



GESTIÓN DE USUARIOS CON KEYCLOAK

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

1. Objetivos.....	3
2. Escenario.....	4
3. Fundamentos teóricos y conceptos.....	5
4. Instalaciones y configuraciones.....	7
4.1 Keycloak.....	7
4.2 Redmine.....	12
4.3 Instalación de plugin OpenID en Redmine.....	12
4.3 Creación de cliente redmine en keycloak.....	15
4.4 Configuración de OpenId Generic en Redmine.....	19
4.5 Cliente wordpress en keycloak.....	22
4.6 Instalación y configuración del plugin OpenId en wordpress.....	25
4.7 Instalación y configuración de OpenID en moodle.....	31
4.8 Implementación de factor de doble verificación.....	35
4.8.1 Configuración FreeOTP en keycloak.....	35
4.8.2 Instalación de aplicación de verificación y prueba.....	37
5. Conclusiones y propuestas.....	39
6. Bibliografía.....	40
6.1 Keycloak.....	40
6.2 REDMINE.....	40
6.3 MOODLE.....	41
6.4 WORDPRESS.....	41
6.5 OAUTH2.....	42
6.6 OPENID CONNECT.....	42
6.7 Factor de doble verificación OTP.....	42

1.Objetivos

Los objetivos que se quieren cumplir en este proyecto son sencillos. Básicamente queremos usar un servidor IAM (Administración de identidades y accesos) como es keycloak, para gestionar los usuarios y sus permisos en nuestros distintos servicios. Para hacerlo más sencillo y familiar he elegido muchos servicios que usamos día a día en nuestro instituto para gestionar distintas tareas y que pueden llegar también a ser útiles en un entorno laboral.

Tales como:

- Redmine
- Wordpress
- Moodle

Estos son los servicios que tengo pensado implementar.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

2. Escenario

La idea era instalarlo en servidores docker, y aunque durante mucho tiempo he estado gestionando todo en docker para diferentes pruebas no fui capaz de sacarlo hacia adelante, así que todos los servicios están en local.

Todos los servicios estarán conectados con nuestro servidor Keycloak obviamente para poder realizar las comprobaciones del proyecto. Y se accederá en local a todos los servicios.

3. Fundamentos teóricos y conceptos

- **Servidor IAM:** Servidor de gestión de accesos e identidades. En nuestro caso usaremos Keycloak, el cual nos permitirá realizar acciones tales como:
 - Inicio de sesión único.
 - Creación de usuarios y roles.
 - Cambio de contraseñas.
 - Monitorización de accesos.
 - Creación de usuarios temporales.
 - Gestionar los permisos de dichos usuarios.
 - Múltiple factor de autenticación.
 - Revocación de accesos.
 - Gestionar olvido de contraseñas.
 - Validación de cuentas.
 - Conexión con sistema LDAP.
 - Configuración en alta disponibilidad.
- **SAML2:** Es lo que nos permitirá realizar un inicio de sesión único. Crearemos los usuarios y roles en keycloak, sin embargo, tendremos SAML como intermediario para realizar el inicio de sesión único. Es un protocolo de autenticación en el que los Proveedores de identidad (IdP) intercambian documentos XML que permiten a un usuario final acceder a un Proveedor de servicios.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- **Oauth2:** Se trata de un protocolo para pasar la autorización de un servicio a otro sin compartir las credenciales de usuario reales, como un nombre de usuario y contraseña. Con esta herramienta un usuario puede iniciar sesión en una plataforma y luego estar autorizado para realizar acciones y ver datos en otra plataforma. Utiliza tokens de acceso. Un Token de acceso es un dato que representa la autorización para acceder a los recursos en nombre del usuario final.
- **OpenID:** Otro intermediario como SAML2, el que probablemente usemos, ya que tiene mayor documentación de soporte en instalaciones en todos los sistemas. Es un protocolo de identidad que utiliza los mecanismos de autorización y autenticación de OAuth 2.0. Realiza la autenticación del usuario, el consentimiento del usuario y la emisión de tokens.
- **Token de ejecución:** Está diseñado principalmente como un medio para conceder acceso a un conjunto de recursos, por ejemplo, API remotas o datos de usuario.
- **OTP:** Es una contraseña que pierde su validez después de su uso, de ahí su denominación. Por lo general, se emplea como parte de una autenticación de doble factor. Nosotros usaremos este servicio para autenticarnos de forma segura y usaremos FreeOTP como aplicación que proporcione los códigos

4. Instalaciones y configuraciones

4.1 Keycloak

- Debemos instalar dependencias como java:

```
apt-get install default-jdk -y
```

- Descargamos keycloak

```
wget https://github.com/keycloak/keycloak/releases/download/15.0.2/keycloak-15.0.2.tar.gz
```

- Lo descomprimos y movemos a /opt

```
tar -xvzf keycloak-15.0.2.tar.gz  
mv keycloak-15.0.2 /opt/keycloak
```

- Añadimos usuarios y grupos necesarios.

```
groupadd keycloak  
useradd -r -g keycloak -d /opt/keycloak -s /sbin/nologin keycloak
```

- Cambiamos los permisos del servicio.

```
chown -R keycloak: /opt/keycloak  
chmod o+x /opt/keycloak/bin/
```

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Procedemos a habilitar los ficheros de configuración:

```
cp /opt/keycloak/docs/contrib/scripts/systemd/wildfly.conf
```

```
/etc/keycloak/keycloak.conf
```

```
cp /opt/keycloak/docs/contrib/scripts/systemd/launch.sh /opt/keycloak/bin/
```

- Cambiamos los permisos.

```
chown keycloak: /opt/keycloak/bin/launch.sh
```

- Ahora entramos en el fichero “/opt/keycloak/bin/launch.sh” que debe quedar de la siguiente forma:

```
#!/bin/bash
```

```
if [ "x$WILDFLY_HOME" = "x" ]; then
```

```
WILDFLY_HOME="/opt/keycloak"
```

```
fi
```

```
if [[ "$1" == "domain" ]]; then
```

```
$WILDFLY_HOME/bin/domain.sh -c $2 -b $3
```

```
else
```

```
$WILDFLY_HOME/bin/standalone.sh -c $2 -b $3
```

```
fi
```

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Ahora crearemos un servicio para que se ejecute nuestro servidor keycloak.

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

- Debe quedar de la siguiente forma:

[Unit]

Description=The Keycloak Server

After=syslog.target network.target

Before=httpd.service

[Service]

Environment=LAUNCH_JBOSS_IN_BACKGROUND=1

EnvironmentFile=/etc/keycloak/keycloak.conf

User=keycloak

Group=keycloak

LimitNOFILE=102642

PIDFile=/var/run/keycloak/keycloak.pid

ExecStart=/opt/keycloak/bin/launch.sh \$WILDFLY_MODE \$WILDFLY_CONFIG

\$WILDFLY_BIND

StandardOutput=null

[Install]

WantedBy=multi-user.target

- Creamos un usuario administrador para poder loggearnos:

```
/opt/keycloak/bin/add-user-keycloak.sh -u admin
```

Gestión de usuarios con keycloak.

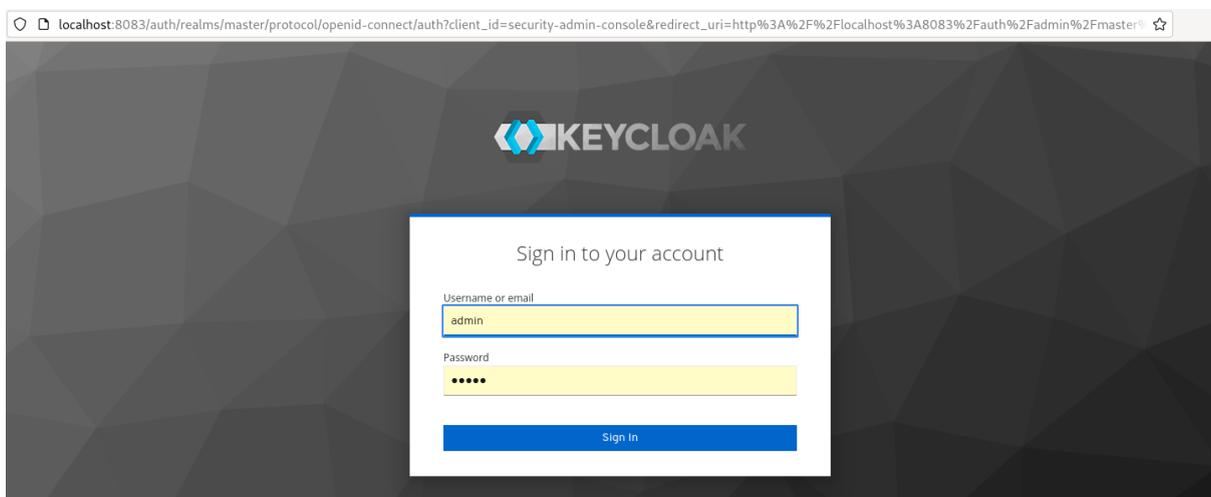
De Alejandro Gutiérrez Valencia.

