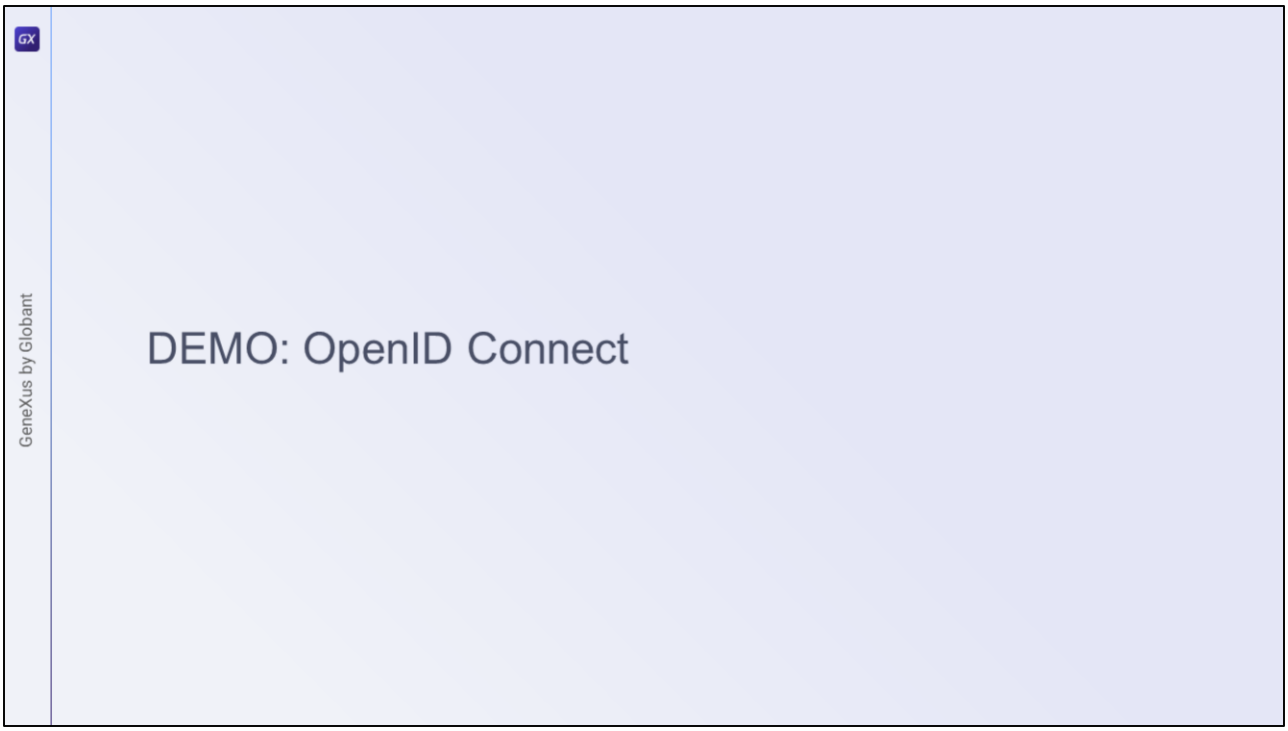
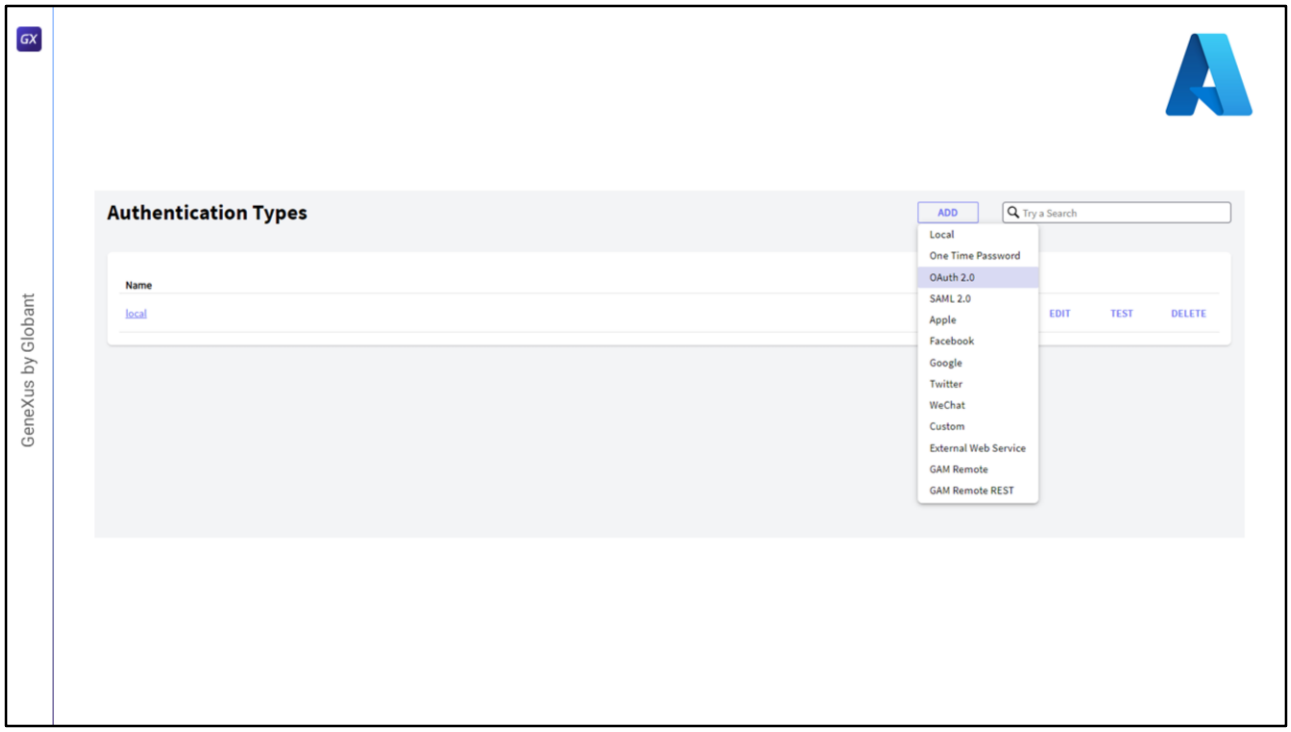


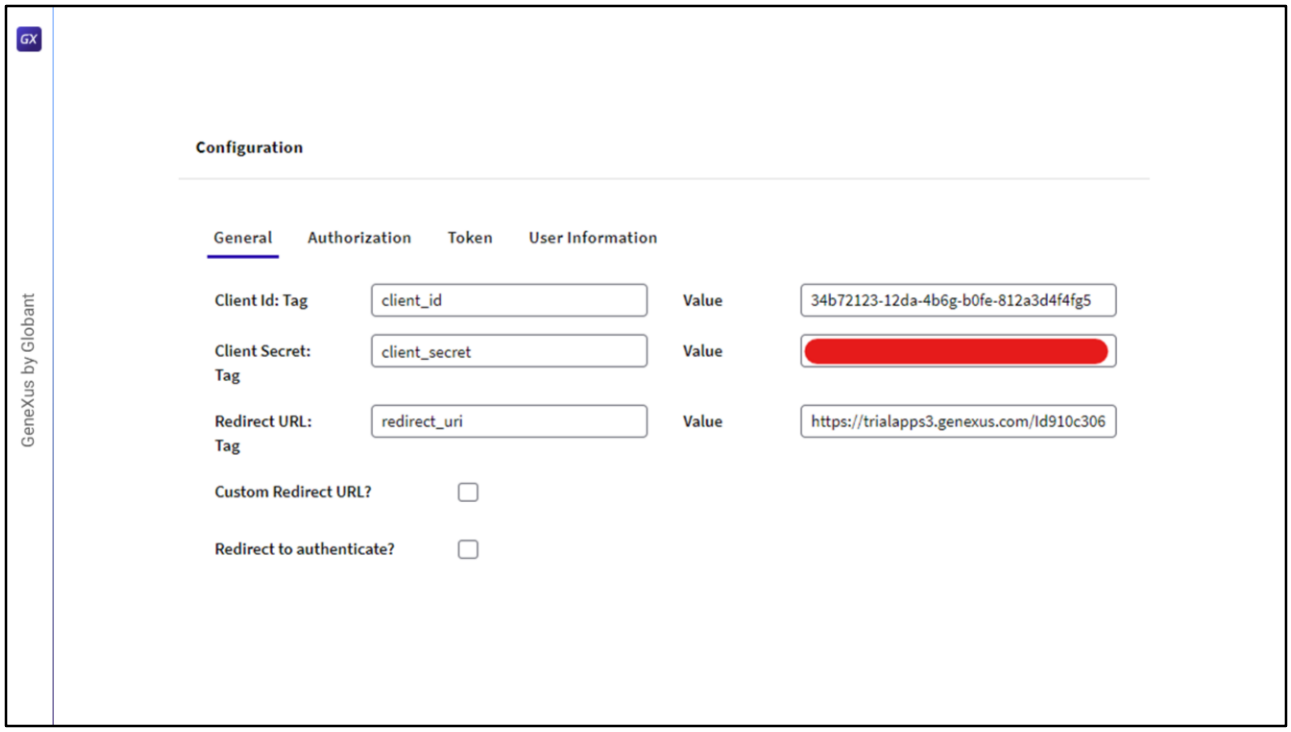
Primer demo: OpenID Connect.



Para esta demo, utilizaremos el protocolo OAuth 2.0 en GAM. Nuestro proveedor de identidad será Azure Active Directory a través de Microsoft.

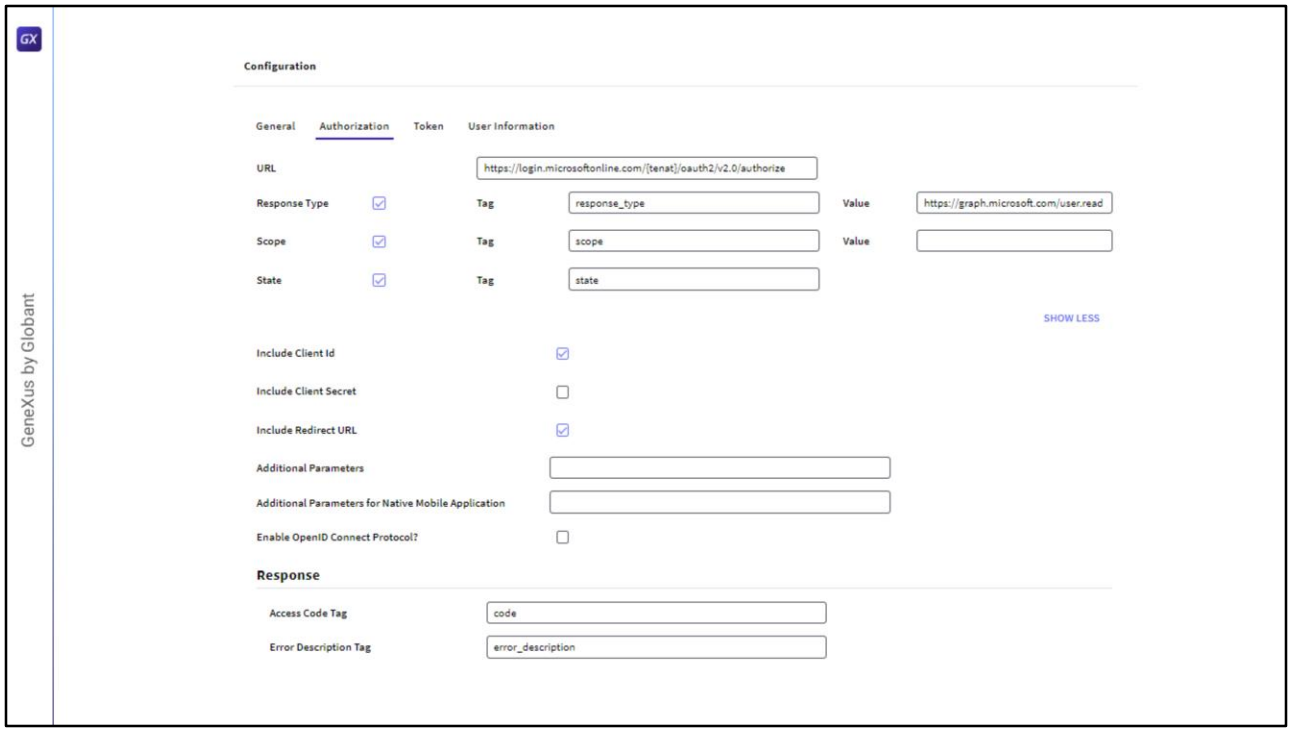
Asumiremos que la configuración desde el lado de Azure ya se realizó correctamente y no entraremos en detalle de la misma. Para ver cómo realizarla pueden encontrar en la Wiki de GeneXus un artículo detallado sobre esto.

