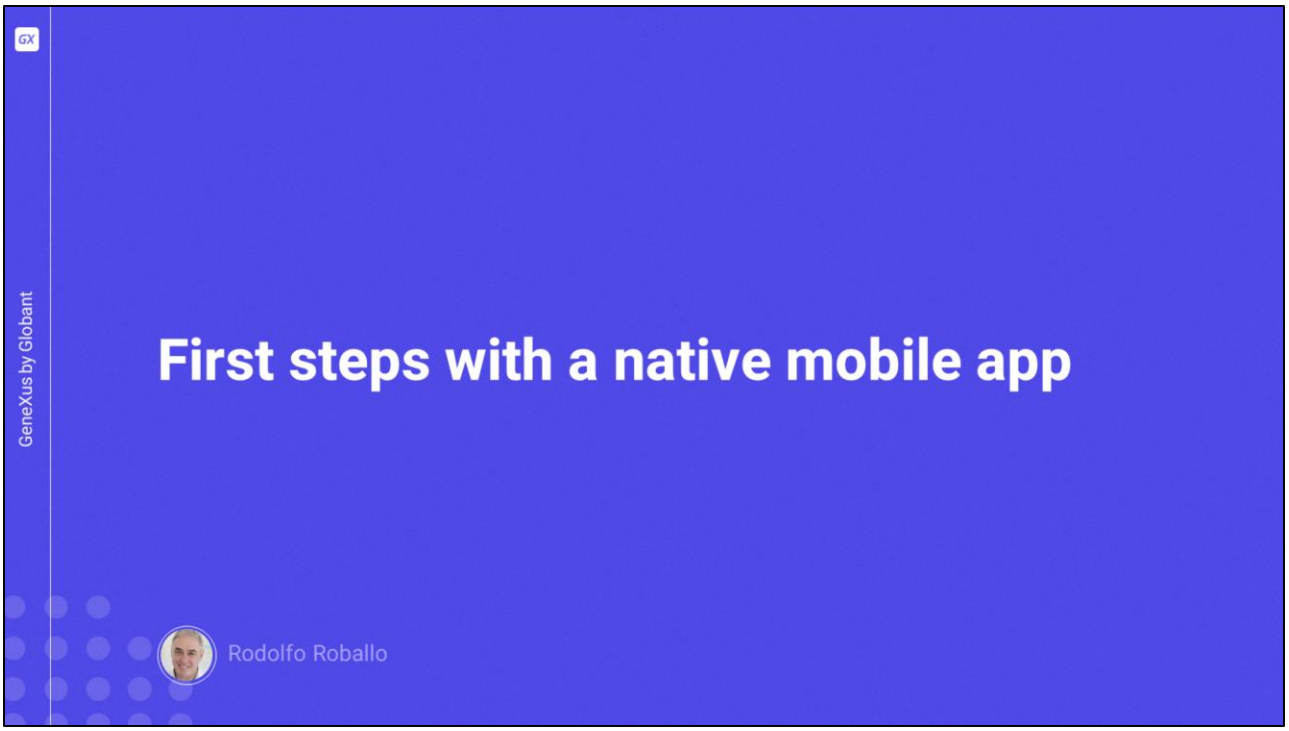
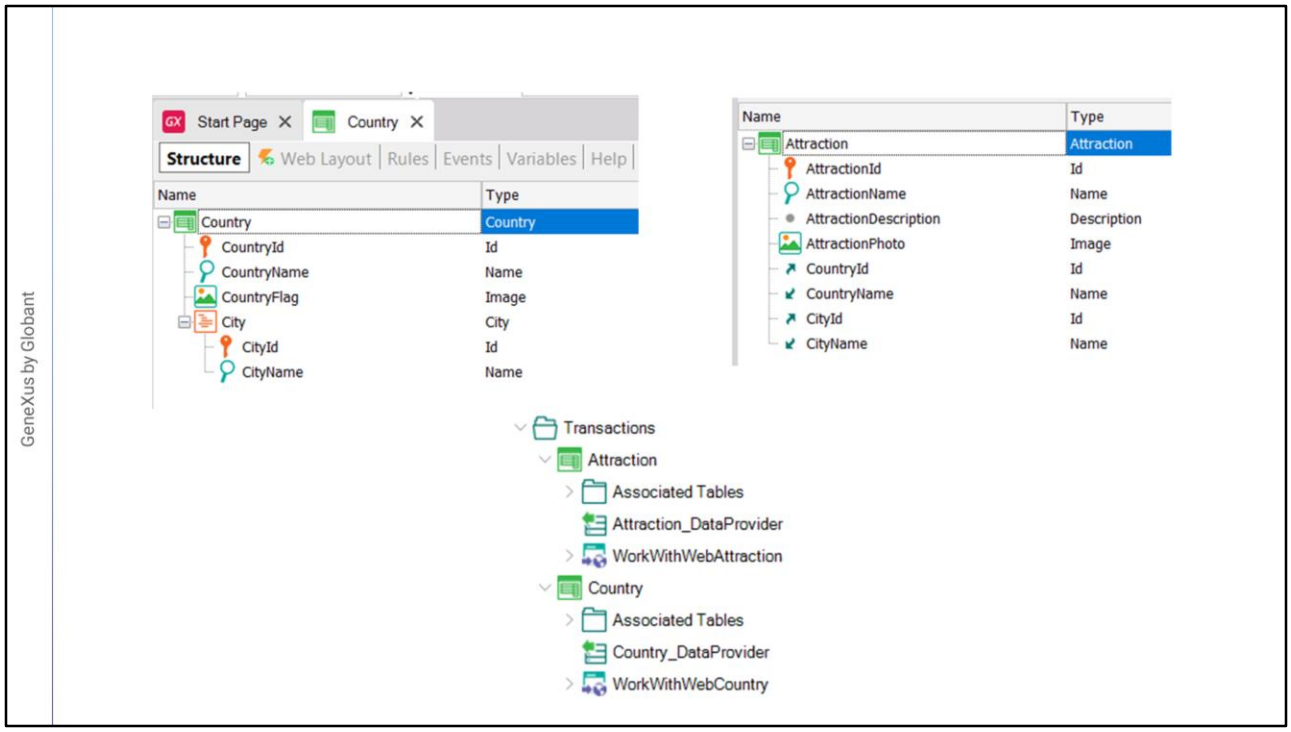