- Para no tener problemas debemos deshabilitar el requerimiento de SSL.

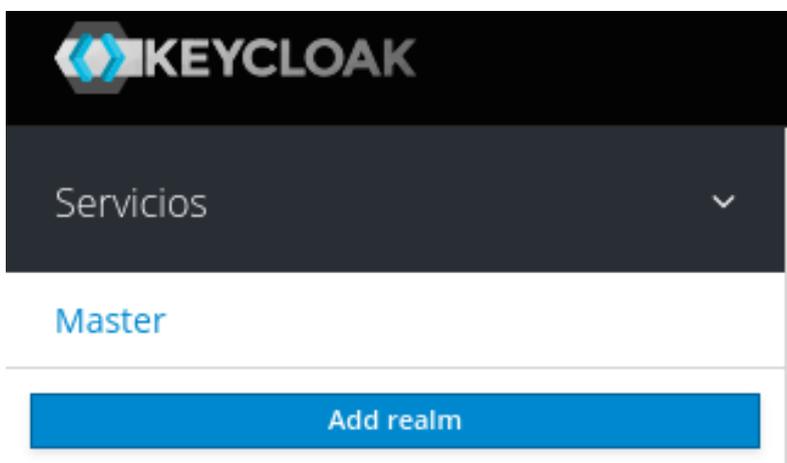
```
/opt/keycloak/bin/kcadm.sh config credentials --server http://localhost:8080/auth  
--realm master --user admin
```

```
/opt/keycloak/bin/kcadm.sh update realms/master -s sslRequired=NONE
```

- Y ya podremos acceder a la página de administración web.



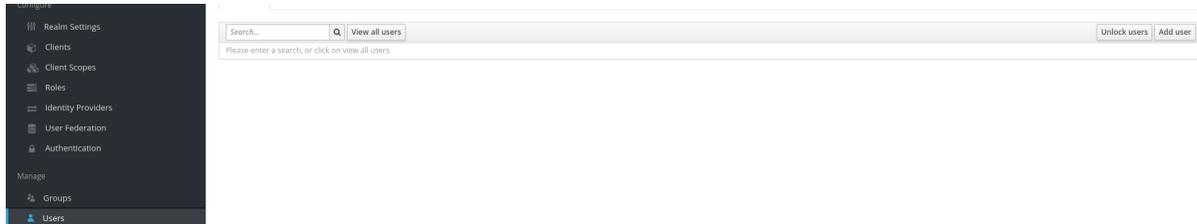
- Vamos a entrar y configurar un nuevo Realm, los realm son dominios dentro de nuestro servidor donde manejaremos los distintos clientes y usuarios.



Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Dentro del realm que hemos creado llamado "Servicios" vamos a crear un usuario. Vamos a la pestaña de "users" y le damos a "add user".



- Lo configuramos a nuestro gusto, en mi caso crearé uno que nos sirva como administrador.

Formulario de configuración de usuario:

- ID:
- Created At:
- Username *:
- Email:
- First Name:
- Last Name:
- User Enabled: ON
- Email Verified: OFF
- Groups:
No group selected
- Required User Actions:

Botones:

- Una vez creado entramos y le ponemos una contraseña. Desde el apartado "Credentials", desactivamos la opción de "temporal", añadimos una contraseña y guardamos.

Formulario de configuración de credenciales:

- Password:
- Password Confirmation:
- Temporary: OFF

Botón:

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

4.2 Redmine

- Instalación paso a paso, primero entramos con admin/admin tal como está configurado:



- Seguidamente se nos pedirá que cambiemos la contraseña:



4.3 Instalación de plugin OpenID en Redmine

- Por ahora entraremos manualmente en el contenedor y lo instalaremos, más adelante crearemos una receta ansible que haga estos pasos sola.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Clonamos el plugin en la carpeta correspondiente.

```
root@30f00b33dd75:/usr/src/redmine/plugins# git clone
https://github.com/devopskube/redmine_openid_connect
Cloning into 'redmine_oidc'...
remote: Enumerating objects: 486, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 486 (delta 28), reused 20 (delta 20), pack-reused 440
Receiving objects: 100% (486/486), 93.33 KiB | 1.35 MiB/s, done.
Resolving deltas: 100% (197/197), done.
```

- Una vez dentro realizamos las siguientes acciones:

```
bundle install
```

```
bundle exec rake redmine:plugins:migrate RAILS_ENV=production
```

Gestión de usuarios con keycloak. De Alejandro Gutiérrez Valencia.

- Reiniciamos el servidor y nos dirigimos a Administración > Extensiones.



- Y ya tendremos nuestro plugin instalado.

Extensiones

Redmine Openid Connect plugin OpenID Connect implementation for Redmine https://github.com/devopskube/redmine_openid_connect	Alfonso Juan Dillera / Markus M. May	0.9.4	Configurar
--	--------------------------------------	-------	----------------------------

[Comprobar actualizaciones](#)

Gestión de usuarios con keycloak. De Alejandro Gutiérrez Valencia.

4.3 Creación de cliente redmine en keycloak.

- Para configurar cada servicio necesitamos crear un cliente en nuestro realm. Para ello obviamente, iremos a la pestaña de “clients” y crearemos uno nuevo.



- Empezaremos con esta simple configuración, el nombre del cliente, el protocolo “OpenId” y la URL de nuestro servicio.

[Clients](#) > [Add Client](#)

Add Client

Import

Client ID

Client Protocol

Root URL

- Una vez guardemos, nos aparecerá una pestaña con este cliente y una configuración mucho más amplia, en primer lugar cambiaremos el protocolo de público a confidencial.

Client Protocol

Access Type

Client Protocol

Access Type

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Esta es la configuración principal:

Client ID ?	<input type="text" value="redmine"/>
Name ?	<input type="text"/>
Description ?	<input type="text"/>
Enabled ?	<input checked="" type="checkbox"/>
Always Display in Console ?	<input type="checkbox"/>
Consent Required ?	<input type="checkbox"/>
Login Theme ?	<input type="text"/>
Client Protocol ?	<input type="text" value="openid-connect"/>
Access Type ?	<input type="text" value="confidential"/>
Standard Flow Enabled ?	<input checked="" type="checkbox"/>
Implicit Flow Enabled ?	<input type="checkbox"/>
Direct Access Grants Enabled ?	<input type="checkbox"/>
Service Accounts Enabled ?	<input checked="" type="checkbox"/>
OAuth 2.0 Device Authorization Grant Enabled ?	<input type="checkbox"/>
OIDC CIBA Grant Enabled ?	<input type="checkbox"/>
Authorization Enabled ?	<input type="checkbox"/>
Root URL ?	<input type="text" value="http://localhost/redmine"/>
* Valid Redirect URIs ?	<input type="text" value="http://localhost/redmine/*"/>
Base URL ?	<input type="text"/>
Admin URL ?	<input type="text" value="http://localhost/redmine"/>
Web Origins ?	<input type="text" value="http://localhost"/>
Backchannel Logout URL ?	<input type="text"/>

- En la pestaña roles crearemos dos nuevos "Admin" y "User".

Redmine

Settings Credentials Keys **Roles** Client Scopes [?](#) Mappers [?](#) Scope [?](#) Revocation Sessions [?](#) Offline Access [?](#) Clustering Installation [?](#) Service Account Roles [?](#)

Role Name	Composite	Description	Actions
Admin	False		Edit Delete
User	False		Edit Delete

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- En “Mappers” debemos añadir todos estos pre-construidos por defecto en keycloak.