En primer lugar, debemos crear un nuevo Tipo de autenticación GAM OAuth 2.0 y definir los conceptos básicos, como lo es el Nombre, su Descripción, etc.



En la pestaña General, se debe definir lo siguiente:
Primero seteamos el Client ID y Client Secret que obtenemos desde Azure.
La URL de redirección debe ser la URL Base del Backend de nuestra aplicación.

Como comentamos en el video anterior, no marcaremos la opción Redireccionar para autenticar ya que queremos loguearnos desde el GAM mismo.

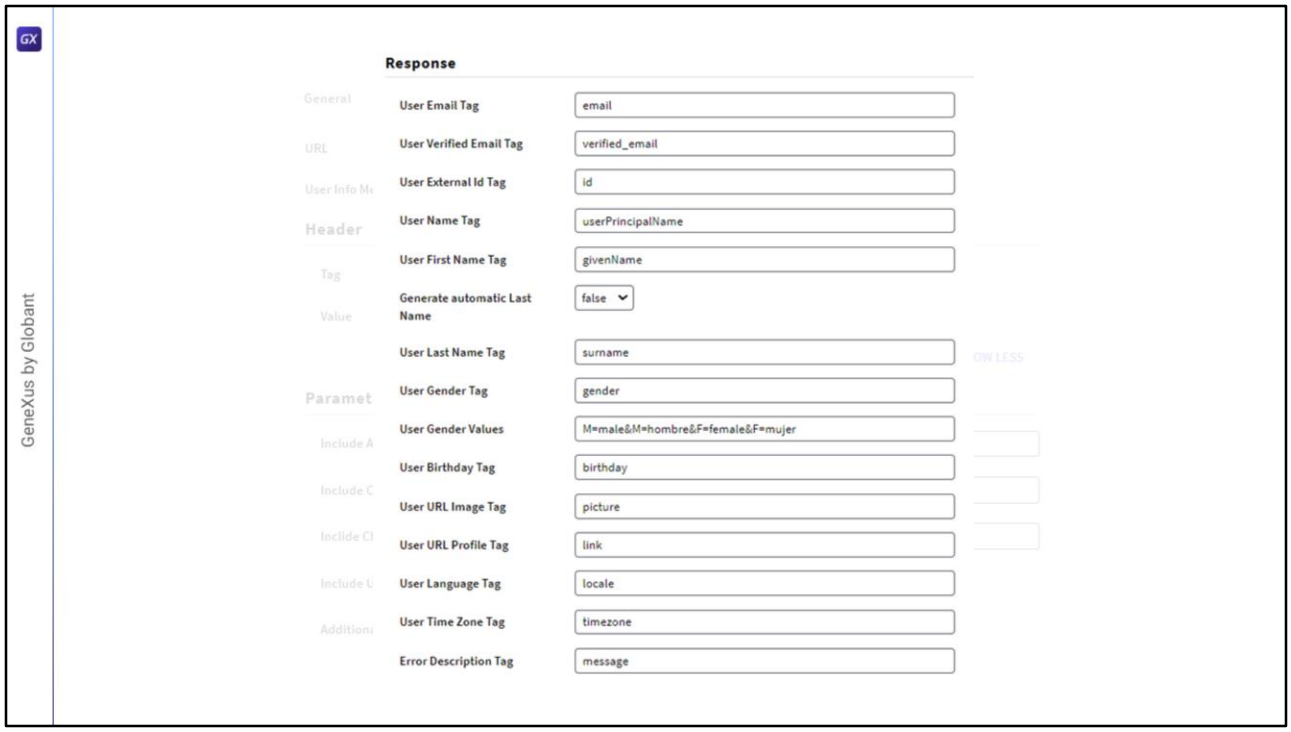


Ahora nos dirigimos a la pestaña Authorization.
Acá debemos setear la URL de Azure obtenida desde su portal, la cual luce de la siguiente manera.
Lo segundo a modificar, debe ser el Response type, el cuál debe contener la URL que vemos en pantalla.

El resto queda todo por defecto.

Ahora tenemos la pestaña Token.
Aca nuevamente seteamos la URL de Azure obtenida desde su portal, y el resto lo dejamos por defecto, excepto los campos Grant Type y Additional Parameters, los cuales seteamos con lo que vemos en pantalla.

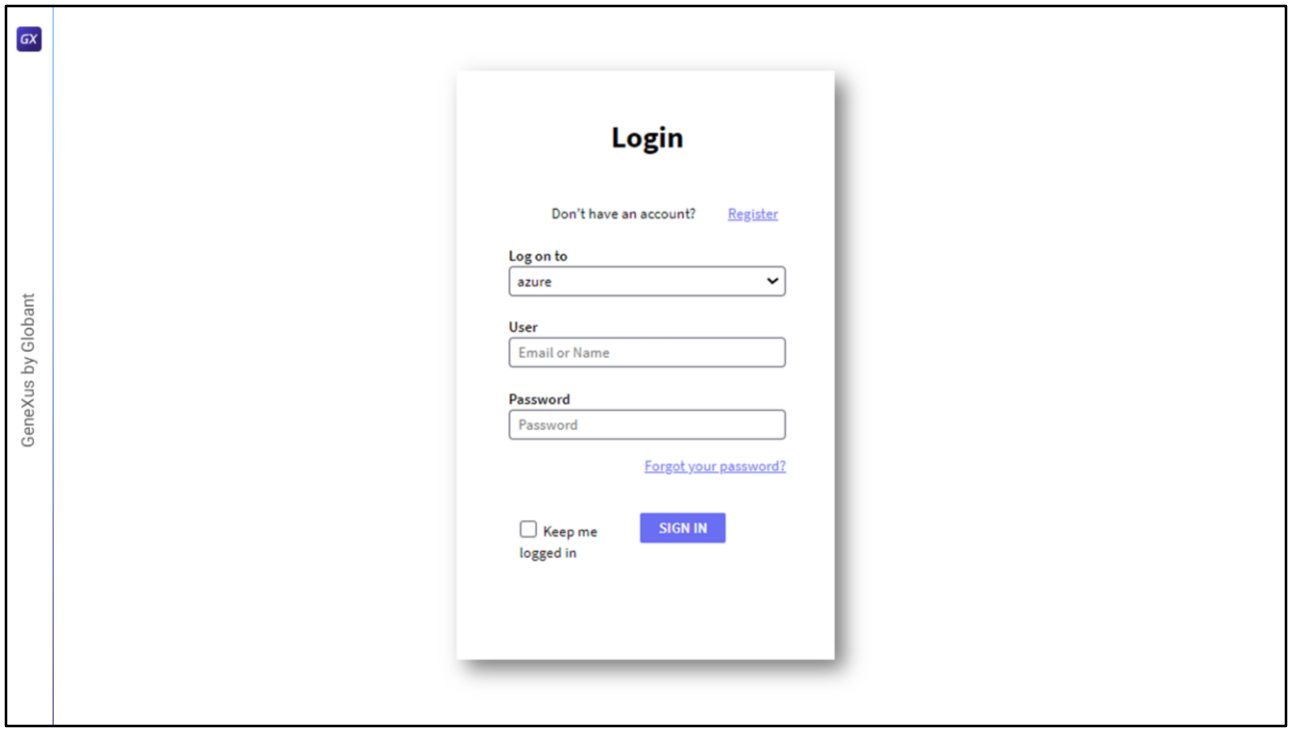
Cabe aclarar que esto último solo se debe cambiar cuando queremos que no se redirija al momento de loguearse y se realice desde el login de GAM.
En caso contrario, el Grant Type se debe dejar con el valor por defecto (que es *authorization_code*) y sin parámetros adicionales.



Finalmente, en la pestaña User Information seteamos la URL que vemos en pantalla (también obtenida desde Azure) y no incluimos nada. Por defecto se incluye el Access Token.
Los campos de Response deben quedar con los siguientes valores.

De esta forma es como se crea el usuario en el GAM local, y es desde donde se mapea la obtención de la información del usuario según el IDP configurado.
El IDP debe retornar un identificador único de usuario, el cual se debe configurar en "User External Id Tag", y por este es que en los sucesivos login de GAM se tiene certezas de que se está autenticando el mismo usuario.

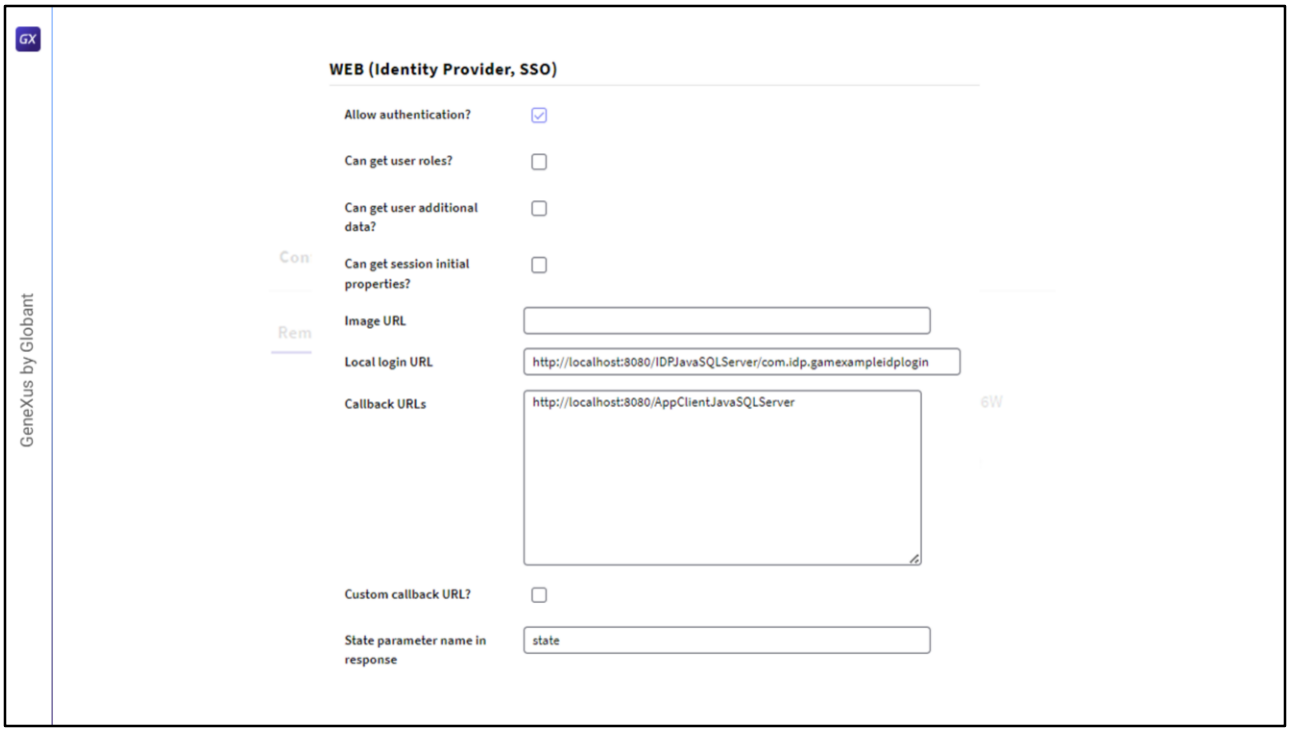
Con esto concluimos la configuración.