Now we will start to develop our first native application for mobile devices.

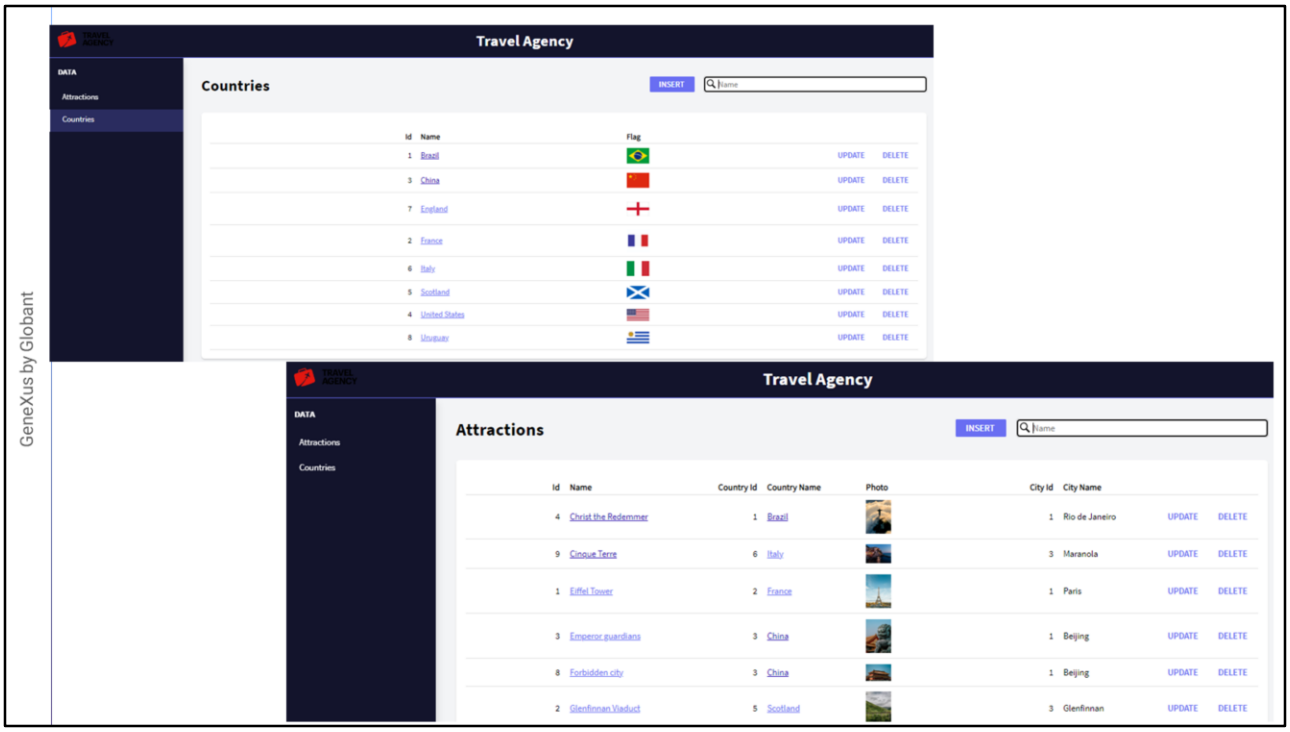


Here we see that there's already a KB with some objects. Since our examples will be based on the reality of a travel agency, we have the Countries transaction with basic data such as their identifier, name, and flag, as well as a subordinate level to model that a country has many cities and to store the identifier and name of each city.

We also have a transaction for the Tourist Attractions promoted by the travel agency with their identifier, name, description, photo, and the country and city to which each attraction belongs.

