

# CREACIÓN DE APLICACIONES WEB SEGURAS EN J2EE CON WILDFLY

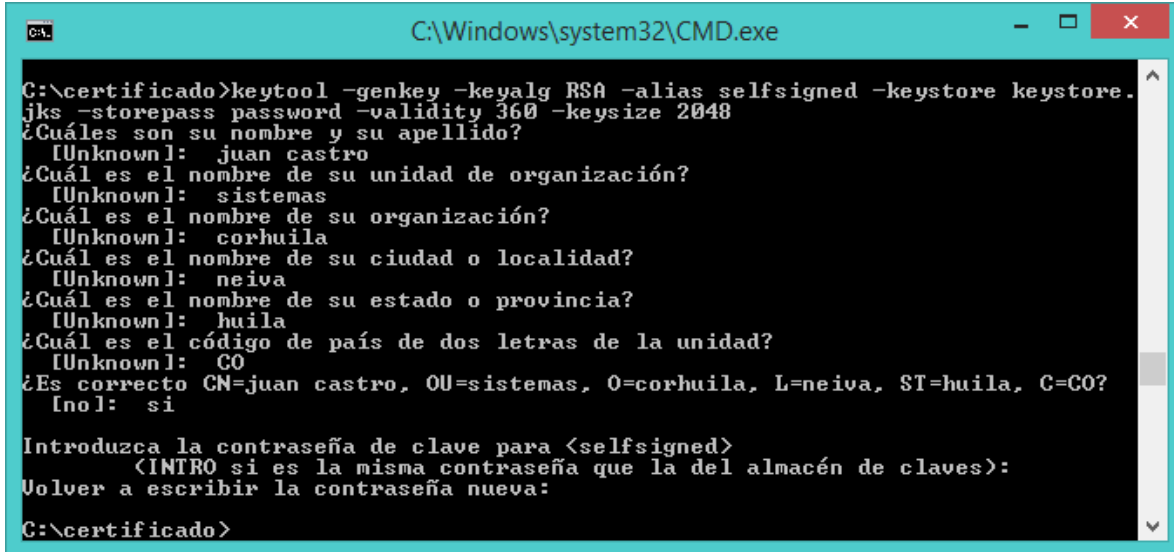
Profesor: Ing. Juan Antonio Castro Silva

Version: 2.0 (11-MAYO-2016-06:52)

## 1. GENERAR UN CERTIFICADO DIGITAL AUTOFIRMADO

Para generar un certificado digital autofirmado utilizamos la herramienta keytool de Java. Cree una carpeta llamada (/certificado) para almacenar el archivo con el certificado que se va a generar keystore.js, ingrese a la carpeta creada, ejecute el siguiente comando en una ventana de DOS (o terminal en Linux), para este ejemplo se debe utilizar como clave la palabra password (luego la puede cambiar).

```
keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks  
-storepass password -validity 360 -keysize 2048
```



```
C:\Windows\system32\CMD.exe  
C:\certificado>keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.  
jks -storepass password -validity 360 -keysize 2048  
¿Cuáles son su nombre y su apellido?  
[Unknown]: juan castro  
¿Cuál es el nombre de su unidad de organización?  
[Unknown]: sistemas  
¿Cuál es el nombre de su organización?  
[Unknown]: corhuila  
¿Cuál es el nombre de su ciudad o localidad?  
[Unknown]: neiva  
¿Cuál es el nombre de su estado o provincia?  
[Unknown]: huila  
¿Cuál es el código de país de dos letras de la unidad?  
[Unknown]: CO  
¿Es correcto CN=juan castro, OU=sistemas, O=corhuila, L=neiva, ST=huila, C=CO?  
[no]: si  
Introduzca la contraseña de clave para <selfsigned>  
(INTRO si es la misma contraseña que la del almacén de claves):  
Volver a escribir la contraseña nueva:  
C:\certificado>
```

Copie el archivo keystore.jks y péguelo dentro de la carpeta de configuración del wildfly (/wildfly/standalone/configuration/).