Ahora simplemente basta con ir al Login, seleccionar loguearnos con el tipo de autenticación recién creado e ingresar las credenciales de un usuario definido en Azure.

Eso es todo.

Segunda demo: IDP.



Para esta demo, volveremos a utilizar el protocolo Oauth 2.0. El GAM a través de él, será nuestro proveedor de identidad.
En primer lugar, debemos configurar nuestra aplicación GAM definida en el servidor del IDP que interactuará como el proveedor.

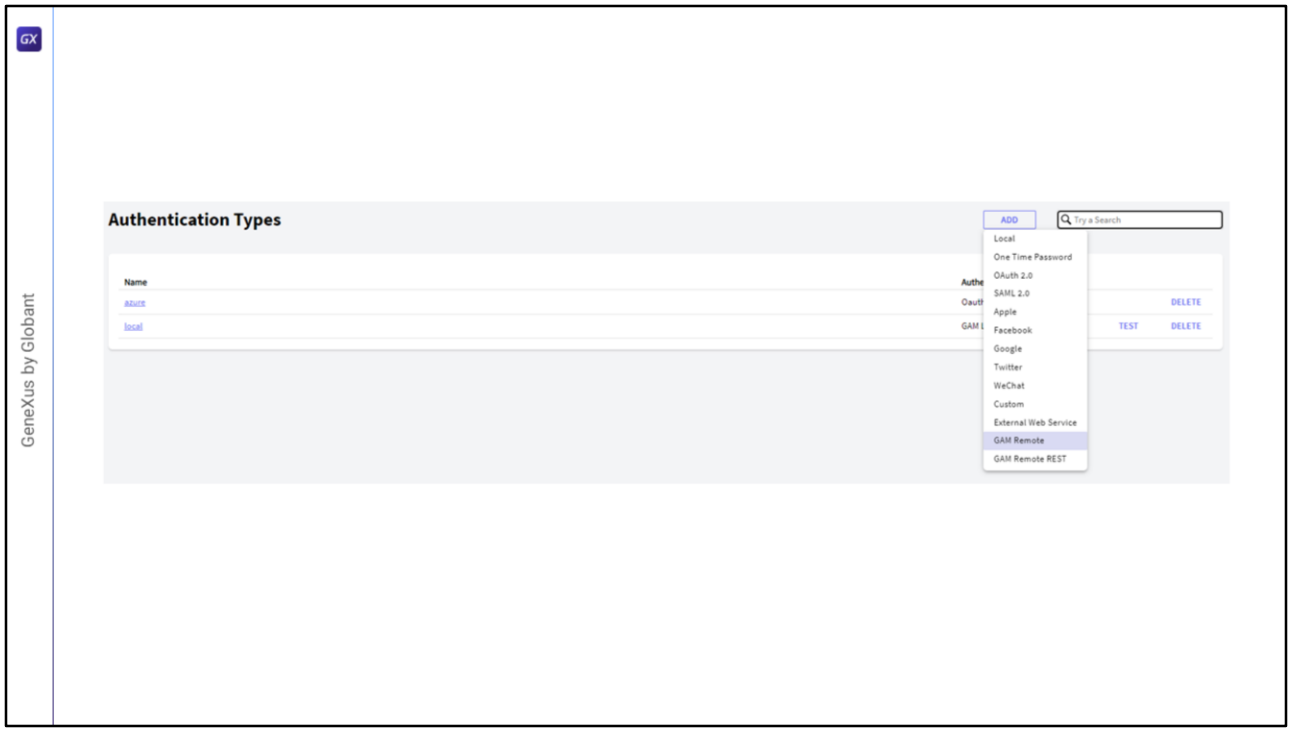
Para esto, debemos dirigirnos a la pestaña “Remote Authentication” en las configuraciones de la aplicación desde el Backend de GAM.
Acá nos guardamos el Client ID y Client Secret para setearlos luego en la aplicación cliente.
Luego, debemos marcar la opción de permitir la autenticación en el apartado WEB (Identity Provider, SSO). Allí, se puede indicar si se quiere compartir con el Cliente los roles de los usuarios, la información adicional, etc.

Lo importante a mostrar en la demo, son las URL de login local y callback.
La primera debe corresponder a la URL del login de la aplicación servidor, que, en nuestro caso, utilizaremos el web panel de ejemplo que nos proporciona GAM llamado **GAMExampleIDPLogin**, el cual será quien realice el proceso de login en el IDP. Una observación a destacar es que en versiones anteriores a GeneXus 18 se utiliza el Web Panel GAMRemoteLogin en vez del GAMExampleIDPLogin que utilizamos en la demo.
La segunda, debe ser el path de la aplicación cliente desde donde se invocará al IDP la cual será llamada después de que haya terminado el proceso de Login. Este último

parámetro puede estar compuesto por más de una URL, las cuales deben ser separadas por punto y coma.

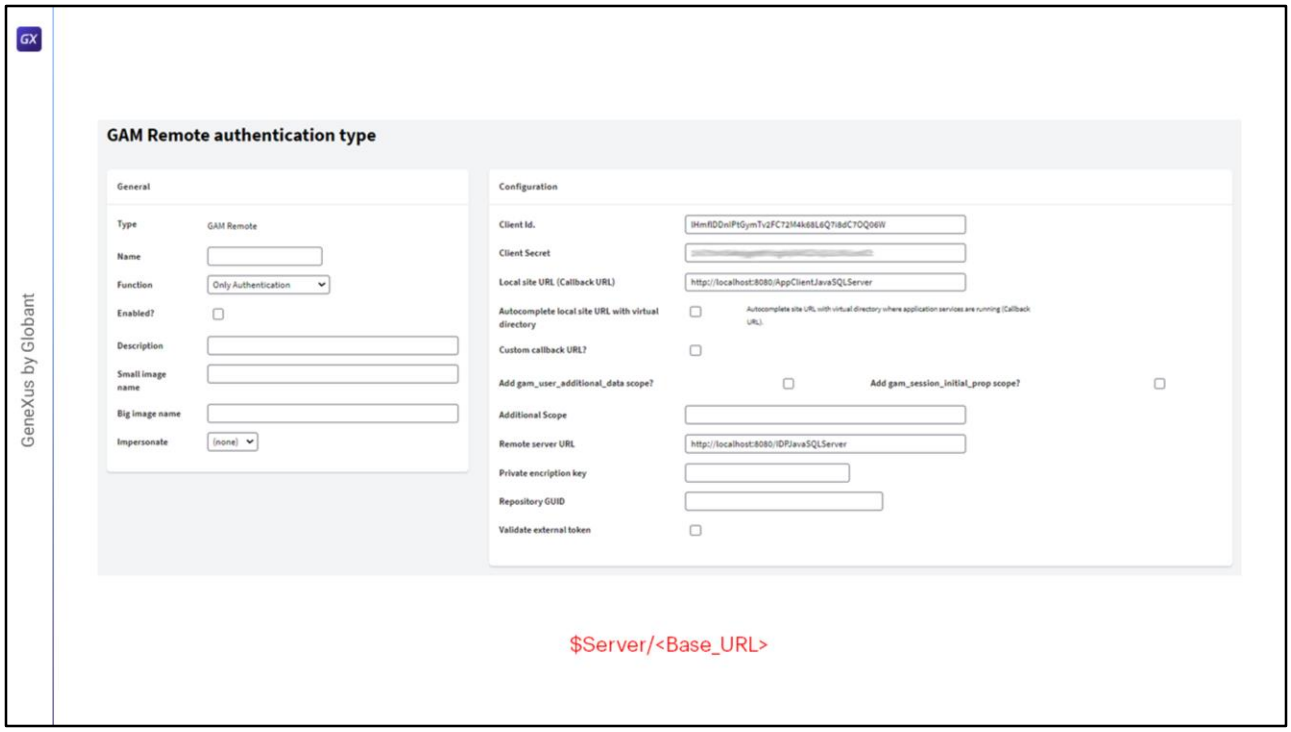
Obviamente es en este GAM donde se deben tener definidos los usuarios que serán utilizados para loguearse en el IDP.

Teniendo esta configuración y un usuario creado, ya finalizamos el proceso desde el lado del IDP.



Veamos el lado del Cliente.

El primer paso es dirigirnos a Tipos de Autenticación desde el menú del backend de GAM, y crear uno de tipo GAM Remote.



Lo importante a configurar aquí es lo siguiente:

La propiedad Function, la setearemos en Only Authentication dado que en el lado del servidor de IDP no le indicamos que se compartan los roles de usuario. En caso de que pongamos la otra opción, Autenticación y roles, al momento de loguearnos nos devolverá un error.

Lo siguiente a configurar, son el Client Id y Client Secret que nos guardamos desde el IDP.

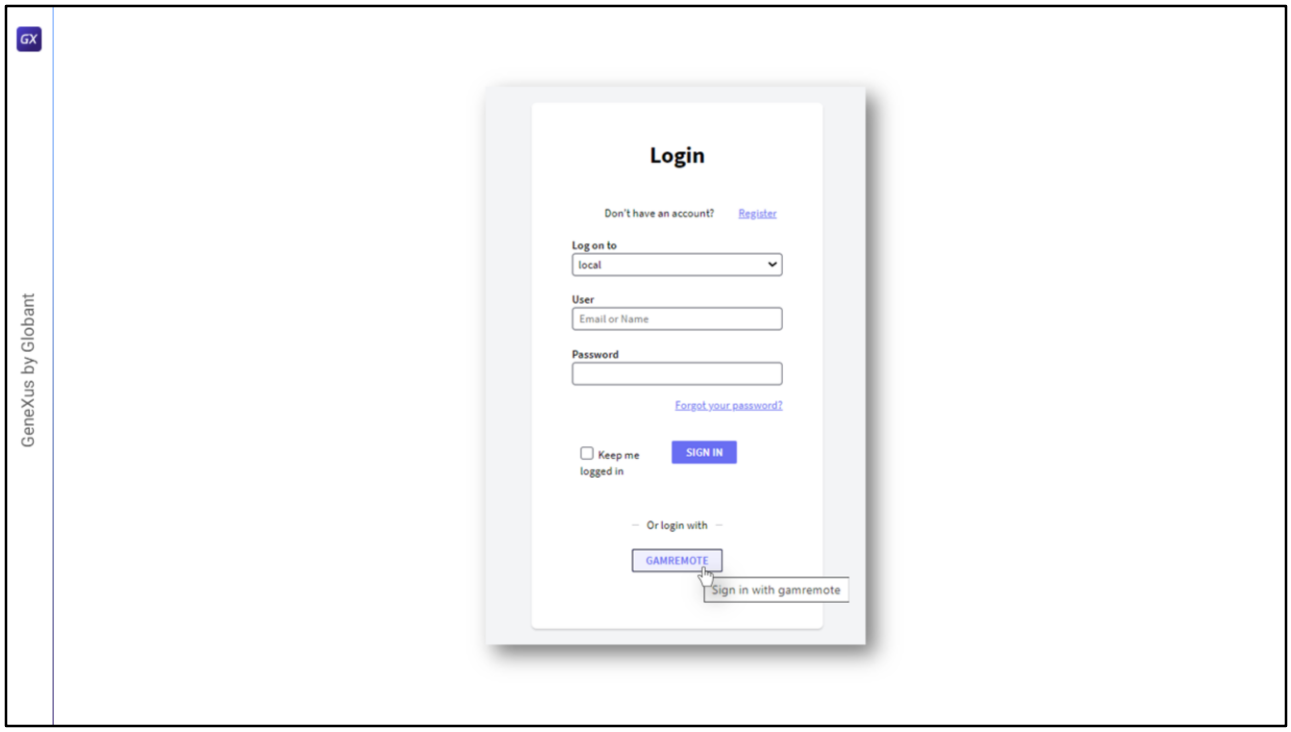
Posteriormente configuraremos la propiedad “Local site URL” con la dirección de nuestra aplicación cliente, la misma que ya especificamos en la Callback URL en el servidor; también la propiedad “Remote server URL” con la dirección del IDP, siguiendo el formato que vemos en rojo.

