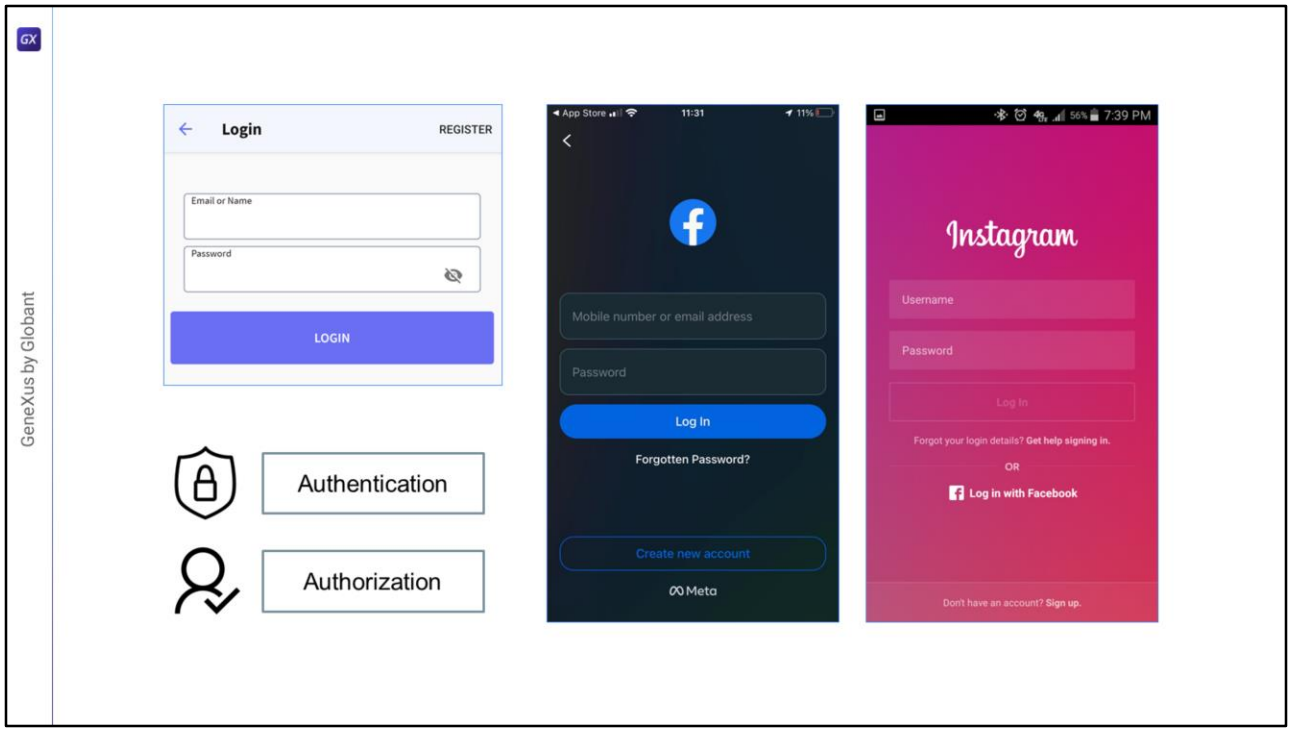
GeneXus by Globant

GeneXus Access Manager

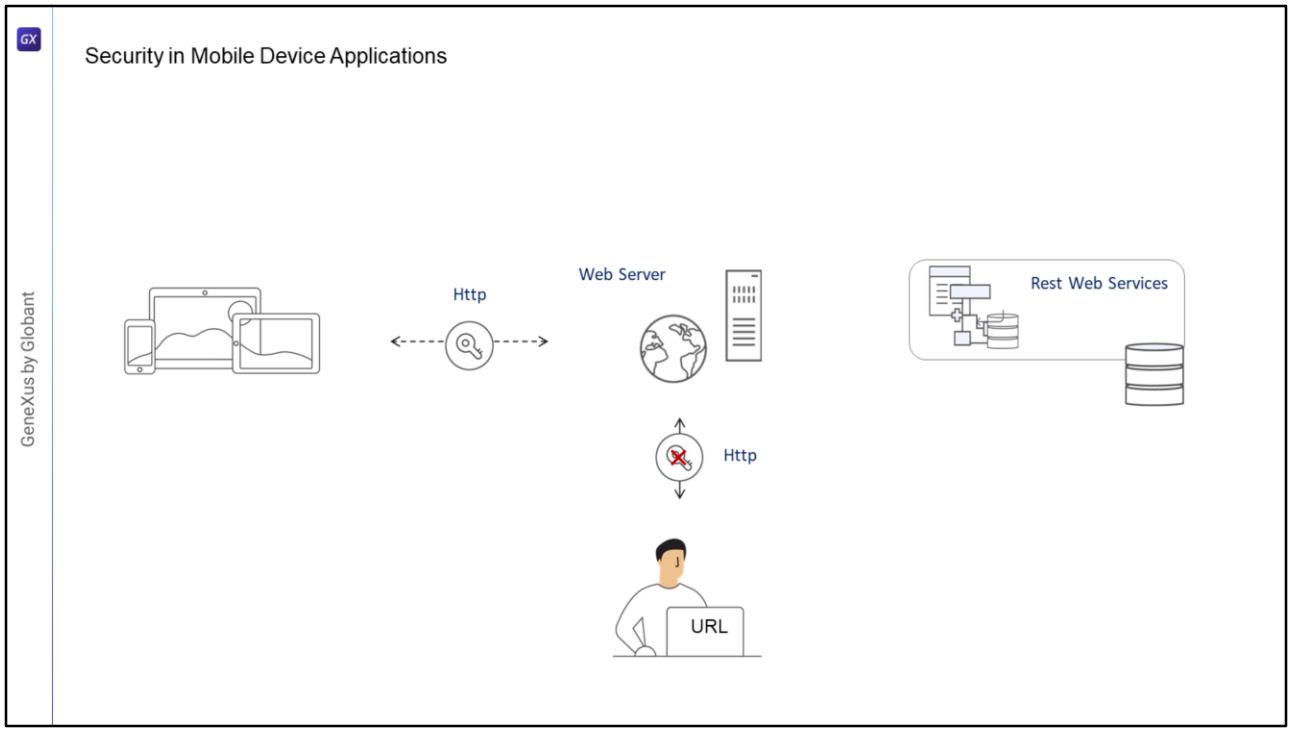
Introduction

Diego Marranghello



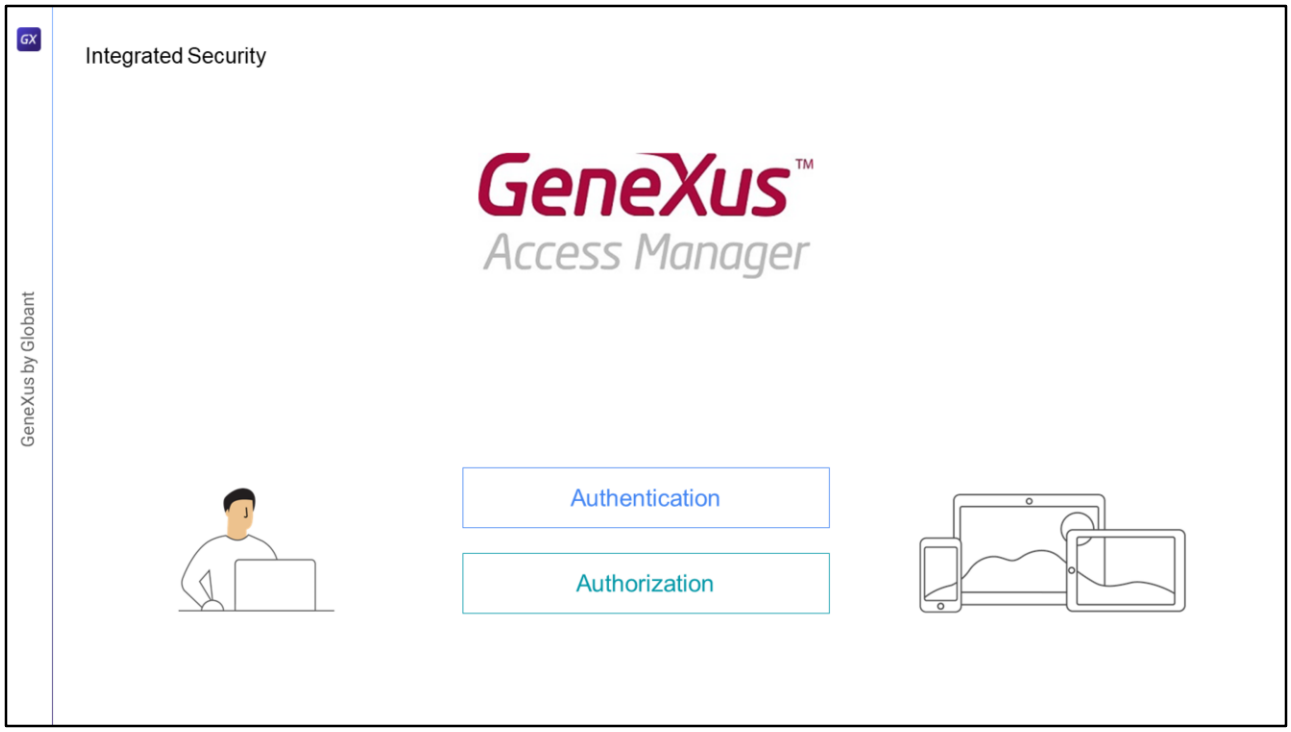
As we know, most current applications require a security scheme so that only allowed users can access them, and also to authorize or restrict access to parts of an application according to the permissions assigned to each user.

This means ensuring that all users who log in are properly authenticated and authorized.



On the other hand, mobile device applications are distributed applications, so they are partially executed on the device itself. The apps' business layer is implemented through Rest services with an access URL, and that's why they are exposed to unwanted access.

Just like for web applications, we verify that only authenticated and authorized users can access the application, so that users without the corresponding permissions are not allowed to run it.

