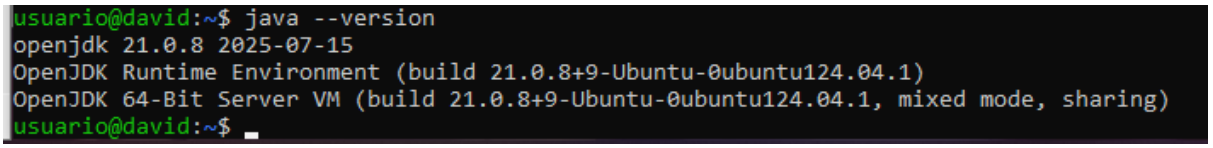
sudo apt update

sudo apt -y install default-jdk



WildFly soporta Java 17 y Java 11, podéis confirmar vuestra versión instalada así:

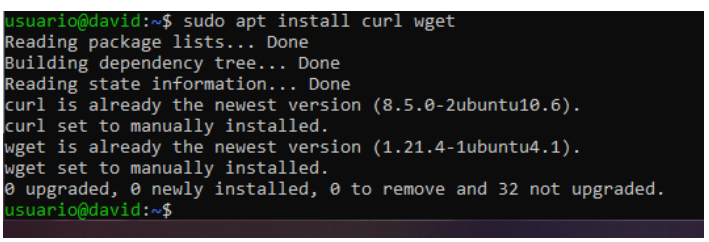
java --version



Descargar el tarball (archivo comprimido) de WildFly

utilizaremos las herramientas *curl* y *wget*. Si no las tuviéramos instaladas, primero lo haremos:

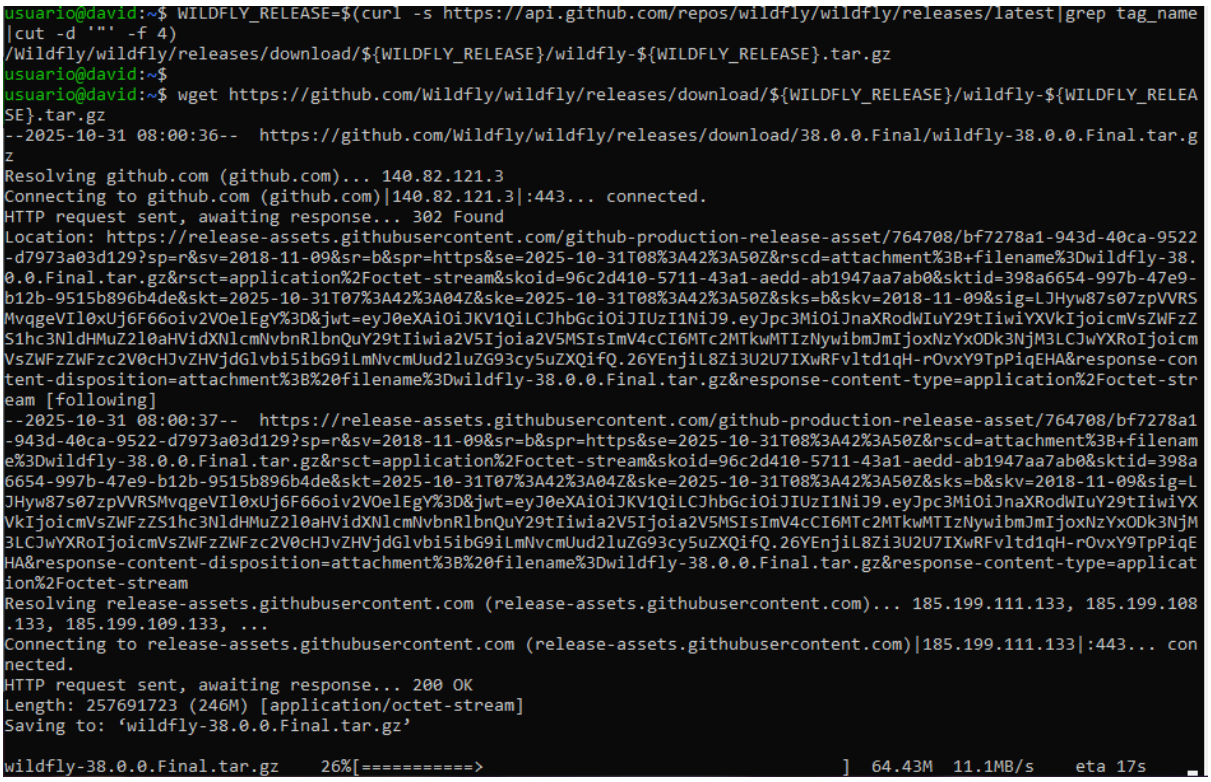
sudo apt install curl wget



Bajamos el archivo comprimido con la última versión de WildFly:

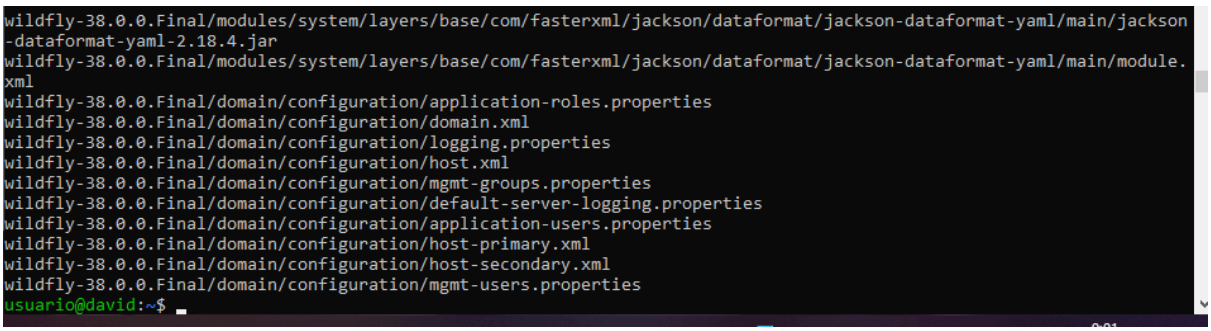
WILDFLY_RELEASE=$(curl -s https://api.github.com/repos/wildfly/wildfly/releases/latest|grep tag_name|cut -d "" -f 4)

wget https://github.com/Wildfly/wildfly/releases/download/${WILDFLY_RELEASE}/wildfly-${WILDFLY_RELEASE}.tar.gz



Descomprimimos:

tar xvf wildfly-${WILDFLY_RELEASE}.tar.gz



Movemos el contenido descomprimido al directorio /opt:

sudo mv wildfly-${WILDFLY_RELEASE} /opt/wildfly

```
usuario@david:~$ sudo mv wildfly-${WILDFLY_RELEASE} /opt/wildfly
usuario@david:~$
```

Configurar WildFly como un servicio más de Systemd

Para empezar añadido el usuario de sistema wildfly que es el que se encargará del servicio:

sudo groupadd --system wildfly

sudo useradd -s /sbin/nologin --system -d /opt/wildfly -g wildfly wildfly

```
usuario@david:~$ sudo groupadd --system wildfly
usuario@david:~$ sudo useradd -s /sbin/nologin --system -d /opt/wildfly  -g wildfly wildfly
usuario@david:~$
```

Creamos el directorio que contendrá las configuraciones de WildFly:

sudo mkdir /etc/wildfly

```
usuario@david:~$ sudo mkdir /etc/wildfly
```

Copiamos el archivo del servicio así como los scripts de inicio al directorio

sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf /etc/wildfly/

sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service /etc/systemd/system/

sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh /opt/wildfly/bin/

sudo chmod +x /opt/wildfly/bin/launch.sh

```
usuario@david:~$ sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf /etc/wildfly/
usuario@david:~$ sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service /etc/systemd/system/
usuario@david:~$ sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh /opt/wildfly/bin/
usuario@david:~$ sudo chmod +x /opt/wildfly/bin/launch.sh
usuario@david:~$
```

Le damos los permisos adecuados al directorio /opt/wildfly:

sudo chown -R wildfly:wildfly /opt/wildfly

```
usuario@david:~$ sudo chown -R wildfly:wildfly /opt/wildfly
usuario@david:~$
```

Recargamos los servicios de systemd (sistema que se encarga de manejar los servicios en nuestra Debian):

sudo systemctl daemon-reload

```
usuario@david:~$ sudo systemctl daemon-reload
usuario@david:~$
```

