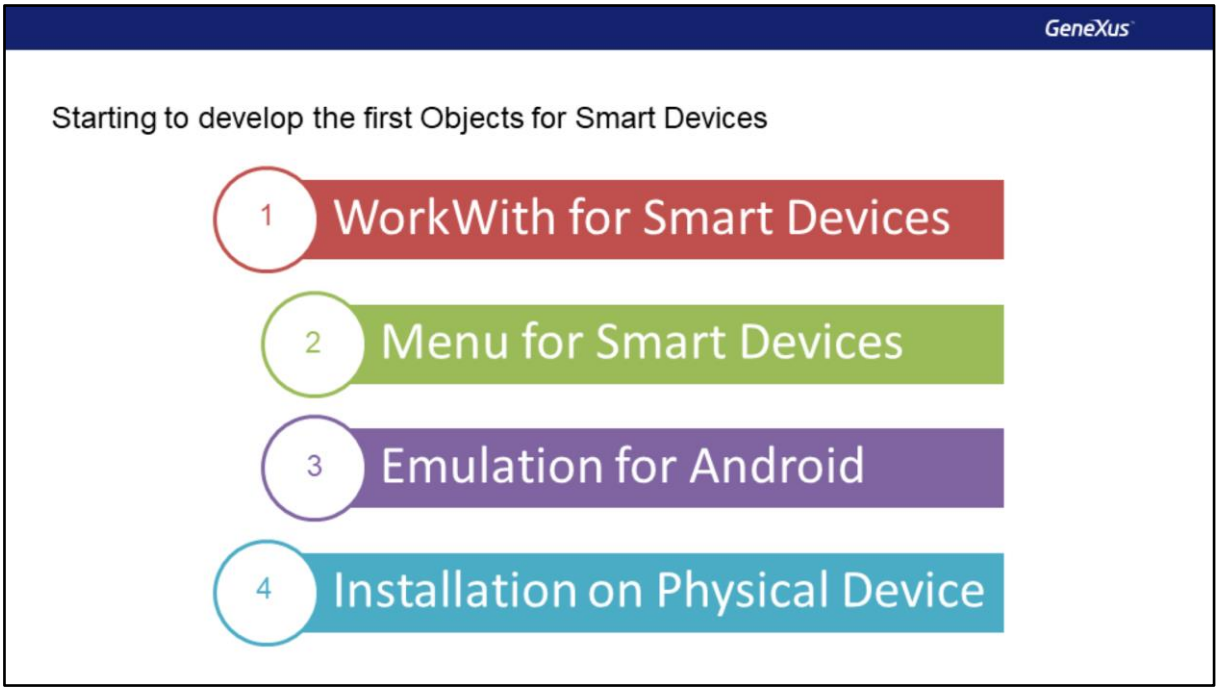

GeneXus™
The power of doing.

