Instalación de Ambiente de Desarrollo

Iliac Huerta Trujillo

1 de octubre de 2020

Resumen

Al presente documento tiene como objetivo servir como guía de instalación para el ambiente de desarrollo del proyecto "Sistema de Monitoreo del Estado de Salud de Pacientes COVID-19 en Hospitales", la guía es perfectible, cualquier comentario útil para mejorarla será bienvenido.

1. Requerimientos iniciales

Es necesario tener instalado Java JDK version 14, pueden descargarlo de la liga:
https://www.oracle.com/java/technologies/javase/jdk14-archive-downloads.html
Wildfly 20 mismo que pueden descargar de :
https://download.jboss.org/wildfly/20.0.1.Final/wildfly-20.0.1.Final.zip
El ID recomendado es NetBeans 12 que pueden descargarlo de: https://netbeans.apache.org/
Aunque cualquier otro IDE que maneje Maven es suficiente.
Declarar las variables de entorno con las rutas relativas a su instalación, en sistemas windows se recomienda instalar el servidor de aplicaciones (wildfly) en el directorio raíz, las variables a declarar son:

- JAVA_HOME
- JBOSS_HOME

Instalar MySql (8), deberá crear un esquema de base de datos y un usuario para el proyecto.
El esquema se deberá llamar **sismhosp**, Seguido de la creación del esquema mencionado, es necesario crear un usuario asociado a ese esquema a fin de evitar el uso del usuario root de mysql, el usuario sugerido se deberá llamar **sismhospdba**, una vez creado el usuario es necesario dar privilegios para crear modificar y eliminar tablas al esquema.
Con el usuario creado, conectarse a la base de datos (se recomienda usar Workbench) de ese esquema y se deberán ejecutar el script de creación de base de datos, el archivo es:
creaBDMHospitalmysql.sql.
Así como el script de carga inicial:
cargaInicialBDRMmysql.sql
Con esto tenemos instalada la base de datos, el esquema, el usuario y tablas de la aplicación.

2. Servidor de aplicaciones

El servidor de aplicaciones(SA) a utilizar es Wildfly version 20, una versión estable y probada.
Es muy importante aclarar que el sustituir, agregar o alterar de forma incorrecta lo siguiente en el documento, puede afectar al SA de forma irreparable, por lo que se debe hacer con sumo cuidado la configuración que se menciona a continuación.

Lo primero es agregar al SA el módulo para trabajar con MySql, esto se hace usando el archivo jar del driver de la base de datos ***mysql-connector-java-8.0.21.jar***, para eso se realizará lo siguiente:
Crear la estructura de ditectorios siguiente:

/WILDFLY_HOME/modules/system/layers/base/com/mysql/main

Copiar el arcjivo jar, dentro del directorio *main* y crear el archivo **module.xml** con el siguiente contenido:

```
<module xmlns="urn:jboss:module:1.5" name="com.mysql">
    <resources>
        <resource-root path="mysql-connector-java-8.0.21.jar" />
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

Lo que instala el módulo al SA.
Es necesario realizar modificaciones al archivo ***standalone.xml*** que se encuentra en:
/WILDFLY_HOME/standalone/configuration/standalone.xml.
Buscar el tag <**divers**> y agregar las lineas para que el tag quede de la siguiente forma:

```
<drivers>
    <driver name="h2" module="com.h2database.h2">
        <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
    </driver>
    <driver name="mysql" module="com.mysql">
        <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
        <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
    </driver>
</drivers>
```

Buscar el tag <**datasources**> y agregar el siguiente nodo:

```
<datasource jndi-name="java:jboss/sismhospDS" pool-name="sismhospDS" enabled="true"
                    use-java-context="true" use-ccm="true">
        <connection-url>jdbc:mysql://localhost:3306/sismhosp?serverTimezone=UTC</connection-url>
    <driver>mysql</driver>
    <security>
            <user-name>sismhospdba</user-name>
        <password>12345678</password>
    </security>
    <validation>
        <valid-connection-checker
        class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
         <background-validation>true</background-validation>
         <exception-sorter
```

```
          class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
      </validation>
</datasource>
```

donde sismhospdba es el usuario de la base de datos que tiene acceso al esquema creado para el proyecto, el password utilizado solo es para fines didácticos, deberán asignar el que definieron para el usuario de base de datos.

3. Configuración de seguridad

La definición de seguridad se hace en el subsistema Elytron, deberán buscar la línea

```
<subsystem xmlns="urn:wildfly:elytron:10.0"
                final-providers="combined-providers"
                disallowed-providers="OracleUcrypto">
```

Dentro del tag <**security-domains**> agregar el siguiente nodo:

```
<security-domain name="sistmhospDomain" default-realm="sistmhospRealm"
                permission-mapper="default-permission-mapper">
    <realm name="sistmhospRealm" role-decoder="from-roles-attribute"/>
</security-domain>
```

Siguiendo en el mismo subsistema, dentro del tag <**security-realms**>, agregar el nodo:

```
<jdbc-realm name="sistmhospRealm">
        <principal-query sql="SELECT CONTRASENIA FROM MH_USUARIO
                                    WHERE ID_USUARIO = ? AND ACTIVO = true"
                         data-source="sismhospDS">
                <clear-password-mapper password-index="1"/>
    </principal-query>
    <principal-query sql="SELECT rol.DESCRIPCION, 'Roles'
                            FROM MH_USUARIO_ROL ur INNER JOIN MH_ROL rol ON ur.ID_ROL = rol.ID_ROL
                            INNER JOIN MH_USUARIO usu ON usu.ID_USUARIO = ur.ID_USUARIO
                            WHERE usu.ID_USUARIO= ?" data-source="sismhospDS">
            <attribute-mapping>
                <attribute to="roles" index="1"/>
        </attribute-mapping>
   </principal-query>
</jdbc-realm>
```

Que define la manera de obtener el password para un usuario y el o los roles asociados al mismo.
Inmediatamente después del nodo <**security-realms**>, se encuentra el nodo <**mappers**>, (Siempre y cuando no exista) dentro agregar el nodo:

```
<simple-role-decoder name="from-roles-attribute" attribute="roles"/>
```

Esto se puede hacer inmediatamente después del tag <**simple-role-decoder** /> que se encuentra ya definido en mappers.
Seguido de mappers, se encuentra el nodo <**http**> dentro de ese nodo, agregar el siguiente nodo:

```
<http-authentication-factory name="sismhosp-db-http-auth"
        security-domain="sistmhospDomain"
        http-server-mechanism-factory="global">
        <mechanism-configuration>
            <mechanism mechanism-name="FORM">
             <mechanism-realm realm-name="sistmhospRealm"/>
        </mechanism>
    </mechanism-configuration>
</http-authentication-factory>
```

Una vez definido el método de autenticación HTTP, bucar el subsistema **urn:jboss:domain:undertow:11.0**, dentro del tag que limita al subsistema **undertow**, se encuentra el nodo <**handlers**> inmediatamente después del nodo de cierre (</**handlers**>) es necesario agregar el siguiente nodo:

```
<application-security-domains>
    <application-security-domain name="sismhospApplicationDomain"
                            http-authentication-factory="sismhosp-db-http-auth"/>
</application-security-domains>
```

De ya existir el nodo <**application-security-domains**>, sólo agregar el nodo interno **application-security-domain**.
Por último se debe iniciar el servidor de aplicaciones, esto se hace ejecutando el archivo ***stanalone.bat*** (en caso de Windows) o ***standalone.sh*** (en caso de linux)
El servidor deberá iniciar sin ningún problema, en caso de presentarse, verifique que la configuración se encuentre bien realizada.