

**GeneXus**<sup>™</sup>  
by **Globant**

# MOBILE

Nicolas Adrién



GeneXus™

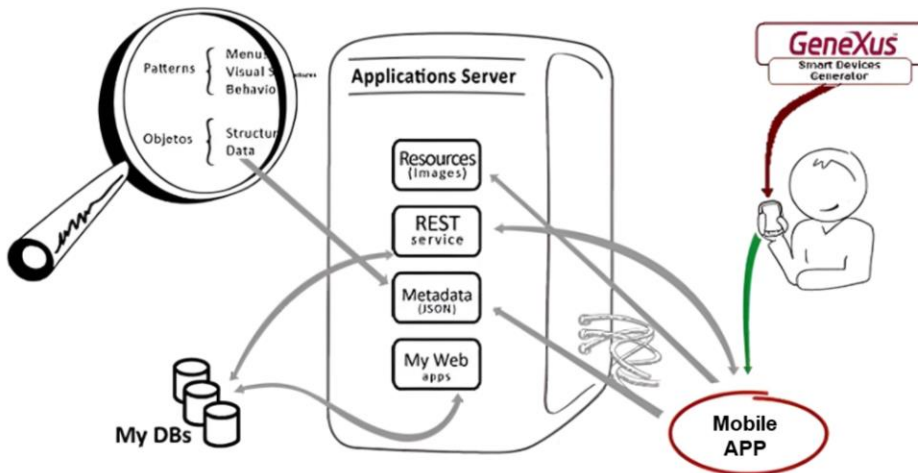
# MOBILE

Authentication and Access Control in Mobile Applications

*GeneXus*<sup>™</sup>

En este video hablaremos la autenticación y el control de acceso en las aplicaciones móviles de GeneXus, usando GAM.

## Native Mobile Authentication



Las aplicaciones móviles nativas instaladas en los dispositivos consumen servicios REST del servidor Web. Estas aplicaciones crean una interfaz de usuario que también se basa en recursos, además de metadatos.

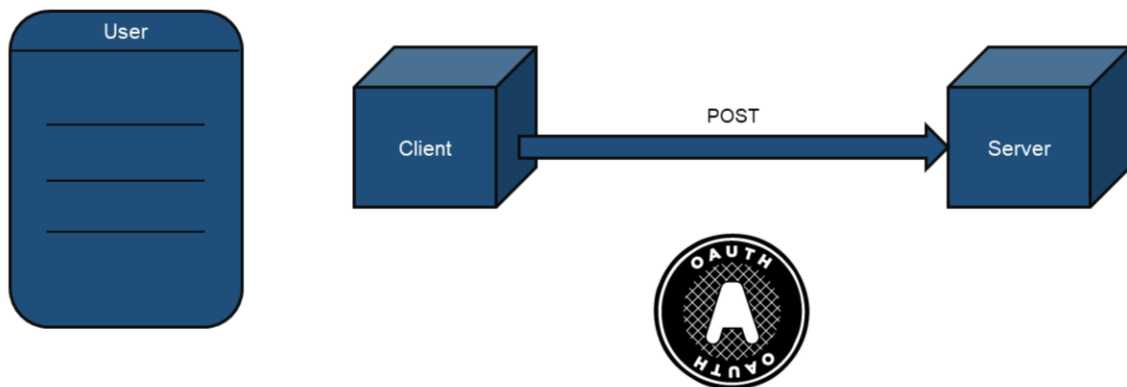
Dada esta arquitectura, es importante contar con un mecanismo de seguridad para evitar que este tipo de acciones sean realizadas por usuarios no autorizados. La forma de hacerlo, por supuesto, es usando GAM.

Al igual que en las aplicaciones web, a través de la configuración de Habilitar la propiedad de seguridad integrada, estamos agregando seguridad automáticamente a la aplicación móvil. Esta seguridad puede ser tanto por Autenticación como Autorización.

Veamos esto con más profundidad.

## Authentication

### Authentication Types - Local



Al igual que en las aplicaciones web, en las aplicaciones móviles nativas un escenario posible es que todos los objetos móviles nativos sean privados.