Each transaction has its own data provider to populate the associated table with data at the time of its creation in the database. We also see that the transactions have the Work With for Web pattern applied, so these objects are part of the travel agency's back office application.

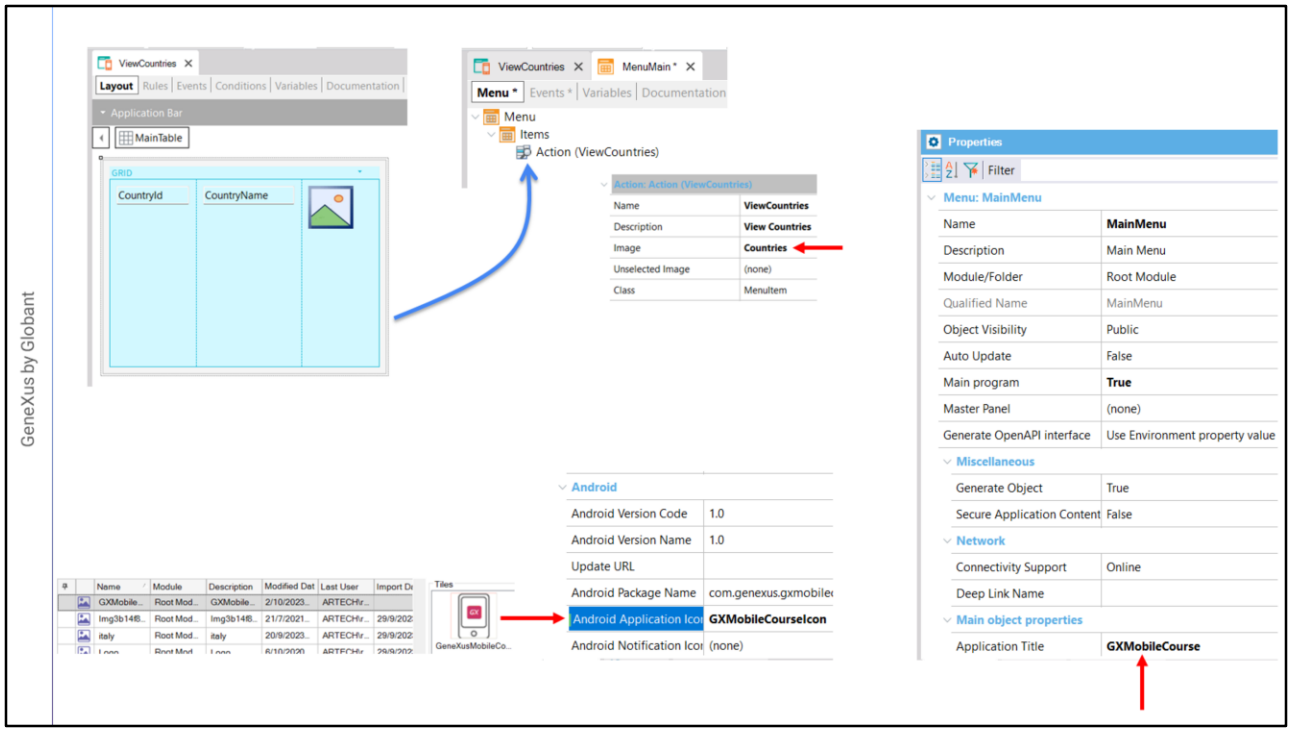
These objects are available in an xpz file called ImportBeforeFirstRun.xpz that you can download from the course page, in case you want to create the KB from scratch and import the xpz to leave it in the initial state of this video.



