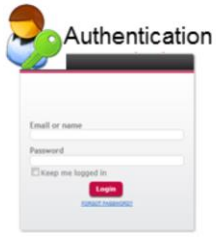
GeneXus X Evolution 3

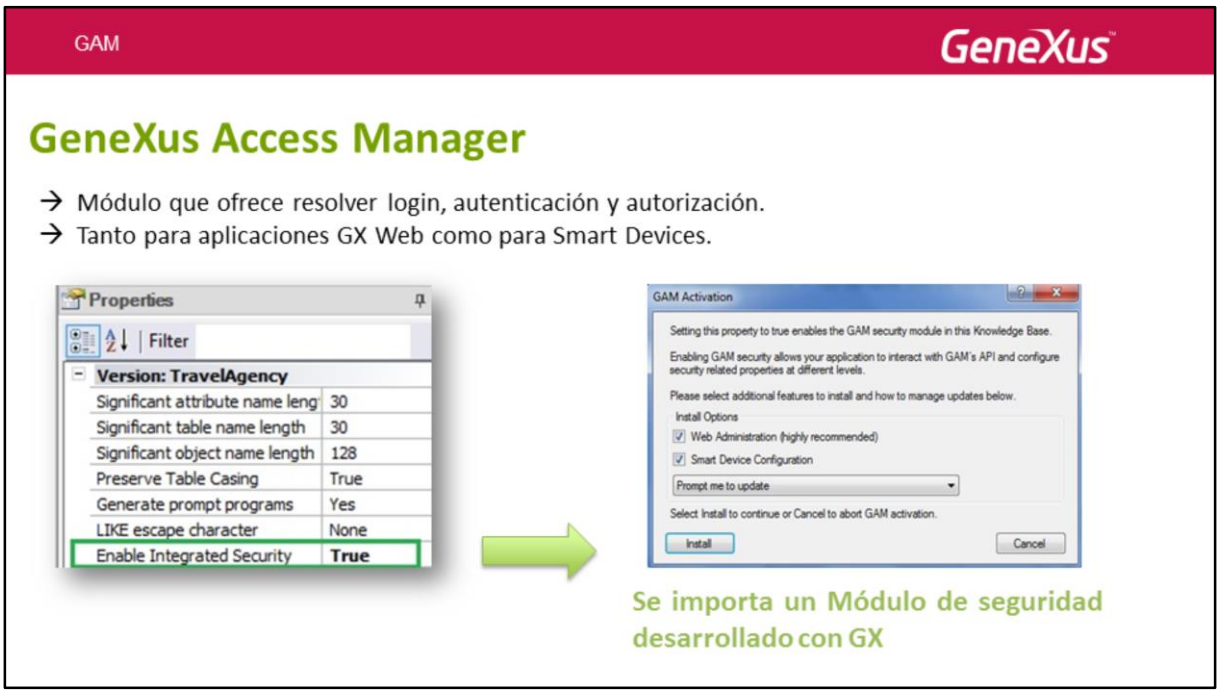
GeneXus

GAM

GAM



GeneXus™
ACCESS MANAGER



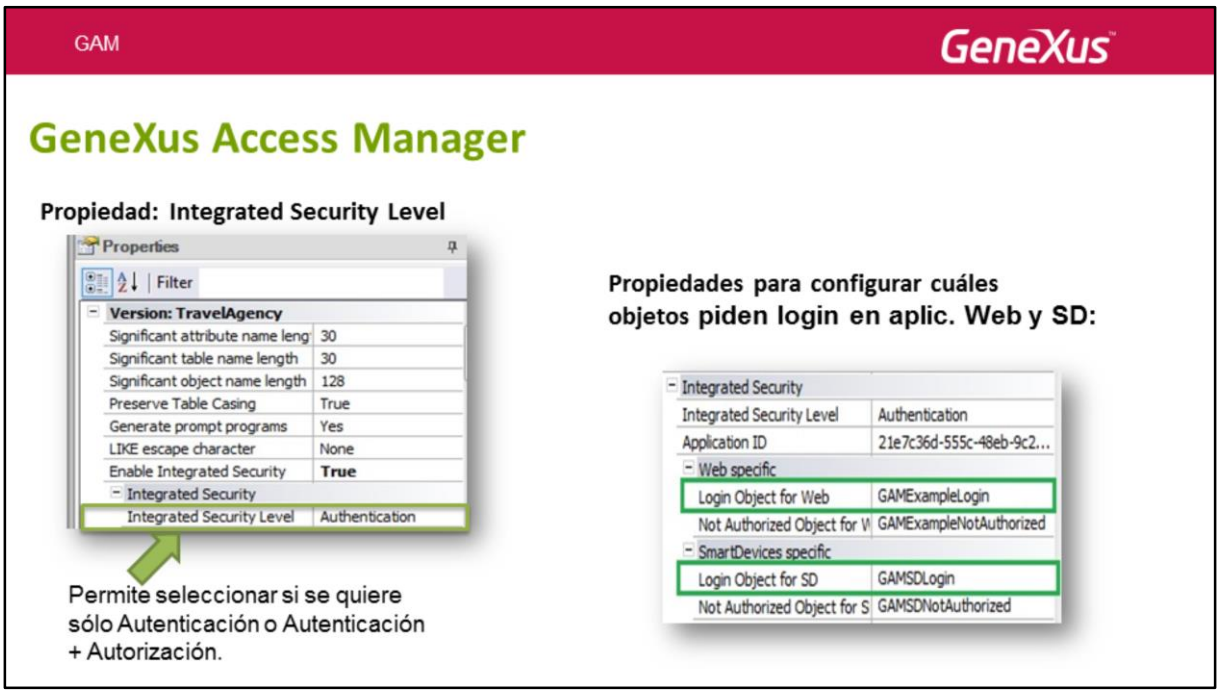
La gran mayoría de las aplicaciones modernas necesitan algún esquema de login, autenticación y autorización.

Para cubrir estas necesidades, GeneXus ofrece un módulo de seguridad, llamado GeneXus Access Manager (GAM) que resuelve las funcionalidades de autenticación y autorización, tanto para aplicaciones Web como para aplicaciones para Smart Devices.