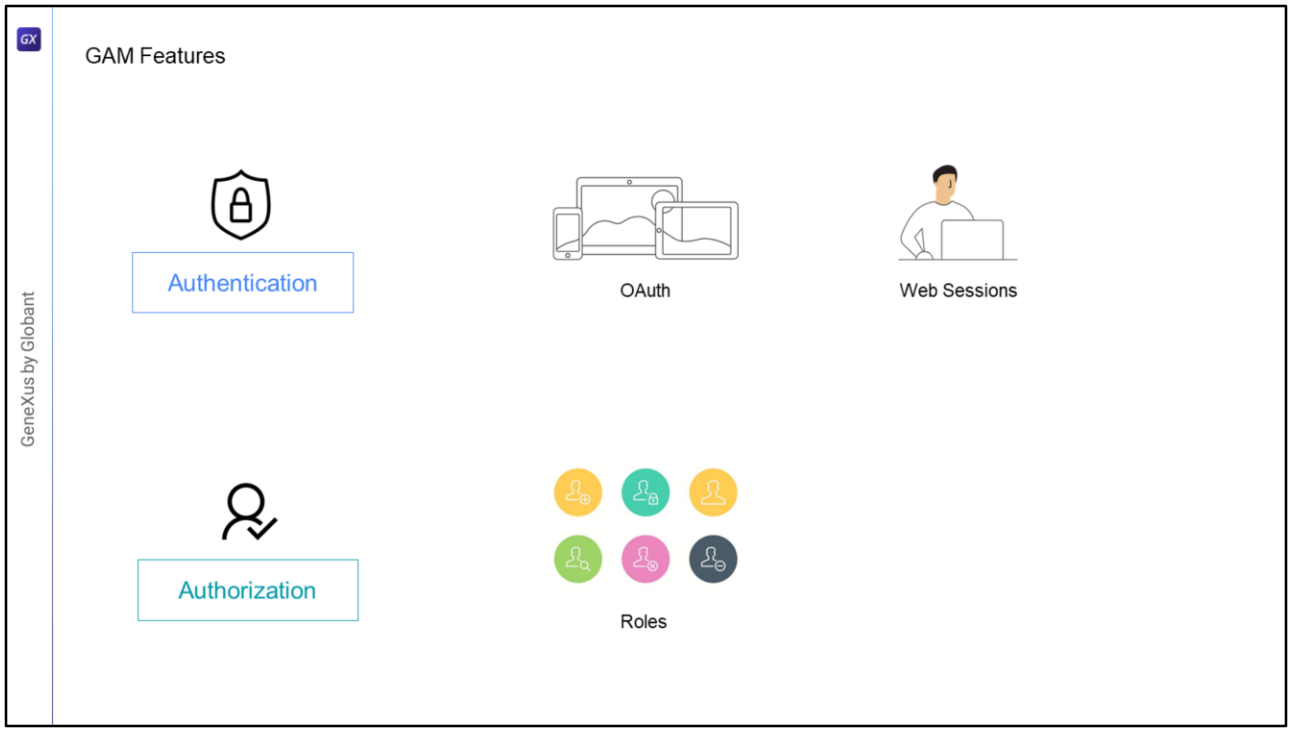


The GeneXus Access Manager (GAM) security module handles authentication and authorization features for both web and mobile device applications.

GAM has been developed with GeneXus, so it can be easily integrated into the application's KB in order to solve everything related to its security in a centralized manner. Its objective is to have the Security solution used in the most declarative way possible within the application, without adding complexity.

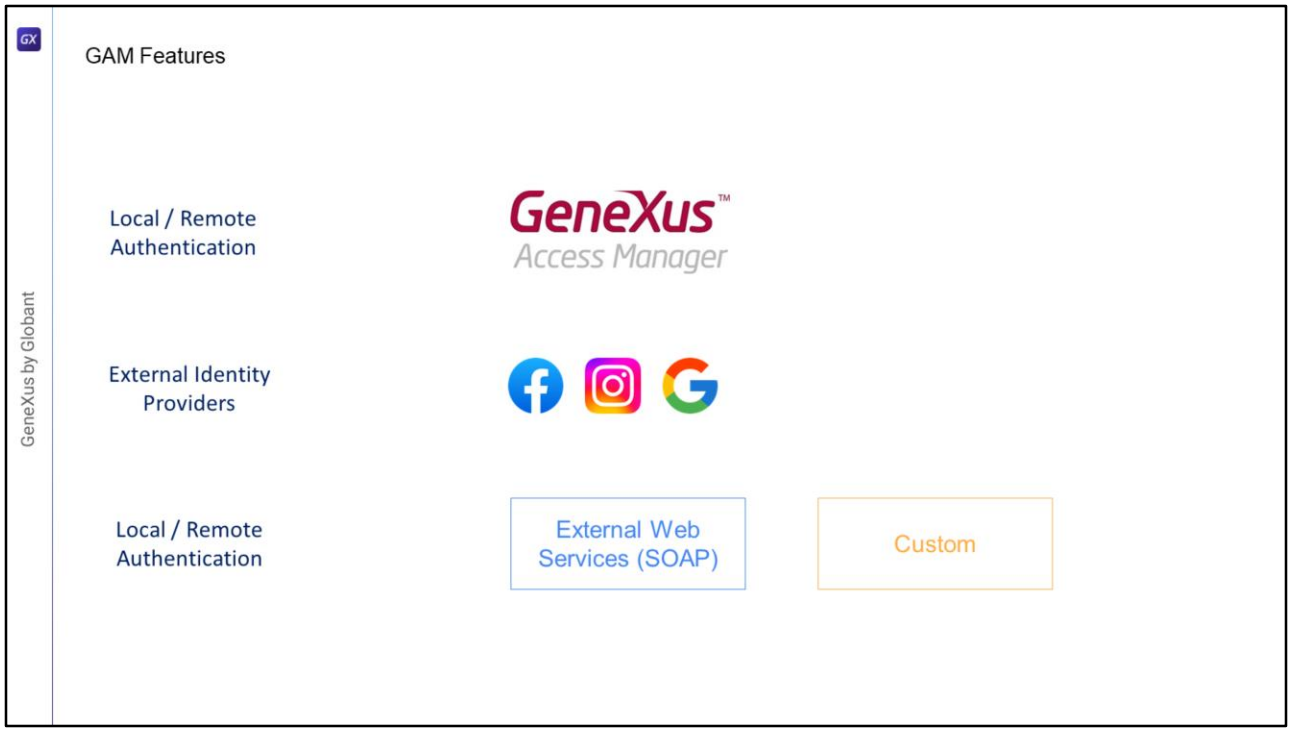
GAM also provides a backoffice that allows creating users, permissions, security policies and access to objects, among other things.