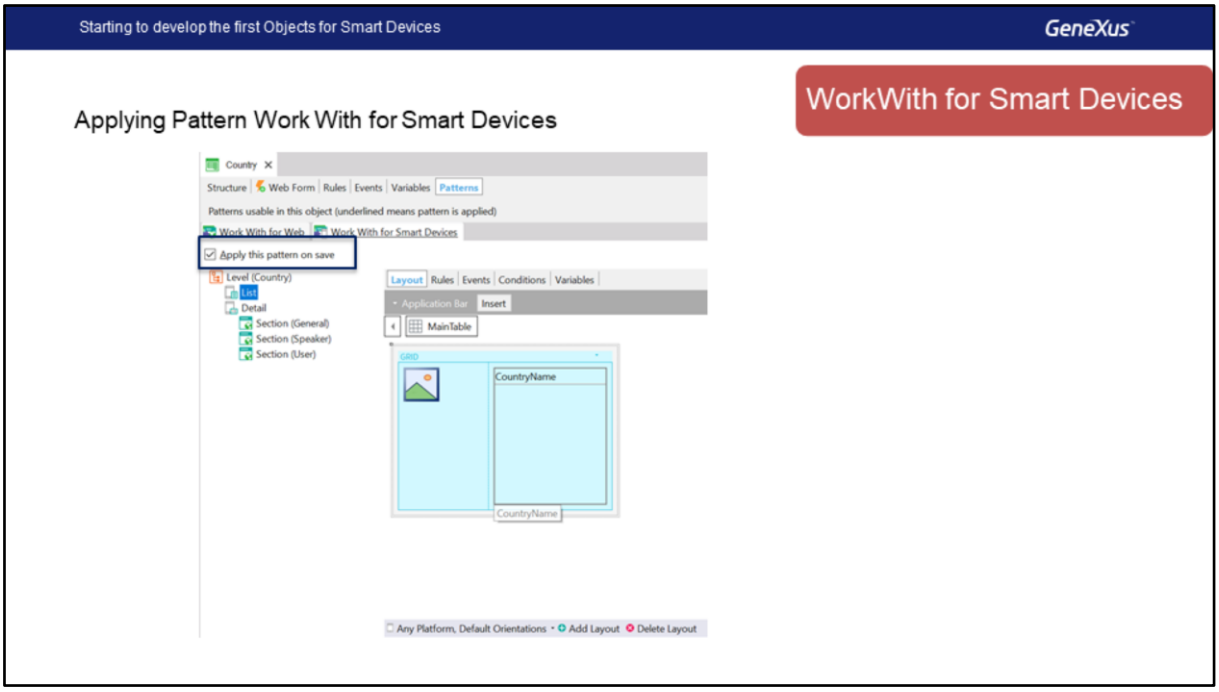
# Introduction

Starting to develop the first Objects for Smart Devices

GeneXus™ 16



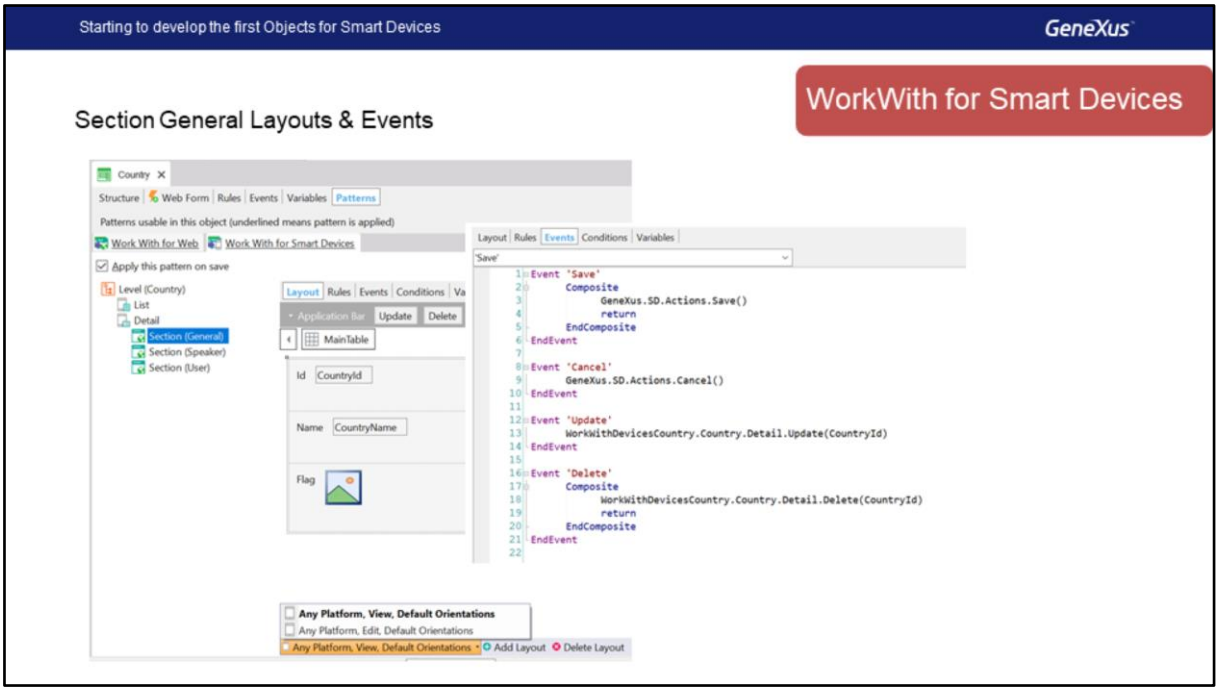

We will start by developing the Smart Device application. To do so, we'll see how to apply the Work With for Smart Devices pattern, create a Menu using the Menu for Smart Devices, and run the application for the first time using the Android Emulator, as well as install the application on a physical device.



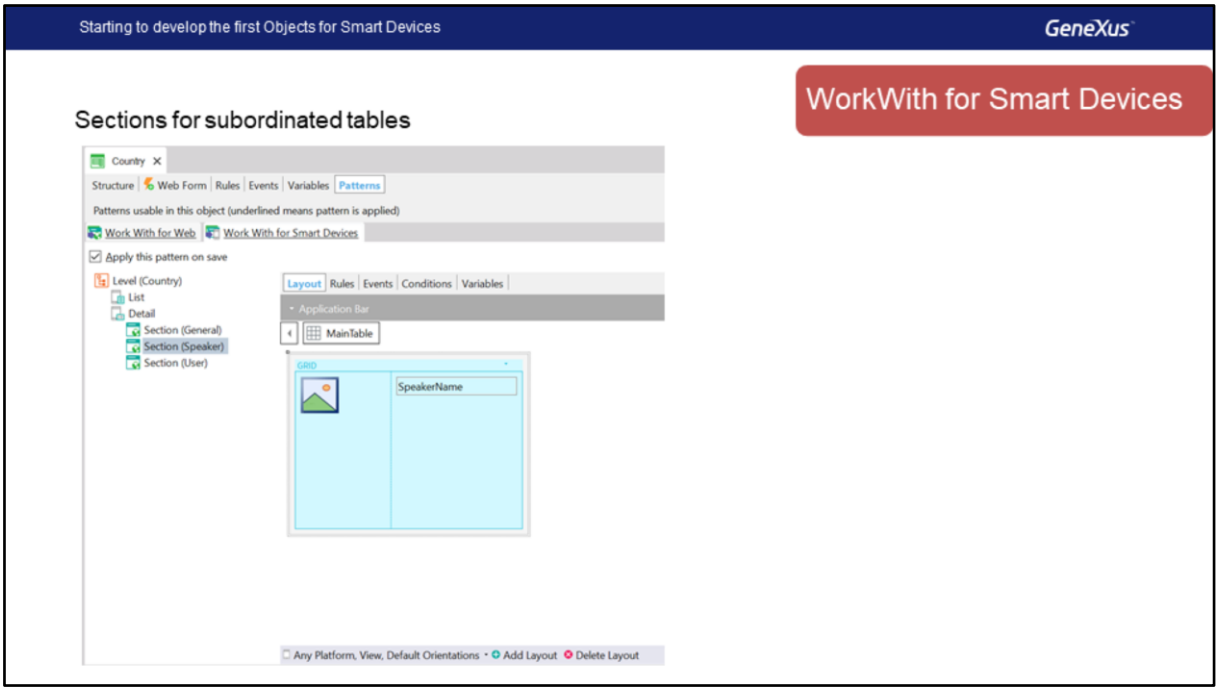

We return to GeneXus and apply the Work With for Smart Devices pattern to the Country transaction first.

Let's first review its properties; we see that in this case it is already a Business Component, and this happens because we've used it as a Data Provider to populate data. If we hadn't done so, it would be enabled when the pattern is applied, just like Expose as Web Service which in this case is set to False.