Configurando nuestra aplicación con el nivel de seguridad integrado en Autenticación, lograremos que todos los objetos requieran de que el usuario se encuentre con una sesión activa.

Veamos ahora cómo se comportan y funcionan los tipos de autenticación en aplicaciones móviles.

Comencemos con el tipo Local.

Este tipo de autenticación funciona al igual que las aplicaciones web, donde las credenciales de los usuarios se almacenan en la tabla "User" del GAM.

Como ya hemos mencionado, la contraseña no es almacenada. En su lugar se almacena un hash de esta utilizando una clave única para cada usuario y el algoritmo SHA-512.

Luego, simplemente se hace un pedido POST desde el cliente al servidor, donde en este caso, el cliente corresponde al dispositivo móvil.

Por detrás en lo que es bajo nivel, éste y todos los métodos de autenticación con GAM utilizan el protocolo OAuth 2.0.

## Authentication

### Authentication Types - Local

The screenshot shows a REST client interface with a POST request to `/oauth/access_token`. The request body is form-data with the following parameters:

KEY	VALUE
<input checked="" type="checkbox"/> client_id	d3b208f72cb84607ab88c4647b898728
<input checked="" type="checkbox"/> client_secret	fe8e07052da449db94af48f394043407
<input checked="" type="checkbox"/> grant_type	GAMLocal
<input checked="" type="checkbox"/> scope	FullControl
<input checked="" type="checkbox"/> username	GAM
<input checked="" type="checkbox"/> password	123

The response body is JSON:

```

1 {
2   "access_token": "85a3006c-0606-4102-980e-223f88463ec2111c4cc059529f93f1c06286d108ae044a14b5e0775d4688ad5059bad375d888602c77fe62f1f78",
3   "token_type": "Bearer",
4   "expires_in": 3600,
5   "refresh_token": "8011040yzmct3v1a59b3U3wGqJjmuSan1wGp3",
6   "scope": "FullControl",
7   "user_guid": "c3910a62-5960-4495-a29e-67867784fc3"
8 }

```

Aquí podemos ver un ejemplo de lo que sería el pedido POST que mencionamos.

El pedido se realiza a la URL de nuestra aplicación, seguido de `/oauth/access_token`. Como parámetros, se le incluyen los típicos `client_id` y `client_secret`, el `grant_type` y `scope`, y finalmente el nombre de usuario y contraseña.

En caso de que el pedido sea correcto, deberíamos recibir un JSON como el siguiente, con todos los datos del acceso e identificador del usuario.

Esto les puede ser de utilidad para probar sus propios tipos de autenticación, donde la prueba la pueden realizar con cualquier herramienta de pruebas de APIs.

## Login

How to

```
Event 'BtnLogin'  
  Composite  
    GeneXus.Common.UI.Progress.ShowWithTitle("Connecting...")  
    GeneXus.SD.Actions.Login(&UserName, &Password)  
    GeneXus.Common.UI.Progress.Hide()  
    Return  
  EndComposite  
EndEvent
```

&amp;LoginExternalAdditionalParameters

En temas anteriores habíamos revisado el mecanismo de Login para las aplicaciones web. Veamos ahora cómo funciona esto en las aplicaciones móviles.

Para ver esto, nos centraremos en el objeto GAMSDLogin, el cual está incluido en las implementaciones de referencia que trae GAM.

En este caso, el inicio de sesión local se realiza mediante el External Object GeneXus.SD.Actions y su método de inicio de sesión llamado Login.

Por atrás, una vez iniciada la sesión se almacena la información en el dispositivo, pero de forma cifrada, guardando un token de seguridad utilizando el KeyStore (en el caso de Android) o Keychain (en el caso de iOS).

Este método de Login perteneciente a Actions está sobrecargado, por lo que permite incluir el parámetro adicional &LoginExternalAdditionalParameters como veíamos en los casos de Login externos hace un momento.

Veamos esto con otro ejemplo y detalle.