## 2. CONFIGURAR EL SERVIDOR WILDFLY PARA UTILIZAR HTTPS

Para activar el protocolo https en el servidor de aplicaciones wildfly, modificamos el archivo de configuración del wildfly (/wildfly/standalone/configuration/standalone.xml) y adicionamos el bloque del `<security-realm name="SecureRealm">` antes de cerrar el bloque `</security-realms>`:

```
<security-realm name="SecureRealm">
  <server-identities>
    <ssl>
      <keystore path="keystore.jks" relative-to="jboss.server.config.dir"
        keystore-password="password"/>
    </ssl>
  </server-identities>
</security-realm>
```

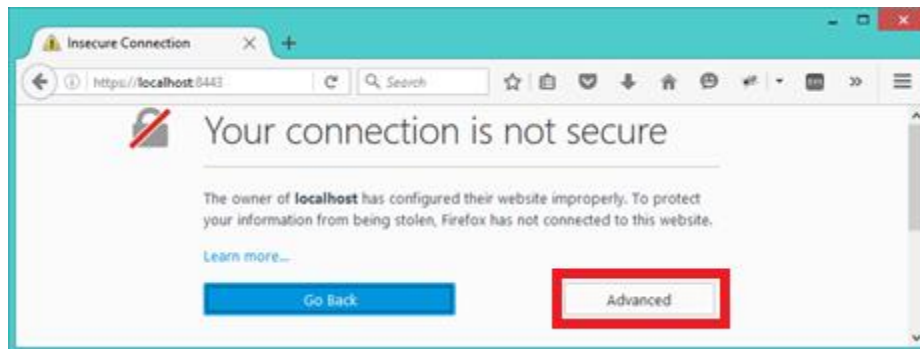
Insertamos la línea del `<https-listener...>` para activar el protocolo https.

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="default" socket-binding="http"
      redirect-socket="https"/>

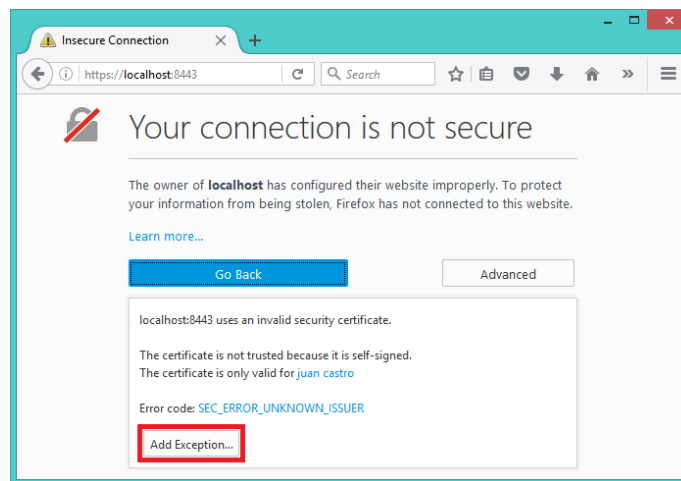
    <https-listener name="https" socket-binding="https"
      security-realm="SecureRealm"/>