In addition, it provides an API to access many of these features programmatically.



To handle Authentication on mobile devices, Oauth is used internally, unlike Web applications where Web Sessions are used.

In the case of Authorization, its implementation is based on Roles.



GAM provides several Authentication Types, which are as follows:

Local authentication using GAM where users and all their credentials are stored in a database that we own. It can also be done remotely, since an application using GAM can become an identity provider. In this case, other applications with GAM can remotely connect to this server and obtain authentication from there.

We can also use other external identity providers that provide authentication based on the Oauth 2.0 protocol such as Facebook, Instagram, Google, etc. In this case, there is no need to define local users.

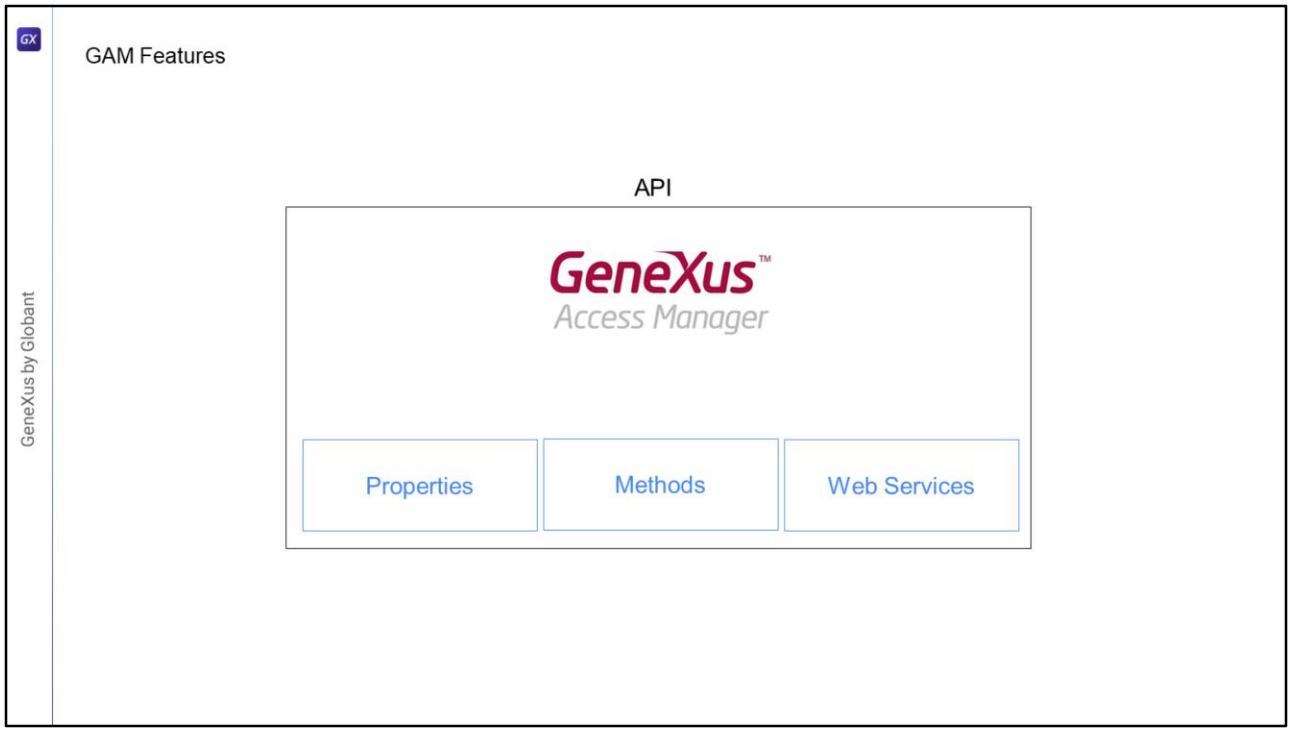
Sometimes it is necessary to integrate our application with others through external authentication.
One example of external authentication is to use a SOAP web service provided by the other application and configure GAM to consume this web service.

The other application may provide an external program for authentication purposes, and it may not be a web service. In this case, it is possible to configure GAM to accept a Custom authentication.