Como comentarios adicionales:
La propiedad “Add gam_user_additional_data scope?” debe activarse cuando queremos pasar datos adicionales del usuario. En el lado del servidor, se debe seleccionar la propiedad Permitir autenticación, en la sección Web (Proveedor de identidad, SSO).

La propiedad “Add gam_session_initial_prop scope?” consiste en pedirle al IDP que

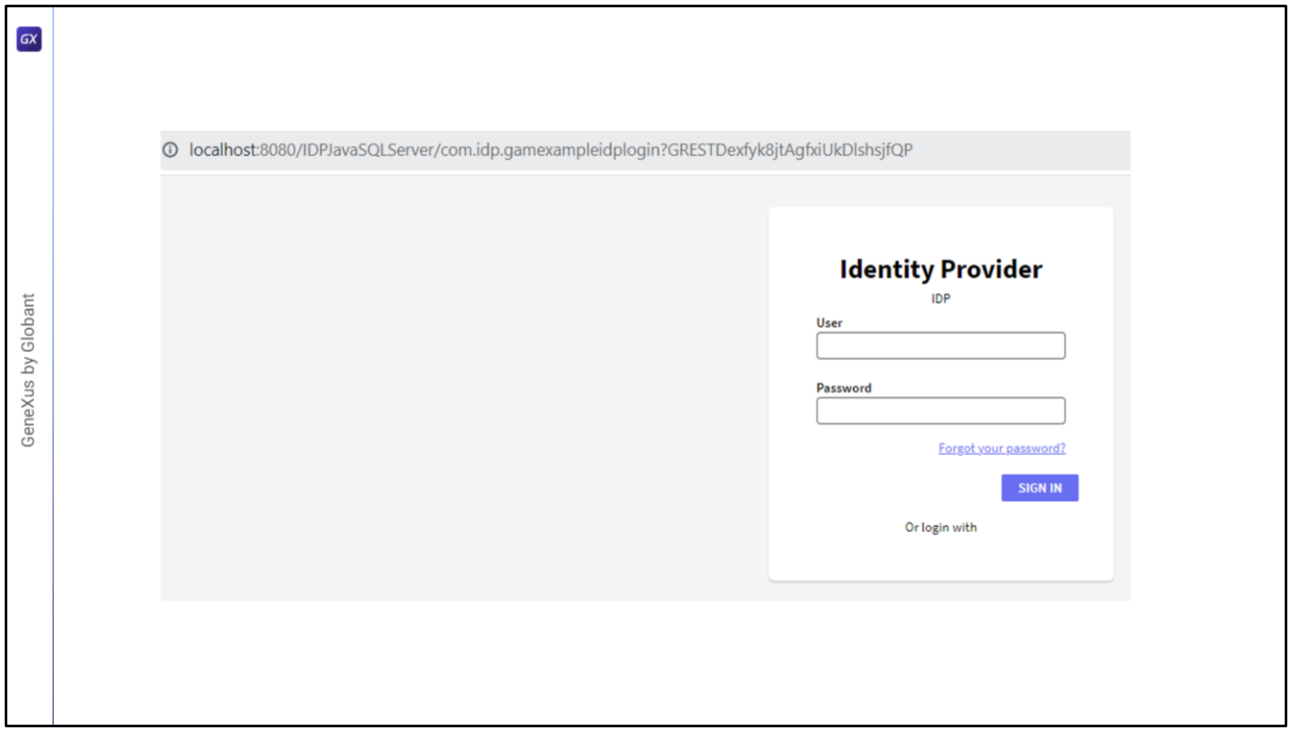
devuelva al cliente las propiedades iniciales establecidas dinámicamente en el inicio de sesión. Por supuesto, en el IDP también se debe configurar que se envíe esta información.

Finalmente, la propiedad “Validate External Token” valida el vencimiento de la sesión según la expiración del token y lo renueva automáticamente sin tener que realizarlo nosotros manualmente.

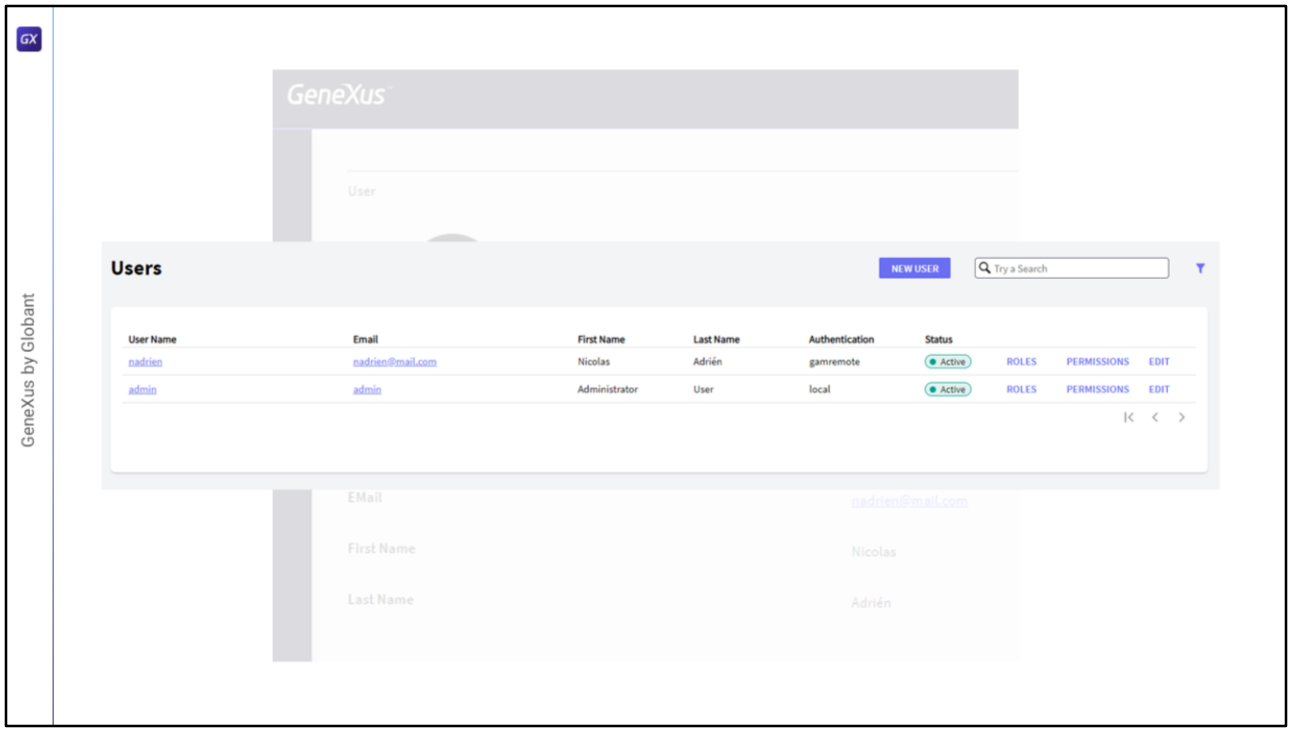


A efectos de la demo, creamos un Web Panel en la aplicación Cliente, donde nos muestra los datos del usuario logueado. Obviamente este objeto tiene activado la seguridad integrada con valor Autenticación.

Cuando queremos acceder a él, dado que no estamos logueados nos redirige al login. Veamos que ahora que tenemos definido el tipo de autenticación Oauth, desde el login tenemos la opción de acceder a través de esa misma.

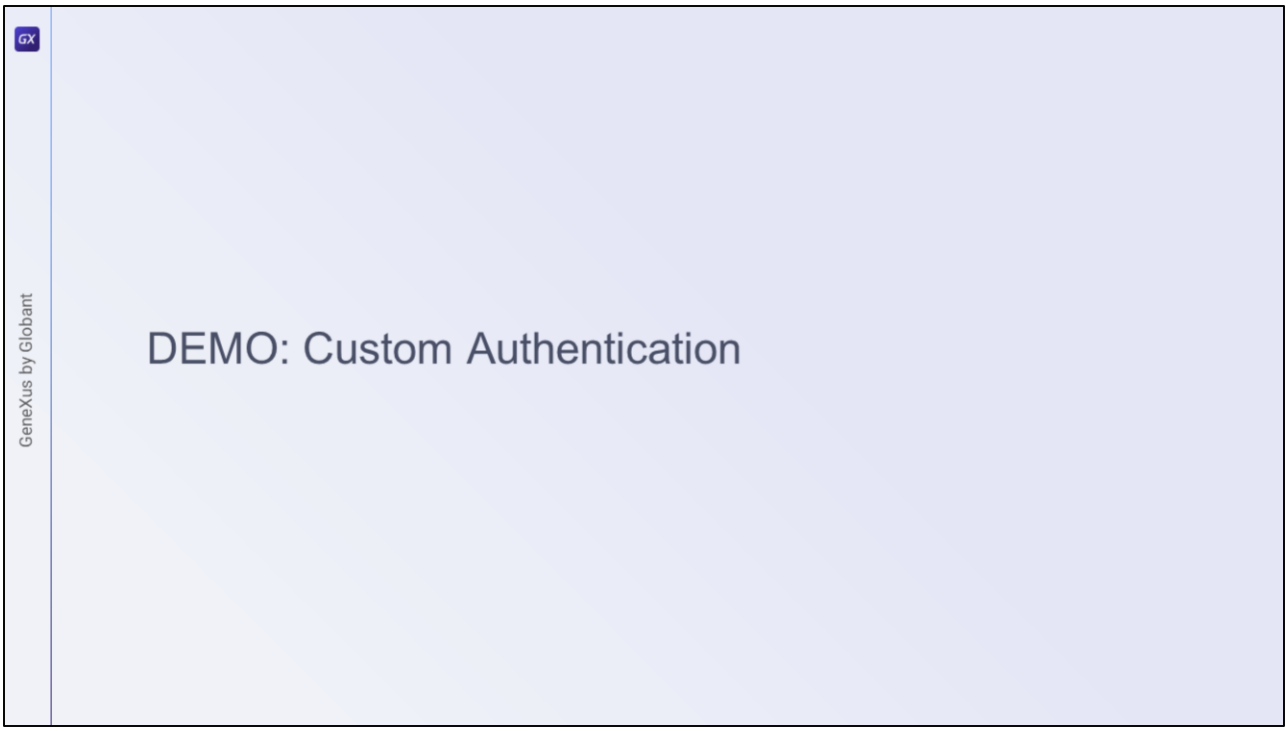


Al hacer clic en esa opción, vemos que nos redirige al IDP y su login remoto. Procedemos a loguearnos con el usuario que habíamos definido en el IDP.



Efectivamente vemos como ahora nos redirigió a nuestro Web Panel con la información del usuario logueado.

Si nos dirigimos al Backend de la aplicación cliente, podemos ver cómo fue creado el usuario que habíamos creado en el IDP con su información.