    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content"/>
      <filter-ref name="server-header"/>
      <filter-ref name="x-powered-by-header"/>
    </host>
  </server>
</subsystem>
```

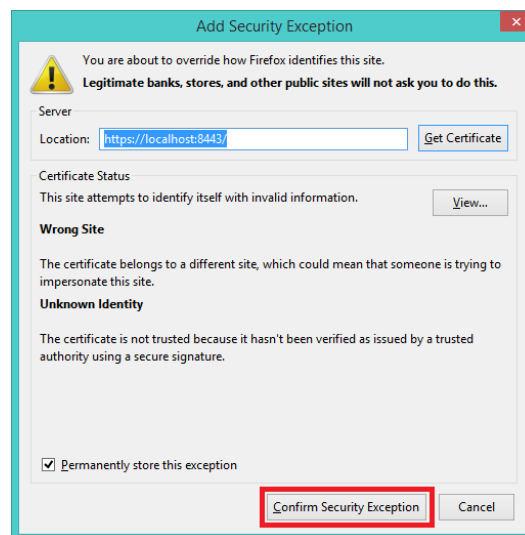
Para probar el protocolo https, corra el servidor de aplicaciones wildfly (standalone.bat o standalone.sh), abra un navegador y digite la dirección del servidor (<https://localhost:8443/>).



Haga click en el boton [Advanced] y añada la excepción [Add Exception].



Finalmente confirme la excepción de seguridad.



Verifique el protocolo https y el puerto 8443 en la url.



De esta manera la información que sea transmitida estará cifrada.

### 3. AUTORIZACION Y AUTENTICACION

Para la creación de aplicaciones seguras con el servidor de aplicaciones wildfly vamos a utilizar JAAS (Java Authentication and Authorization Service).

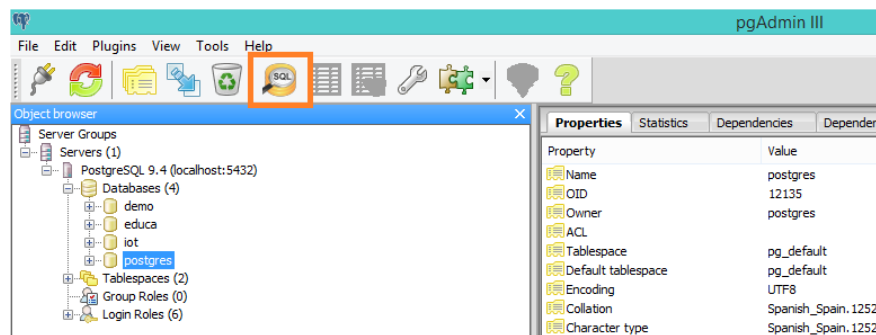
Para implementar el mecanismo de seguridad vamos a crear una aplicación web y a seguir los siguientes pasos:

1. Creación de la Base de Datos.
2. Configuración del Servidor de Aplicaciones – J2EE – WildFly.
3. Configuración de la Aplicación Web.