Before creating our first mobile screen, let's press F5 to run the back office screens and get to know the data we are going to use.

We open the launchpad and run the WWCountry object to see the countries data. We select a country and see its cities.

On the left side of the screen, a menu is available that allows us to work with other entities, such as tourist attractions, so we open it and see that there are several attractions entered with their main data.

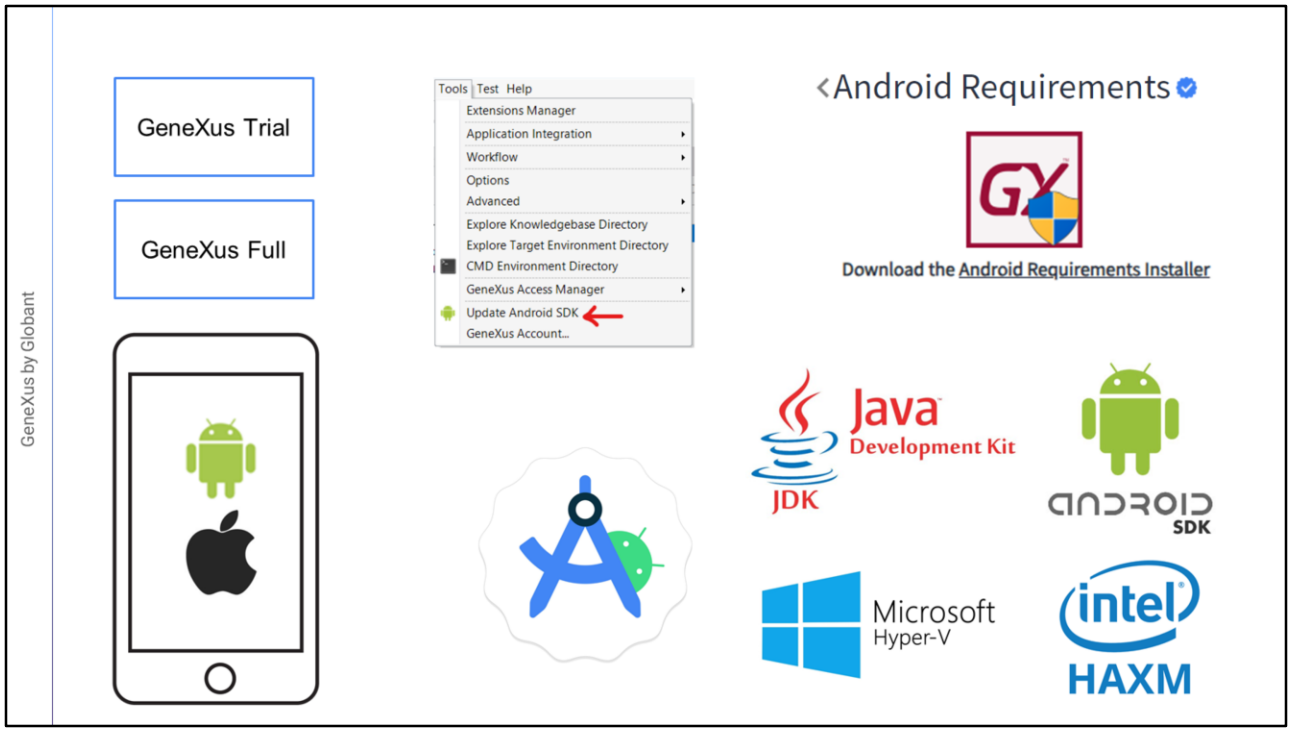


Now let's start developing the screens of our mobile application. We create a panel object called ViewCountries, drag a grid control to the form, and select the attributes CountryId, CountryName, and CountryFlag. We save.

This object could be set as main, but it makes sense to invoke it from another object that opens when running the application; for example, a menu.