Tercer demo: Autenticación Custom.

```
Parm(in:&StrInput, out:&StrOutput); //&StrInput and &StrOutput are varchar(256)


&Key = '03E1E1AAA5BCA19FBA8C42058B4ABF28'
&GAMWSLoginIn.FromJson(&StrInput) // &GAMWSLoginIn is &GAMWSLoginInSDT data type

//Decrypt parameters
&UserLogin = Decrypt64( &GAMWSLoginIn.GAMUsrLogin, &Key )
&UserPassword = Decrypt64( &GAMWSLoginIn.GAMUsrPwd, &Key )
&GAMWSLoginOut = New GAMWSLoginOutSDT() //&GAMWSLoginOut is &GAMWSLoginOutSDT data type
&GAMWSLoginOut.WSVersion = GAMAutExtWebServiceVersions.GAM10
&GAMWSLoginOut.User = New GAMWSLoginOutUserSDT()

Do 'ValidUser'

&StrOutput = &GAMWSLoginOut.ToJson()

Sub 'ValidUser'
    If &UserLogin = !"user"
       If &UserPassword = !"password"
          &GAMWSLoginOut.WSStatus = 1
          &GAMWSLoginOut.User.Code = !"code"
          &GAMWSLoginOut.User.FirstName = !"FirstName"
          &GAMWSLoginOut.User.LastName = !"LastName"
          &GAMWSLoginOut.User.EMail = !"name2@domain.com"
          Do 'GetRoles' //optional
       Else
          &GAMWSLoginOut.WSStatus = 3
       EndIf
    Else
       &GAMWSLoginOut.WSStatus = 2
    EndIf
EndSub

Sub 'GetRoles'
    &GAMWSLoginOutUserRol = New()
    &GAMWSLoginOutUserRol.RoleCode = "role_1"
    &GAMWSLoginOut.User.Roles.Add(&GAMWSLoginOutUserRol)
    &GAMWSLoginOutUserRol = New()
    &GAMWSLoginOutUserRol.RoleCode = "role_2"
    &GAMWSLoginOut.User.Roles.Add(&GAMWSLoginOutUserRol)
EndSub
```

Para realizar una autenticación Custom, debemos crear un procedimiento.
En la Wiki de GeneXus pueden encontrar el ejemplo que vemos en pantalla, con una lógica muy simple ya definida. Queda a cargo del desarrollador modificarla bajo sus condiciones.

En primer lugar, vemos que como reglas se definen dos Varchar: uno de entrada y otro de salida, los cuales traerán los datos ingresados por el usuario y devolverán el resultado del login con determinada información del usuario (esto último en caso de éxito por supuesto).

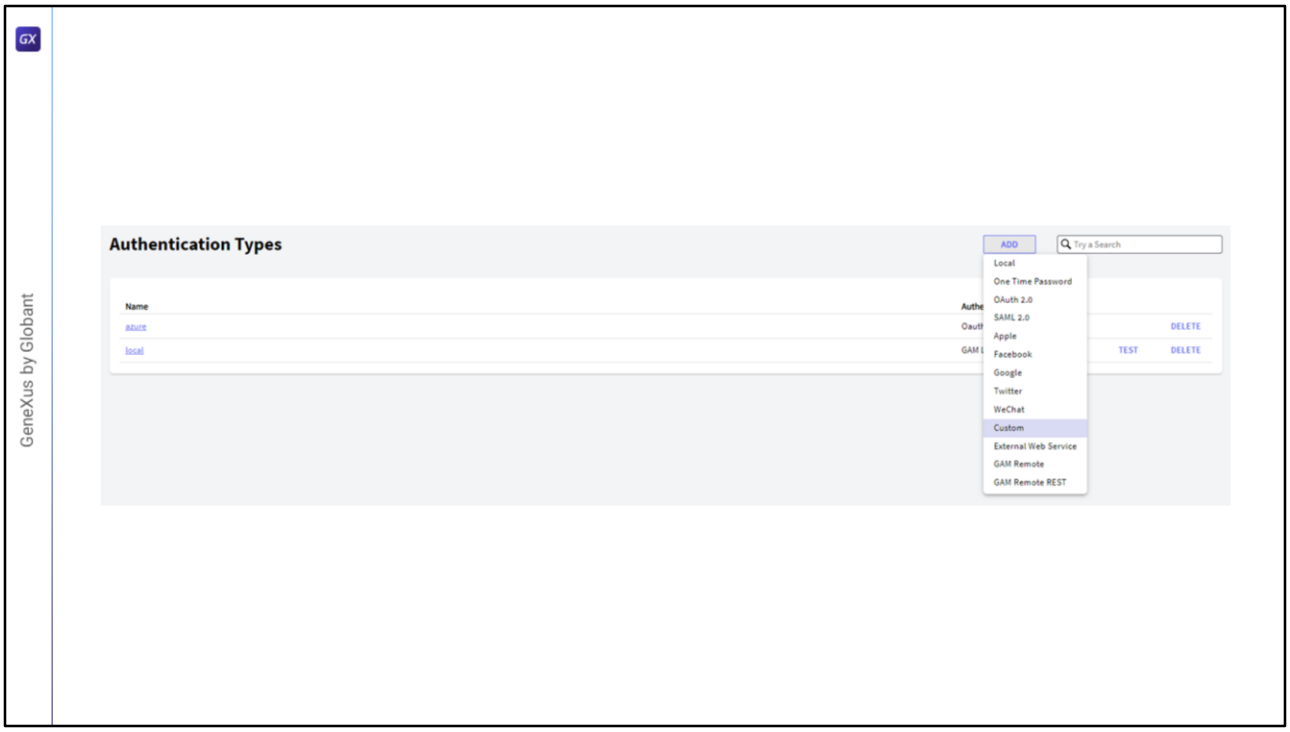
Luego, se define una clave que más adelante entraremos en detalle, y se desencriptan los parámetros de ese parámetro de entrada procedente de las reglas, además de crearse un data type que será cargado en el parámetro de salida al finalizar.

Posteriormente, en el método ValidUser se realiza la validación del nombre de usuario y contraseña, el cual está hecho a modo de ejemplo verificando que el nombre de usuario es “user” y la contraseña es “password”. En caso contrario, se devuelven distintos errores según la falla.
Este método debe cambiarse por una lógica de login más segura y que no haga diferenciación entre los errores según usuario o contraseña.

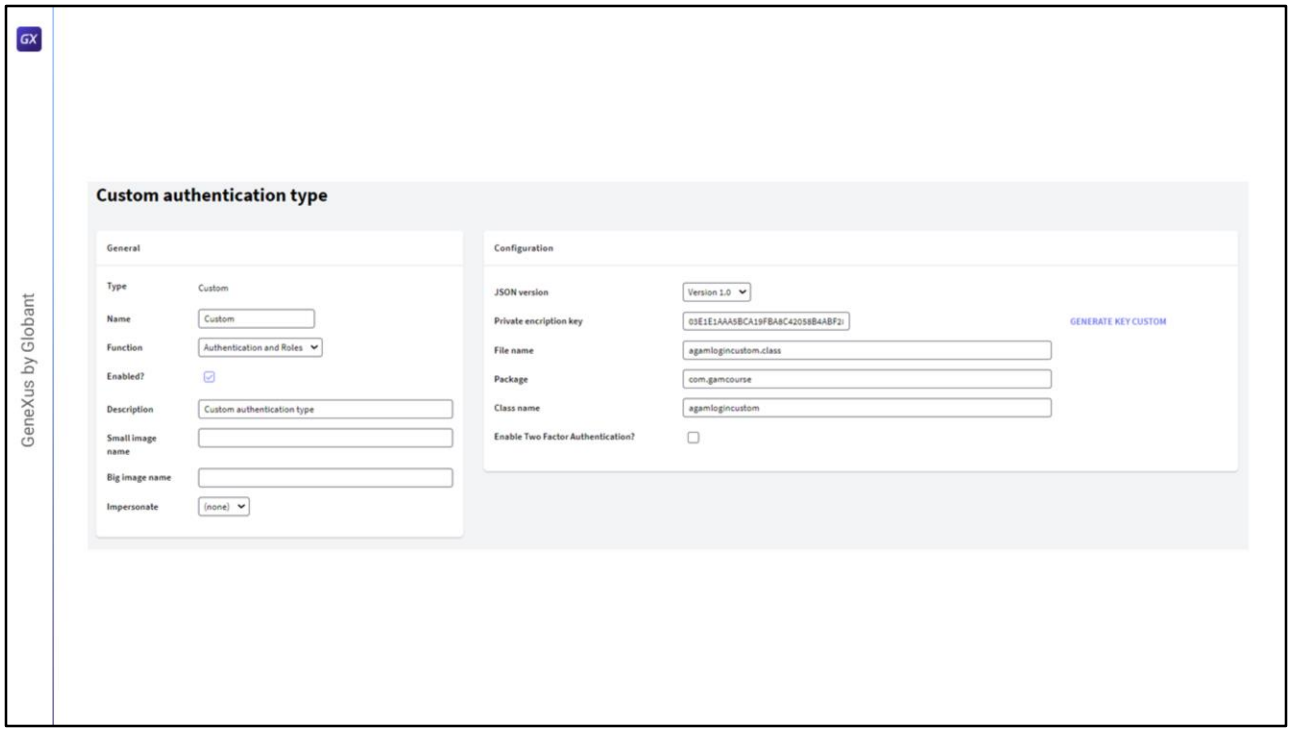
Opcionalmente, se puede utilizar el método GetRoles para definirle determinados roles

al usuario logueado.

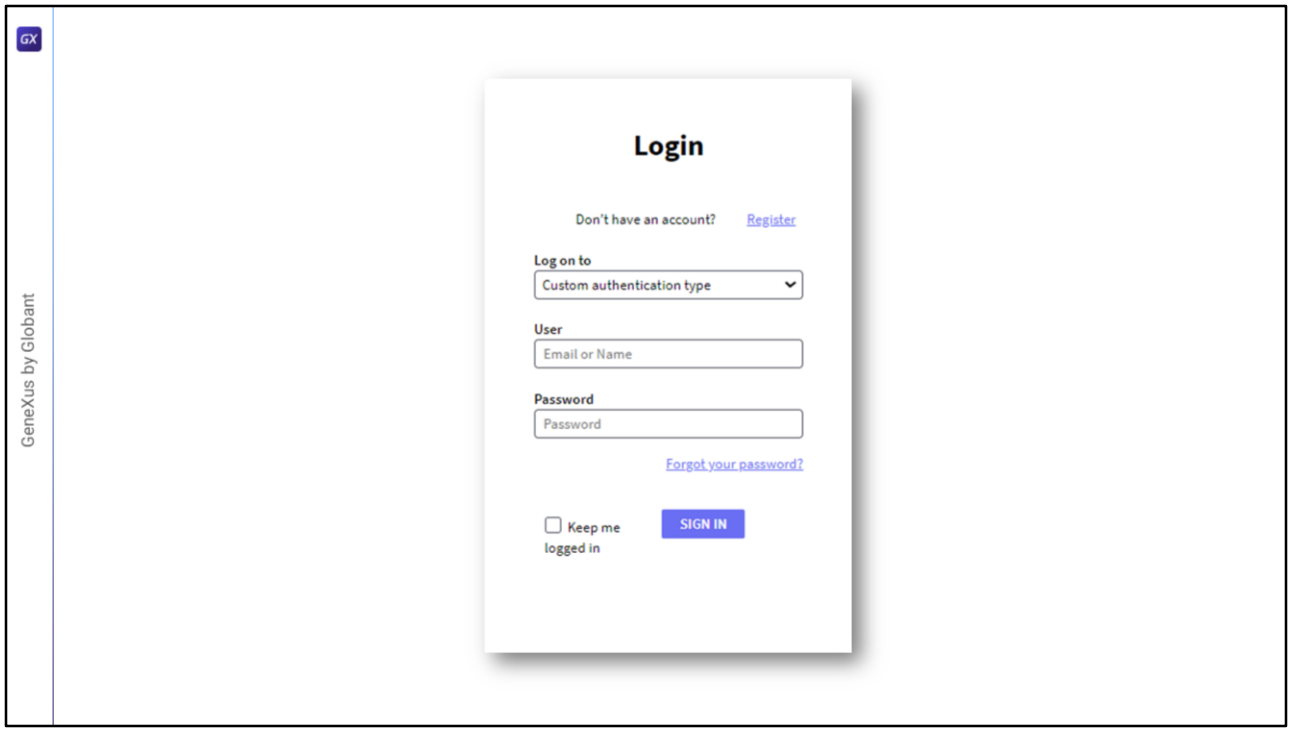
Este método es de utilidad cuando queremos programar nosotros como validamos la contraseña de un usuario, ya sea para validarlo contra una base de datos local, contra un LDAP o contra otro lugar donde se encuentren almacenadas las credenciales de los usuarios.