#### 3.1 CREACION DE LA BASE DE DATOS - POSTGRESQL

##### 3.1.1 Crear un login role en Postgresql:

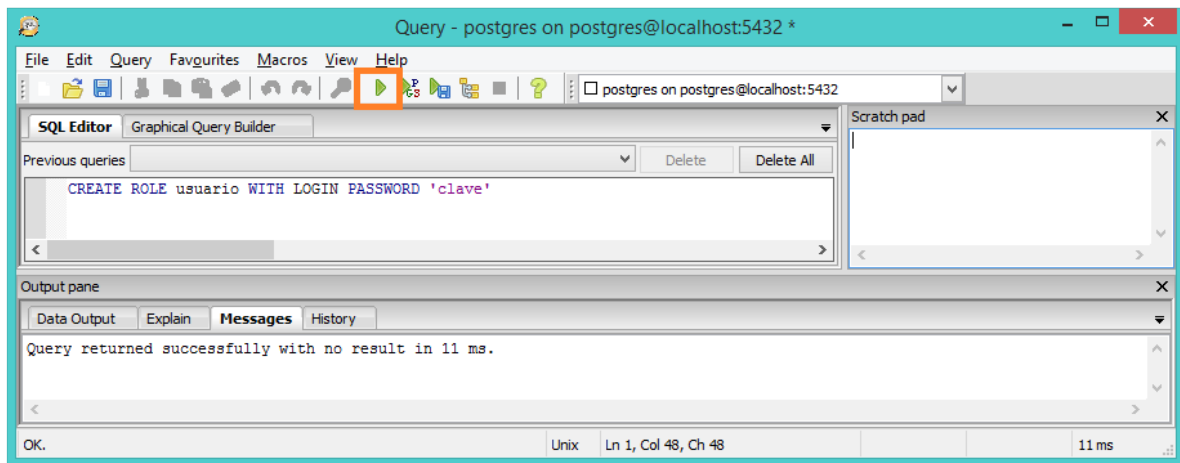
Abra la ventana de ejecución de consultas SQL:



La siguiente sentencia SQL permite crear un nuevo role (usuario), que será el propietario de la base de datos:

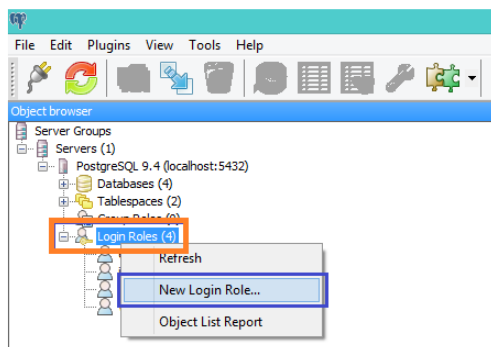
```
CREATE ROLE web201 WITH LOGIN PASSWORD 'web201';
```

Escriba la consulta SQL en el editor y haga click en el boton [Ejecutar], arriba en la barra de herramientas.

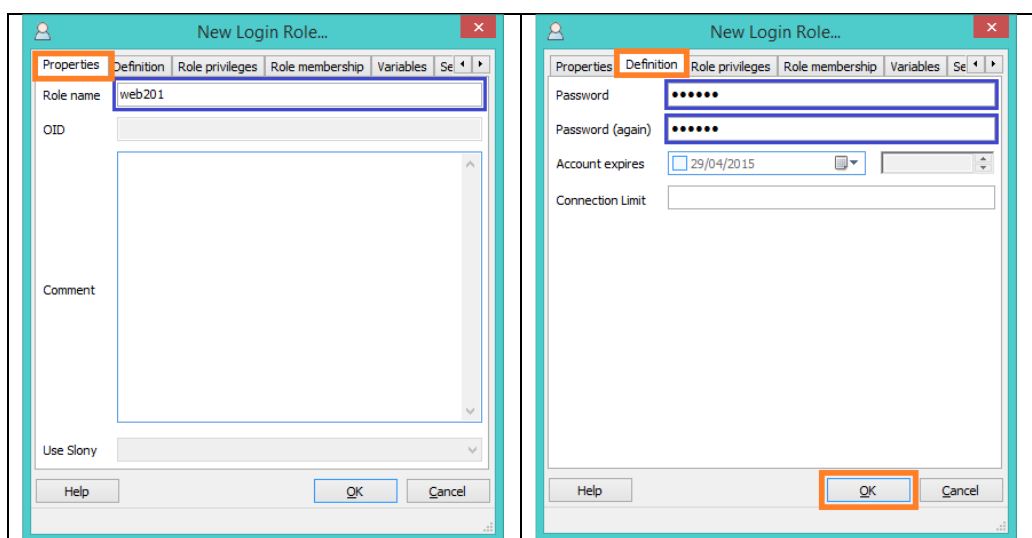


También es posible utilizar el editor gráfico como se muestra a continuación:

Haga click con el boton derecho del mouse en la opción de Login Roles y seleccione el ítem [New Login Role].



En la ventana de New Login Role, en la ficha de Properties (propiedades) digite el Role name (nombre del rol), seleccione la ficha de Definition (definición), digite el password (la clave) y confírmelo, finalmente haga click en el boton [OK].

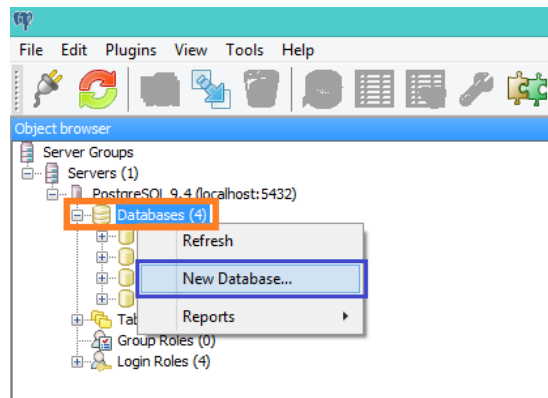


### 3.1.2 Crear la base de datos (web201):

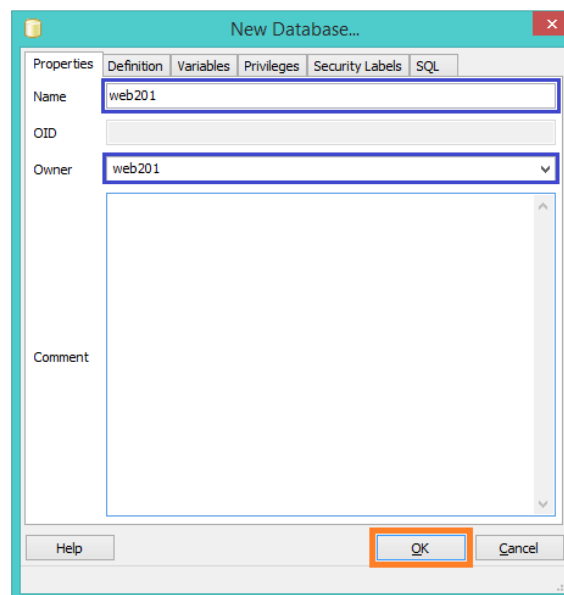
Para crear la base de datos ejecute la siguiente sentencia en el editor de SQL:

```
CREATE DATABASE web201
WITH OWNER = web201
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'es_CO.UTF-8'
LC_CTYPE = 'es_CO.UTF-8'
CONNECTION LIMIT = -1;
```

Si lo prefiere hacer por el entorno gráfico, haga click en la opción Databases y seleccione el ítem [New Database...].



En la ventana de New Database, digite el nombre de la base de datos (web201) y seleccione como Owner (propietario) a web201, finalmente haga click en el boton de [OK].





### 3.1.3 Crear las tablas:

Para crear las tablas (grupo, usuario y usuario\_grupo) ejecute las siguientes sentencias en el editor de SQL:

#### Tabla grupo

```
CREATE TABLE grupo
(
  gru_codigo serial NOT NULL,
  gru_nombre character varying (100),
  gru_id character varying (100),
  CONSTRAINT grupo_pkey PRIMARY KEY (gru_codigo)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE grupo
OWNER TO web201;
```

#### Tabla usuario

```
CREATE TABLE usuario
(
  usu_codigo bigserial NOT NULL,
  usu_nombre character varying (50),
  usu_login character varying (20),
  usu_password character varying (64),
  CONSTRAINT usuario_pkey PRIMARY KEY (usu_codigo)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE usuario
OWNER TO web201;
```

#### Tabla usuario\_grupo

```
CREATE TABLE usuario_grupo
(
  usg_codigo bigserial NOT NULL,
  usg_usuario bigint,
  usg_grupo bigint,
  CONSTRAINT usuario_grupo_pkey PRIMARY KEY (usg_codigo),
  CONSTRAINT usuario_grupo_usg_grupo_fkey FOREIGN KEY (usg_grupo)
REFERENCES grupo (gru_codigo) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT usuario_grupo_usg_usuario_fkey FOREIGN KEY (usg_usuario)
REFERENCES usuario (usu_codigo) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE usuario_grupo
OWNER TO web201;
```

### 3.1.4 Alimentar la base de datos:

Ejecutar en el editor de SQL las siguientes consultas (queries).

```
-- datos de la tabla grupo
insert into grupo (gru_nombre, gru_id)
values ('Administrador', 'ADMINISTRADOR');
insert into grupo (gru_nombre, gru_id)
values ('Cliente', 'CLIENTE');
```

```
-- datos de la tabla usuario
insert into usuario (usu_nombre, usu_login, usu_password)
values ('Pedro Perez', 'pedro', 'perez');
insert into usuario (usu_nombre, usu_login, usu_password)
values ('Maria Cardenas', 'maria', 'cardenas');
insert into usuario (usu_nombre, usu_login, usu_password)
values ('Jose Lopez', 'jose', 'lopez');
```

```
-- datos de la tabla usuario_grupo
-- se asumen los siguientes valores:
-- usuarios 1=pedro, 2=maria, 3=jose
y para los grupos 1=ADMINISTRADOR y 2=CLIENTE
insert into usuario_grupo (usg_usuario, usg_grupo)
values (1, 1);
insert into usuario_grupo (usg_usuario, usg_grupo)
values (2, 2);
insert into usuario_grupo (usg_usuario, usg_grupo)
values (3, 1);
insert into usuario_grupo (usg_usuario, usg_grupo)
values (3, 2);
```

## 4. CRIPTOGRAFIA

Para activar la librería pgcrypto en el motor de base de datos Postgresql, ejecutamos el siguiente comando en la ventana de SQL.

```
CREATE EXTENSION pgcrypto;
```

Para encriptar una clave con el algoritmo md5 y mostrar el resultado con el encode base64:

```
SELECT encode(digest('clave a encriptar', 'md5'), 'base64');
```

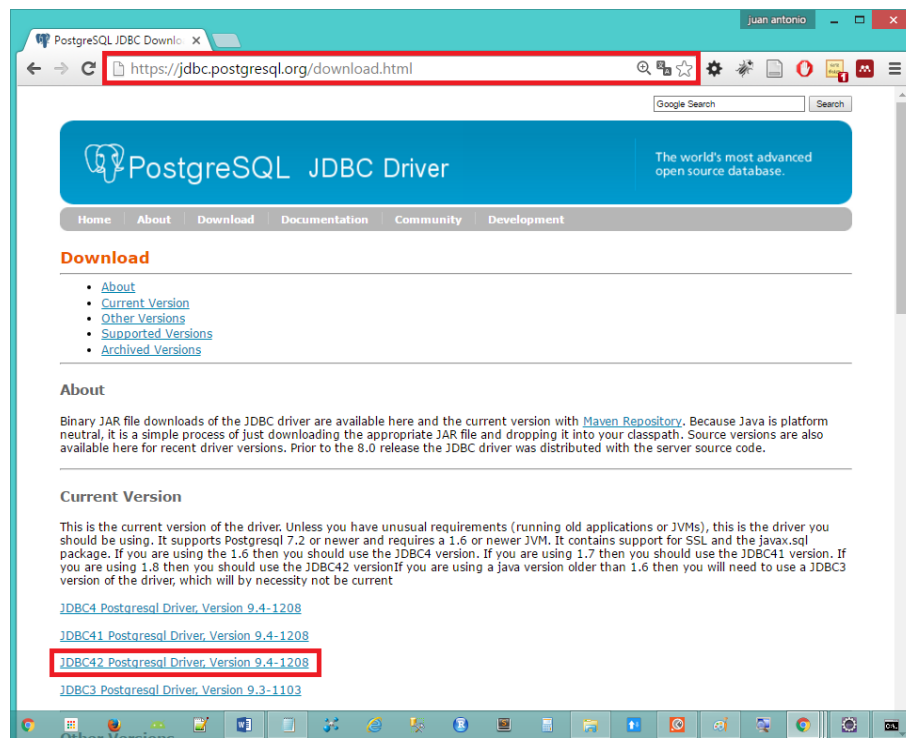
Actualizamos las claves de los usuarios:

```
UPDATE usuario set usu_password = encode(digest('perez', 'md5'),  
'base64') where usu_login = 'pedro';  
UPDATE usuario set usu_password = encode(digest('cardenas', 'md5'),  
'base64') where usu_login = 'maria';  
UPDATE usuario set usu_password = encode(digest('lopez', 'md5'),  
'base64') where usu_login = 'jose';
```

## 5. CONFIGURACION DEL SERVIDOR DE APLICACIONES WILDFLY

### 5.1 INSTALACION DEL DRIVER JDBC

En este proyecto para acceder a la base de datos (postgresql) debe instalar el driver JDBC, lo puede descargar del sitio web (<https://jdbc.postgresql.org/download.html>). En esta página web se advierte que si se utiliza JDK 1.7 se debe usar el Driver versión JDBC41, y se usa el JDK 1.8 se debe utilizar el Driver versión JDBC42.



Copie el driver JDBC (postgresql-9.4-1208.jar) en la carpeta de publicaciones del servidor de aplicaciones (/wildfly/standalone/deployments/).

### 5.2 CREAR UN DATASOURCE

En el archivo de configuración del wildfly /standalone/configuration/standalone.xml en la sección de <datasources> antes de la etiqueta <drivers>, pegue el nuevo datasource, el cual indica en la url el motor de la base de datos (postgresql), la dirección ip (127.0.0.1) , el puerto (5432) y el nombre de la base de datos (web201). Muestra de igual forma el driver que se va a utilizar para la conexión a la base de datos (**postgresql-9.4-1208.jar**) y en el nodo de seguridad se especifica el usuario y la clave.

```
<datasource jta="false" jndi-name="java:/web201DS" pool-name="web201DS"
enabled="true" use-ccm="false">
```

```

<connection-url>jdbc:postgresql://127.0.0.1:5432/web201</connection-
url>
<driver-class>org.postgresql.Driver</driver-class>
<driver>postgresql-9.4-1208.jar</driver>
<security>
<user-name>web201</user-name>
<password>web201</password>
</security>
</datasource>

```

## 5.3 CREAR EL DOMINIO DE SEGURIDAD

En la sección de `<security-domains>` antes de la etiqueta de cierre `</security-domains>` adicione el dominio de seguridad, el cual define el nombre (seguridadWeb201), la información de la conexión a la base de datos (java:/web201DS), la sentencia SQL que retorna el password del usuario (principalsQuery) y los grupos- perfiles (rolesQuery):