Therefore, we create a menu object named MainMenu and drag it to the ViewCountries panel to its Items section. In the properties of the created action, we assign the Image property to the Countries image.

If we now go to the properties of the menu object, we see that by default it is a main object. We assign a name to the Application Title property, and under the Android section we choose the image that will be the application icon. Lastly, we set the menu object as a startup object.

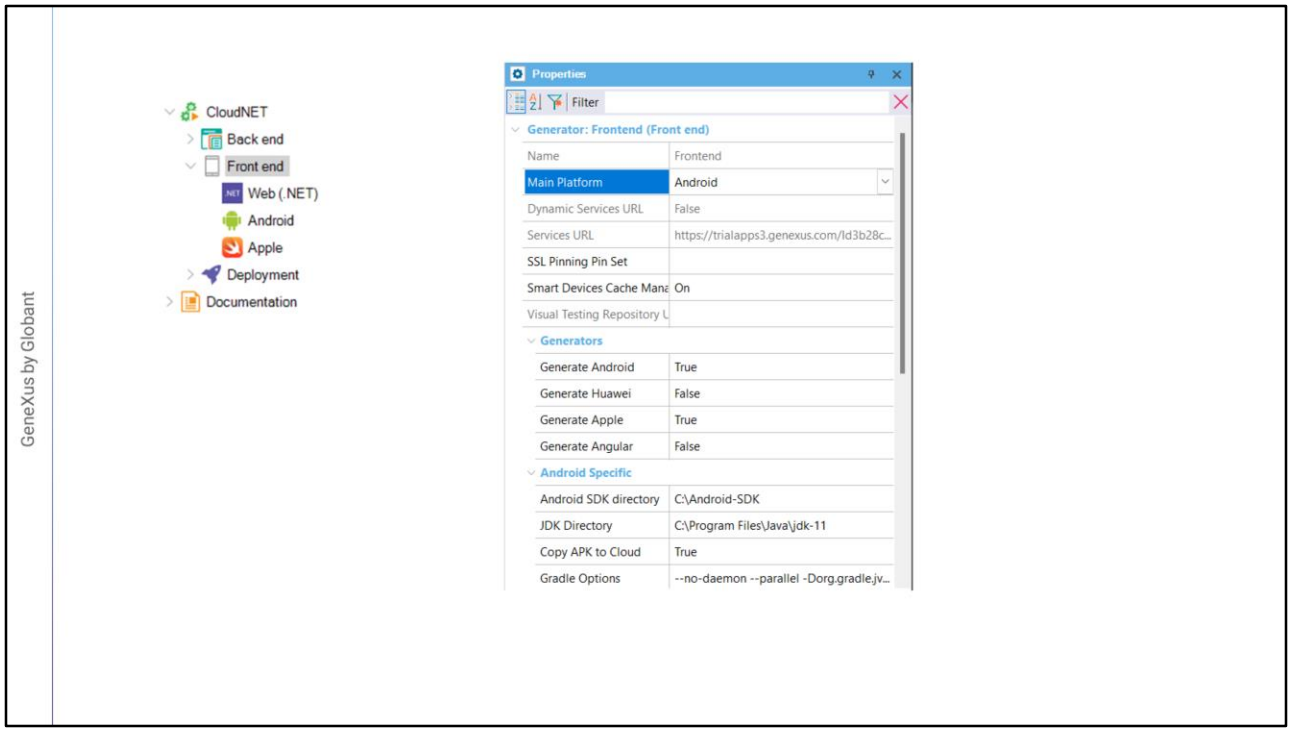


Before running the application for the first time, we must verify that we meet the requirements to generate a mobile application. In this course, we will generate for Android and Apple. We will start prototyping with an Android emulator, although later on we will see how to prototype on Android and Apple physical devices.

To do this course, we just need to install GeneXus Trial, which will offer to automatically install the software needed to generate applications on Android. But if you have installed the GeneXus Full version or you did not accept the installation of the Android environment, the GeneXus IDE provides the Update Android SDK tool that will install all the necessary requirements of the Android SDK.

Another option is to search for "Android Requirements" in the wiki, where we will find an automatic installer of the Android requirements. In addition, if necessary, there is detailed information to manually install, for example, the Oracle JDK and the Android SDK, as well as the Hardware Accelerated Execution Manager (HAXM) from Intel or the Windows Hypervisor Platform (WHPX) from Windows to make the emulator work faster.