Ahora que ya contamos con un procedimiento personalizado para la autenticación, debemos configurarlo en GAM.
El primer paso es dirigirse a Authentication Types, y crear uno nuevo de tipo Custom.



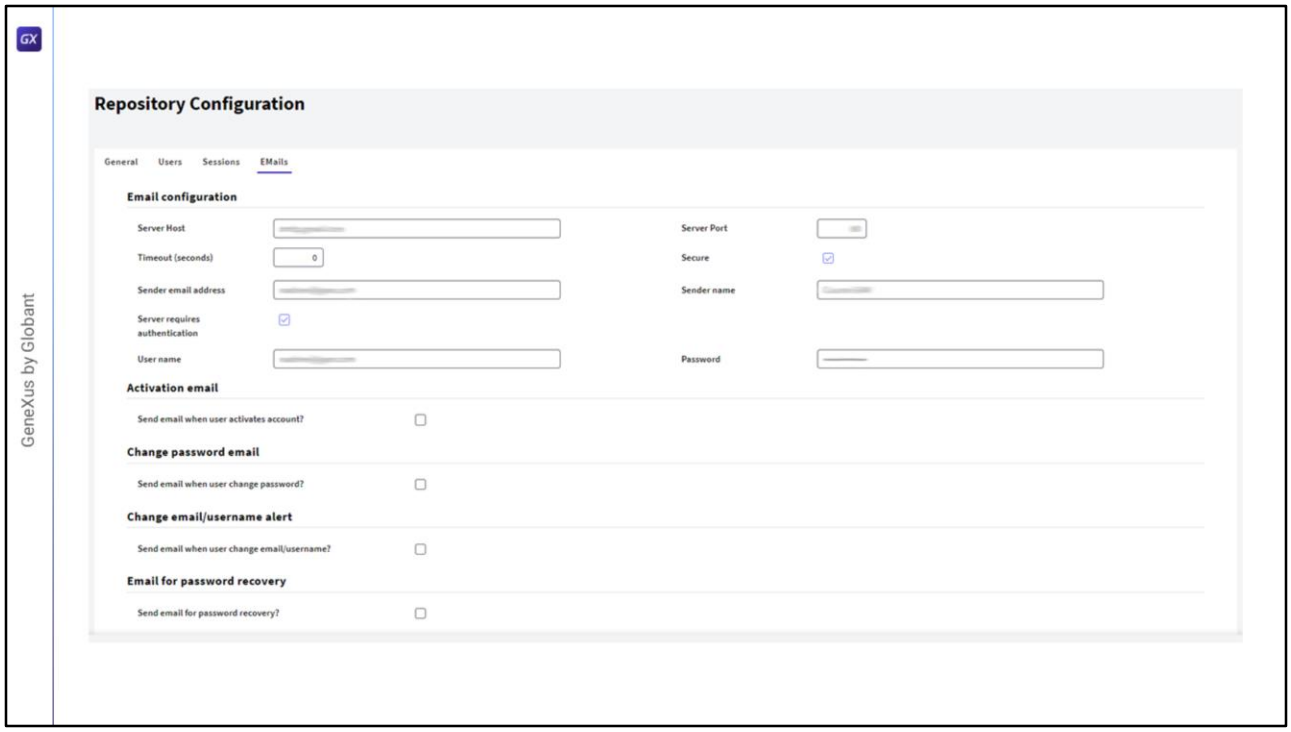
Aquí, las configuraciones a resaltar son las siguientes:
Función: Permite especificar si queremos que el tipo de autenticación sea Autenticación y roles, o solo Autenticación. En nuestro caso dejamos la primera opción.
Private encription key, aquí se debe configurar la clave de cifrado utilizada en el procedimiento para descifrar el usuario y contraseña recibido. Si recuerdan en la slide del procedimiento GeneXus que mostré anteriormente se definía una clave la cual es la que ingresamos en esta propiedad. Esta es útil porque la función de encriptación de GeneXus la utiliza para cifrar el nombre de usuario y contraseña cuando se pasan al programa.
Nombre de archivo: aquí se especifica el nombre del archivo que corresponde al procedimiento externo. En el caso de Java es opcional.
Paquete: aquí se especifica en el caso de modelos Java el mismo valor de propiedad de nombre de paquete de Java, y en el caso de modelos .NET el valor de la propiedad de espacio de nombres de la aplicación. Esta propiedad es opcional y depende si el procedimiento o programa utilizado tiene paquete o no.
Finalmente, Nombre de la clase, que es una propiedad obligatoria que especifica el nombre de la clase del procedimiento.



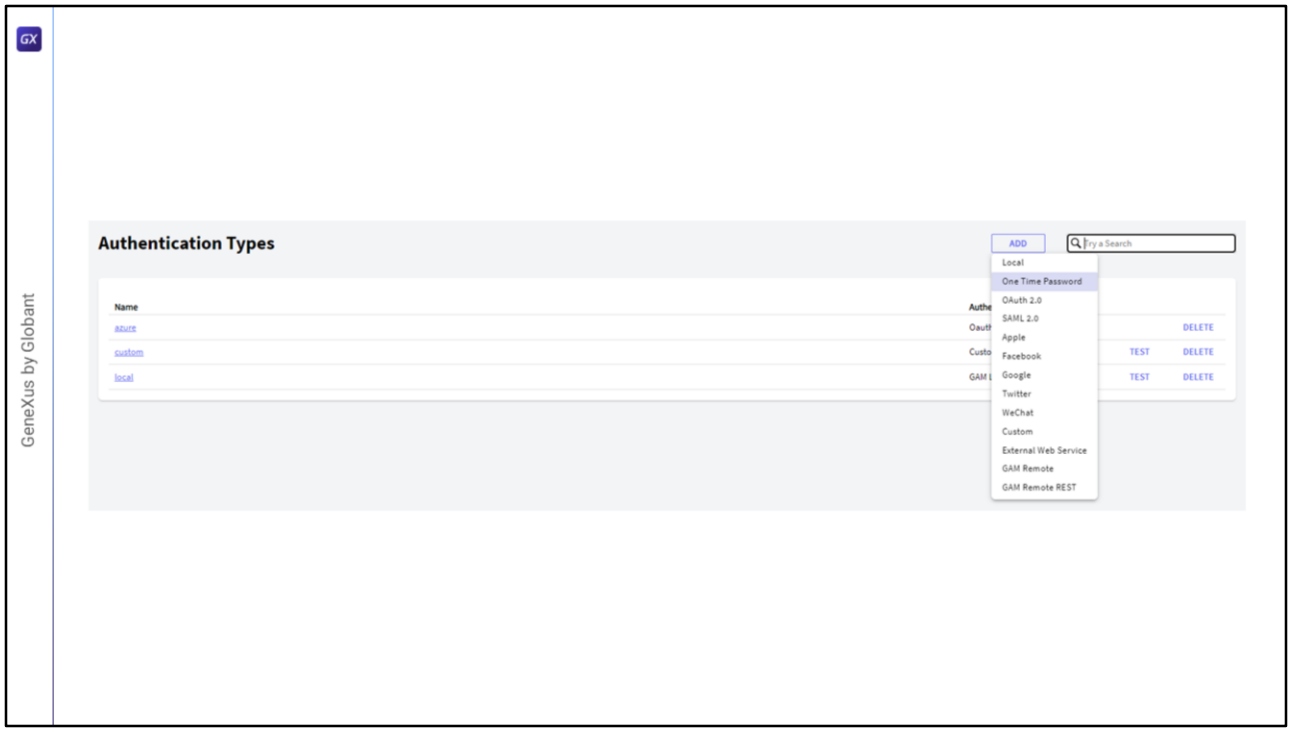
Luego de tener todo configurado, simplemente en nuestro login se setea el tipo de autenticación custom, y se realiza el login.

Un detalle a mencionar es que en este caso el tipo de autenticación se selecciona a través del combo resaltado debido a que le indicamos que no se redireccione al IDP. En caso contrario, el tipo de autenticación se muestra como un icono en la parte inferior del login como veíamos en la Demo del IDP.

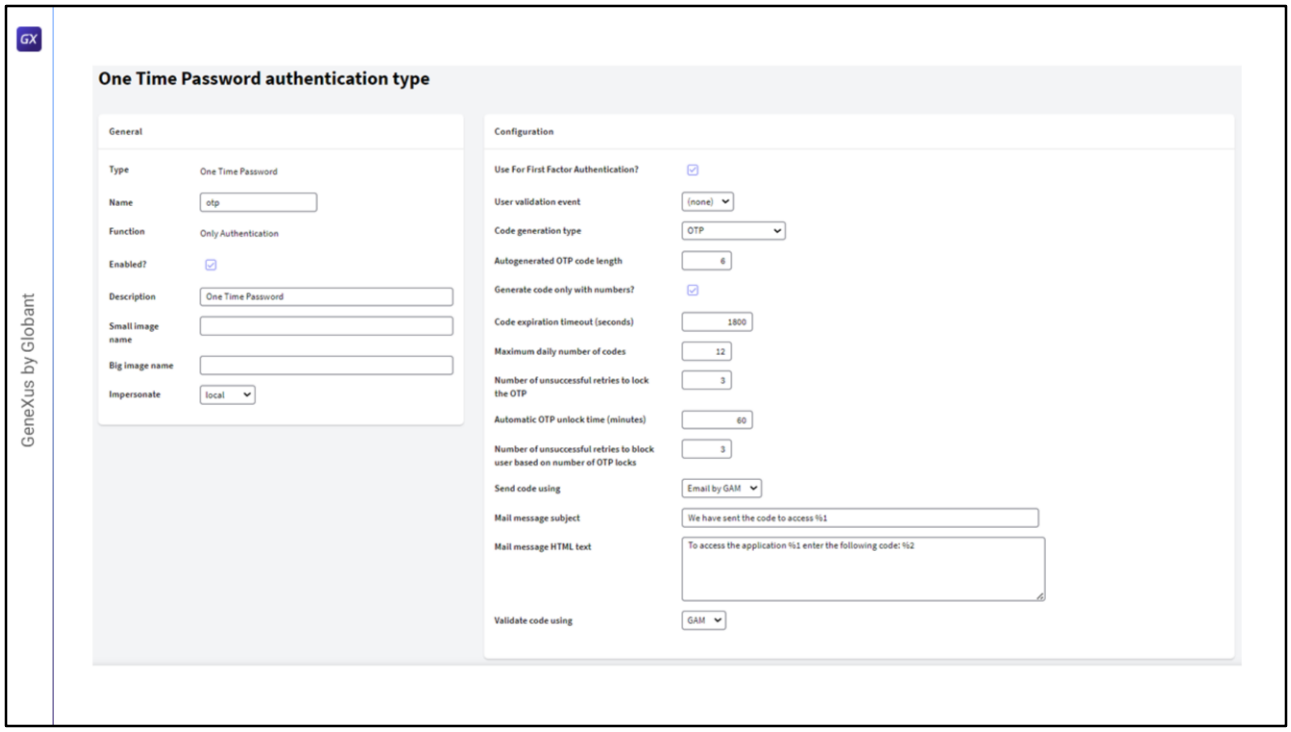
OTP.



Un requisito previo para hacer funcionar OTP, es que el repositorio debe tener configurado el servicio de mail para enviar los códigos.
Esto se configura en la opción “Configuración del Repositorio” del backend de GAM.