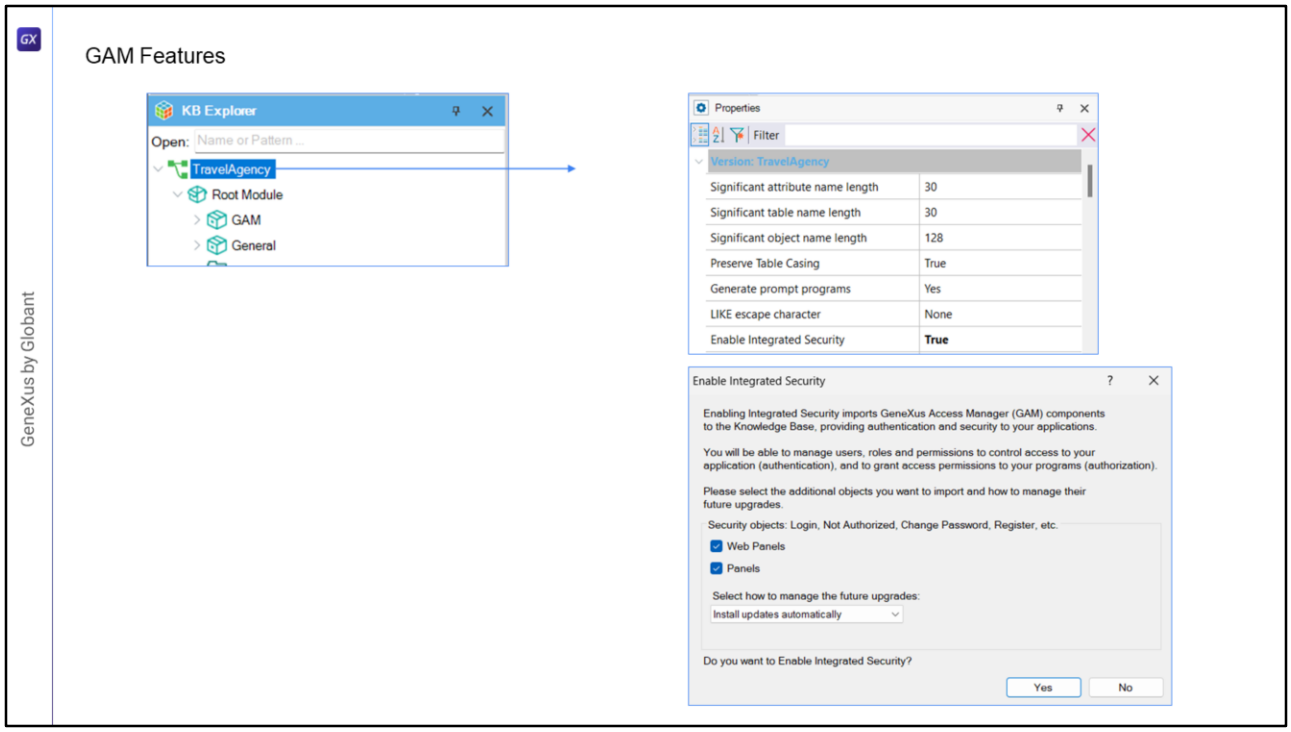
For Authorization, we define the permissions and execution of the objects and operation modes of transactions.

This is defined by granting permissions to each role for every object, and the actual

permissions over an object will depend on the roles assigned to users. For example, in mobile devices they are the WorkWith and Panel objects.

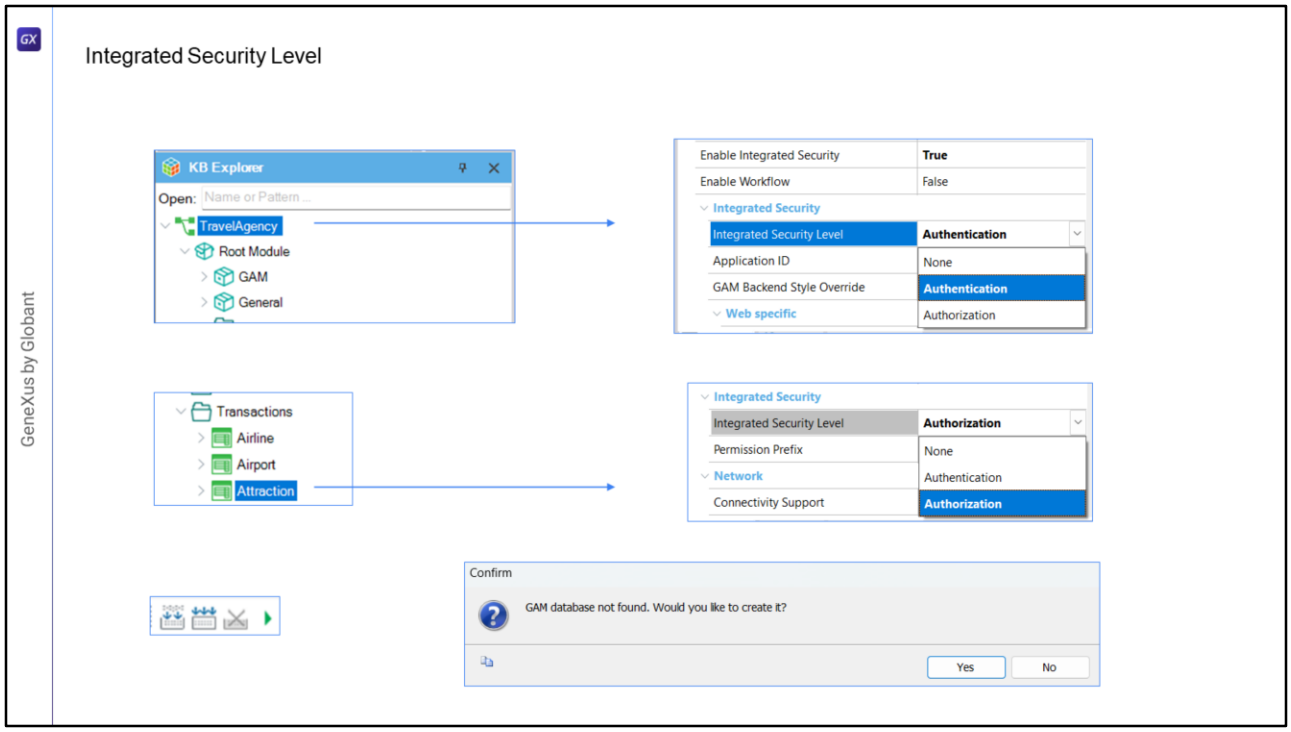


GAM also exposes an API to access its properties and methods in case it is necessary to do so from our application, and a series of Web services that can be used from other applications. We will not go into detail about this functionality in this video.