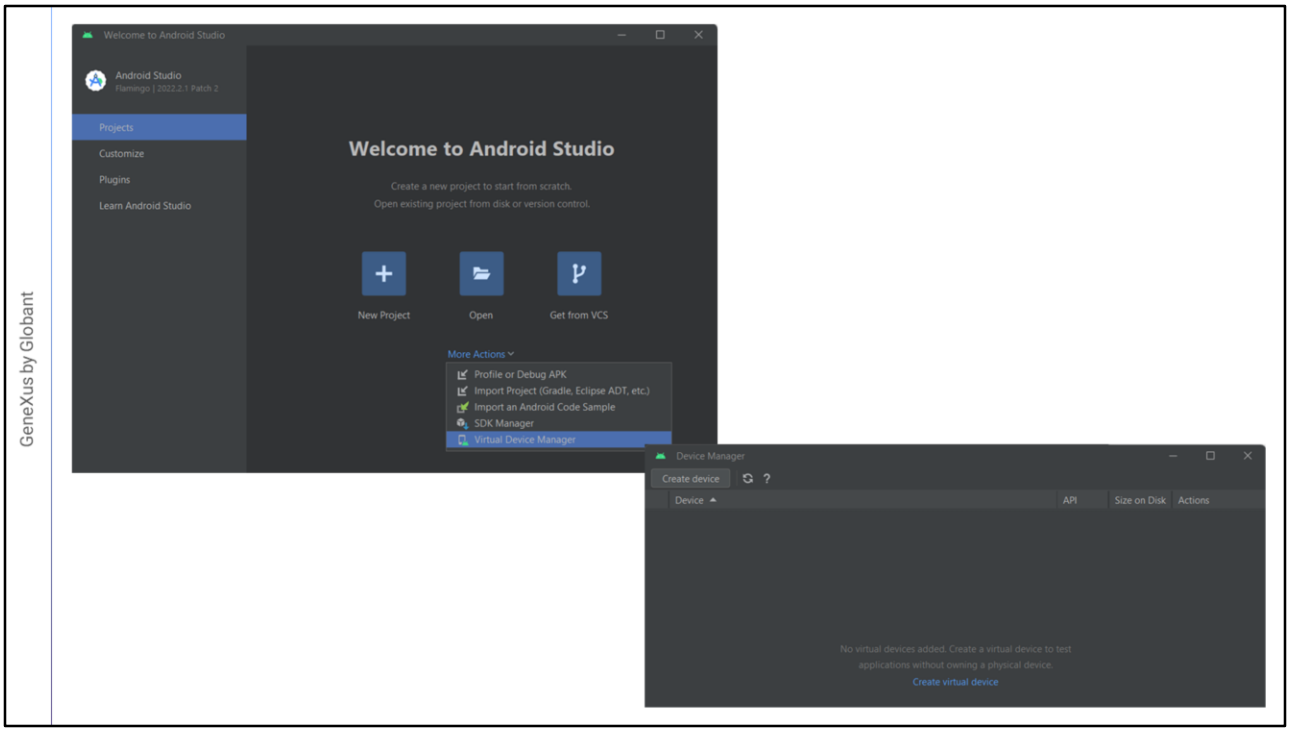
Even though GeneXus can automatically create an emulator of an Android device, it is convenient to also install Android Studio, which will allow us to create customized virtual devices and monitor the performance of our generated application.



Once we have all the software installed, we are going to run our mobile application for the first time, but first we must verify that our front end settings are properly configured to do so.