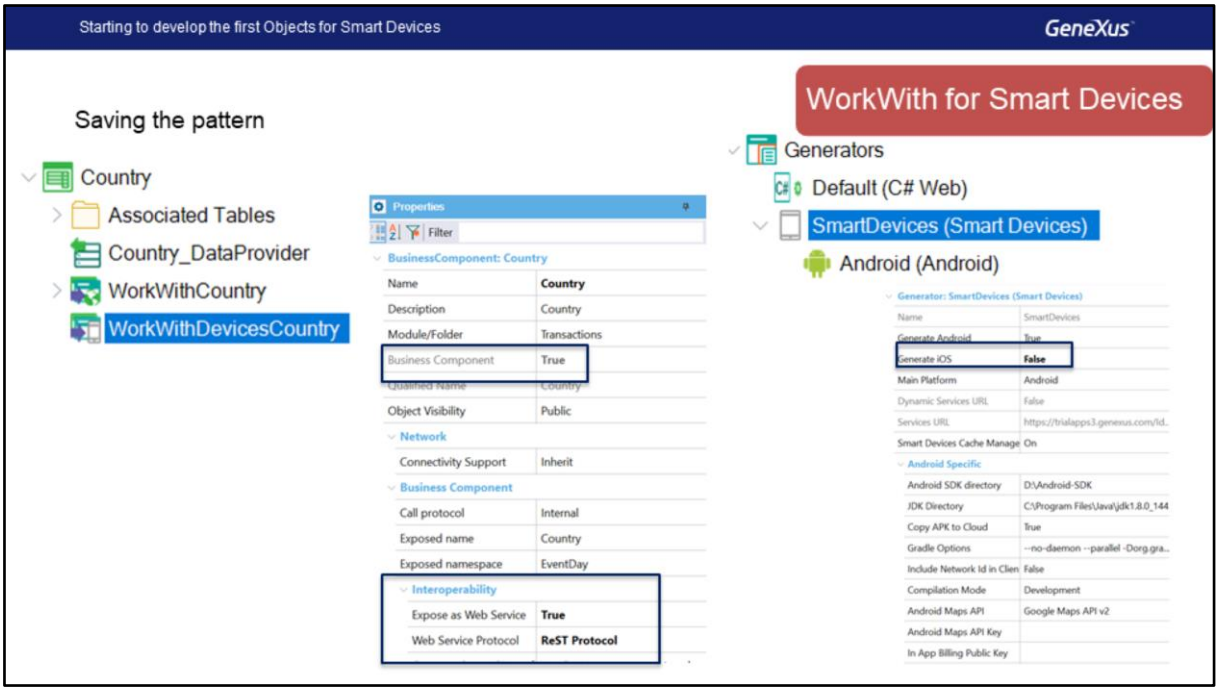
We select the checkbox Apply this pattern on save and review, for example, the events already defined in the Country level, the list that implements the list of countries with the Insert button in the Application Bar. In Detail a series of sections will be implemented and they will be automatically changed at runtime.




The first section we are going to see is the "General" Section that has Update and Delete options and a specific layout for edition that will implement the Save and Cancel options. We see that all the events are already implemented. We return to the Layout.

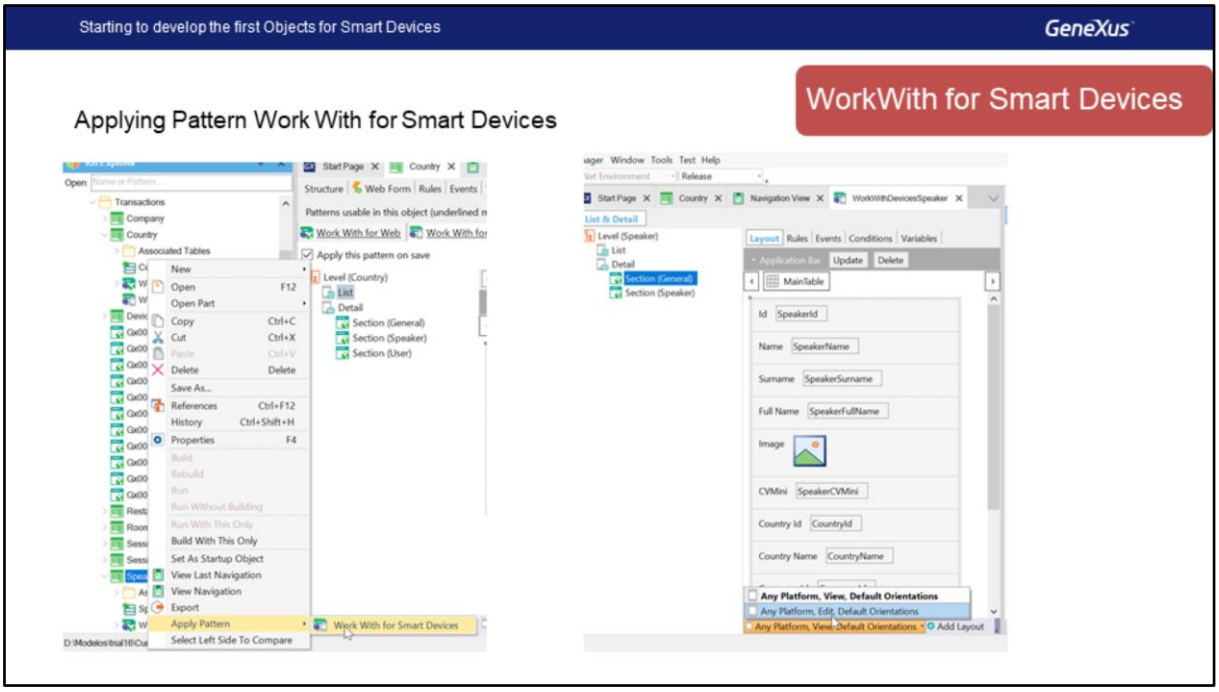



Also, we'll see a Section with the Speakers of that country, which will be a grid in this case. The same happens with the users of that country.