## Login

How to

Name	Type	Description	Is Collection
LoginExternalAdditionalParameters		Login External Additional Parameters	<input checked="" type="checkbox"/>
Repository	Character(40)	Repository	<input type="checkbox"/>
AuthenticationTypeName	Character(60)	Authentication Type Name	<input type="checkbox"/>
Properties		Properties	<input checked="" type="checkbox"/>
Property			
Id	Character(40)	Id	<input type="checkbox"/>
Value	Character(40)	Value	<input type="checkbox"/>

```

Event 'BtnLogin'
  Composite
    GeneXus.Common.UI.Progress.ShowWithTitle("Connecting...")
    &LoginExternalAdditionalParameters.Repository = !"1f41a6bb-bc52-451b-b521-c4bcd4a9ac8b"
    GeneXus.SD.Actions.Login(&UserName, &Password, &LoginExternalAdditionalParameters)
    GeneXus.Common.UI.Progress.Hide()
  Return
EndComposite
EndEvent

```

Ese parámetro que mencionamos, debe corresponder al objeto llamado de la misma manera, el cual luce así.

Un ejemplo de utilización de esta sobrecarga en el método de Login SD puede ser el siguiente.

El objeto LoginExternalAdditionalParameters permite establecer el GUID del Repositorio al que queremos conectarnos. Esto es útil cuando es una aplicación MultiTenant.

Cuando le damos un valor para la propiedad Repository del parámetro &LoginExternalAdditionalParameters, se puede establecer la conexión a ese Tenant (en particular a ese cliente), representada por un Repositorio y su GUID.

Además de esta opción que utilizamos, tenemos otras que pueden ser el Nombre del tipo de autenticación, y una lista de propiedades dinámicas Id y Value que podría necesitar la aplicación.



## Login

How to

The screenshot shows a REST client interface for a POST request to `/oauth/access_token`. The request body is set to `x-www-form-urlencoded`. The parameters are as follows:

Parameter	Value
<input checked="" type="checkbox"/> client_secret	fe8e07052da449db94af48f394043407
<input checked="" type="checkbox"/> grant_type	GAMlocal
<input checked="" type="checkbox"/> scope	FullControl
<input checked="" type="checkbox"/> username	GAM
<input checked="" type="checkbox"/> password	123
<input checked="" type="checkbox"/> additional_parameters	<pre>{   "AuthenticationTypeName": "local",   "Repository": "",   "Properties": [     { "Id": "Company", "Value": "GeneXus" },     { "Id": "Branch", "Value": "12" }   ] }</pre>
Key	Value

Volviendo a las pruebas manuales, podemos replicar el pedido de login que veíamos antes, pero esta vez agregando estos parámetros adicionales que comentábamos. Para hacer esto, agregamos al pedido el parámetro `additional_parameters`, y en él, incluimos los datos que queremos adicionar al pedido.

En este caso vemos que se incluye el nombre del tipo de autenticación que queremos utilizar, y dos propiedades de las que mencionábamos anteriormente como (Id, Valor) donde en este caso es el nombre de una compañía y una sucursal.



## Authentication

Authentication Types - Google

### Google authentication type

```

Event 'Google'
  Composite
    &LoginExternalAdditionalParameters = new()
    &LoginExternalAdditionalParameters.AuthenticationTypeName = !"googleb"
    GeneXus.SD.Actions.LoginExternal(GAMAuthenticationTypes.Google, &UserName, &Password, &LoginExternalAdditionalParameters)
  Return
EndComposite
EndEvent

```

Enabled?	<input type="checkbox"/>	Local site URL	<input type="text" value="https://trialapps3.genexus.com"/>
Description	<input type="text"/>	Additional Scope	<input type="text"/>
Small image name	<input type="text" value="GAMButtonGoogleSmall"/>		
Big image name	<input type="text"/>		
Impersonate	<input type="text" value="(none)"/>		

Como ya hemos visto en el tema de Autenticación, podemos utilizar a Google como proveedor de Identidades para el inicio de sesión.

Veamos una configuración de esto.

En el sitio de Google debemos crear una aplicación de cliente y obtener la información y secreto del cliente. La URL de este sitio es la siguiente.

Primero vamos a la sección API y Servicios y hacemos clic en la sección Credenciales, seleccionando "ID de cliente OAuth".