To enable GAM, go to the KB's active version level and set the Enable Integrated Security property to True.

When doing so, a dialog box will be opened informing that the GAM module will be installed in the KB, with the solution ready to run on the Web and on mobile devices.



Once GAM is enabled, we see another property called Integrated Security Level, which allows setting the default value for the security of KB objects.

This property is also at each object level, so it will be possible to customize the security of each object.

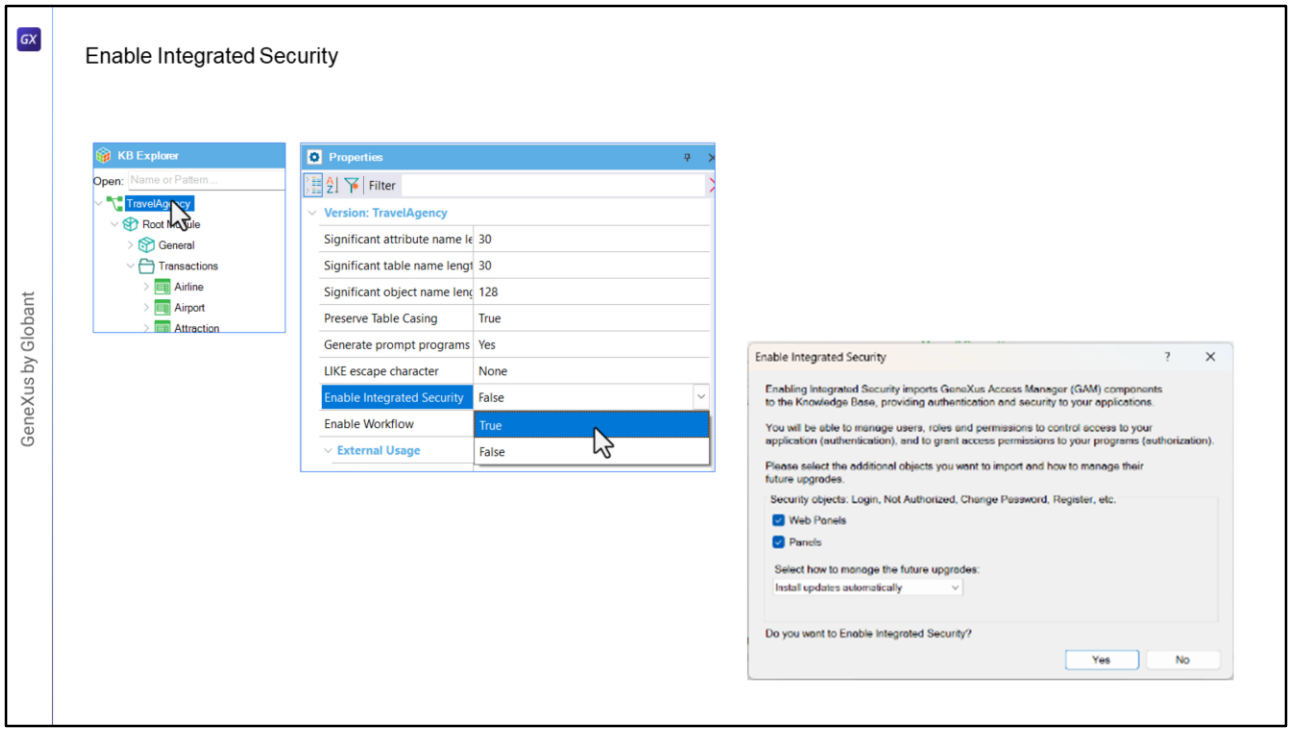
It has three possible values:

None: indicates that the object will be public; that is to say, it will have no security features.
Authentication: indicates that only authenticated users will be able to run it.
Authorization: indicates that, in addition to being authenticated, users will have to be authorized to run this object. That is to say, they must have the corresponding role to run it.

Once we have these security properties configured, GAM objects will be automatically imported into the KB and we will have to do a Rebuild All.

GAM will request to create a database that is independent from the application database. It will be associated with an independent Data Store in the KB, so its whole configuration is independent.

Let's see all this in an example.



We have created part of an application for a travel agency.