```

<security-domain name="seguridadWeb201">
  <authentication>
    <login-module code="Database" flag="required">
      <module-option name="dsJndiName" value="java:/web201DS"/>
      <module-option name="principalsQuery"
        value="select usu_password from usuario where usu_login=?"/>
      <module-option name="rolesQuery"
        value="SELECT grupo.gru_id, 'Roles' FROM public.usuario,
        public.usuario_grupo, public.grupo
        WHERE usuario.usu_codigo = usuario_grupo.usg_usuario
        AND grupo.gru_codigo = usuario_grupo.usg_grupo
        AND usu_login=?"/>
      <module-option name="hashAlgorithm" value="MD5"/>
      <module-option name="hashEncoding" value="base64"/>
    </login-module>
  </authentication>
</security-domain>

```

## 6. CONFIGURACION DE LA APLICACION WEB

### 6.1 CREAR LAS CARPETAS – FOLDERS

Ahora creamos una aplicación web (wildfly/standalone/deployments/web201.war) con la siguiente estructura de carpetas y archivos.

Carpeta – Folder o Archivo	Role – Grupo o Perfil
/pedido/	CLIENTE
/admin/	ADMINISTRADOR
/producto/	ADMINISTRADOR
/usuario/	ADMINISTRADOR, CLIENTE
/tienda/	RECURSO PUBLICO – NO ASEGURADO
/index.jsp	RECURSO PUBLICO – NO ASEGURADO
/login.jsp	RECURSO PUBLICO – NO ASEGURADO
/error.jsp	RECURSO PUBLICO – NO ASEGURADO
/salir.jsp	RECURSO PUBLICO – NO ASEGURADO

### 6.2 CREAR LOS ARCHIVOS

>> Cree el archivo /index.jsp con el siguiente contenido:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sitio Seguro :: Bienvenido</title>
</head>
<body>
<h2>Bienvenido</h2>
<p><a href="/usuario">Login</a></p>
</body>
</html>
```

Aquí se hace un enlace a la carpeta /usuario, la cual está protegida (solo para usuarios con el perfil ADMINISTRADOR o CLIENTE) y de esta manera se activará el mecanismo de seguridad (dominio) que obliga al usuario a loguearse.

>> Cree el archivo de /login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sitio Seguro :: Login</title>
</head>
<body>
<h2>Ingresar al Sistema</h2>
<form action="j_security_check" method="post">
<p>Usuario: <input type="text" name="j_username"></p>
<p>Clave: <input type="password" name="j_password"></p>
<p><input type="submit" value="Ingresar al Sistema">
</form>
</body>
</html>
```

En este archivo se crea un formulario de login que enviará los parámetros j\_username y j\_password con los datos del usuario (usuario y clave respectivamente), a un Servlet del Servicio de Autenticación y Autorización de Java (JAAS) llamado j\_security\_check.

>> Cree el archivo /error.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sitio Seguro :: Error</title>
</head>
<body>
<h2>Usuario o Clave Errada</h2>
</body>
</html>
```

Aquí se muestra al usuario un mensaje cuando se comete un error al digitar el usuario o la clave, este recurso es lanzado automáticamente por el mecanismo de seguridad.

>> Cree el archivo /usuario/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sitio Seguro :: Usuario</title>
</head>
<body>
<h2>Módulo para usuarios</h2>
<p><a href="../salir.jsp">Cerrar Sesión</a></p>
<%
String usuario = request.getUserPrincipal().getName();
out.println("Bienvenido: " + usuario);
if(request.isUserInRole("CLIENTE")){
out.println("<p><a href='../pedido'>Pedidos</a></p>");
}
if(request.isUserInRole("ADMINISTRADOR")){
out.println("<p><a href='../admin'>Administración</a></p>");
}
%>
</body>
</html>
```

En este archivo se muestra el nombre del usuario que se encuentra logueado (`request.getUserPrincipal().getName()`) y se comprueba – verifica si un usuario pertenece a un grupo (perfil) determinado (`request.isUserInRole("PERFIL_ID")`). También se crea un enlace a la página que cierra la sesión, la invalida y por lo tanto borra todos los datos, creando una salida segura.

>> Cree el archivo /salir.jsp con el siguiente contenido:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%
session.invalidate();
response.sendRedirect("index.jsp");
%>
```

Es necesario que el usuario cierre la sesión, cuando se llama a este recurso se invalidan (borran) todos los datos almacenados en la sesión.