Iniciamos el servicio WildFly y lo habilitamos para que se inicie automáticamente al arrancar la máquina:

sudo systemctl start wildfly

```
usuario@david:~$ sudo systemctl start wildfly
usuario@david:~$
```

sudo systemctl enable wildfly

```
usuario@david:~$ sudo systemctl enable wildfly
Created symlink /etc/systemd/system/multi-user.target.wants/wildfly.service → /etc/systemd/system/wildfly.service.
usuario@david:~$
```

Comprobamos que se está ejecutando:

sudo systemctl status wildfly

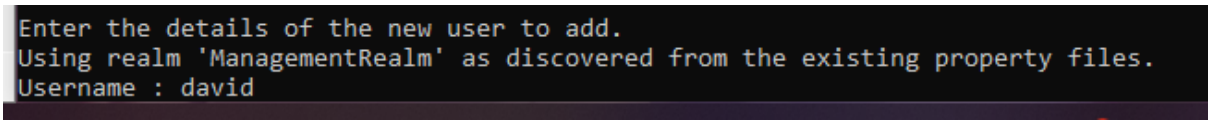
```
usuario@david:~$ sudo systemctl status wildfly
● wildfly.service - The WildFly Application Server
     Loaded: loaded (/etc/systemd/system/wildfly.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-10-31 08:12:10 UTC; 1min 2s ago
   Main PID: 6886 (launch.sh)
      Tasks: 64 (limit: 11882)
     Memory: 274.4M (peak: 284.3M)
        CPU: 9.970s
     CGroup: /system.slice/wildfly.service
             ├─6886 /bin/bash /opt/wildfly/bin/launch.sh standalone standalone.xml 0.0.0.0
             ├─6887 /bin/sh /opt/wildfly/bin/standalone.sh -c standalone.xml -b 0.0.0.0
             └─7008 java "-D[Standalone]" "-Djdk.serialFilter=maxbytes=10485760;maxdepth=128;maxarray=100000;maxrefs>

Oct 31 08:12:10 david systemd[1]: Started wildfly.service - The WildFly Application Server.
lines 1-13/13 (END)
```

WildFly nos proporciona un archivo para administrar los usuarios, así que en primer lugar y para crear un usuario:

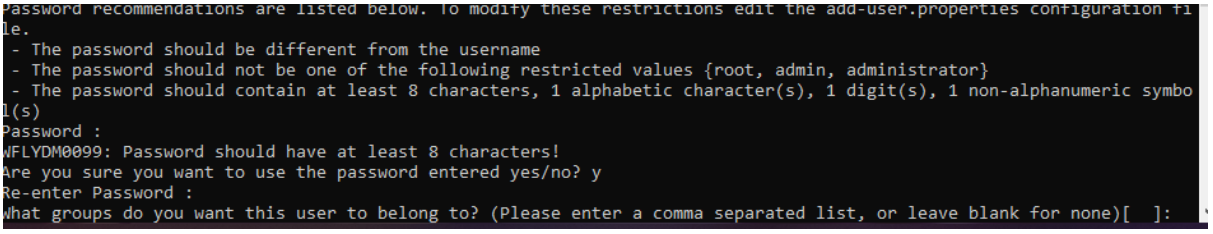
sudo /opt/wildfly/bin/add-user.sh

```
usuario@david:~$ sudo /opt/wildfly/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
```

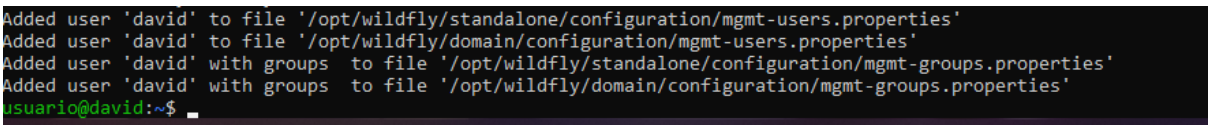
Ponemos el nombre de usuario



Ponemos la contraseña y le damos enter y luego yes



Vemos que se ha creado correctamente



Para ser capaces de ejecutar los scripts de WildFly desde nuestra sesión actual, añadimos el directorio /opt/wildfly/bin/ a nuestro PATH:

cat >> ~/.bashrc <<EOF

export WILDFLY_BIN="/opt/wildfly/bin/"

export PATH=\$PATH:\$WILDFLY_BIN

EOF

Y para no tener que reiniciar nuestra sesión actual SSH, aplicamos los cambios en la sesión actual así:

source ~/.bashrc

Y validamos que nos podemos conectar sin problemas:

jboss-cli.sh --connect

```
usuario@david:~$ jboss-cli.sh --connect
Authenticating against security realm: ManagementRealm
Username: david
Password:
[standalone@localhost:9990 /] exit
```