Luego seleccionamos "Aplicación Web" en el Tipo de aplicación.

Una vez hecho esto, debemos cambiar las URI de redirección. Ahí podemos especificar la URI completa de nuestra aplicación, incluido /oauth/gam/signin, como se ve en la imagen. Esto último es importante y aplica sin importar si nuestra aplicación es Java o .NET.

Siempre se debe especificar la URL completa de la aplicación, incluyendo el directorio virtual, seguido de /oauth/gam/signin.

Luego de confirmar esto, obtendremos nuestra información del cliente y terminamos desde este lado.

Ahora configuremos nuestra aplicación desde el backend de GAM.

Para eso creamos un nuevo tipo de autenticación de Google, y completamos la información con la proporcionada por Google.

Algo a destacar en este paso, es que en el campo Local site URL solo se necesita ingresar el dominio del servidor que ejecuta la aplicación. No es necesario ingresar la URL completa del sitio, pero en caso de ingresar, no se debe incluir el "/servlet" en Java.

En este momento ya tenemos configurado el lado de Google y del Backend. Pero como estamos en una aplicación móvil, necesitamos una lógica distinta en el Login de nuestra aplicación por lo que debemos agregar un evento en él que nos autentique con Google.

Vemos que esta lógica ahora utiliza el método LoginExternal del External Object Actions, y como primer parámetro se le indica que es un tipo de autenticación de Google a través del dominio GAMAuthenticationTypes.

Luego el nombre de usuario y contraseña serán ignorados, y por último tenemos a LoginExternalAdditionalParameters, que con el podemos indicar información adicional al login que nos sea de utilidad. En este caso le definimos el nombre del tipo de autenticación.

Esto es de utilidad en el caso de que tengamos más de un tipo de autenticación de Google en el repositorio y con este nombre vamos a poder seleccionar el que deseamos. Si no es así, podemos hacer el llamado de login sin utilizar este parámetro.

Más adelante veremos con más detalle este atributo.

Y con esto terminamos el proceso de autenticación con Google.

## Authentication



### Authentication Types - Facebook

```
Event 'Facebook'  
  Composite  
    &LoginExternalAdditionalParameters = new()  
    &LoginExternalAdditionalParameters.AuthenticationTypeName = !"facebook"  
    GeneXus.SD.Actions.LoginExternal(GAMAuthenticationTypes.Facebook, &UserName, &Password, &LoginExternalAdditionalParameters)  
    Return  
  EndComposite  
EndEvent
```

Ahora veamos el caso de Facebook.

En el curso introductorio mostramos cómo hacer un tipo de autenticación con Facebook, por lo cual no repetiremos esto. Puede consultar como hacer esto en dicho curso o en la Wiki de GeneXus.

Al igual que en el caso de Google, una vez que tenemos nuestro cliente y el tipo de autenticación en el backend de GAM creados y configurados, necesitamos una lógica personalizada en la aplicación.

En este caso, nuevamente agregamos un evento nuevo con lo siguiente.

La explicación es la misma que la de Google, con la diferencia que ahora en el primer parámetro de LoginExternal le indicamos que el tipo es Facebook.

## Authorization



<prefix>\_Execute



<prefix>\_Services\_Execute

<prefix>\_Services\_Insert  
<prefix>\_Services\_Update  
<prefix>\_Services\_Delete

<prefix>\_Services.FullControl

### Autorización.

En el caso de los paneles SD, GAM comprueba si el usuario tiene permiso para ejecutarlos. Para eso verifica que el usuario tenga el permiso <prefix>\_Execute, donde prefix es el Prefijo de Permiso definido para el objeto.

En caso de que se detecte un error del permiso, se mostrará en pantalla el mismo o se redirigirá al objeto especificado como no autorizado para la propiedad SD.

En los paneles y WorkWith SD, la lógica de negocio se resuelve utilizando Web Services REST.

Cuando se generan estos objetos, también se generan proveedores de datos expuestos como servicios web REST.

Estos servicios web se vuelven privados cuando declara la seguridad sobre los objetos mencionados.