Redmine

Settings Credentials Keys Roles Client Scopes **Mappers** Scope Revocation Sessions Offline Access Clustering Installation Service Account Roles

Name	Category	Type	Priority Order	Actions
family name	Token mapper	User Property	0	Edit Delete
email	Token mapper	User Property	0	Edit Delete
Client IP Address	Token mapper	User Session Note	0	Edit Delete
full name	Token mapper	User's full name	0	Edit Delete
Client ID	Token mapper	User Session Note	0	Edit Delete
Client Host	Token mapper	User Session Note	0	Edit Delete
username	Token mapper	User Property	0	Edit Delete
given name	Token mapper	User Property	0	Edit Delete

- Solo debemos cambiar el mapper “username” que normalmente viene con el campo “token claim name” como “preferred_username”, debemos cambiar a “user_username”

Username

Protocol: openid-connect

ID: d911e0d0-8a76-49b2-9047-1186aa0418bd

Name: username

Mapper Type: User Property

Property: username

Token Claim Name: user_username

Claim JSON Type: String

Add to ID token: ON

Add to access token: ON

Add to userinfo: OFF

Save Cancel

- Y crear uno nuevo llamado “member of”

Clients > redmine > Mappers > Create Protocol Mappers

Create Protocol Mapper

Protocol: openid-connect

Name: member_of

Mapper Type: User Client Role

Client ID: redmine

Client Role prefix:

Multivalued: ON

Token Claim Name: member_of

Claim JSON Type: String

Add to ID token: ON

Add to access token: ON

Add to userinfo: OFF

Save Cancel

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- En scope debemos añadir “offline_access” y “uma_authorization” al “realm roles”

Redmine

Settings Credentials Keys Roles Client Scopes Mappers **Scope** Revocation Sessions Offline Access Clustering Installation Service Account Roles

redmine Scope Mappings

Full Scope Allowed OFF

Realm Roles	Available Roles	Assigned Roles	Effective Roles
	default-roles-servicios	offline_access uma_authorization	offline_access uma_authorization

Client Roles: redmine

Available Roles	Assigned Roles	Effective Roles
		Admin User

- Ya hemos terminado con el cliente, ahora para añadir a nuestro usuario Administrador entramos al usuario creado en “Role Mappings”, en el desplegable de “Client Roles” ponemos nuestro cliente y seleccionamos en este caso el rol de administrador.

Users > administrador

Administrador

Details Attributes Credentials **Role Mappings** Groups Consents Sessions

Realm Roles	Available Roles	Assigned Roles	Effective Roles
		default-roles-servicios	default-roles-servicios offline_access uma_authorization

Client Roles: redmine

Available Roles	Assigned Roles	Effective Roles
User	Admin	Admin

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

4.4 Configuración de OpenId Generic en Redmine

- Vamos a configurar nuestro OpenId con el plugin que instalamos anteriormente. Entramos y aparecerá esta página:

Extensiones » Redmine Openid Connect plugin

OpenID Connect Configuration
Enabled
Client ID
OpenID Connect server url
Client Secret
OpenID Connect scopes (comma-separated)
Authorized group (blank if all users are authorized)
Admins group (members of this group are treated as admin)
How often to retrieve openid configuration (default 1 day)
Disable Ssl Validation
Login Selector
Create user if not exists
Users from the following auth sources will be required to login with SSO

Acceptar

- Empecemos poco a poco, en primer lugar necesitamos el "Client Secret", esto es una clave que sacamos de nuestro cliente en keycloak, en el apartado "Credentials"

Redmine 

Settings **Credentials** Keys Roles Client Scopes ⓘ Mappers ⓘ Scope ⓘ Revocation Sessions ⓘ Offline Ac

Client Authenticator ⓘ

Secret

Registration access token ⓘ

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Lo pegamos en "client secret" y rellenamos los demás campos, en "server url" la dirección del servidor keycloak, en "authorized group" uno para los usuarios normales a los que calificamos como rol en keycloak por "User" y en el otro a los administradores, a quienes calificamos como "Admin"

Extensiones » Redmine Openid Connect plugin

OpenID Connect Configuration

Enabled

Client ID

OpenID Connect server url

Client Secret

OpenID Connect scopes (comma-separated)

Authorized group (blank if all users are authorized)

Admins group (members of this group are treated as admin)

How often to retrieve openid configuration (default 1 day)

Disable Ssl Validation

Login Selector

Create user if not exists

Users from the following auth sources will be required to login with SSO

Aceptar

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Una vez hecho esto salimos y vemos que nos aparece la opción de loggearnos con SSO.



The image shows a login interface. At the top, there is a label 'Identificador' above a text input field. Below the input field is a dropdown menu with two visible options: 'admin' and 'usuario1', both with the text 'De este sitio web' underneath them. Below the dropdown is a button labeled 'Ver credenciales guardadas'. Below this entire section is another button labeled 'Login with SSO'.

- Sin embargo, hay un problema y no puedo loggearme.

Internal error

An error occurred on the page you were trying to access.

If you continue to experience problems please contact your Redmine administrator for assistance.

If you are the Redmine administrator, check your log files for details about the error.

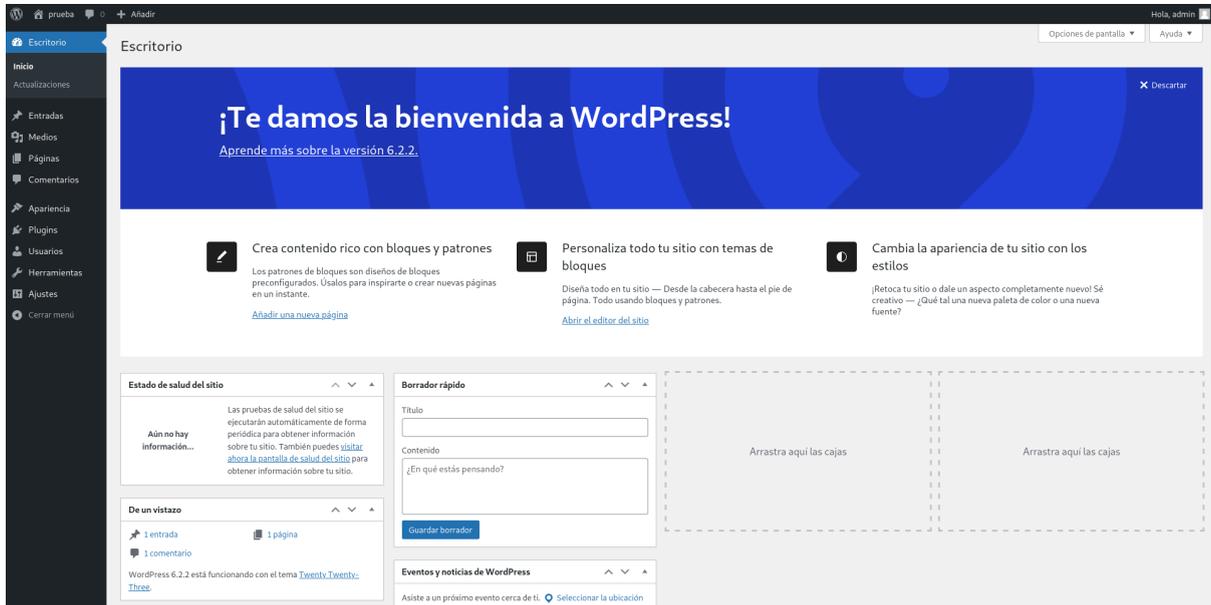
[Back](#)

Gestión de usuarios con keycloak.

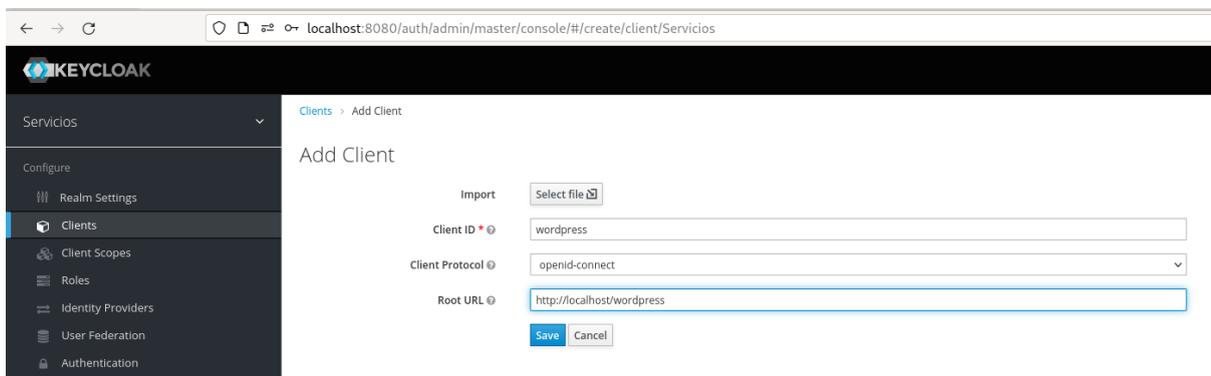
De Alejandro Gutiérrez Valencia.