Para disponer de este módulo con todos los controles de seguridad que ofrece, simplemente hay que configurar en nuestra base de conocimiento, a nivel de la versión activa, la propiedad Enable integrated security con el valor True.

Al hacerlo, aparece el diálago para la activación del GAM.

Hay que presionar “Install” y como consecuencia se importará un Módulo de seguridad desarrollado con GeneXus, que se integra a nuestra aplicación, permitiendo así resolver lo referente a la seguridad de la misma.



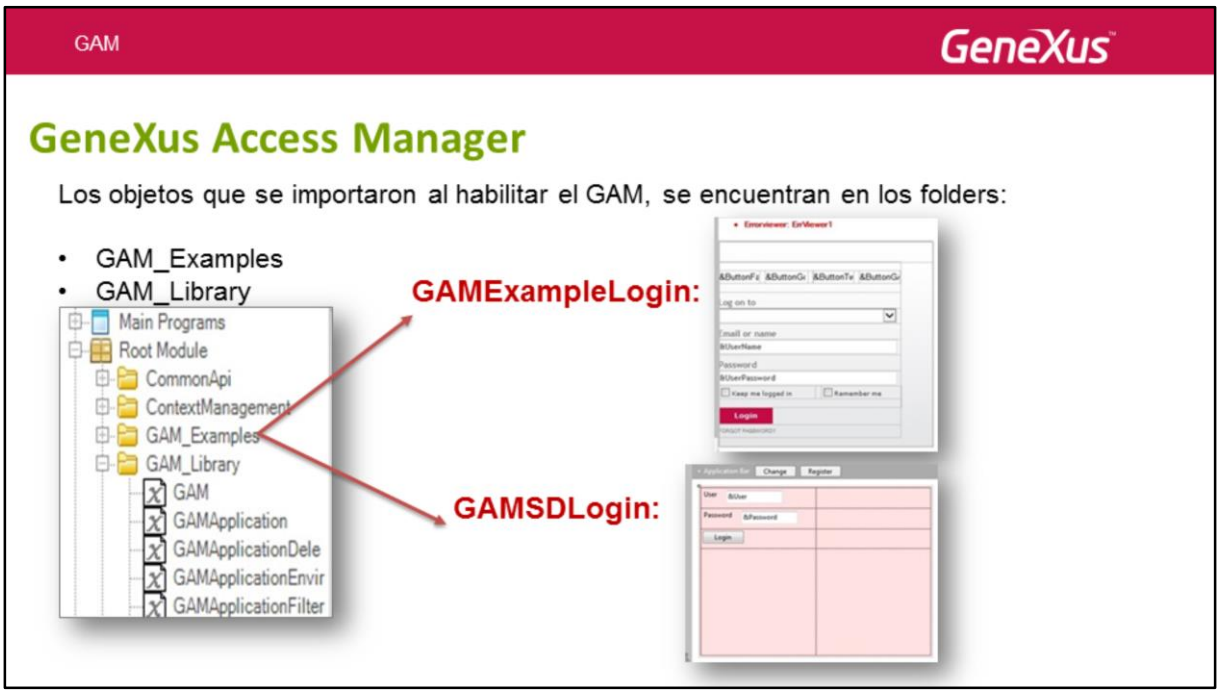
Una vez habilitada la seguridad, se puede seleccionar si se quiere solamente Autenticación o Autenticación+Autorización.