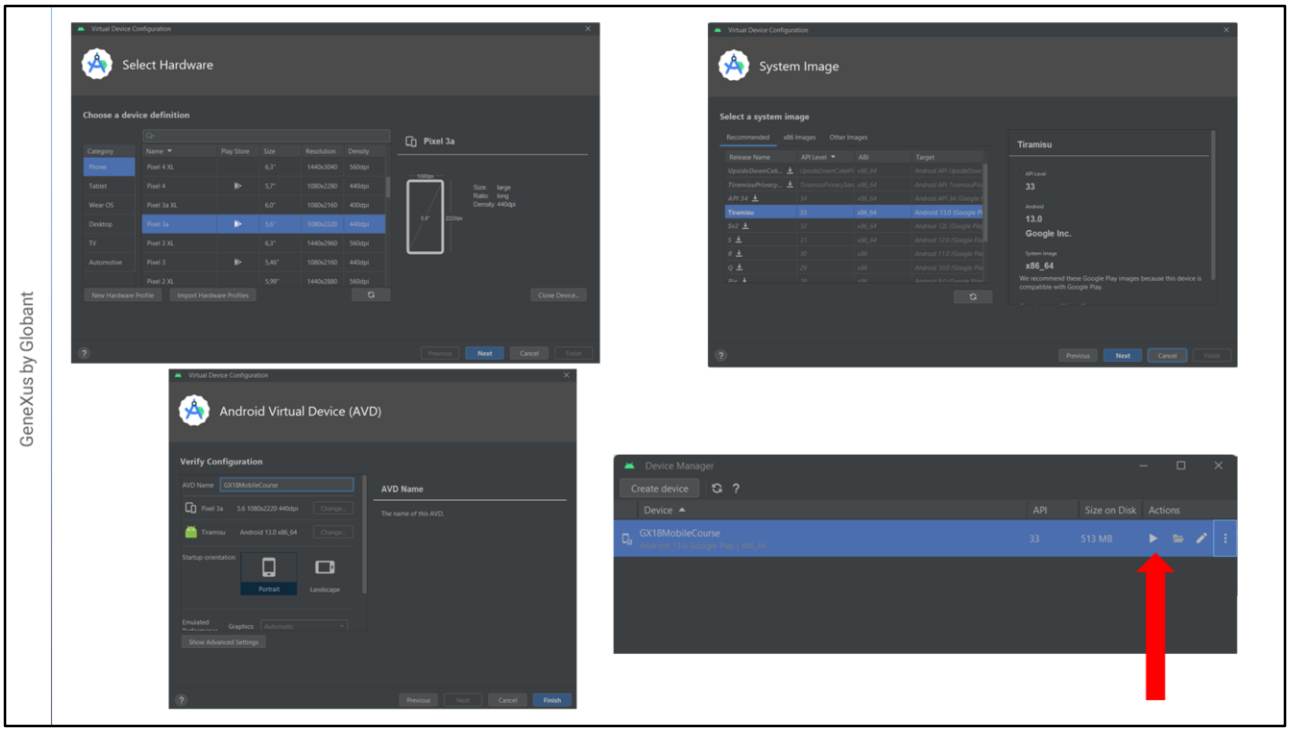
We go to the KB explorer and open the Front end section. By default, Android and Apple are displayed, and if we go to the node properties we see that this is because the Generate Android and Generate Apple properties are set to True by default.

We also see that the Main Platform property is set to Android and that, below the Android Specific section, the Android SDK directory property is automatically set to the directory where the Android SDK is installed; the same goes for the directory where the JDK is located.



Before running, let's open Android Studio. We click on More Actions and run the Virtual Device Manager.

We will create an emulator that we like, but above all, that is suitable for what we are going to do in the course.