4.5 Cliente wordpress en keycloak

- Primero instalaremos un servidor wordpress normal y corriente.



- Una vez instalado vamos a crear el cliente de keycloak:



Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Esta será la configuración principal:

Client ID	<input type="text" value="wordpress"/>
Name	<input type="text"/>
Description	<input type="text"/>
Enabled	<input checked="" type="checkbox"/>
Always Display in Console	<input type="checkbox"/>
Consent Required	<input type="checkbox"/>
Login Theme	<input type="text"/>
Client Protocol	<input type="text" value="openid-connect"/>
Access Type	<input type="text" value="confidential"/>
Standard Flow Enabled	<input checked="" type="checkbox"/>
Implicit Flow Enabled	<input type="checkbox"/>
Direct Access Grants Enabled	<input type="checkbox"/>
Service Accounts Enabled	<input checked="" type="checkbox"/>
OAuth 2.0 Device Authorization Grant Enabled	<input type="checkbox"/>
OIDC CIBA Grant Enabled	<input type="checkbox"/>
Authorization Enabled	<input checked="" type="checkbox"/>
Root URL	<input type="text" value="http://localhost/wordpress"/>
* Valid Redirect URIs	<input type="text" value="http://localhost/wordpress/*"/> <input type="button" value="-"/> <input type="button" value="+"/>
Base URL	<input type="text"/>
Admin URL	<input type="text" value="http://localhost/wordpress"/>
Web Origins	<input type="text" value="http://localhost"/> <input type="button" value="-"/> <input type="button" value="+"/>
Backchannel Logout URL	<input type="text"/>

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Nuevamente crearemos un rol Admin y User, aunque aquí no nos servirá tanto como en redmine. Ya que como veremos más adelante, no podremos especificar qué usuarios serán administradores y cuáles no en wordpress. Sin embargo, sí que podemos añadir el usuario al cliente wordpress.

The screenshot shows the Keycloak administration interface for the 'administrador' user. The 'Role Mappings' tab is active, displaying role assignments for both Realm Roles and Client Roles.

Realm Roles:

- Available Roles:** Empty list with an 'Add selected >' button.
- Assigned Roles:** Contains 'default-roles-servicios' with a '« Remove selected' button.
- Effective Roles:** Contains 'default-roles-servicios', 'offline_access', and 'uma_authorization'.

Client Roles (Client: wordpress):

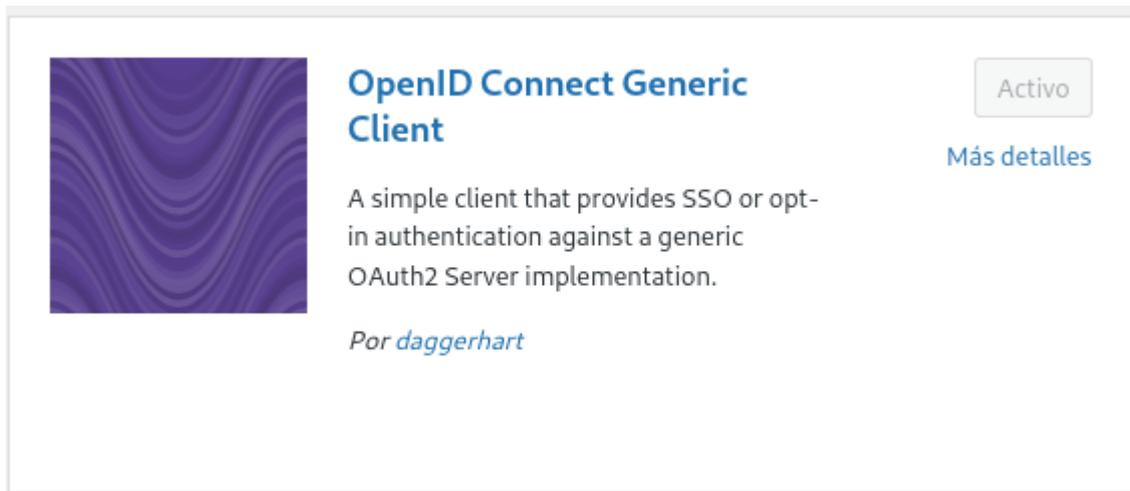
- Available Roles:** Empty list with an 'Add selected >' button.
- Assigned Roles:** Contains 'Admin' and 'uma_protection' with a '« Remove selected' button.
- Effective Roles:** Contains 'Admin' and 'uma_protection'.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

4.6 Instalación y configuración del plugin OpenId en wordpress

- Ya hemos configurado el cliente de keycloak, ahora vamos a instalar el plugin de OpenId en wordpress que nos permitirá conectar nuestro keycloak. Para ello vamos a “plugins > añadir nuevo” y buscamos e instalamos el siguiente plugin:



The screenshot shows the WordPress plugin directory interface for the 'OpenID Connect Generic Client' plugin. On the left is a purple abstract icon. The title 'OpenID Connect Generic Client' is in blue. Below it is a description: 'A simple client that provides SSO or opt-in authentication against a generic OAuth2 Server implementation.' The author is listed as 'Por daggerhart'. In the top right corner, there is a grey button labeled 'Activo' and a blue link labeled 'Más detalles'.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Una vez activado nos vamos a “Ajustes > OpenId Connect Client” y ahí podremos configurar nuestro OpenId para que se conecte a keycloak. Vamos a ver la configuración:

Login Type	<input type="text" value="OpenID Connect button on login form"/> ▼
	Select how the client (login form) should provide login options.
Client ID	<input type="text" value="wordpress"/>
	The ID this client will be recognized as when connecting the to Identity provider server. Example: <code>my-wordpress-client-id</code>
Client Secret Key	<input type="text" value="469966a7-c148-4725-b9ca-1776ff4d419b"/>
	Arbitrary secret key the server expects from this client. Can be anything, but should be very unique
OpenID Scope	<input type="text" value="openid"/>
	Space separated list of scopes this client should access. Example: <code>email profile openid offline_access</code>
Login Endpoint URL	<input type="text" value="http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/auth"/>
	Identify provider authorization endpoint. Example: <code>https://example.com/oauth2/authorize</code>
Userinfo Endpoint URL	<input type="text" value="http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/userinfo"/>
	Identify provider User information endpoint. Example: <code>https://example.com/oauth2/UserInfo</code>
Token Validation Endpoint URL	<input type="text" value="http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/token"/>
	Identify provider token endpoint. Example: <code>https://example.com/oauth2/token</code>
End Session Endpoint URL	<input type="text" value="http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/logout"/>
	Identify provider logout endpoint. Example: <code>https://example.com/oauth2/logout</code>
ACR values	<input type="text"/>
	Use a specific defined authentication contract from the IDP - optional.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

Disable SSL Verify	<input checked="" type="checkbox"/>	Do not require SSL verification during authorization. The OAuth extension uses certificate verification. Not recommended for production sites.
HTTP Request Timeout	<input type="text" value="20"/>	Set the timeout for requests made to the IDP. Default value is 5. Example: 30
Nickname Key	<input type="text" value="preferred_username"/>	Where in the user claim array to find the user's nickname. Possible standard value Example: preferred_username
Email Formatting	<input type="text" value="{email}"/>	String from which the user's email address is built. Specify "{email}" as long as the email claim is present. Example: {email}
Display Name Formatting	<input type="text" value="{given_name} {family_name}"/>	String from which the user's display name is built. Example: {given_name} {family_name}
Identify with User Name	<input checked="" type="checkbox"/>	If checked, the user's identity will be determined by the user name instead of the email address.
State time limit	<input type="text"/>	State valid time in seconds. Defaults to 180
Enable Refresh Token	<input checked="" type="checkbox"/>	If checked, support refresh tokens used to obtain access tokens from supported IDPs.
WordPress User Settings		
Modify the interaction between OpenID Connect and WordPress users.		
Link Existing Users	<input checked="" type="checkbox"/>	If a WordPress account already exists with the same identity as a newly-authenticated user, link the two accounts.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

Create user if does not exist If the user identity is not linked to an existing WordPress user, it is created. If this setting is not enabled, and

Redirect Back to Origin Page After a successful OpenID Connect authentication, this will redirect the user back to the page on which they
example, users logging in through the default wp-login.php page would end up on the WordPress Dashboa

Redirect to the login screen when session is expired When enabled, this will automatically redirect the user back to the WordPress login page if their access toke

Authorization Settings

Control the authorization mechanics of the site.

Enforce Privacy Require users be logged in to see the site.

Alternate Redirect URI Provide an alternative redirect route. Useful if your server is causing issues with the default admin-ajax met

Log Settings

Log information about login attempts through OpenID Connect Generic.