Accediendo a la consola de administración desde la interfaz web

Debemos modificar el archivo /opt/wildfly/bin/launch.sh/opt/wildfly/bin/launch.sh y dejarlo así:

```
  GNU nano 7.2                                /opt/wildfly/bin/launch.sh *
#!/bin/bash

if [ "x$wildfly_HOME" = "x" ]; then
    wildfly_HOME="/opt/wildfly"
fi

if [[ "$1" == "domain" ]]; then
    $wildfly_HOME/bin/domain.sh -c $2 -b $3
else
    $wildfly_HOME/bin/standalone.sh -c $2 -b $3 -bmanagement=0.0.0.0
fi
```

Reiniciamos el servicio tras el cambio:

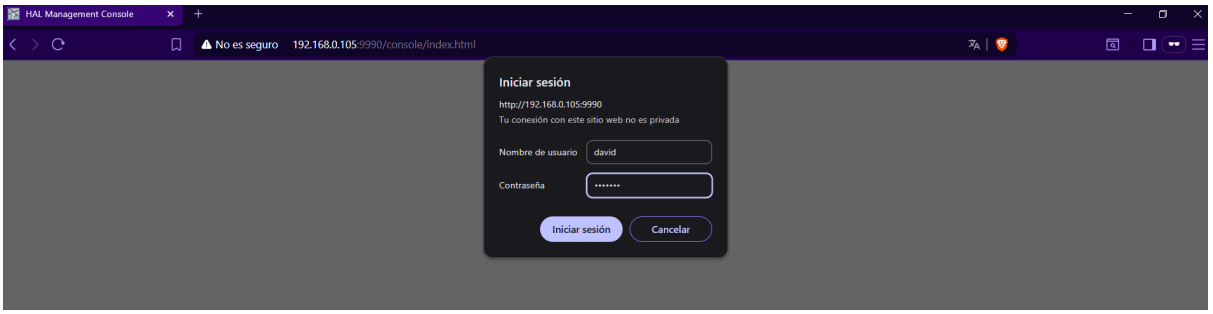
sudo systemctl restart wildfly

Comprobamos su estado:

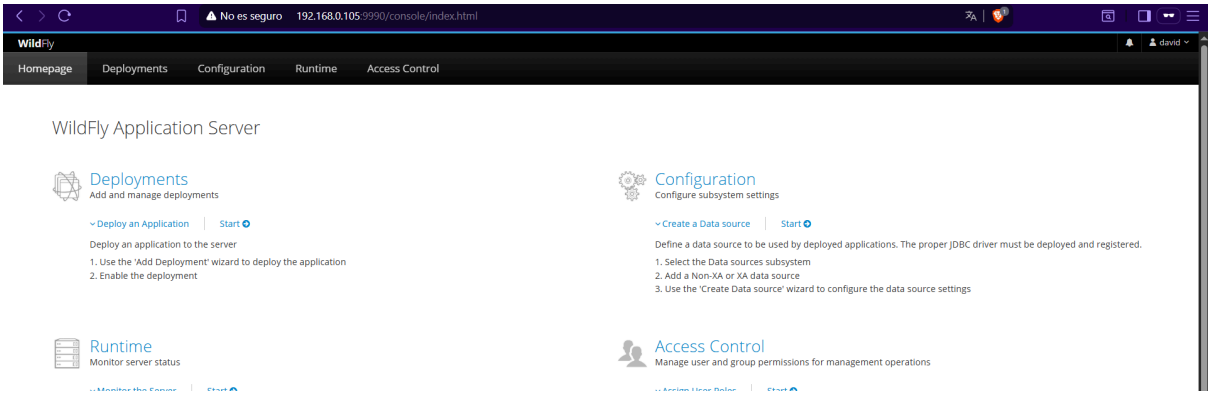
systemctl status wildfly

```
usuario@david:~$ sudo systemctl restart wildfly
usuario@david:~$ sudo systemctl status  wildfly
● wildfly.service - The WildFly Application Server
     Loaded: loaded (/etc/systemd/system/wildfly.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-10-31 08:23:20 UTC; 15s ago
   Main PID: 7453 (launch.sh)
      Tasks: 139 (limit: 11882)
     Memory: 275.5M (peak: 276.0M)
        CPU: 9.793s
     CGroup: /system.slice/wildfly.service
             ├─7453 /bin/bash /opt/wildfly/bin/launch.sh standalone standalone.xml 0.0.0.0
             ├─7454 /bin/sh /opt/wildfly/bin/standalone.sh -c standalone.xml -b 0.0.0.0 -bmanagement=0.0.0.0
             └─7582 java "-D[Standalone]" "-Djdk.serialFilter=maxbytes=10485760;maxdepth=128;maxarray=100000;maxrefs>

Oct 31 08:23:20 david systemd[1]: Started wildfly.service - The WildFly Application Server.
lines 1-13/13 (END)
```