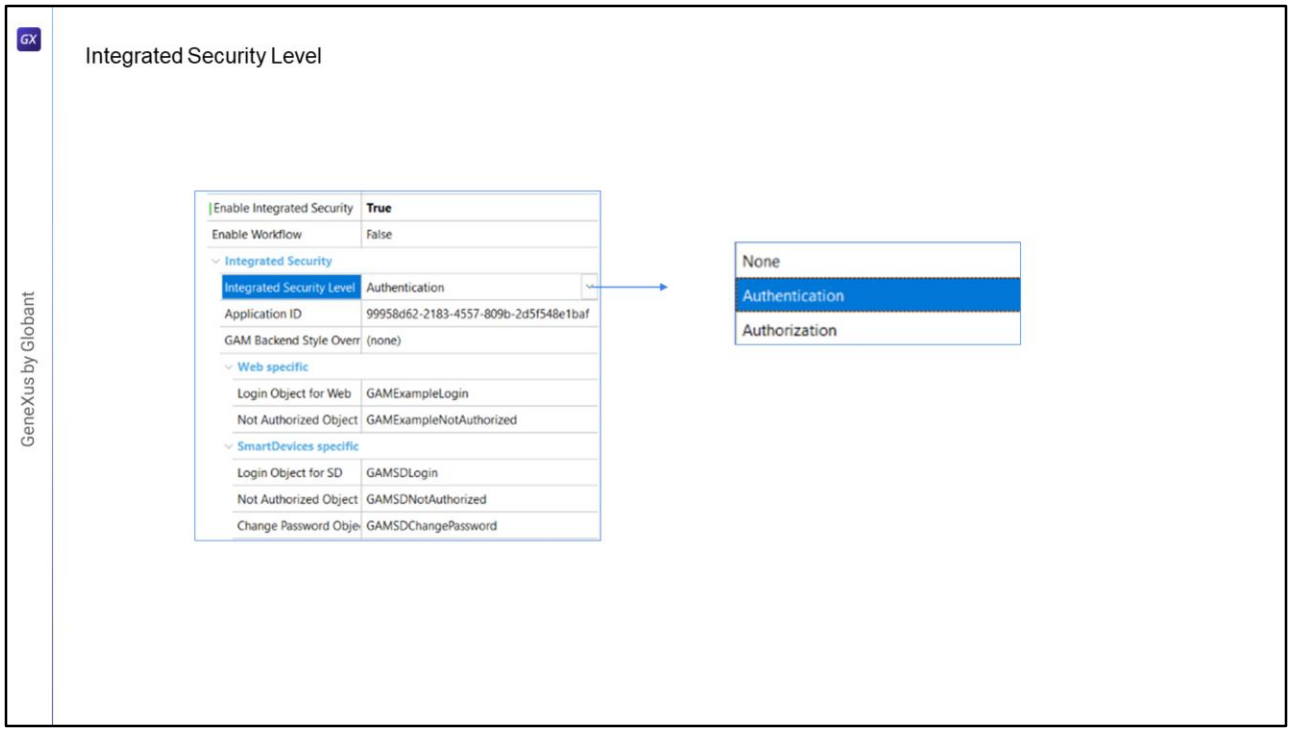
We are going to go to the Knowledge Base properties, click on TravelAgency, the name of the KB, and we are going to enable GAM by setting the Enable Integrated Security property to True.

Here we can indicate if we want it to be integrated in the objects used for Web applications, and/or in case we have some object generated for mobile devices, as in the case of the Panels or the WorkWith used for mobile, we can indicate that it is integrated to this type of applications; in this case, it must be selected because we are interested in applying it to a mobile application.

With this combo, we can choose how we want this module to be updated; it can be automatic, we can choose to be asked, or that it is never updated.

We confirm.

This is where GAM objects begin to be imported.



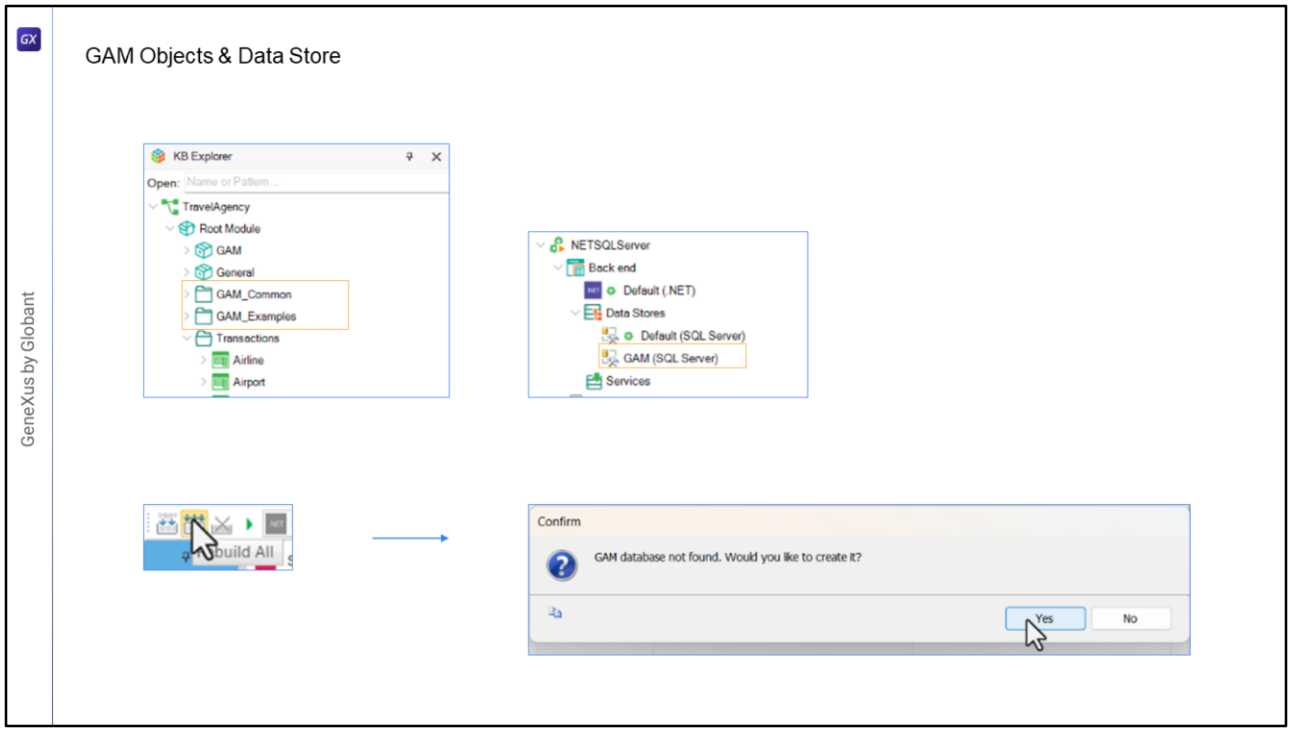
Now the Integrated Security Level property is available.