Enable Logging Very simple log messages for debugging purposes.

Log Limit Number of items to keep in the log. These logs are stored as an option in the database, so space is limited.

- Recordemos que el “cliente secret” lo sacamos del cliente que hemos creado en nuestro realm de keycloak en el apartado “credentials” y que las URL las sacaremos de [“http://localhost:8080/auth/realms/Servicios/.well-known/openid-configuration”](http://localhost:8080/auth/realms/Servicios/.well-known/openid-configuration)

The screenshot shows a web browser window with the address bar containing the URL `localhost:8080/auth/realms/Servicios/.well-known/openid-configuration`. The page content displays a JSON object with the following structure:

```
{  "issuer": "http://localhost:8080/auth/realms/Servicios",  "authorization_endpoint": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/auth",  "token_endpoint": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/token",  "introspection_endpoint": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/token/introspect",  "userinfo_endpoint": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/userinfo",  "end_session_endpoint": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/logout",  "jwks_uri": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/certs",  "check_session_iframe": "http://localhost:8080/auth/realms/Servicios/protocol/openid-connect/login-status-iframe.html",  "grant_types_supported": []}
```

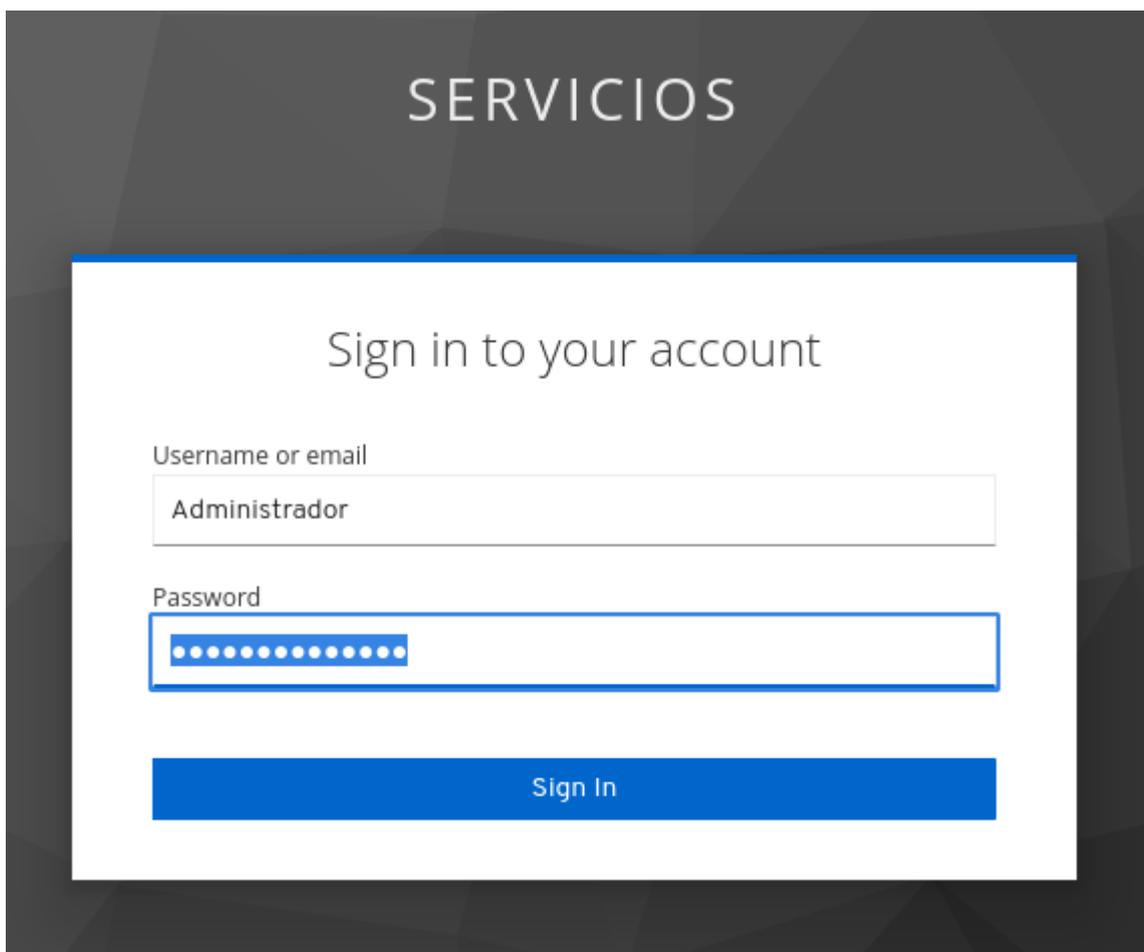
Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Una vez hecho esto podemos salir y logearnos con la opción de OpenId



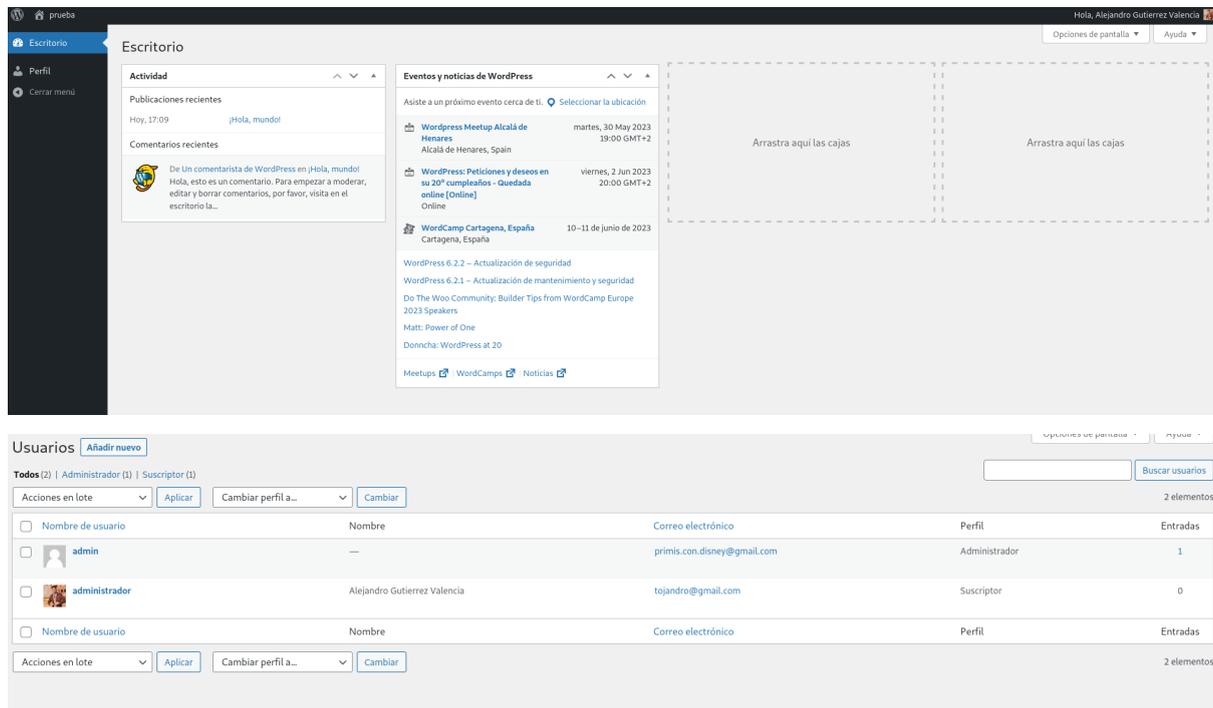
- Se nos abrirá una pestaña de keycloak para introducir nuestras credenciales:



Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Ahora ya estamos dentro. Y podemos ver que se ha creado un usuario nuevo.



The image shows two screenshots from a WordPress dashboard. The top screenshot is the 'Escritorio' (Dashboard) view, showing recent activity, WordPress events, and a sidebar with navigation options. The bottom screenshot is the 'Usuarios' (Users) management page, displaying a table of users.