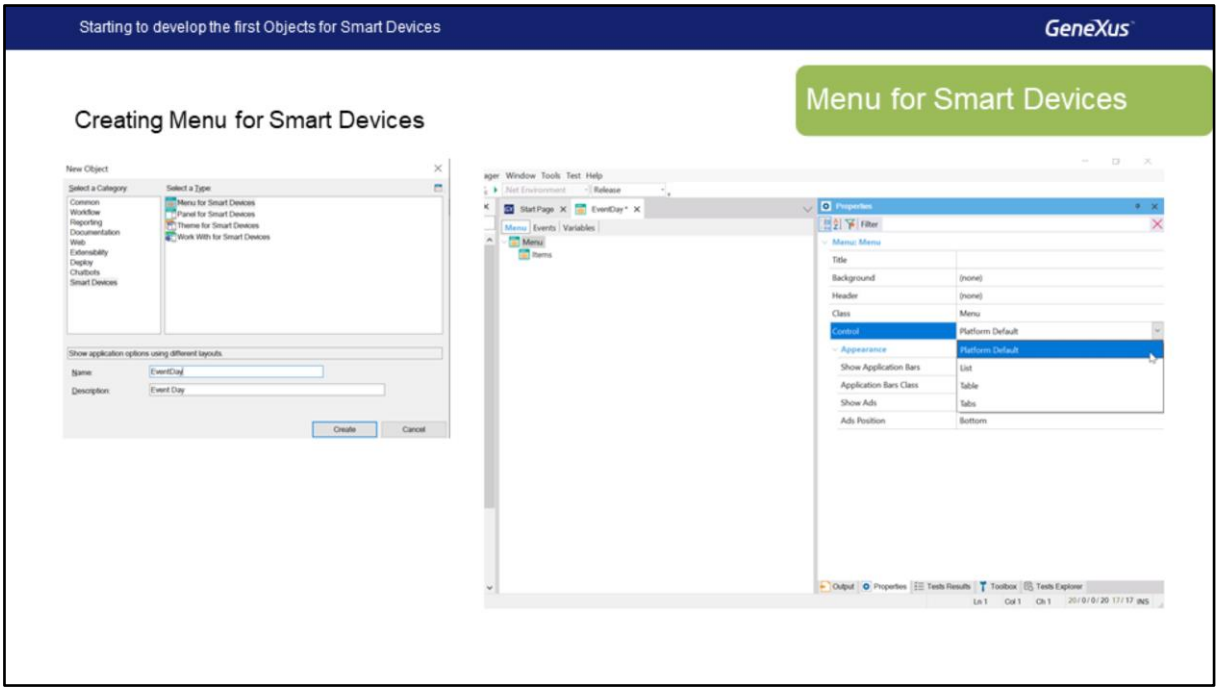

We are going to save and see what things change after saving. First, we see the object associated with Country, WorkWithDevicesCountry. The property Expose as Web Service has been enabled and the protocol used will be REST; we'll see this when we talk about application architecture.

Now we have the Smart Devices generator, but we will only use Android for now (we leave the property Generate iOS set to False). Its properties are where the Android SDK is installed.



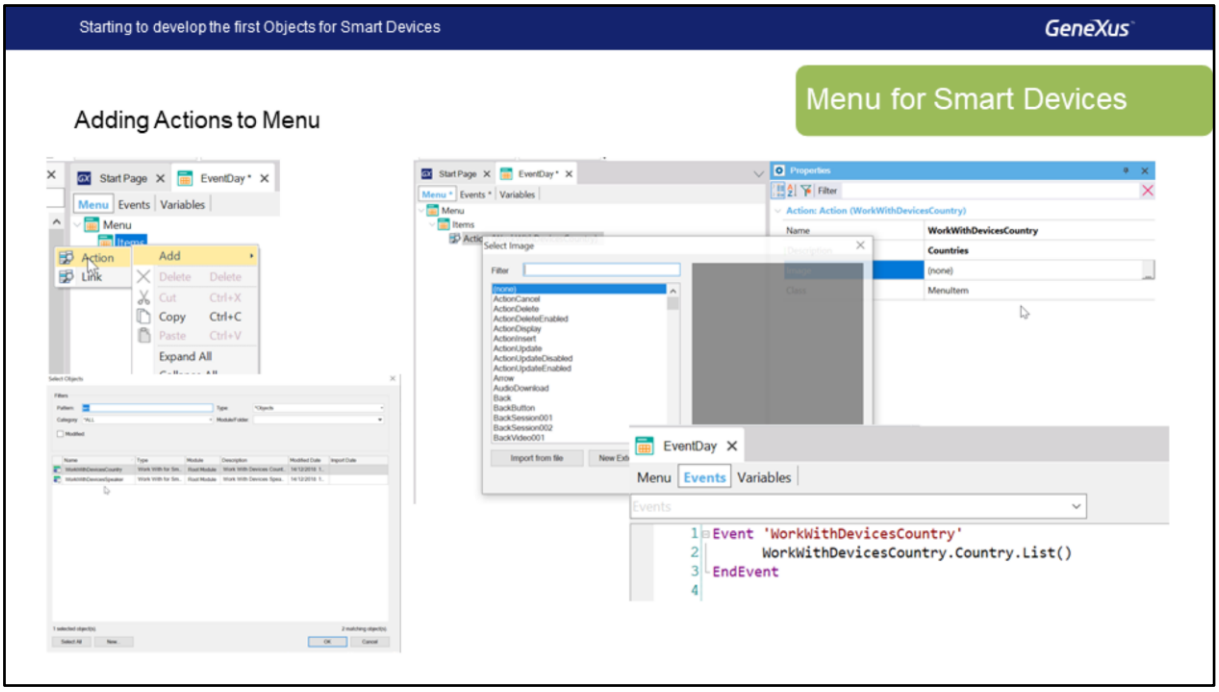

Now we will apply the same pattern to Speaker. We may do so in the same way, but we will use another way to apply the pattern, which is to right-click (on the transaction), Apply Pattern -> Work With Smart Devices.

The pattern is now generated and we can access it directly. Also, the list is implemented, in this case the Speakers, and the Section General with View and Edit mode for each speaker. Also, we see that a section is implemented where the sessions for that speaker will be displayed.