Esto se logra configurando la propiedad I**ntegrated Security Level**.

Por ahora vamos a trabajar solamente con Autenticación.

Cuando se habilita el GAM, además de importarse objetos, se realizan varios cambios. Por ejemplo, se habilitan propiedades para configurar cuál será el objeto para Login tanto para aplicaciones Web como para Smart Devices:

Observemos la propiedad **Login Object for Web**: Tiene el valor GAMExampleLogin para indicar que se utilizará ese objeto para el login de las aplicaciones Web.
Y la propiedad **Login Object for SD**: tiene el valor GAMSDLogin, indicando el nombre del objeto que realizará el login de las aplicaciones para Smart Devices.



Los objetos que se importaron al habilitar el GAM, podemos encontrarlos en los folders GAM_Examples y GAM_Library.

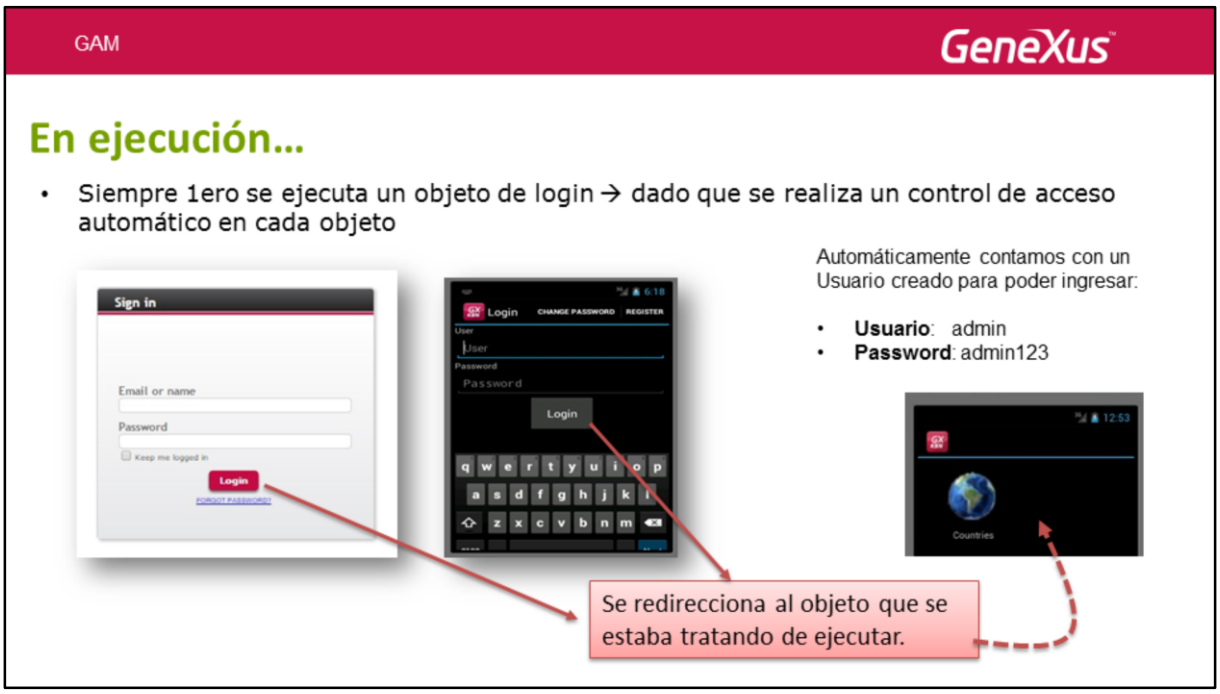
El folder GAM_EXamples contiene todos los objetos de ejemplo que se importan. Observemos que contiene Web Panels y Panels for Smart Devices . Estos objetos van a ser utilizados para la autenticación y autorización de los usuarios.