<input type="checkbox"/>	Nombre de usuario	Nombre	Correo electrónico	Perfil	Entradas
<input type="checkbox"/>	admin	—	primis.con.disney@gmail.com	Administrador	1
<input type="checkbox"/>	administrador	Alejandro Gutierrez Valencia	tojandro@gmail.com	Suscriptor	0

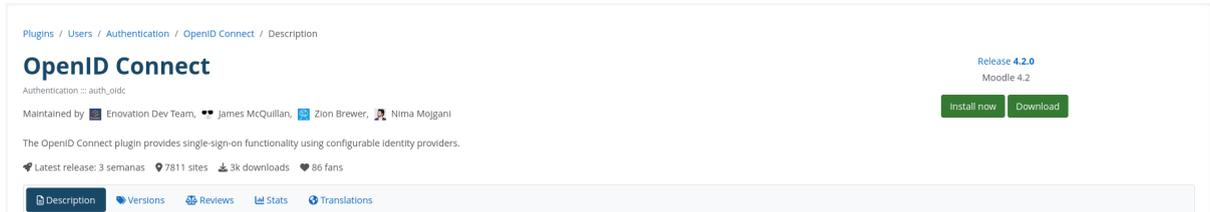
- Aunque vemos que no es administrador. Para que lo fuese deberíamos entrar como un usuario administrador y añadir este usuario específico como administrador.

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

4.7 Instalación y configuración de OpenID en moodle.

- Descargamos el plugin de [la página oficial de moodle](#)



- Lo instalamos en nuestro sitio moodle, para ello nos dirigimos a “Zona de administración > Extensiones > Instalar complemento” y arrastramos nuestro plugin aquí.

Pruebas Moodle Local Usuarios

General Usuarios Cursos Calificaciones Extensiones Apariencia Servidor Informes Desarrollo

Instalador de complemento

Instalar complementos desde el directorio de extensiones de Moodle

▼ Instalar complemento desde un archivo ZIP

Paquete ZIP



Seleccione un archivo...

auth_oidc_moodle42_2023042400.zip

Tipos de archivo aceptados:

Archivo (ZIP) .zip

Mostrar más...

Instalar complemento desde archivo ZIP

Requerido

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Ahora tendremos que realizar la configuración de nuestro OpenID, rellenando los datos tal como lo hicimos en los otros dos servicios.

Nuevos ajustes - Conexión OpenID

Nombre del proveedor
auth_oidc | opname Valor por defecto: Conexión OpenID

Esta es una etiqueta para que el usuario final identifique el tipo de credenciales que debe utilizar para el acceso. Esta etiqueta se utiliza para identificar al proveedor.

ID de cliente
auth_oidc | clientid Valor por defecto: Vacío

Su ID de cliente registrado en el proveedor de identidad

Secreto de cliente
auth_oidc | clientsecret Valor por defecto: Vacío

Su secreto de cliente registrado en el proveedor de identidad. En algunos proveedores, también se conoce como clave.

Extremo de autorización
auth_oidc | authendpoint Valor por defecto: https://login.microsoftonline.com/common/oauth2/authorize

La URI del extremo de autorización del proveedor de identidad que va a utilizar.

Extremo de ficha
auth_oidc | tokenendpoint Valor por defecto: https://login.microsoftonline.com/common/oauth2/token

La URI del extremo de ficha del proveedor de identidad que debe utilizar.

Recurso
auth_oidc | oidcresource Valor por defecto: https://graph.microsoft.com

El recurso de OpenID Connect para el cual enviar la solicitud.

Scope
auth_oidc | oidcscope Valor por defecto: openid profile email

El alcance de OIDC a utilizar.

Forzar redirigir
auth_oidc | forcedredirect Valor por defecto: No

Si se habilita, se saltará la página del índice para ingresar al sitio y redireccionará a la página para Conectar a OpenID. No puede ser sobrepasada con ?noredirect=1 URL param

Auto-Anexar
auth_oidc | autoappend Valor por defecto: Vacío

Anexe automáticamente esta cadena cuando los usuarios inicien sesión mediante el flujo de inicio de sesión de nombre de usuario/contraseña. Esto es útil cuando el proveedor de identidad requiere un dominio común, pero no desea solicitar a los usuarios que lo escriban cuando inician sesión. Por ejemplo, si el usuario completo de OpenID Connect es

- Por supuesto hemos creado un cliente nuevo en keycloak.

The screenshot shows the Keycloak administration interface. On the left is a navigation sidebar with 'Servicios' at the top and a menu for 'Clients' (Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication, Manage, Groups, Users, Sessions, Events, Import, Export). The main content area is titled 'Clients > moodle' and shows the configuration for a client named 'moodle'. The 'Settings' tab is active, displaying various configuration options:

- Client ID: moodle
- Name: (empty)
- Description: (empty)
- Enabled: ON
- Always Display in Console: OFF
- Consent Required: OFF
- Login Theme: (dropdown)
- Client Protocol: openid-connect
- Access Type: confidential
- Standard Flow Enabled: ON
- Implicit Flow Enabled: OFF
- Direct Access Grants Enabled: OFF
- Service Accounts Enabled: ON
- OAuth 2.0 Device Authorization Grant Enabled: OFF
- OIDC CIBA Grant Enabled: OFF
- Authorization Enabled: OFF
- Root URL: (empty)
- Valid Redirect URIs: http://localhost/moodle/auth/oidc/
- Base URL: (empty)

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Nos dirigimos de nuevo a extensiones y esta vez a gestionar autenticación, y pondremos OpenID como nuestra primera opción al loggearnos.

Gestionar la autenticación

Extensiones (plugins) de identificación disponibles

Nombre	Usuarios	Habilitar	Arriba/Abajo	Configuración	Probar la configuración	Desinstalar
Cuentas manuales	2			Configuración		
No hay sesión	0					
Conexión OpenID	0		↓	Configuración		Desinstalar
Identificación basada en Email	0		↑ ↓	Configuración		Desinstalar
OAuth 2	0		↑	Configuración		Desinstalar

- Ahora nos deslogueamos y tratamos de volver a loggearnos con OpenID.

Entrar a Pruebas Moodle Local Usuarios

admin

●●●●●●●●●●●●●●●●

Acceder

[¿Ha extraviado la contraseña?](#)

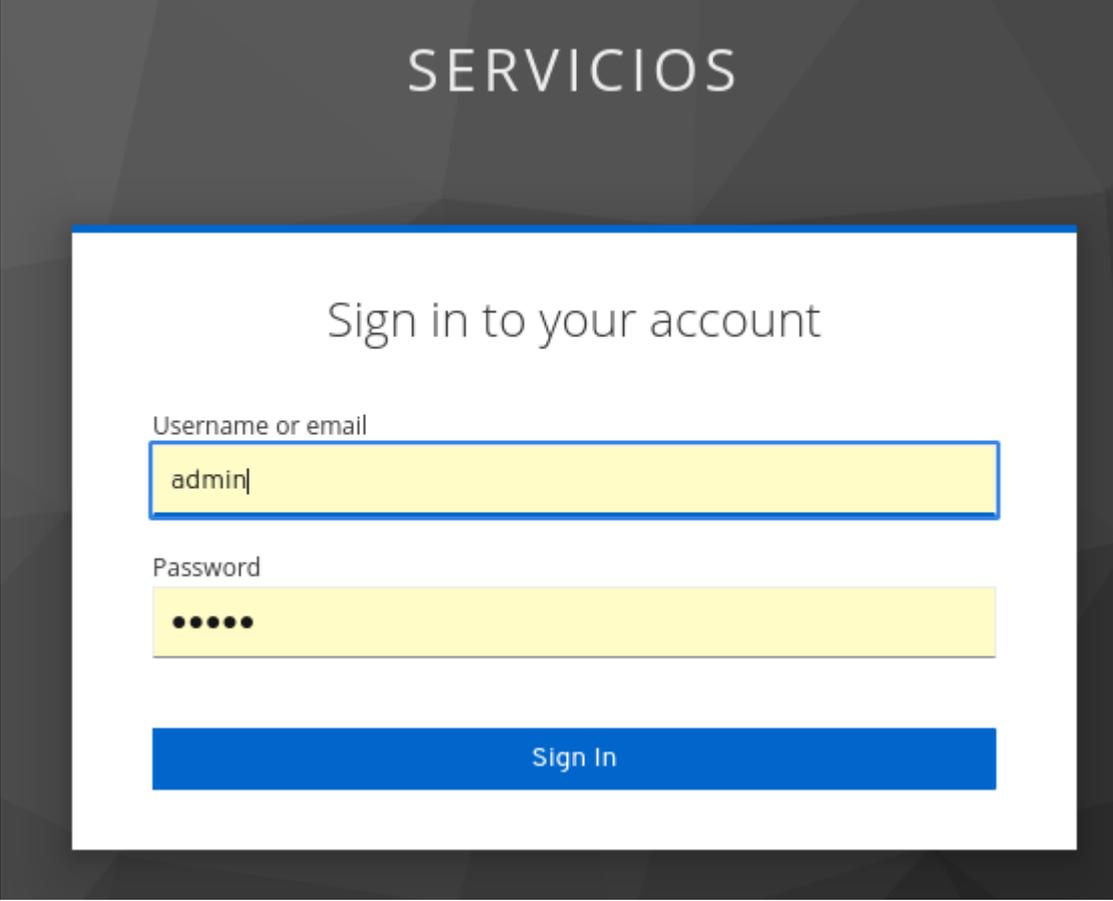
Identifíquese usando su cuenta en:



Conexión OpenID

Gestión de usuarios con keycloak. De Alejandro Gutiérrez Valencia.