Un detalle a tener en cuenta es que las acciones en los paneles de los WorkWithSD están relacionadas directamente con el Business Component asociado con el WorkWith. Eso significa que si se aplica el patrón Trabajar con a una Transacción, esta se guarda automáticamente como Business Component expuesto como servicio web REST.

Para controlar los permisos sobre las acciones de insertar, actualizar y eliminar, se debe declarar los permisos sobre el propio Business Component.

Dado que la idea es validar si el usuario tiene derechos para ejecutar un objeto dependiendo del método con el que se emite la llamada, el método GET requiere el permiso <prefix>\_Services\_Execute (donde en este caso prefix es el prefijo de permiso definido para el Business Component), y este es el permiso mínimo requerido.

El resto de los permisos son:

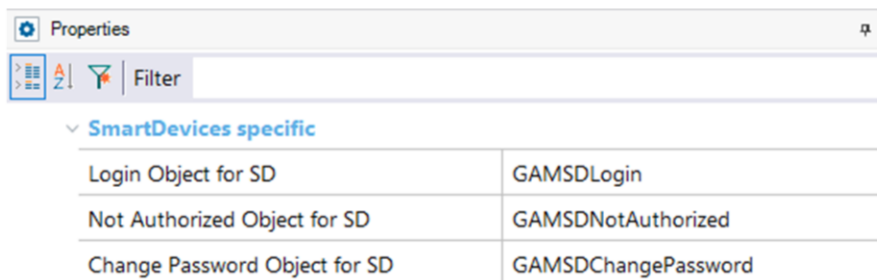
<prefix>\_Services\_Insert

<prefix>\_Services\_Update

<prefix>\_Services\_Delete

Acá como siempre, también tenemos el FullControl que abarca a todos los permisos.

## Object configuration



The screenshot shows the 'Properties' window in GeneXus. It features a search bar with 'Z' and a 'Filter' button. Below the search bar, there is a section titled 'SmartDevices specific' which contains a table with three rows of configuration data.

SmartDevices specific	
Login Object for SD	GAMSDLLogin
Not Authorized Object for SD	GAMSDNotAuthorized
Change Password Object for SD	GAMSDChangePassword

Anteriormente nombramos que se podía redirigir al usuario hacia el objeto de no autorizado en caso de que no tuviera permisos para acceder a un recurso. Este objeto lo podemos configurar en el nivel de versión de la KB, bajo la sección SmartDevices specific.

Y no solo podemos configurar ese, si no que también desde allí es donde debemos definir qué objetos serán los asociados al login y al cambio de contraseña.

Veamos una particularidad de este último.

## Change Password

GAMSDChangePassword



← Change Password BACK

Email or Name  
admin

Current password  
Current password

New password  
New password

Confirm password  
Confirm new password

CHANGE PASSWORD

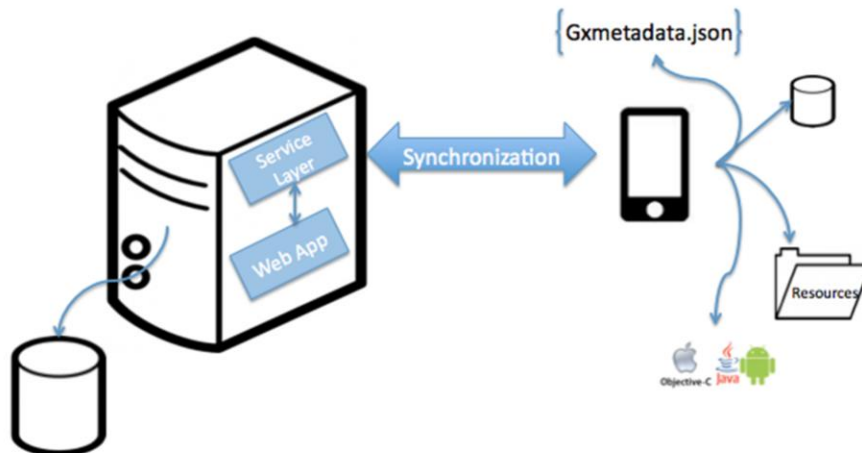
Al igual que en las aplicaciones web, cuando a un usuario se le expira o debe cambiar su contraseña, se lo redirige automáticamente al objeto de cambio de contraseña. En particular para Mobile, tenemos el objeto GAMSDChangePassword, dentro de los objetos de ejemplo ya incorporados por GAM, y se puede utilizar para este cometido.