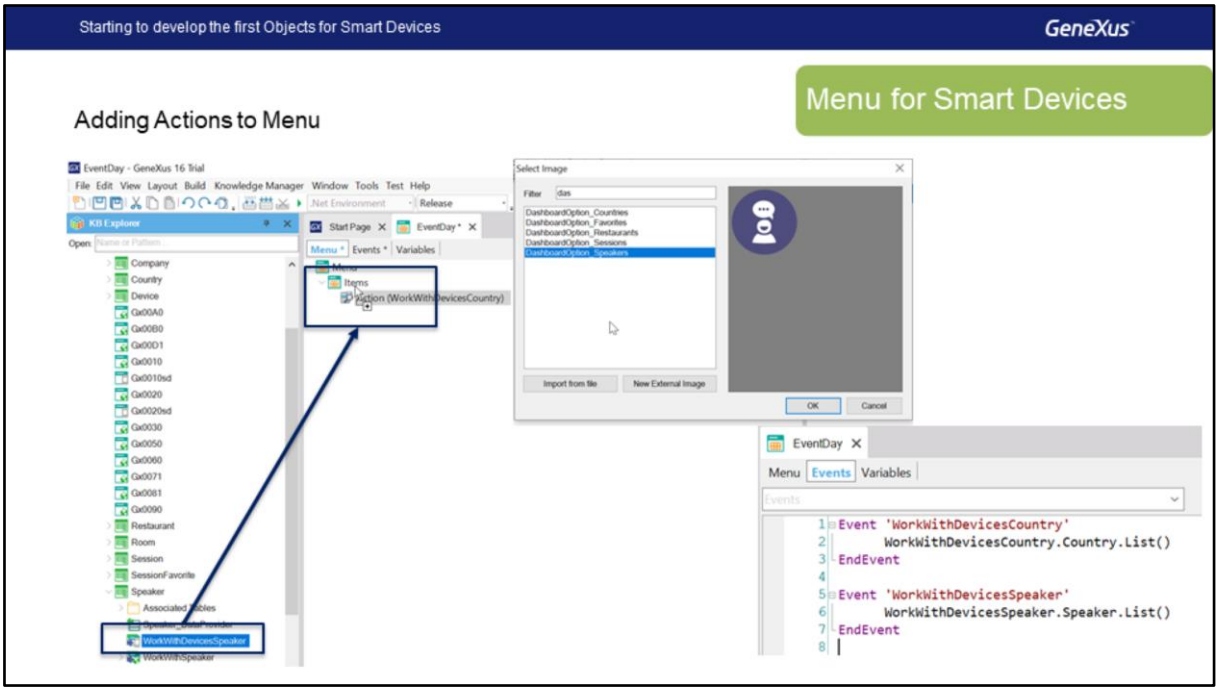



Well, let's continue. To run our application, we are missing the menu, so we create an object of Menu for Smart Devices type, select File, New Object –note that all the objects of that type are grouped in Smart Devices–, select Menu For Smart Devices, call it "EventDay," and click on Create.

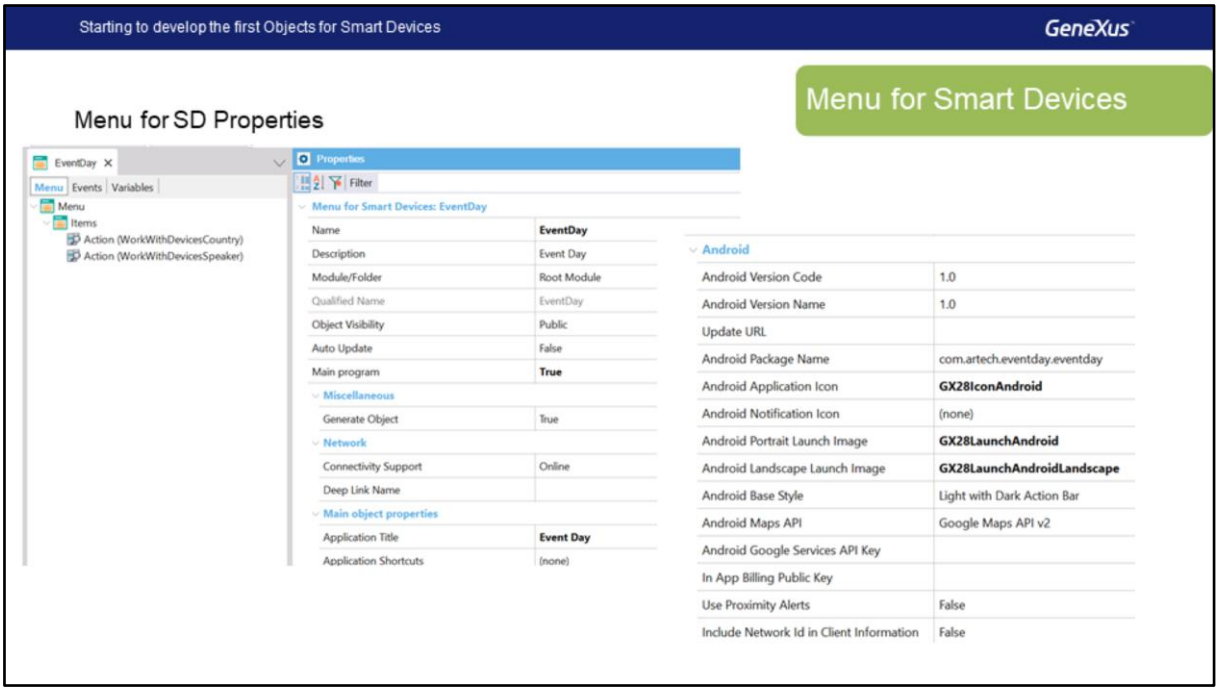
As we've said before, first we will see this Control property of the menu, which will allow us to choose how the menu will be displayed. It can be a List, Table or Tabs. For now, we leave it as Platform Default, which for Android will be a table.




We'll add items by right-clicking and selecting Add, Action. The object selection dialog will be displayed, and we'll select WorkWithDevicesCountry, click on OK and the action will be added. In addition, we see that the event corresponding to that action is also added, with the name of the action which in this case takes the same name as the object, and the call to workwithdevicescountry.country.list() that is the list of countries.
We change the description to make it more user-friendly and call it Countries. Also, we select an icon.
You will already find some icons in your KB after importing the xpz.
Ok, we select "dashboardOption_countries."