- Aparecerá la página de keycloak.



- Aunque vemos que no es posible la conexión.

debian11.local.moodle [Página Principal](#)

Pruebas Moodle Local Usuarios

Error en OpenID Connect. Por favor, revise las bitácoras para más información.

[Más información sobre este error](#)

[Continuar](#)

- Si miramos el log encontramos este error.

Sistema	Conexión OpenID	Mensaje de depuración	auth_oidutils::process_json_response:291: Bad response received Data: (string)Esta URL está bloqueada.	web	127.0.0.1
---------	-----------------	-----------------------	--	-----	-----------

- El error es debido a una de las URL de redirección de keycloak, la cual no existe en este servicio, esto se debe a que el plugin de OpenID para Moodle está pensado para implantarse con Microsoft 365.

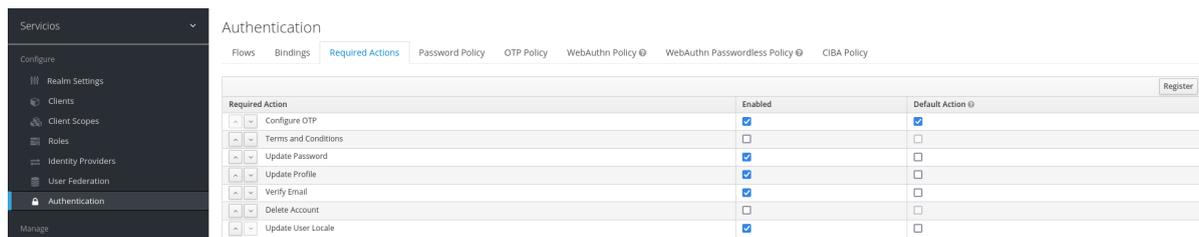
Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

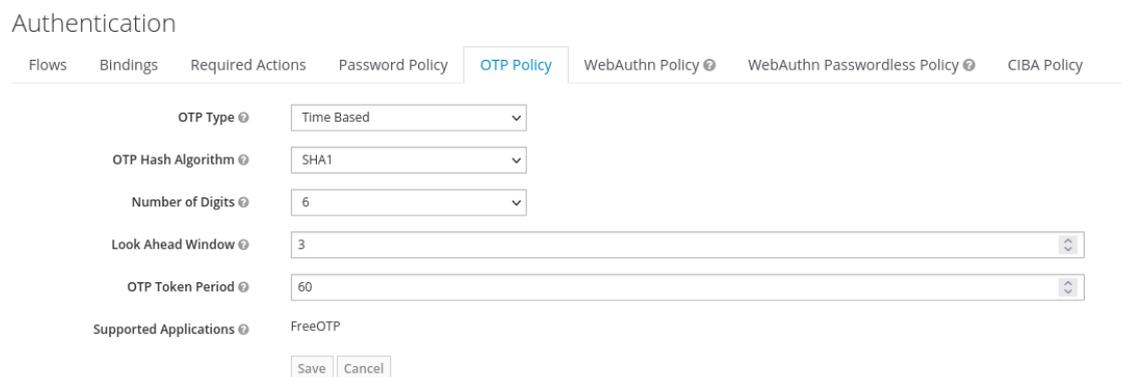
4.8 Implementación de factor de doble verificación

4.8.1 Configuración FreeOTP en keycloak

- Primero realizamos la configuración keycloak. Para ello nos vamos al panel de administración y nos dirigimos a "Authentication > Required Actions" y en "Configure OTP" seleccionamos las dos casillas, para habilitarlo y para añadir las opciones por defecto.



- Ahora configuraremos a nuestro gusto esta herramienta cambiando a la pestaña "OTP Policy", en mi caso está basado en el tiempo, el token dura 60 segundos se podrá usar en 3 pestañas a la vez y serán 6 dígitos. Por supuesto el algoritmo de encriptación estará basado en SHA1.



Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

- Ahora debemos implementar esta funcionalidad en nuestros usuarios, vamos a crear uno nuevo para ver cómo se configuraría. Como vemos en “Required User Actions” hemos añadido la opción Configure OTP.

Add user

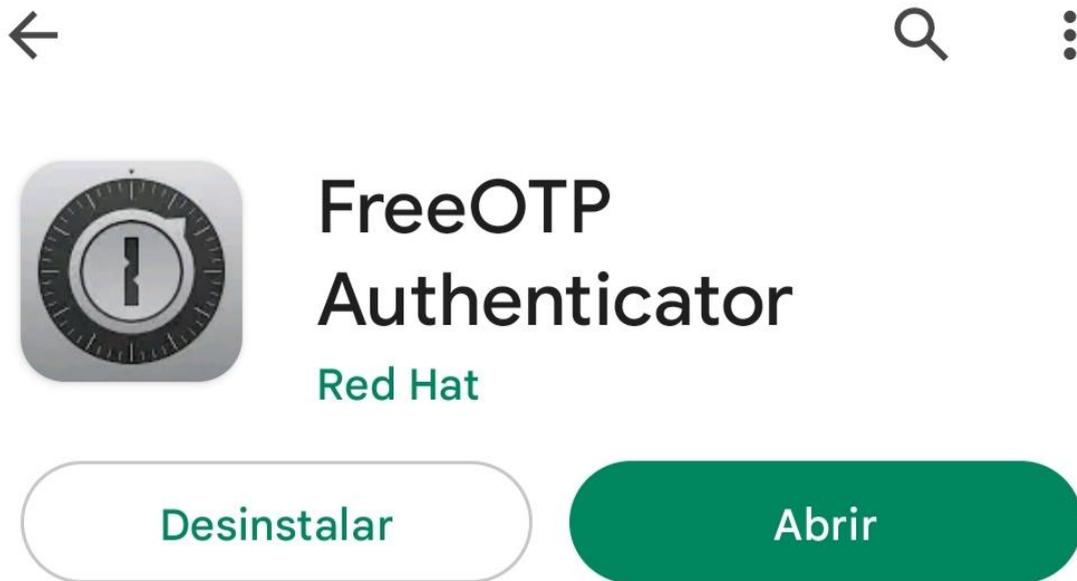
ID	<input type="text"/>
Created At	
Username *	<input type="text" value="usuario1"/>
Email	<input type="text" value="prueba@gmail.com"/>
First Name	<input type="text" value="Usuario1"/>
Last Name	<input type="text" value="Usuario Prueba"/>
User Enabled ⓘ	<input checked="" type="checkbox"/> ON
Email Verified ⓘ	<input type="checkbox"/> OFF
Groups ⓘ	<input type="text" value="Select existing group..."/> No group selected
Required User Actions ⓘ	<input type="text" value="x Configure OTP"/>
	<input type="button" value="Save"/> <input type="button" value="Cancel"/>

- Tenemos nuestro usuario configurado con OTP, ahora debemos instalar nuestra aplicación de verificación y ver una prueba.

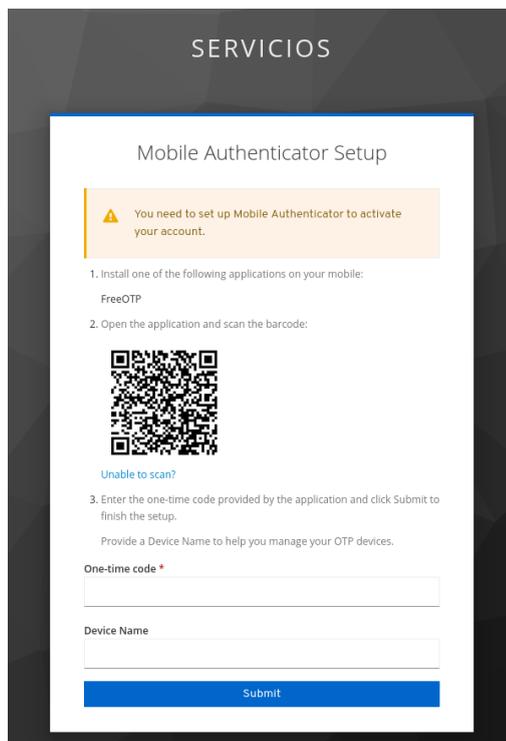
Gestión de usuarios con keycloak. De Alejandro Gutiérrez Valencia.