Como dijimos antes, si se dan las condiciones en las cuales el usuario debe cambiar su contraseña, no podrá hacer nada más hasta que no produzca el cambio, ya que las aplicaciones siempre lo van a redirigir al objeto que se configuró para cambiar la contraseña.



## Online and Offline Apps

### Offline App



Con GeneXus tenemos la posibilidad de tener aplicaciones tanto Online como Offline. Veamos las diferencias que estas tienen, junto a sus objetos y métodos.

Comencemos con las Offline.

La arquitectura de estas aplicaciones móviles tiene que considerar dos situaciones en las que estas aplicaciones deberían funcionar: cuando la aplicación está desconectada y cuando tiene una conexión.

En el primer caso, la aplicación debe poder procesar datos e interactuar con una base de datos sin conexión a la red.

En el segundo caso, la aplicación puede sincronizar datos con el servidor invocando servicios web.

Sin importar si se está conectado o no, el procesamiento de la aplicación se realiza en el dispositivo actualizando su base de datos local. Una vez que se establece una conexión, la aplicación ejecutará la sincronización de sus cambios que se hicieron mientras estaba desconectado.

Una aplicación móvil de estas la podemos dividir en dos componentes: un componente local y un componente de servidor.

El servidor puede tener un backend para la aplicación, la base de datos y una capa de servicios para la sincronización de datos con los dispositivos.

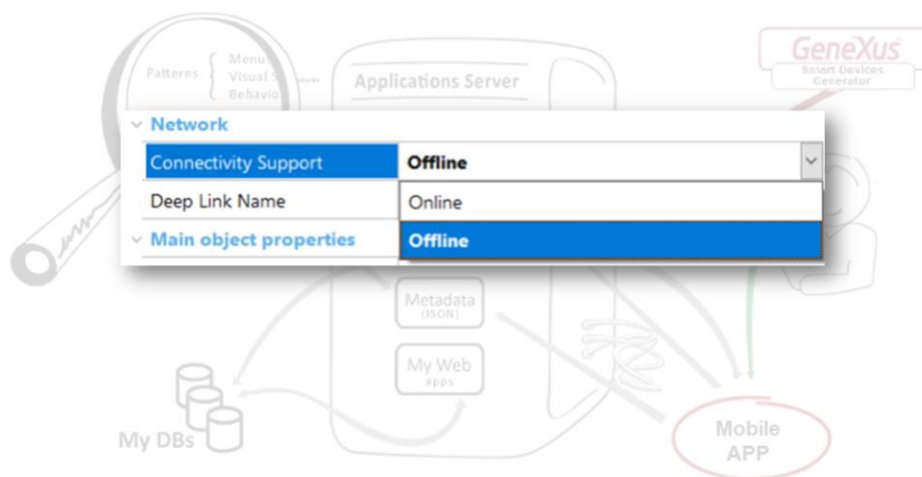
En los dispositivos, las aplicaciones también tienen una base de datos con un

subconjunto de los datos del servidor y toda la lógica que debe ejecutarse localmente. A su vez, la aplicación también tiene todos los metadatos necesarios para el diseño de la interfaz de usuario y los eventos del usuario.

Ambos componentes, se comunican a través de servicios REST para realizar la sincronización necesaria para completar la base de datos local y enviar las modificaciones realizadas en el dispositivo al servidor.

## Online and Offline Apps

### Online App



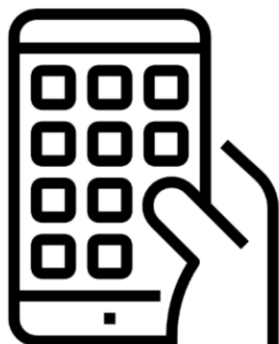
Veamos ahora las aplicaciones con conexión, y para eso remontamos a la ilustración que veíamos al principio.