>> Cree el archivo /admin/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sitio Seguro :: Administración</title>
</head>
<body>
<h2>Módulo para el Administrador</h2>
</body>
</html>
```

>> Cree el archivo /pedido/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sitio Seguro :: Pedidos</title>
</head>
<body>
<h2>Módulo para clientes</h2>
</body>
</html>
```

## 6.3 CONFIGURAR EL DOMINIO DE SEGURIDAD

Cree un archivo llamado /WEB-INF/jboss-web.xml y pegue el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
<security-domain>java:/jaas/seguridadWeb201</security-domain>
</jboss-web>
```

En este archivo se especifica que dominio de seguridad va a utilizar la aplicación web, en nuestro caso **seguridadWeb201**.

## 6.4 ASEGURE LOS RECURSOS DE LA APLICACIÓN WEB

Adicione al archivo /WEB-INF/web.xml al final del archivo, antes de la etiqueta de cierre </web-app> el siguiente contenido:

```
...
...
<security-constraint>
<display-name>Restricción de Seguridad - Administrador</display-name>
<web-resource-collection>
<web-resource-name>Area de Administrador</web-resource-name>
<url-pattern>/admin/*</url-pattern>
<http-method>DELETE</http-method>
<http-method>GET</http-method>
<http-method>POST</http-method>
<http-method>PUT</http-method>
</web-resource-collection>

<auth-constraint>
<role-name>ADMINISTRADOR</role-name>
</auth-constraint>

<user-data-constraint>
<transport-guarantee>NONE</transport-guarantee>
</user-data-constraint>

</security-constraint>

<security-constraint>
<display-name>Restricción de Seguridad - Cliente</display-name>

<web-resource-collection>
<web-resource-name>Area de Cliente</web-resource-name>
<url-pattern>/pedido/*</url-pattern>
<http-method>DELETE</http-method>
<http-method>GET</http-method>
<http-method>POST</http-method>
<http-method>PUT</http-method>
</web-resource-collection>

<auth-constraint>
<role-name>CLIENTE</role-name>
</auth-constraint>

<user-data-constraint>
<transport-guarantee>NONE</transport-guarantee>
</user-data-constraint>

</security-constraint>

<security-constraint>
<display-name>Restricción de Seguridad - Usuarios</display-name>
<web-resource-collection>
<web-resource-name>Area de Usuarios</web-resource-name>
```

```

<url-pattern>/usuario/*</url-pattern>
<http-method>DELETE</http-method>
<http-method>GET</http-method>
<http-method>POST</http-method>
<http-method>PUT</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>ADMINISTRADOR</role-name>
<role-name>CLIENTE</role-name>
</auth-constraint>
<user-data-constraint>
<transport-guarantee>NONE</transport-guarantee>
</user-data-constraint>
</security-constraint>

<login-config>
<auth-method>FORM</auth-method>
<realm-name>seguridadWeb201</realm-name>
<form-login-config>
<form-login-page>/login.jsp</form-login-page>
<form-error-page>/error.jsp</form-error-page>
</form-login-config>
</login-config>

<!-- roles-(perfiles o grupos) válidos para la aplicación web -->
<security-role>
<role-name>ADMINISTRADOR</role-name>
</security-role>
<security-role>
<role-name>CLIENTE</role-name>
</security-role>

```

En la sección de `<security-constraint>...</ security-constraint>` se define una restricción de seguridad, es decir se asegura uno o más recursos, de modo que solo los perfiles autorizados en esta sección podrán hacer uso de ellos. En el nodo `<url-pattern>` se especifica el recurso que se quiere proteger, por ejemplo `/admin/*` significa que se desea proteger la carpeta `/admin` y todo el contenido que pueda tener (archivos y subcarpetas).

```

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

```

Se indican los métodos HTTP que se quieren restringir:

```

<http-method>DELETE</http-method>
<http-method>GET</http-method>
<http-method>POST</http-method>
<http-method>PUT</http-method>

```

También se restringe el acceso solo a determinados perfiles (`<role-name>`), por ejemplo:

```

<auth-constraint>
<role-name>ADMINISTRADOR</role-name>
</auth-constraint>

```

Indica que los recursos definidos en el patrón url solo podrán ser utilizados por los usuarios que tengan el perfil de ADMINISTRADOR.

## 7. PRUEBA DE LA APLICACIÓN WEB

Para probar la aplicación Web abra un navegador (browser) y digite la url (<http://localhost:8080/web201/>) y haga las siguientes pruebas:

- Intente ingresar con un usuario o clase errónea, el sistema no debe permitir el ingreso.
- Intente ingresar a la url <http://localhost:8080/web201/admin/>, el sistema no lo debe dejar ingresar y debe obligarlo a loguearse como un usuario del sistema y que perfil este autorizado (solo como ADMINISTRADOR).
- Ingrese con el perfil de CLIENTE (maria, cardenas) e intente ingresar a la url de administración <http://localhost:8080/web201/admin/>, el sistema debe arrojar un error de permiso porque el perfil de CLIENTE no está autorizado para esta url (recurso).