En particular están los objetos, GAMExampleLogin y GAMSDLogin que son los que quedan configurados, como vimos antes, en las propiedades Login Object for Web y Login Object for Smart Devices.

Pero además hay varios objetos que conforman el Backend del GAM (el Backend es una aplicación Web que se utiliza para administrar y configurar los usuarios, sus roles, permisos, etc.).

En el Folder GAM_Library observemos que hay external objects los cuales tienen las configuraciones necesarias para ejecutar APIs del GAM. Las APIS son funciones que permiten que se comunique nuestra KB con la base de datis del GAM, que es otra base de datis diferente de la asociada a nuestra aplicación. La base de datis del GAM contiene la información de los usuarios, roles, etc.

Algo importante de tener en cuenta, es que cuando habilitamos el GAM, luego debemos ejecutar la acción **Rebuild all**.
Es en este momento que nos solicitan crear la base de datos asociada al GAM.
Ponemos Yes.



Al ejecutar la aplicación (ya sea Web o para Smart Devices) lo primero que se ejecutará es un objeto de login.

La ejecución de este objeto es automática.

En este caso como no hemos definido ningún tipo de autenticación a utilizar como por ejemplo podrían ser: facebook o twitter, por defecto solamente está habilitada la autenticación local y podemos ingresar con el usuario: “admin” y password: “admin123”.

El objeto de login se ejecuta por el simple hecho de haber configurado las propiedades para habilitar el GAM y no hemos tenido que programar nada más. Esto es así porque haciendo uso del GAM, se realiza un **control de acceso automático en cada objeto**.

Una vez que se ingresan los datos de login, se redirecciona al objeto que se estaba tratando de ejecutar, en el caso de la aplicación para Smart Devices al Dashboard.

Este es entonces el comportamiento por defecto tanto para aplicaciones Web como para Smart Devices.