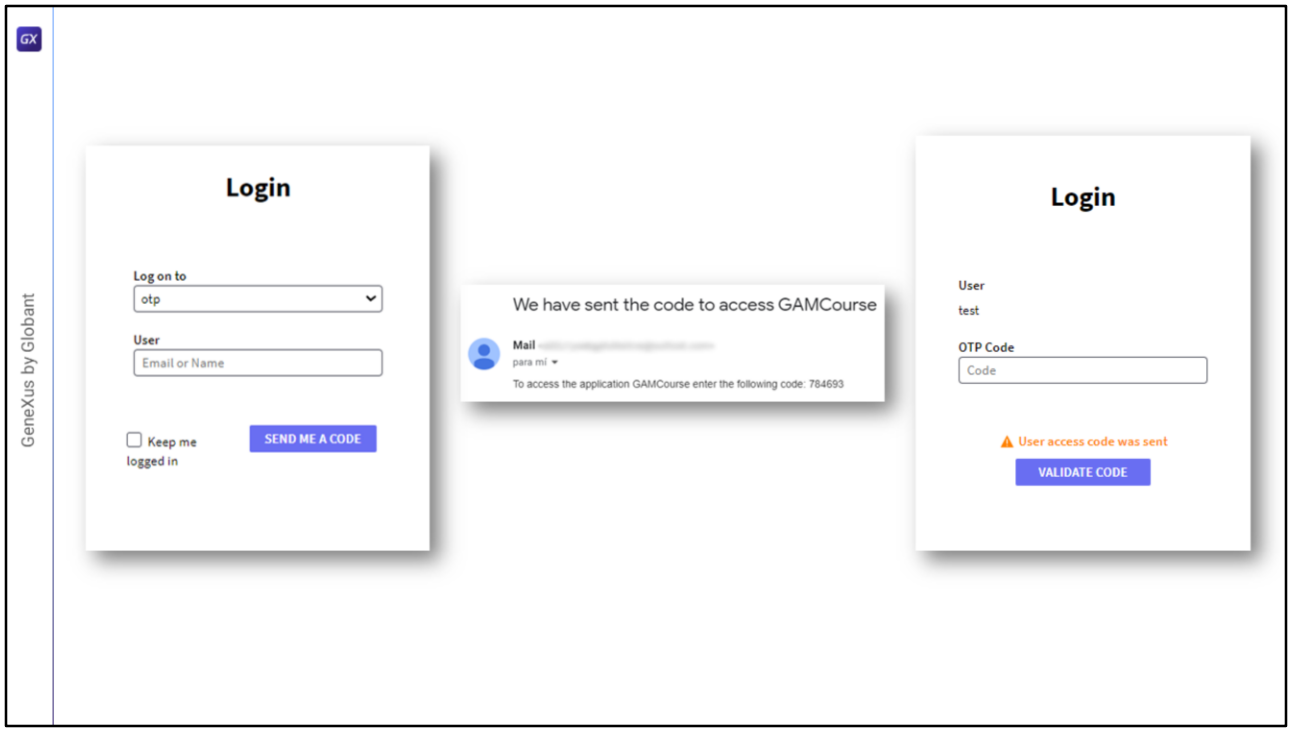


Ahora sí, para definir este tipo de autenticación, todo se realiza y configura nuevamente a través del backend de GAM.
Como en la Demo anterior, nos dirigimos a Authentication Types y damos de alta un nuevo tipo.
En este caso, seleccionamos el tipo One Time Password.

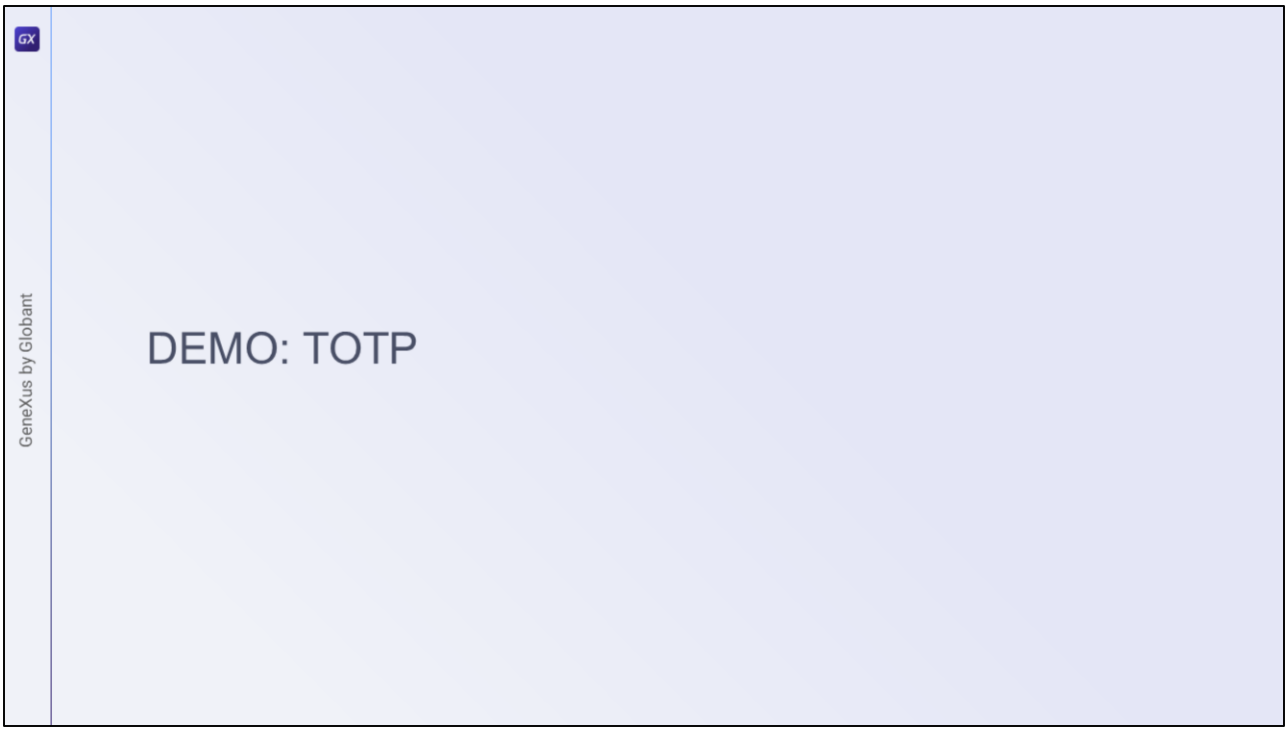


Pasemos a describir las propiedades más importantes:
Impersonate: Aquí se especifica el tipo de autenticación donde los usuarios van a ser validados al usar OTP. Como mencioné anteriormente en el teórico de esto, los usuarios ya deben existir. Este es el único tipo de autenticación que se requiere configurar esta propiedad ya que los usuarios deben existir en la base de datos de GAM.
Usar como autenticación de primer factor: Si no se configura esta propiedad, OTP solo podría usarse como un segundo factor. En nuestro caso, lo habilitamos.

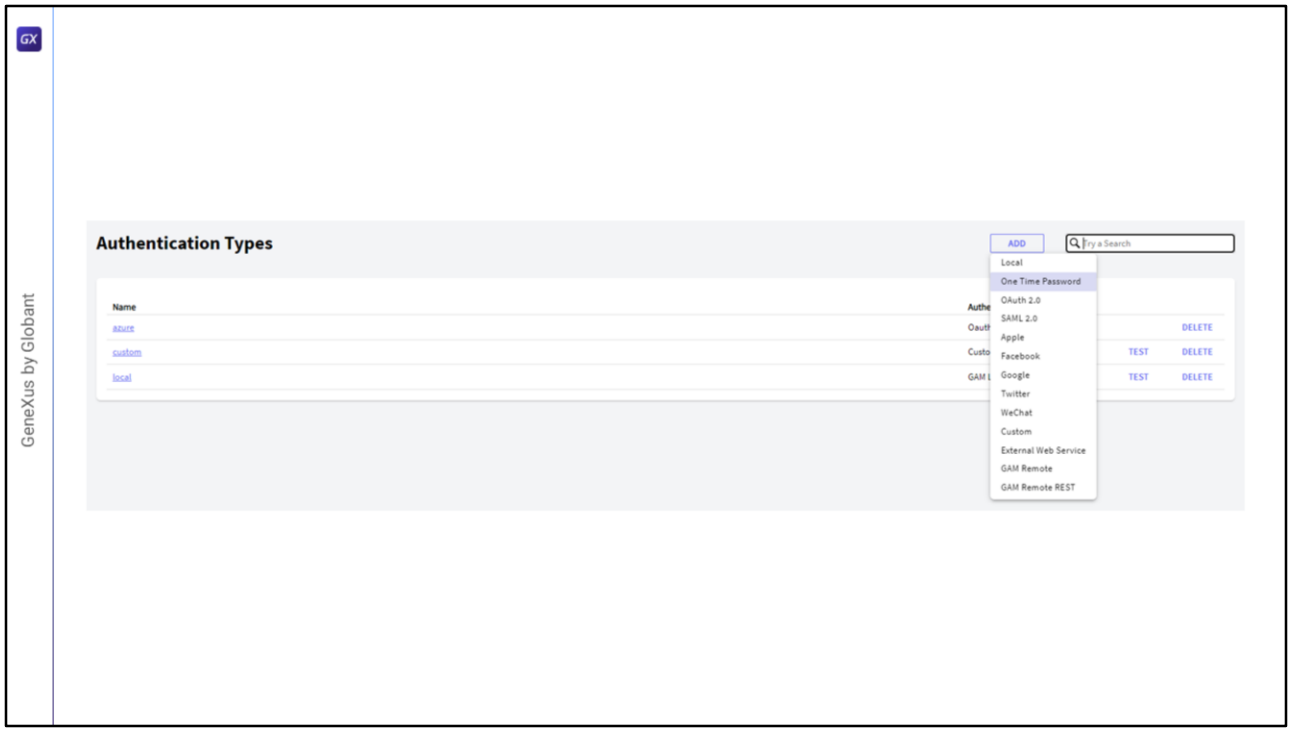
El resto de las propiedades, se utilizan para definir propiedades del código a enviar y el formato del mail.
En nuestro caso usaremos el por defecto de GAM que es el mail, pero recordar que existe la posibilidad del envío del código a través de un SMS. Si el desarrollador decide optar por esta segunda forma, debe implementar y configurar el evento de GAM que luego debe seleccionar en la propiedad "Send code using“.



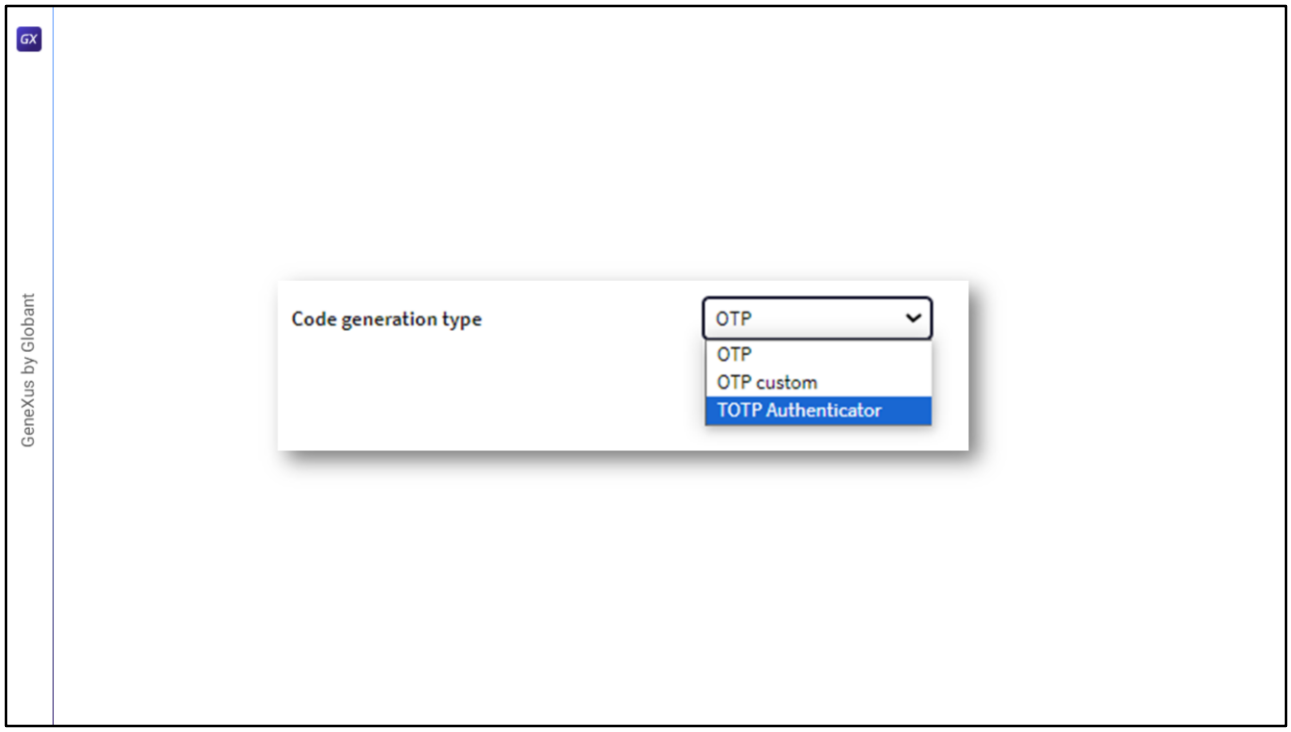
Finalmente yendo al Login, seleccionamos el tipo OTP, donde vemos que nos pedirá solamente el nombre de usuario para el envío de código.
Luego de llegado el código a través del mail, simplemente se ingresa en el login para autenticarse en el sistema.