Ponemos http://Nuestra_ip:9990

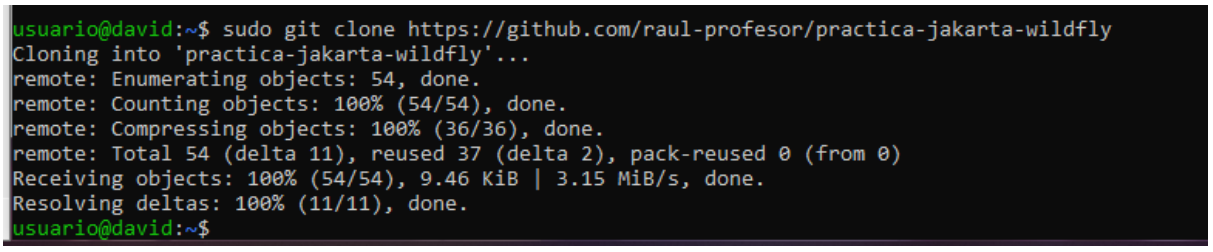


Aparecerá esto

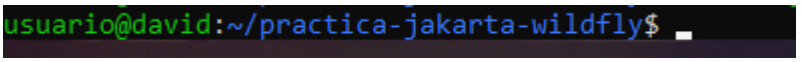


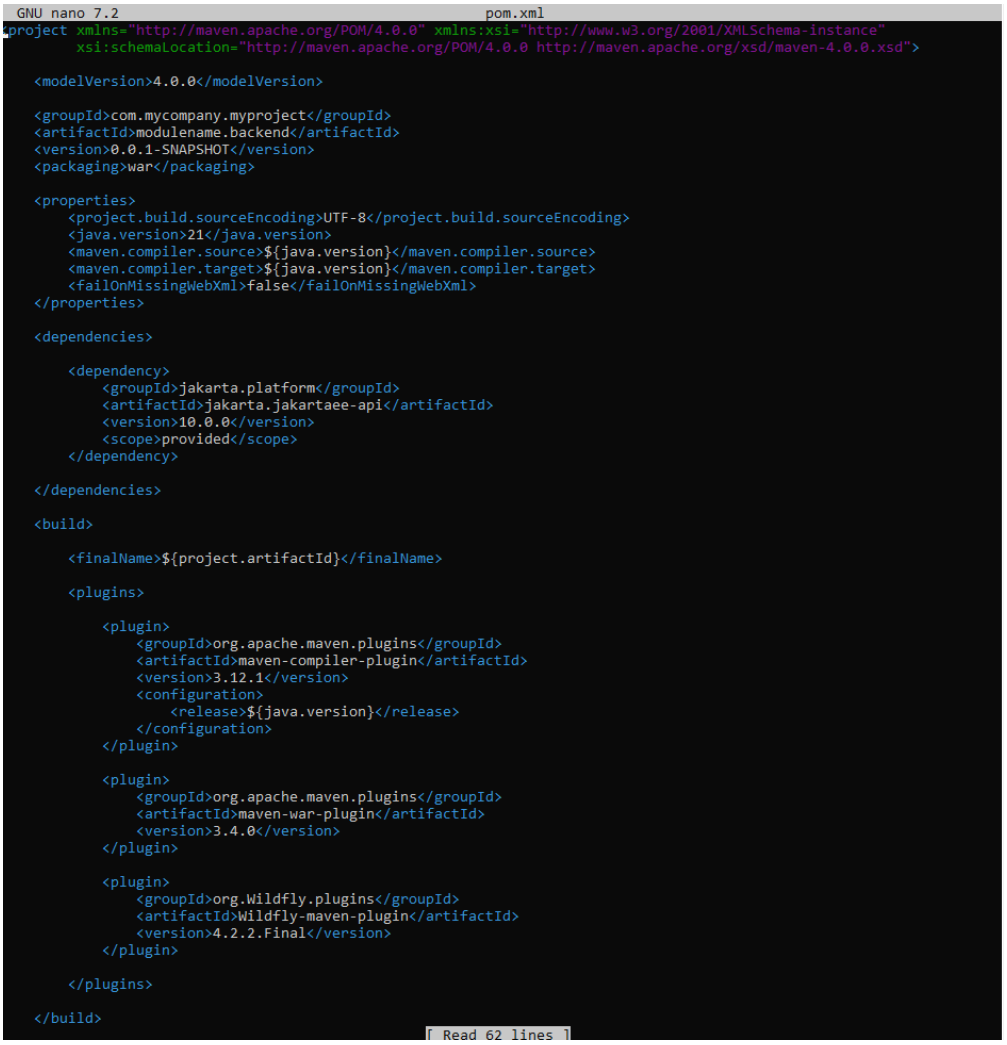
Clonamos el repositorio

sudo git clone **https://github.com/raul-profesor/practica-jakarta-wildfly**



Nos movemos aquí y creamos el pom.xml





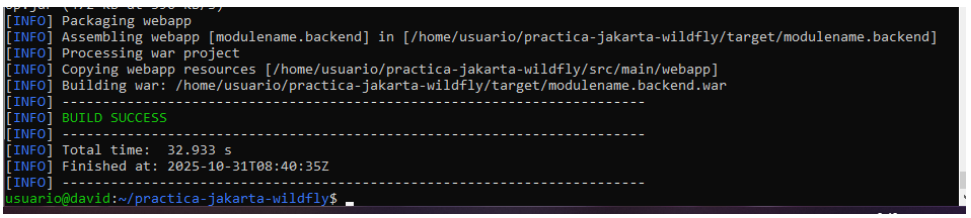

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany.myproject</groupId>
    <artifactId>modulename.backend</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>21</java.version>
        <maven.compiler.source>${java.version}</maven.compiler.source>
        <maven.compiler.target>${java.version}</maven.compiler.target>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </properties>

    <dependencies>

        <dependency>
            <groupId>jakarta.platform</groupId>
            <artifactId>jakarta.jakartaee-api</artifactId>
            <version>10.0.0</version>
            <scope>provided</scope>
        </dependency>

    </dependencies>

    <build>

        <finalName>${project.artifactId}</finalName>

        <plugins>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.12.1</version>
                <configuration>
                    <release>${java.version}</release>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.4.0</version>
            </plugin>

            <plugin>
                <groupId>org.Wildfly.plugins</groupId>
                <artifactId>Wildfly-maven-plugin</artifactId>
                <version>4.2.2.Final</version>
            </plugin>