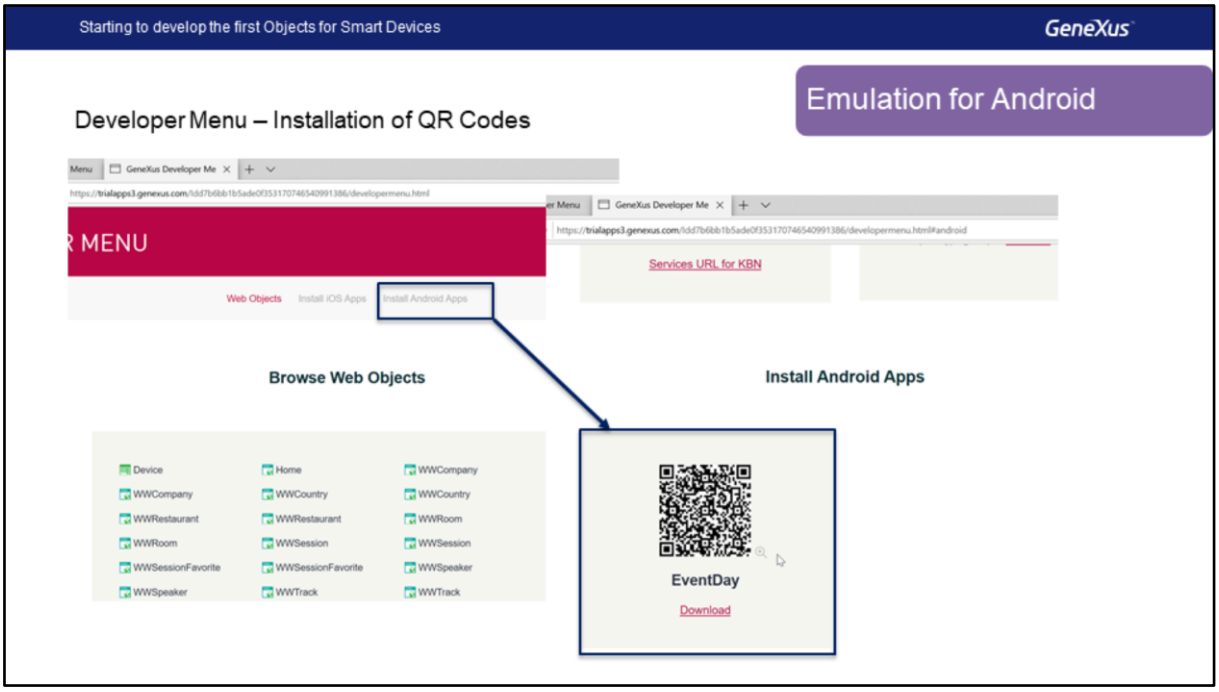



Now we have to do the same to call Speakers. Another way to do so is by selecting the object workwithdevicesspeaker in the KB Explorer. It could also be a Panel for Smart Devices: we Drag and Drop over items, and automatically see that an event is called again with the call to the Speakers List.
We change its name for Speakers, select the image, dashboardOption_speakers and we already have the menu.




Next, we need to enter a title for our application, so we call it “Event Day” again;
this is the name with which the application will appear in the device.
Let's select an icon for this application, in this case we will select Gx28IconAndroid. Also, we are going to select two images, which will be shown as the application is launched, one for Portrait and the other for Landscape mode: Gx28Launchandroid and Gx28Launchandroidlandscape.

We have everything ready, and save the menu.
Now we move the menu, with this icon or with F5; let’s wait for a moment as the changes are transferred to the cloud.

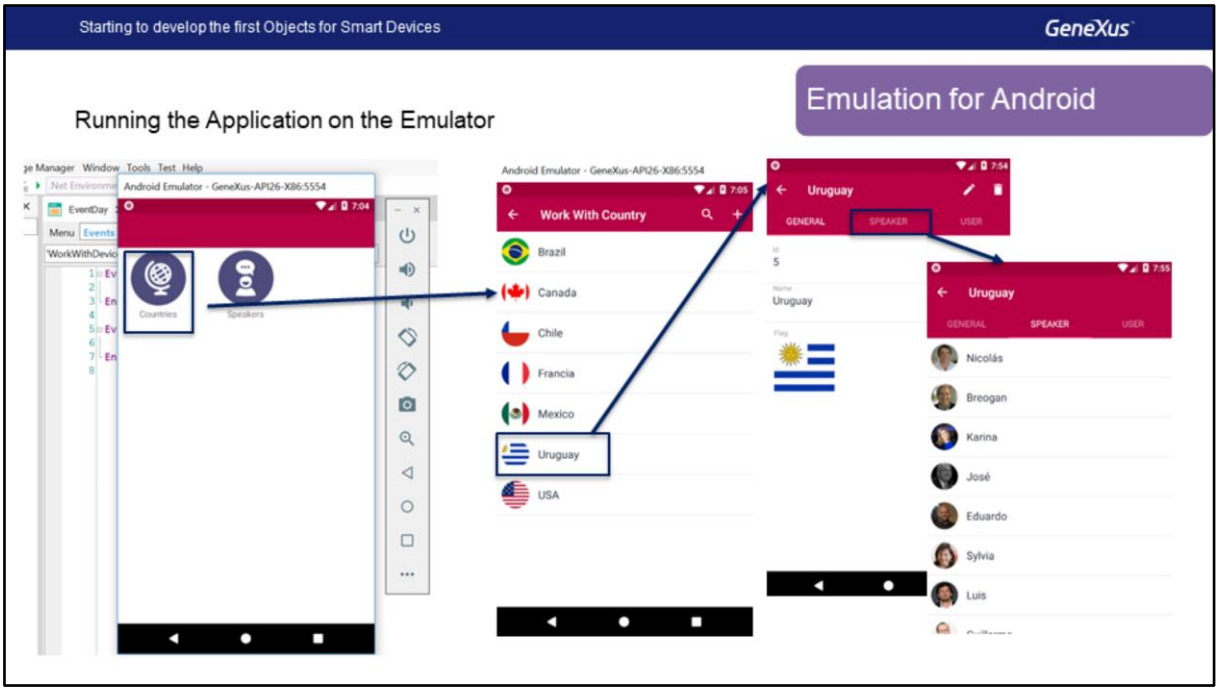



We can see that the Developer Menu is started.

I'm going to mention something that appeared as a mistake: when we have an Android environment (or IOS), to run the menu it must be a Startup Object, and we can do this on the object (right-click), Set As Startup Object. The other option is to directly select RUN for this object (right-click and select RUN) so that it doesn't show this message, which is simply a Warning.

Here is the Developer Menu, now we see that in addition to the web objects we had the links; there are two links above, we go to Install Android Apps. GeneXus provides us with a QR code, if we scan this QR code into a physical device, this QR is actually a link and will allow us to download the Android APK to the device. We'll see how to run it later.

First let's run it on the emulator, then let's select RUN directly on the EventDay menu.
Now the APK will be compiled, and the emulator will run on our machine. Here we can see that the emulator has been started.
When using the emulator, remember to leave it open so that it runs faster. The next step is to install the APK in the emulator, which directly executes it.