GAM

GeneXus

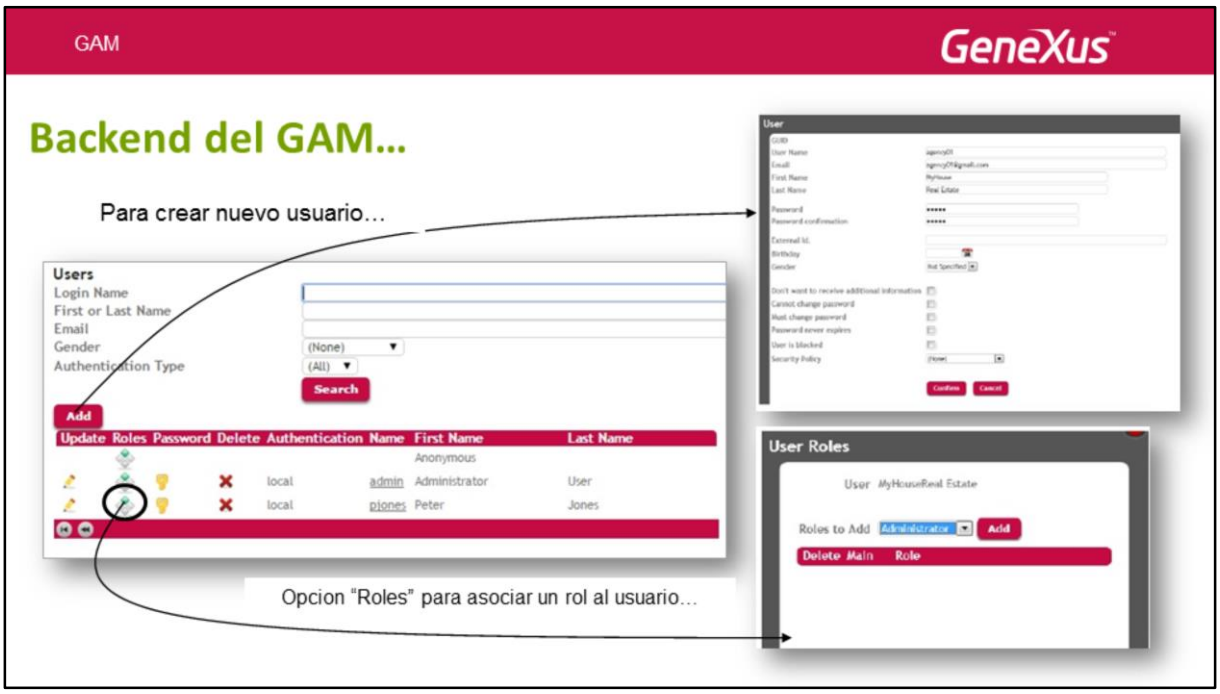
Backend del GAM...

Aplicación Web que permite administrar y configurar:

- Usuarios
- Roles
- Permisos
- Etc.

Como comentábamos antes, entre los objetos que se importan al habilitar el GAM, hay un grupo que implementa el Backend para administrar usuarios, roles, etc.

Para acceder al Backend en tiempo de ejecución, desde el Developer Menu, debemos ejecutar el GAMHome que es el objeto principal del Backend del GAM.



Ingresemos a la opción Users.

Aquí vemos todos los usuarios definidos. Por defecto solo está el usuario admin, que es el que se crea automáticamente con la aolicación del GAM.

Si seleccionamos la opción **Authentication Types**, vemos que por defecto solamente está habilitada la autenticación local.

Aquí es donde podemos definir los diferentes tipos de autenticación que queremos utilizar en nuestra aplicación como por ejemplo, facebook o twitter.

Local: las credenciales del usuario estás almacenadas en el GAM database respository
Facebook: la autenticación es realizada utilizando Facebook como proveedor de identidades
Twitter: la autenticación es realizada utilizando Twitter como proveedor de identidades
Google: la autenticación es realizada utilizando Google como proveedor de identidades
External: hay dos posibilidades:

External Web service: la autenticación es a través de un web service SOAP con ciertos estándares.

Custom: la autenticación es realizada utilizando utilizando algún programa externo que tiene que aplicar ciertos estándares en los parámetros de entrada y de salida.

Remote: la autentiación es realizada usando otra aplicación GAM.

Las APIs son funciones, que en este caso en el que se ha habilitado el GAM en una KB, hacen posible la comunicación con la base de datos del GAM, que es la que contiene la información de los usuarios, roles, etc. (dado que se trata de una base de datos diferente a la asociada a la aplicación descripta en la KB).

Habíamos implementado un menú de opciones haciendo uso del user control: SlideMenu. Este menú se encuentra siempre visible en la aplicación en ejecución, ya que lo hemos incluido en la master page.