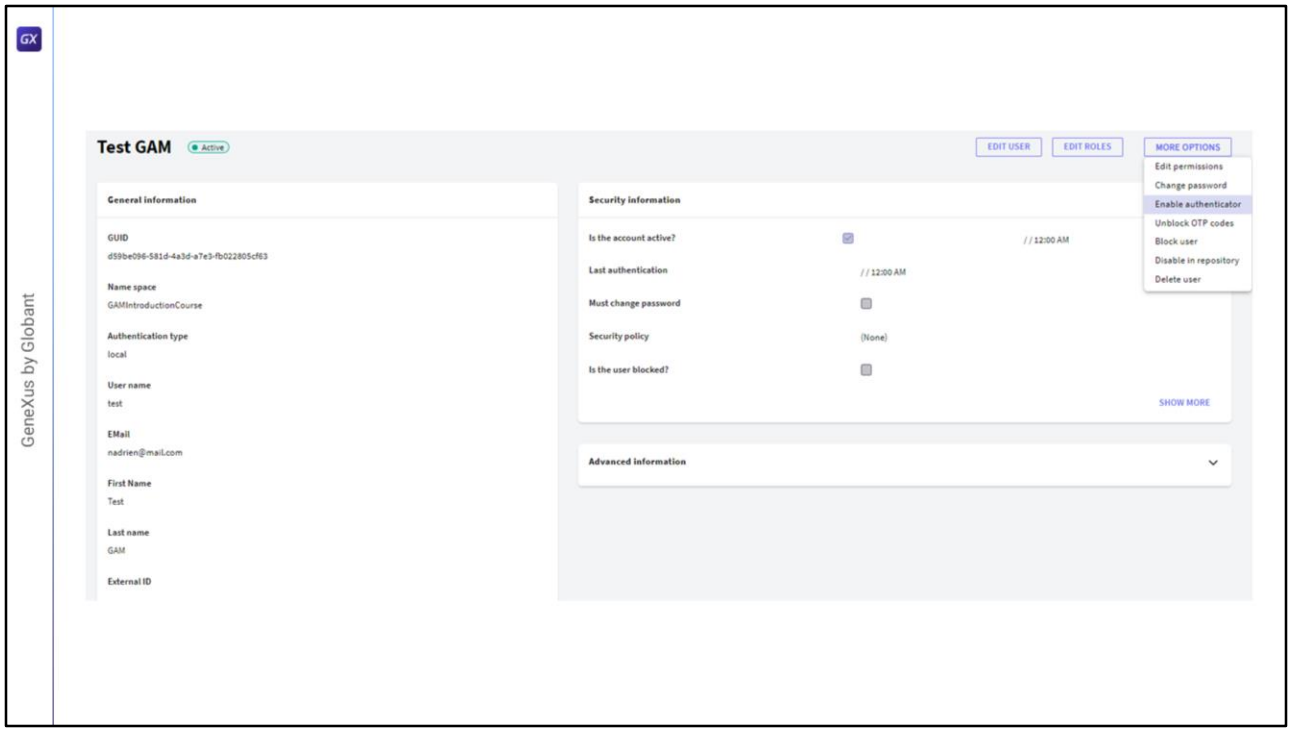
En esta demo, los pasos para configurar un nuevo tipo de autenticación TOTP son prácticamente los mismos que OTP, excepto por una salvedad.



Para definir este tipo de autenticación, nuevamente nos dirigimos a Authentication Types y damos de alta un nuevo tipo.
En este caso, también seleccionamos el tipo One Time Password.



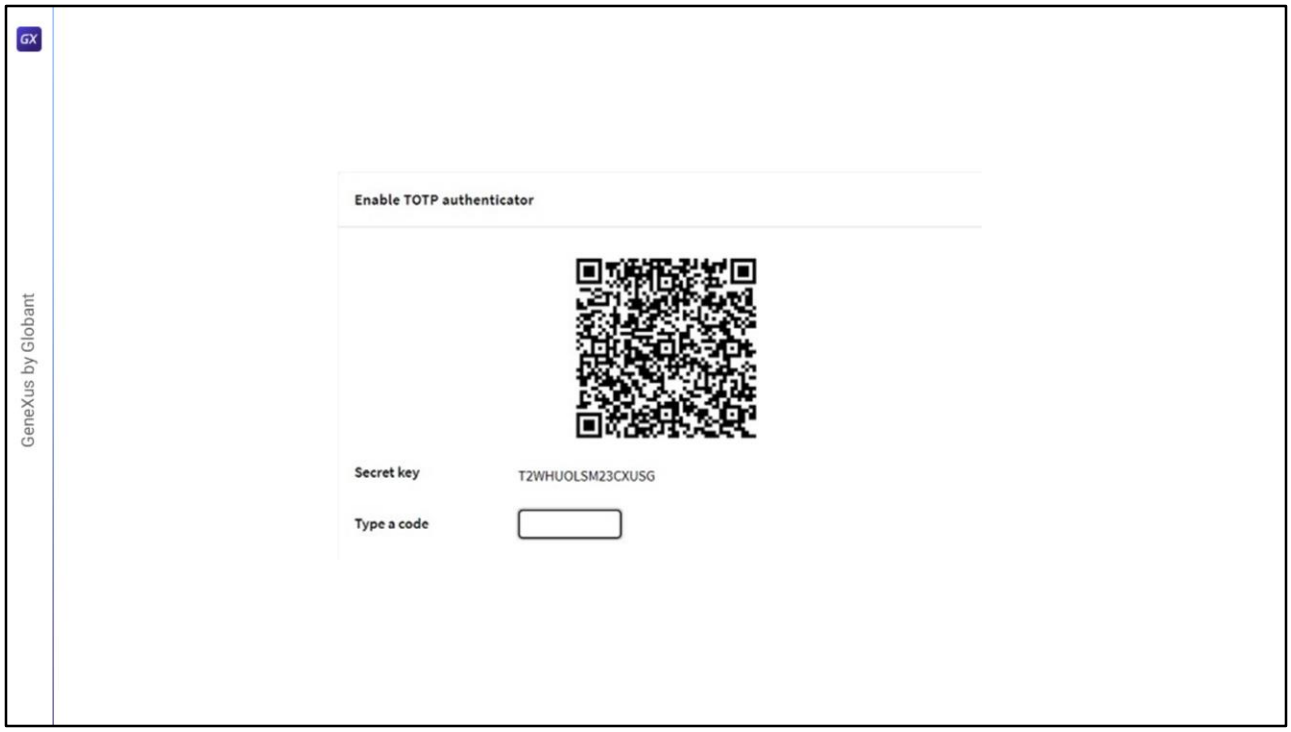
La diferencia con OTP es la propiedad mostrada en pantalla, donde en este caso elegimos TOTP Authenticator.
El resto de las propiedades son para configuraciones del código que no vienen al caso.



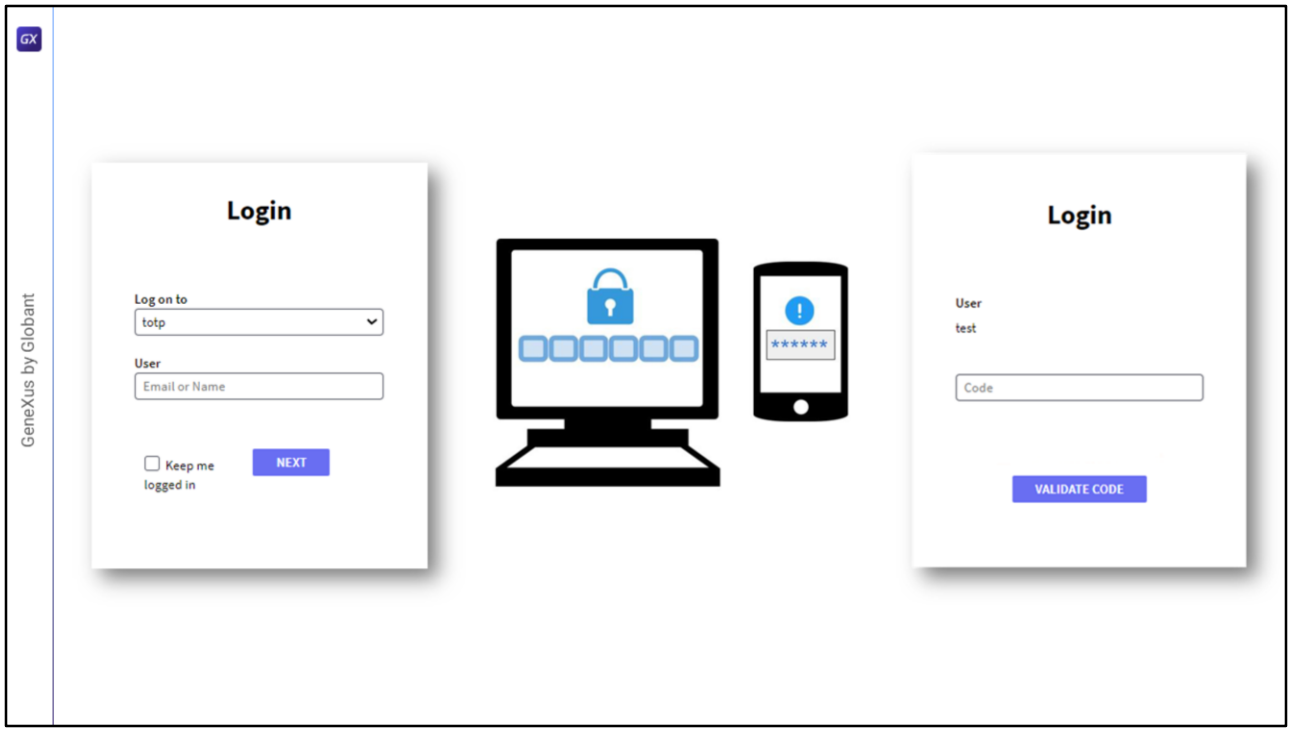
Veamos la salvedad más importante que tiene con OTP.

Cada usuario debe activar la autenticación a través de sus configuraciones. La diferencia más importante es que este algoritmo del código está basado en el tiempo y los códigos los generan las diferentes aplicaciones autenticadoras.

A efectos de la demo, se creó utilizando el usuario administrador del backend de GAM para un usuario “test” de una aplicación de ejemplo.
Los pasos a realizar para esta manera se basan en dirigirse al usuario en cuestión y presionar en Enable authenticator.



Una vez allí, se proporcionará el código QR para configurarse en un software o aplicación móvil basada en autenticación con contraseña de un solo uso, donde luego de haberlo leído, nos retornará el código a ingresar en el campo “Type a code”.



Finalmente, el login es igual que todos los tipos anteriores, donde en este caso existe una aplicación intermediaria la cual nos proporciona el código de acceso.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com