4.8.2 Instalación de aplicación de verificación y prueba

- Debemos instalar una aplicación la cual nos servirá para obtener el código de verificación correspondiente. Se llama "FreeOTP Authenticator".



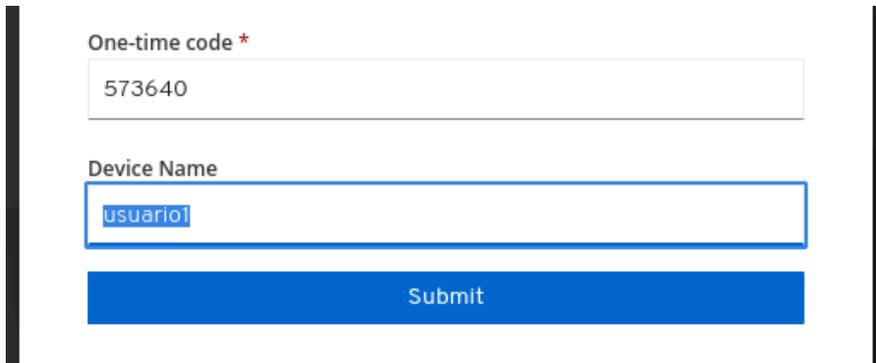
- Tenemos un problema y es que por seguridad la aplicación no permite realizar capturas de pantalla dentro de la misma, sin embargo el proceso es muy sencillo.



Gestión de usuarios con keycloak.

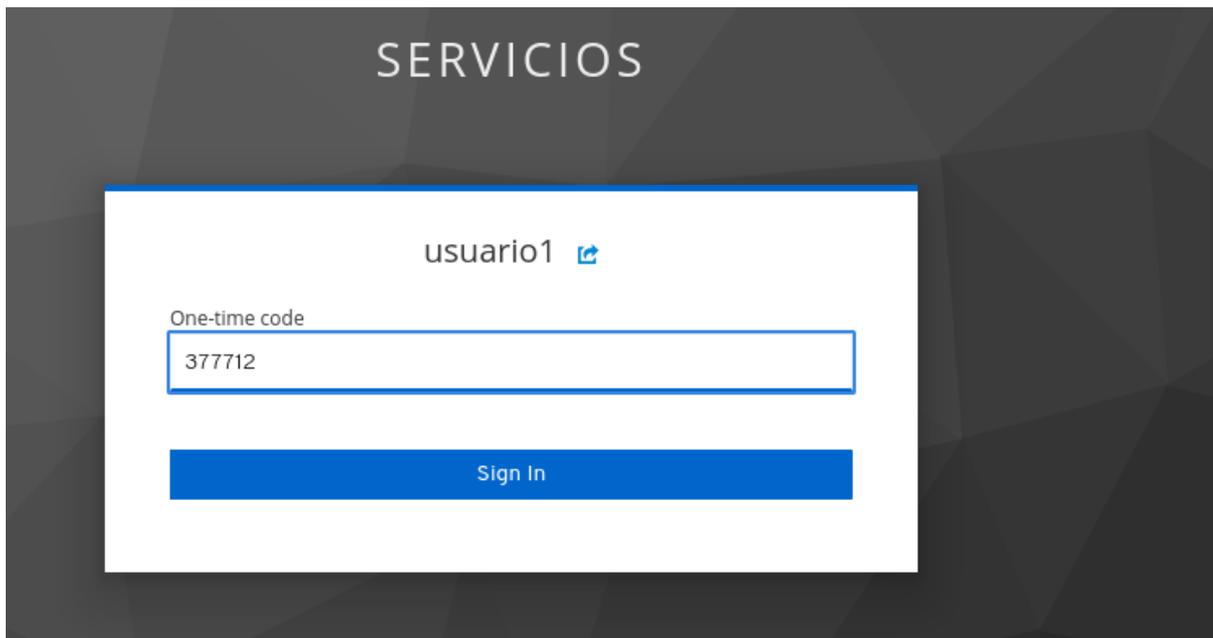
De Alejandro Gutiérrez Valencia.

- Con la aplicación pulsaremos en un icono "+" y escaneamos el código qr que aparece y nos aparecerá directamente un código, lo escribimos y ya estamos loggeados.



A screenshot of a login form. At the top, it says "One-time code *". Below that is a text input field containing the number "573640". Underneath is the label "Device Name" followed by another text input field containing "usuario1". At the bottom of the form is a blue button labeled "Submit".

- Si nos conectamos de nuevo no tendremos que escanear el código, simplemente entramos en nuestra aplicación y tapeamos el código que habrá cambiado.



A screenshot of a login form titled "SERVICIOS" in large white letters on a dark background. Below the title, the text "usuario1" is displayed with a small blue icon to its right. Underneath is the label "One-time code" followed by a text input field containing the number "377712". At the bottom of the form is a blue button labeled "Sign In".

5. Conclusiones y propuestas

Sin duda queda mucho por explorar en todas las herramientas, apenas he rascado la superficie de las posibilidades técnicamente de Keycloak y sus funcionalidades.

En conclusión. Keycloak es un servicio más complejo de lo que imaginaba al comenzar que, para expresar su funcionalidad, depende de otras muchas pequeñas piezas que van dando forma poco a poco a lo que queremos conseguir. Que es: tener una gestión de usuarios y accesos centralizada de nuestros diferentes servicios.

Para realizar esta labor hay que profundizar en las diferentes herramientas intermedias de las que hemos hablado como: SAML2, OpenId, Oauth2. Todas tienen sus ventajas e inconvenientes.

6. Bibliografía

6.1 Keycloak

<https://youtu.be/W38qJEodeKs?list=PL4bT56Uw3S4wEZ0Sp7jrGAX8DMS-MKowg>

<https://www.enmilocalfunciona.io/acelerando-los-desarrollos-con-contenedores-keycloak/>

<https://www.enmilocalfunciona.io/acelerando-los-desarrollos-con-contenedores-keycloak-parte-2/>

<https://blog.desdelinux.net/keycloak-una-solucion-de-gestion-de-acceso-e-identidad-de-codigo-abierto/>

<https://www.chakray.com/es/como-utilizar-keycloak-proveedor-identidades-wso2-api-manager/>

<https://apim.docs.wso2.com/en/latest/administer/key-managers/configure-keycloak-connector/>

<https://cloudinfrastructureservices.co.uk/install-keycloak-sso-on-ubuntu-20-04/>

6.2 REDMINE

<https://stackoverflow.com/questions/75984580/redmine-openid-connct-with-keycloak>

Gestión de usuarios con keycloak.

De Alejandro Gutiérrez Valencia.

<https://labarta.es/como-instalar-redmine-usando-docker-compose/>

https://github.com/Contargo/redmine_oidc

https://github.com/devopskube/redmine_openid_connect

<http://devopsku.be/setup/redmine-keycloak/>

<https://chachocool.com/como-instalar-redmine-en-debian-11-bullseye/>

6.3 MOODLE

<https://stackoverflow.com/questions/69490574/how-to-connect-moodle-with-keycloak>

<https://plugins.miniorange.com/moodle-saml-single-sign-on-sso-for-keycloak>

https://moodle.org/plugins/auth_oidc

https://docs.moodle.org/310/en/Microsoft_365#Plugin_Installation

6.4 WORDPRESS

<https://westergaard.eu/2018/05/sso-lets-talk-about-single-sign-on-for-wordpress-and-gitlab-using-keycloak-redhat-sso-featuring-special-guest-duo-security-for-2fa/>

<https://dev.to/vishalraj82/using-https-in-docker-for-local-development-nc7>

Gestión de usuarios con keycloak.
De Alejandro Gutiérrez Valencia.

<https://www.rubenortiz.es/2022/02/18/habilitar-https-en-wordpress-local-con-do-cker/>

6.5 OAUTH2

<https://linuxhint.com/oauth-linux-installation/>

<https://www.albertcoronado.com/2020/09/24/login-en-tus-aplicaciones-de-micro-servicios-con-oauth2-proxy/>

6.6 OPENID CONNECT

<https://www.youtube.com/watch?v=nNVlewjKQEQ>

6.7 Factor de doble verificación OTP.

<https://ultimatesecurity.pro/post/2fa/>

<https://mailchimp.com/es/help/set-up-a-two-factor-authentication-app-at-login/>

<https://repositorio.unican.es/xmlui/bitstream/handle/10902/26830/444290.pdf?sequence=1>