Los datos requeridos por la aplicación que residen en el dispositivo, se obtendrán consumiendo servicios REST, que accederán a la base de datos del usuario y devolverán la información solicitada a la aplicación.

Para determinar si una aplicación será Online u Offline, se debe configurar la propiedad Connectivity Support en el objeto Main, con el valor que queramos. Como vemos, esta propiedad se encuentra debajo del ítem Network.

Si la aplicación es Offline no se realiza un chequeo de permisos automáticos en el dispositivo, esto solo ocurre en las aplicaciones Online ya que se accede a un servicio Rest en el Servidor.

App with anonymous user



Custom Authentication

External Web Services Authentication

Local Authentication

Aplicación con usuario anónimo, ¿que significa esto?

Este es un concepto que tenemos en el desarrollo de aplicaciones móviles con seguridad de GAM, que permite brindarle a los usuarios la posibilidad de utilizar la aplicación con las mismas o similares funcionalidades que tienen los usuarios registrados.

Por detrás, lo que está haciendo GAM es registrando automáticamente al usuario anónimo por cada dispositivo.

Para GeneXus 15 U10 o versiones anteriores, esto ocurre solo con los tipos de autenticación Custom, External Web Services y Local. Desde GeneXus 15 U11 en adelante, es válido para cualquier tipo de autenticación.

Veamos

cómo

funciona.

## App with anonymous user

How it works



Cada dispositivo cuenta con un identificador para el mismo, entonces cuando un usuario empieza a utilizar la aplicación, automáticamente se crea un "usuario" de GAM usando dicho identificador y pide un token.

Este usuario creado tiene las siguientes características:

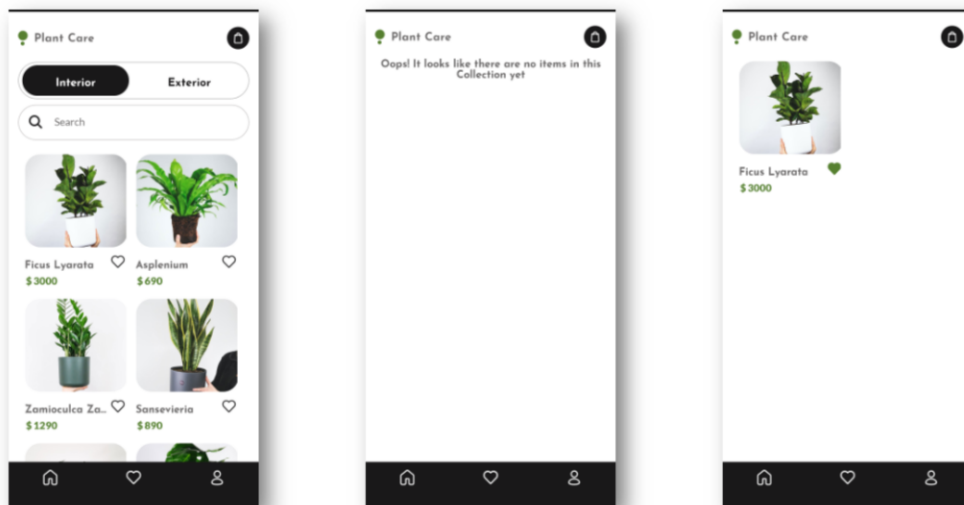
- Es visto por la aplicación como cualquier otro usuario pero tiene la propiedad `GAMUser.isAnonymous = True`
- No se conocen, solicitan ni guardan datos personales de la persona que opera el dispositivo, por lo que queda completamente anónimo
- Su sesión nunca caduca, sin importar la caducidad del token de OAuth que se haya definido para las sesiones

Si el usuario decide registrarse en la aplicación, al utilizar el método `&GAMUser.Save()` se creará el nuevo usuario con sus datos de registro, y le asignará el mismo identificador que tenía el usuario registrado automáticamente.

Esto permite que cualquier información relacionada con el usuario registrado automáticamente que haya guardado la aplicación, permanezca asociada al nuevo usuario registrado, sin tener que realizar cambios en las relaciones usuario-datos que se hayan grabado previamente.

## App with anonymous user

How it works: Example



Veamos un ejemplo para entender en profundidad esto.