        </plugins>

    </build>
```

Despliegue

Build

Para decirle a Maven que haga el *build* del archivo *war*, usaremos el siguiente comando, para limpiar restos de compilaciones anteriores y realizar la nueva de forma "limpia":

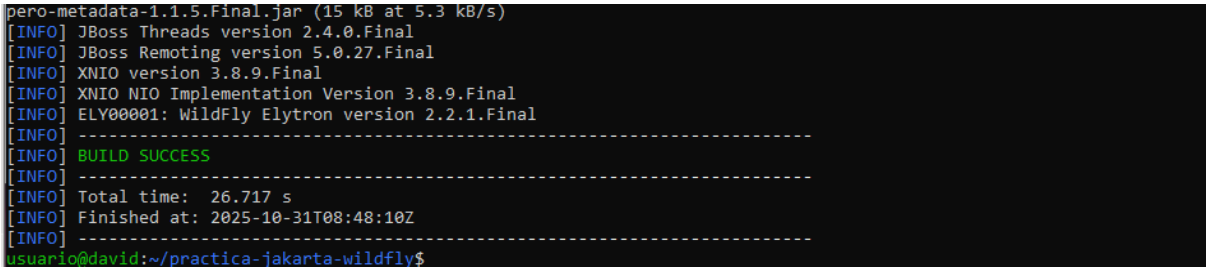
mvn clean package



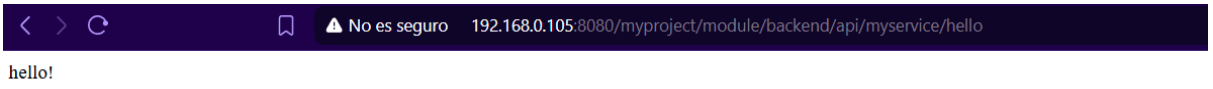
Despliegue en WildFly

Ya tenemos WildFly corriendo desde el principio de este proceso. Para realizar el despliegue, basta con meter el siguiente comando:

mvn wildfly::deploy

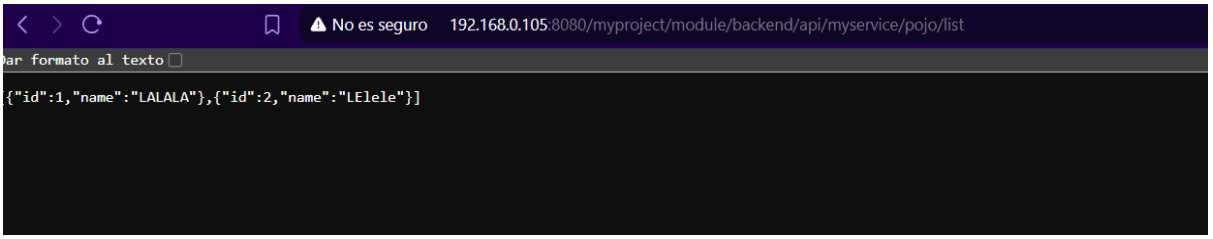


Ponemos http://192.168.0.105:8080/myproject/module/backend/api/myservice/hello



hello!

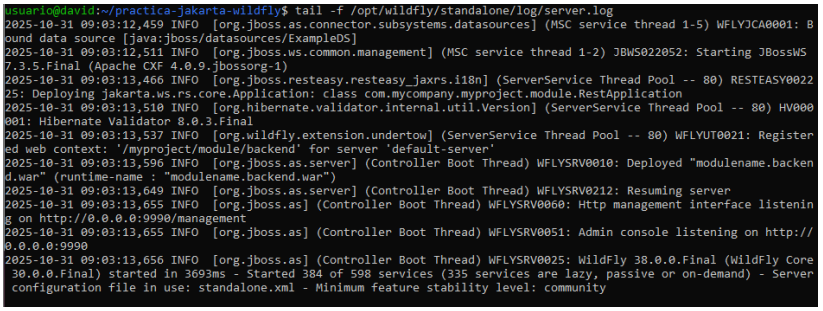
http://192.168.0.105:8080/myproject/module/backend/api/myservice/pojo/list



Comprobad que ocurre lo mismo con /pojo/find.

Ejecuta el siguiente comando para poder ver a tiempo real el log de Wildfly:

tail -f /opt/wildfly/standalone/log/server.log



Crear nueva entrada (pon tu nombre, no el que viene en este comando):

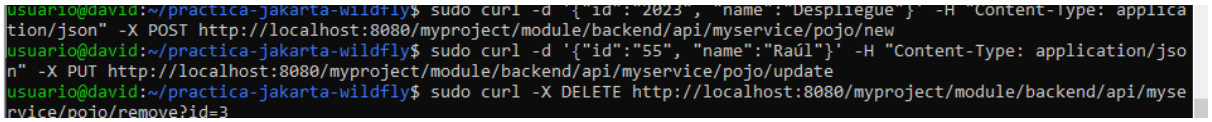
curl -d '{"id":"2023", "name":"Despliegue"}' -H "Content-Type: application/json" -X POST http://localhost:8080/myproject/module/backend/api/myservice/pojo/new

Actualizar una entrada (pon tu nombre, no el mío):

curl -d '{"id":"55", "name":"Raúl"}' -H "Content-Type: application/json" -X PUT http://localhost:8080/myproject/module/backend/api/myservice/pojo/update

Eliminar una entrada

curl -X DELETE http://localhost:8080/myproject/module/backend/api/myservice/pojo/remove?id=3



Muestra las entradas de los logs que se corresponden con estas peticiones