In this property, we can indicate if we want to enable only authentication, which is the default value, if we want authorization, or if we don't want to have security.

We will leave Authentication, which means that to access any object with security enabled, we will be asked for access credentials.

In addition, an Application ID is assigned that will be used in the GAM repository to identify the application.

Now we have the properties where we indicate the login objects, one in case of authorization error and another to change the password.

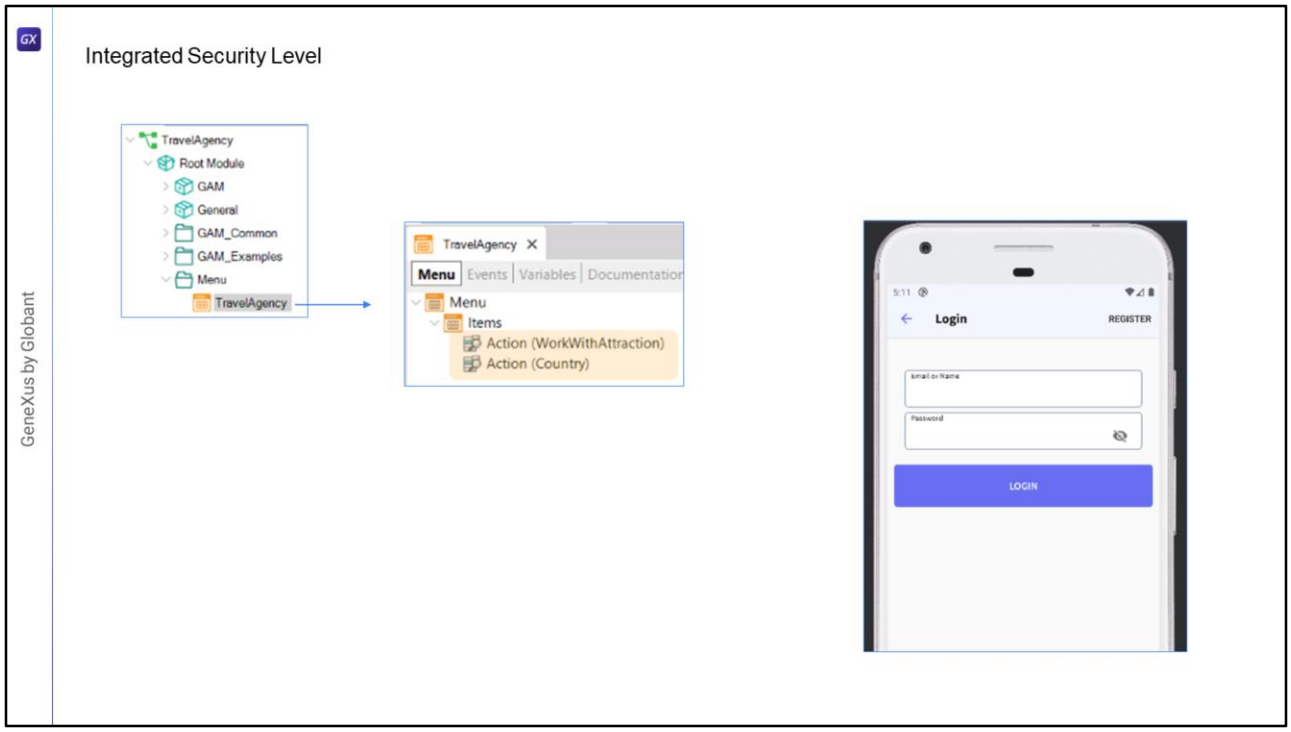


In the KB, we can already see that some folders were created in the root module.

We also have a new Data Store –GAM– with the information of that connection.

Well, after finishing the import process, we are going to do a Rebuild All.

GeneXus tells us that the GAM database was not found and whether we want to create it.
We select Yes.
The database is created with all the tables and then it is initialized.



In our application, we have a Menu object named TravelAgency that we set as startup object, and has these objects as items so that we can access it.

When we run it, the first thing we see is the login screen, since we configured Authentication as security for the whole application.
And if we want to enter without a user, we get an error. All this is provided automatically by GAM.

We will enter with a user that is created by default: “admin”, with the password “admin123”.
Then, we can access the application.

This is the end of the introduction to the GAM object for mobile devices.


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