Supongamos que tenemos una aplicación de venta de plantas. Dicha aplicación tiene la posibilidad de registrarse e iniciar sesión con una cuenta. Además, podemos marcar como favorito las plantas que deseemos para luego por ejemplo, poder comprarlas.

Ahora, lo ideal sería que si aún no he iniciado sesión en la aplicación y selecciono alguna planta como favorito, al momento de registrarme esos favoritos sigan estando.

Esto es lo que nos permite la funcionalidad de usuario anónimo de GAM.

Si la activamos, la persona podrá marcar todos los favoritos que quiera sin preocuparse si ya inició sesión o no en la aplicación, ya que en caso de no haberla hecho, al momento de registrarse los favoritos se pasarán automáticamente a su cuenta.

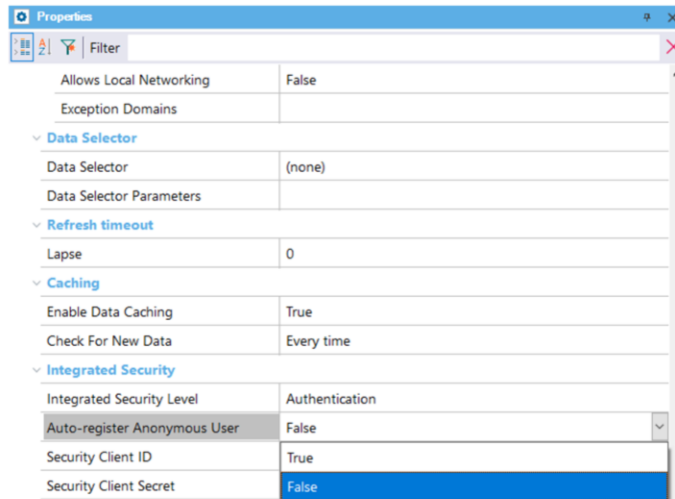
Esto mismo se podría aplicar en el carrito de compras por ejemplo, manteniéndole a la persona los elementos seleccionados en el carrito cuando se registre para hacer el check-out.

Un detalle a destacar, es que si el usuario ya estaba registrado los favoritos no se pasan, porque lo que hace es un Login y no un Usuario nuevo. Esto solo funciona con

el registro.

## App with anonymous user

How to use it



Esta propiedad está disponible para los objetos Mobile que son Main (por ejemplo el Panel), cuando la propiedad de habilitar la seguridad integrada se establece en verdadero.

El valor de esta propiedad determina el comportamiento cuando un usuario anónimo intenta ejecutar el primer objeto de la aplicación que tiene la propiedad Nivel de seguridad integrada en Autenticación o Autorización.

Como vemos en la imagen, los posibles valores para la propiedad de auto registro son Verdadero o Falso.

Si seleccionamos Falso y tenemos definido el nivel de seguridad integrado en Autenticación o Autorización, al usuario anónimo se le desplegará un diálogo indicándole que necesita iniciar sesión o registrarse.

Si seleccionamos Verdadero, en lugar de solicitar iniciar sesión o registrarse, la aplicación va a crear automáticamente un usuario en función de la información del dispositivo móvil del usuario.



## App with anonymous user

How to identify auto-registered users

```
If &GAMSession.User.IsAutoRegisteredUser
    //Anonymous User
else
    //Registered User
EndIf
```

```
GAMUser.isAnonymous()
```

```
&GAMUser = GAMUser.Get()
&GAMUser.IsAutoRegisteredUser
```

En caso de que queramos identificar a estos usuarios registrados automáticamente, tenemos las siguientes formas.

Una opción es a través del usuario de la sesión GAM, preguntando directamente si es un usuario auto registrado.

Otra forma parecido en cuanto a usuario, es la siguiente, donde nos devolverá un booleano indicando si el usuario actual es anónimo o no.

Y la ultima opción es la siguiente, donde primero se obtiene el usuario de GAM y luego se consulta si es auto registrado.

**GeneXus**<sup>™</sup>  
by **Globant**

[training.genexus.com](http://training.genexus.com)  
[wiki.genexus.com](http://wiki.genexus.com)