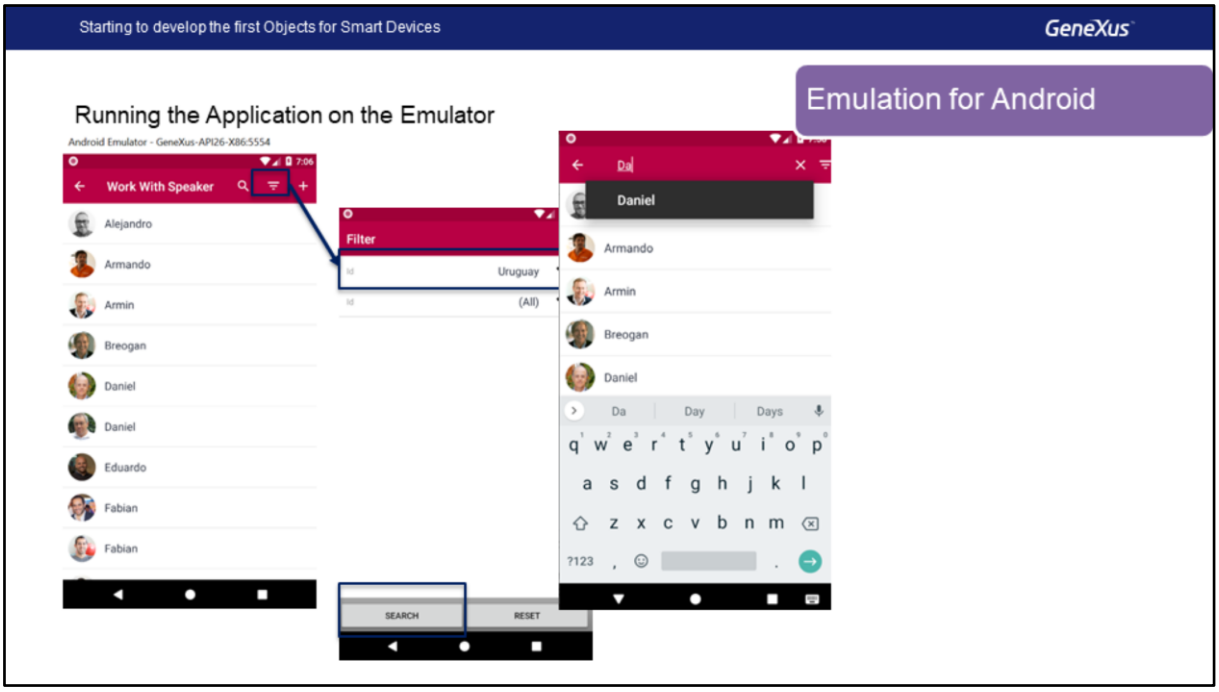



Now we have the application in the Android emulator, let's check how our objects look like. We see the two icons that we had specified, and we are going to enter Countries. Here we see the list of countries, each one with its flag. We select Uruguay, for example, to see its general information.

Here we're in the View of Work With for Smart Devices for Countries. Remember that we had a General section, a section with the speakers of that country, and a section with the users. For now, we don't have any users.

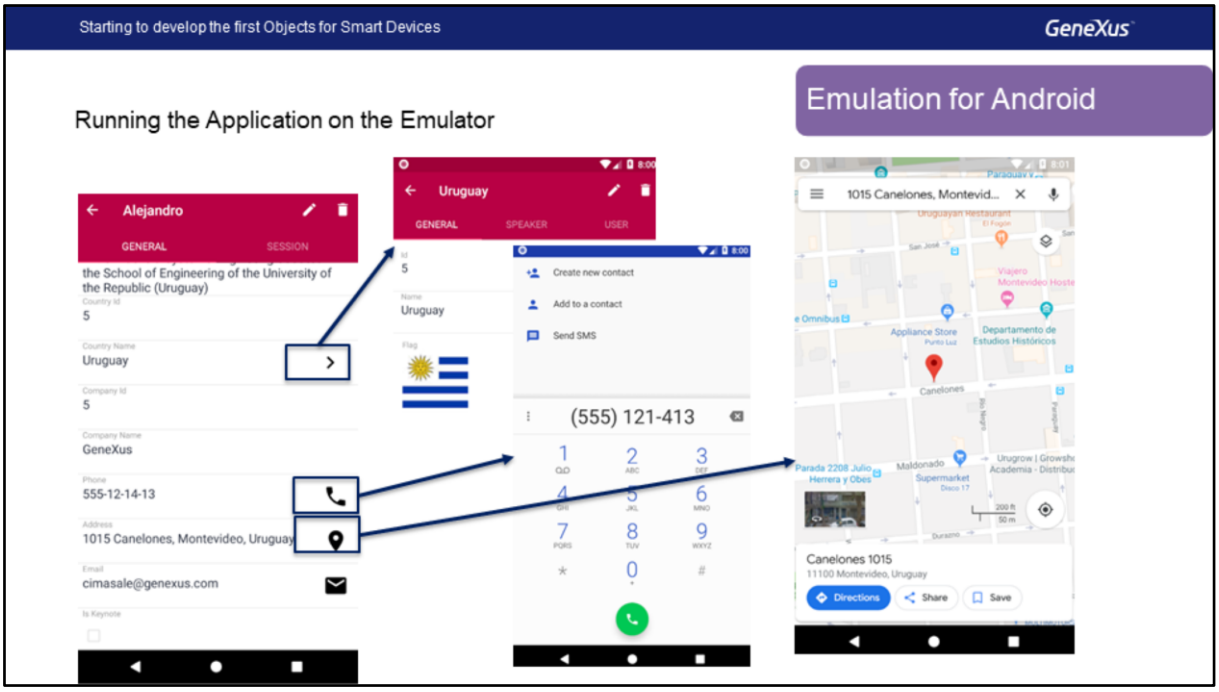
We see that we have icons to update and delete a country.

Also, there is an option to insert in the List.




In addition, we have Speakers, and here is the List of Speakers, with the option to insert. Here appears an icon that in the other case did not appear. This icon provides advanced filters; for example, it will allow us to select the country and to look for the speakers of that country. It indicates that a filter is being applied.