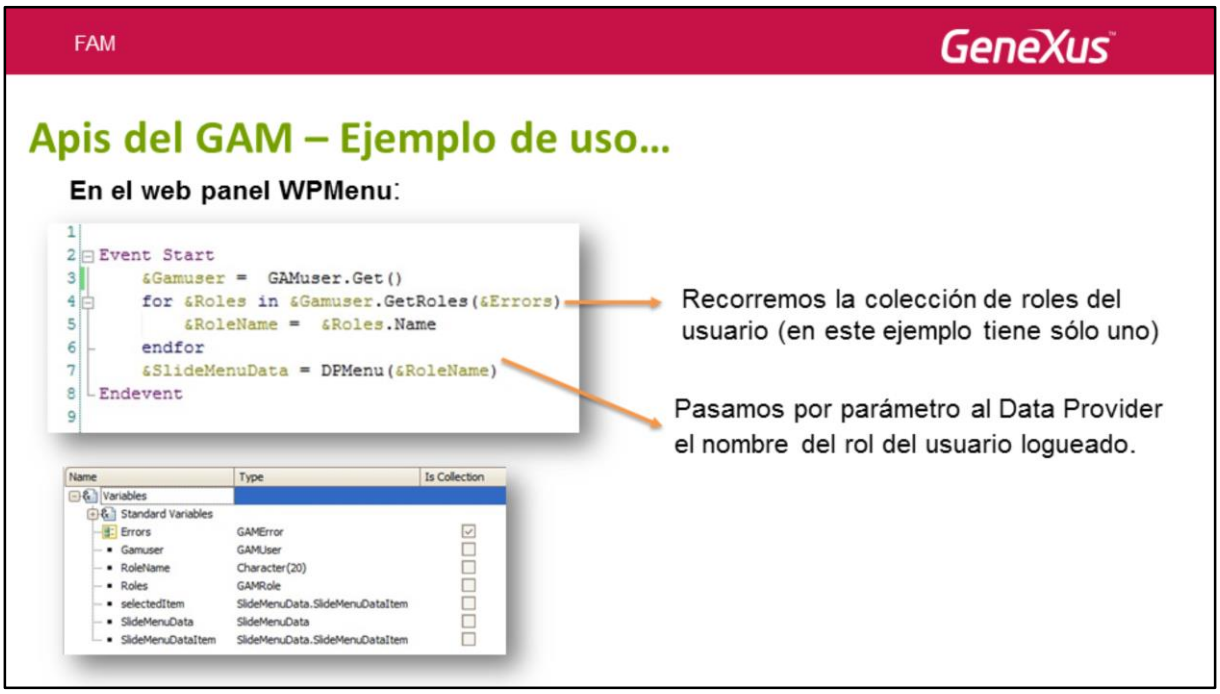
Supongamos que nos solicitan ahora un nuevo requerimiento, **y es que las opciones presentes en el menú, varíen según el rol del usuario logueado**.

Ya hemos aplicado el GAM a nuestra aplicación.

Utilizando el backend del GAM creamos el nuevo rol "SaleAgent", y un nuevo usuario "pjones" con dicho rol asociado.

El objetivo es:

- Si el usuario que se loguea tiene el rol Administrador, entonces se mantendrán todas las mismas opciones que hasta ahora incluíamos en el menú.
- Si en cambio se loguea un usuario con el rol SaleAgent, entonces solamente se ofrecerán determinadas consultas.



En el evento Start del web panel WPMenu estamos invocando al Data Provider que genera la carga con las opciones del menú.

Nuestra propuesta es modificar ese código para que ahora identifiquemos el rol del usuario logueado. Para eso utilizamos la función **GAMuser** provista por las Api del GAM (recordemos que estas Api las encontramos dentro de la carpeta GAMLibrary).

A través del objeto externo GAMUser es que podemos utilizar los métodos disponibles y obtener, entre otras cosas, los roles que tiene el usuario asociado.