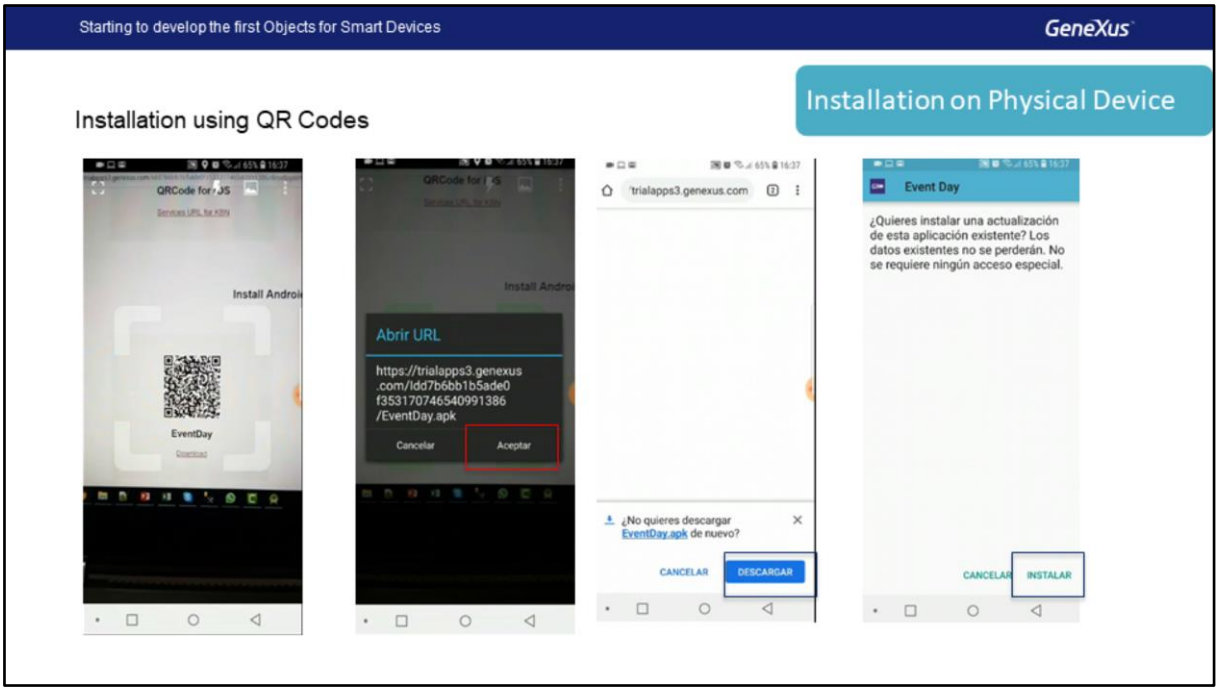
We could also search directly, and also provides a suggestion -Daniel. Here Florencia is displayed and we will see that it not only searches by name, the Workwithdevicesspeaker in the List. In the grid we will see that the Search has a filter by Country, by Company ID, in advanced filters, and in Search the system will directly look by SpeakerName, SpeakerFullname, Curriculum Vitae, Address or Email. Then the search can be done by several attributes.



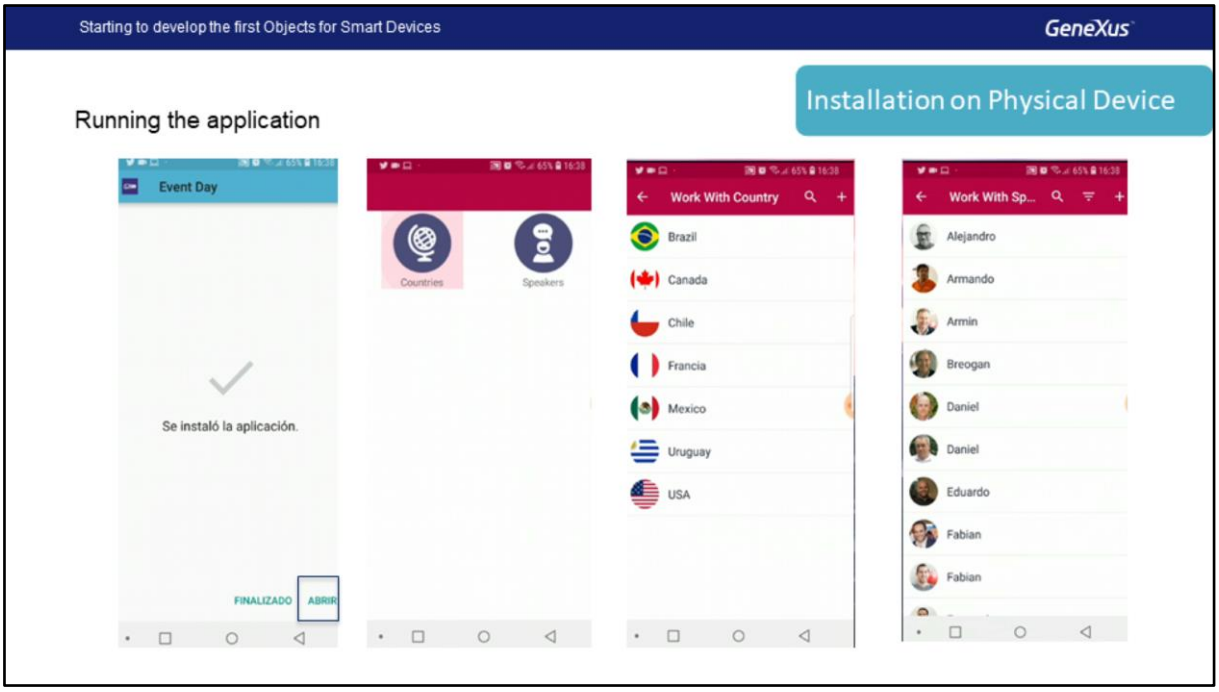

Let's see a speaker, Alejandro in this case. We access that speaker's View; note that here we have these icons, which are automatically displayed when they are Foreign Key and takes us to the View of that Foreign Key.

These icons appear automatically when dealing with some semantic domains, for example, when we use the semantic domain Phone we see that this icon appears and when you TAP on that icon it will directly open the application, in this case to make a phone call.
Or, for example, if we look at the address, the icon will take us to a map. The default application to handle this type of information is opened and there we see the icon that appears with the address. The same for email, which will automatically open the application to send an email.

Let's edit Alejandro's details. For example, we'll change his phone number, which ends in 12, and change it to 13. We confirm. A message indicates that the information was updated and we already see the information here.
In the web backend Work With Speaker, we look for Alejandro, and see that the information has been changed. This happens because the application is working online, and automatically all the changes we make to the device will be reflected on the web, directly in the database and vice versa.
The next step will be to run the same application on a physical device.




Now let's see how to install the application on a physical device using these QR codes.
Here we are seeing a cell phone, now I am going to scan with a QR reader the code that shows the Developer Menu. This will start the download.
Please note that we need to enable the developer options in Android and also allow applications from unknown sources because by default this is disabled so that we can only download and install applications from Google Play in the case of Android.
When the download is complete, we open the file. This file is an Android APK. We install it.




Well, now we can open it and see that the application is exactly the same as we saw in the emulator with the icons, and accessing the same data.
Many times this is the best option for prototyping that will be available, because the physical devices have features that the emulator doesn't have. Also, they will run faster.

Another option is to connect the device to the PC using a USB cable. We need to have the drivers installed, GeneXus in this case will detect the device and will install the application in it without using the emulator and with no need to scan the QR code.


| | |
|---|---|
| Videos | training.genexus.com |
| Documentation | wiki.genexus.com |
| Certifications | training.genexus.com/certifications |