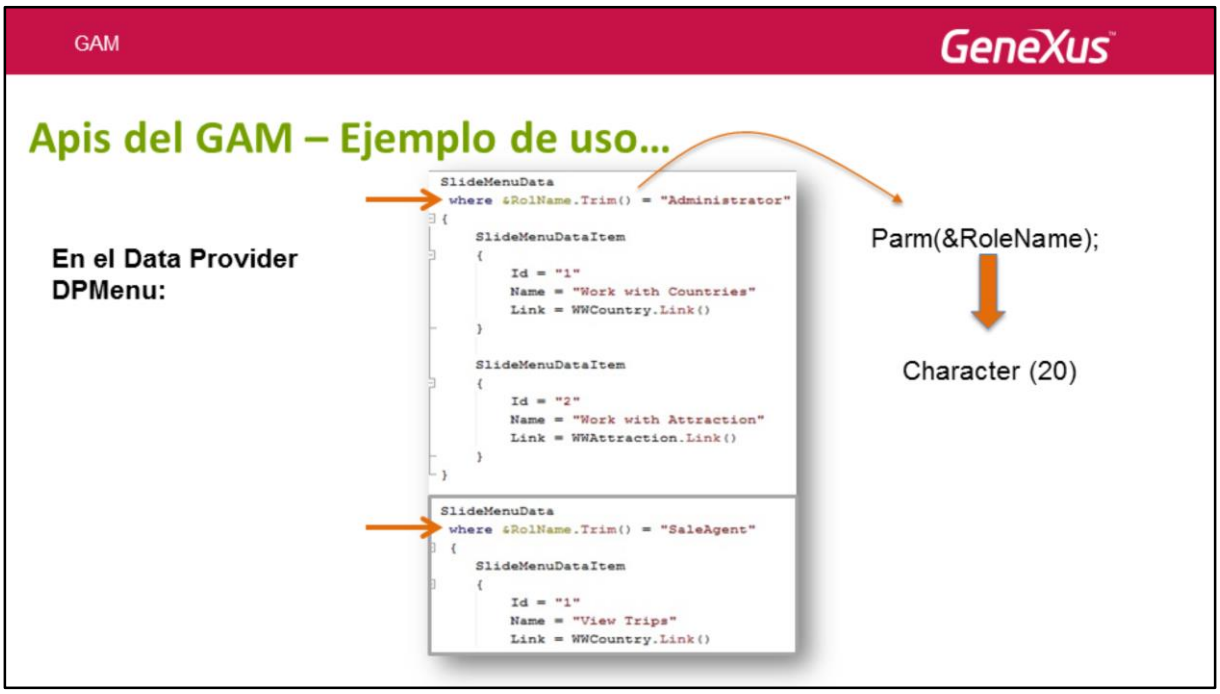
Definimos entonces una variable de nombre &Gamuser en nuestro web panel WPMenu, de tipo de dato GAMuser. Este tipo de dato, como otros, fueron creados automáticamente cuando habilitamos el GAM.

Si bien en nuestro caso cada usuario tiene solamente un rol asociado, podría tener varios, así que tenemos que recorrer una colección de roles y del único rol que tenga el usuario, obtendemos el nombre.

De modo que definimos 2 variables:
La variable &Role: de tipo de dato GAMRole (que es un tipo de dato que representa una instancia de la colección de roles)
Y la variable &RolName como character de 20

Cuando se recorre la colección de roles asociados al usuario es necesario indicar una variable para guardar los posibles errores que se puedan suceder. Así que definimos a la variable &Errors del tipo de dato GAMError, y la marcamos como una colección.

Finalmente llamamos al Data Provider que carga el menú, pasándole por parámetro el rol del usuario logueado. Salvamos.



Ahora en el Data Provider DPMenu, endremos que recibir al rol del usuario logueado.
Así que definimos a la variable RoleName de tipo Character de 20 y declaramos la correspondiente regla Parm.

Hasta el momento tenemos definida la carga de varias opciones genéricas para ser presentadas en el menú.

Ahora vamos a condicionar cuáles opciones cargaremos, dependiendo del rol recibido.

- Si el usuario que se loguea tiene el rol Administrador, entonces se mantendrán todas las mismas opciones que hasta ahora incluíamos en el menú.
- Si en cambio se loguea un usuario con el rol SaleAgent, entonces solamente se ofrecerán determinadas consultas.

Filtramos que si el rol es “Administrator”, se muestren todas estas opciones.
Indicamos entonces Where &RolName.Trim() =“Administrator”

Para definir que si se recibe el rol “SaleAgent”, se carguen las opciones Work With Countries y Work With Attractions, repetimos todo el grupo SlideMenuData con las opciones que corresponden y condicionamos que el RoleName recibido sea “SaleAgent”.

Al presionar F5…

- Si nos logueamos con el usuario: “admin” y password: “admin123” → vemos el menú con las opciones completas.

- Si salimos de la aplicación e ingresamos ahora con el usuario: “pjones” y password: “pjones123” → las opciones del menú han cambiado.


GeneXus