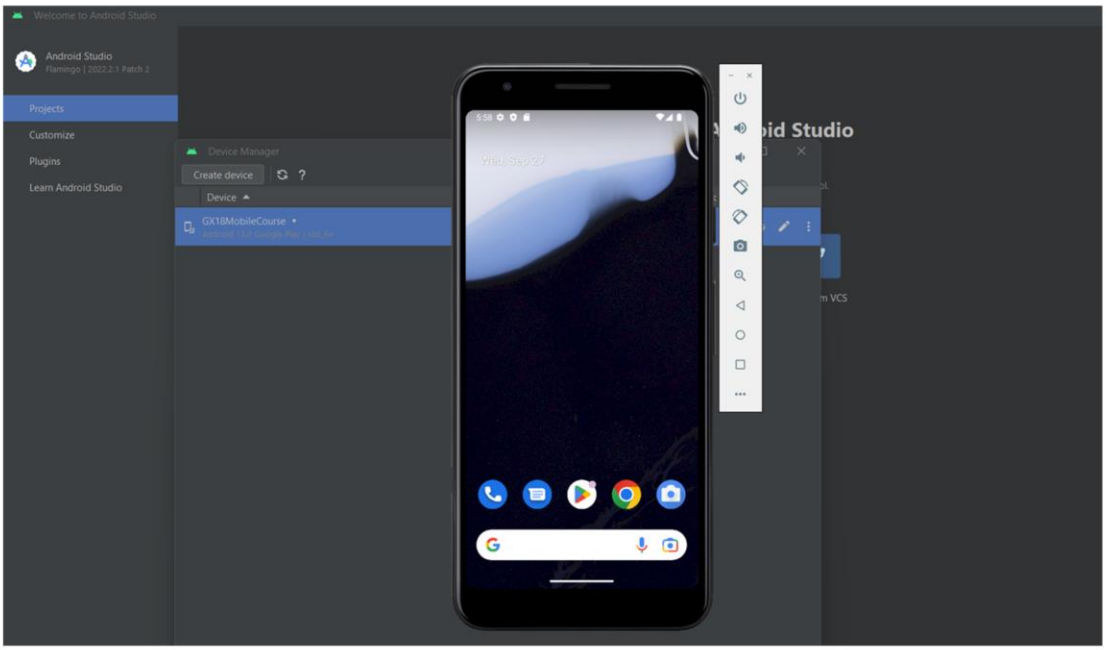


We click on Create Virtual Device and then click on Next. We choose the one called Pixel 3a and click on Next.

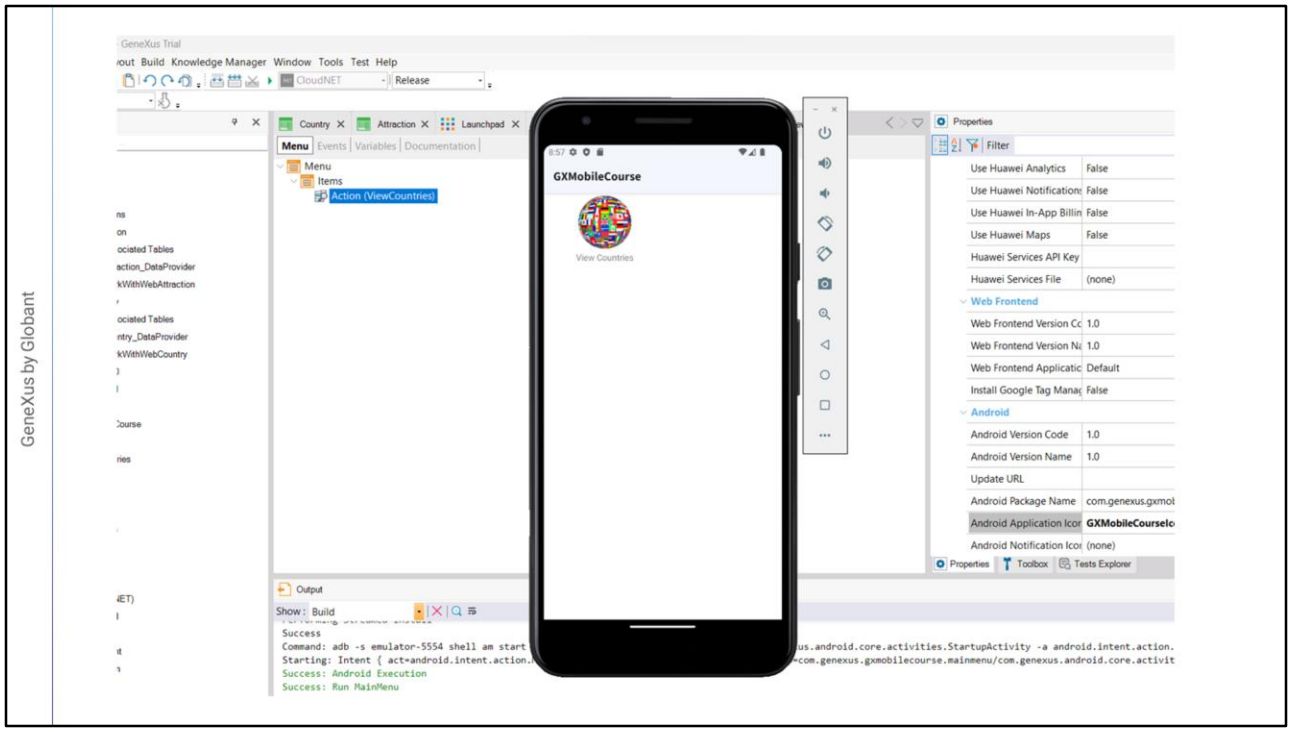
We select the line corresponding to API 33 and click on Next.

We enter the name GXMobileCourse for our Android Virtual Device and click on Finish.

To test it, we press the play button ....

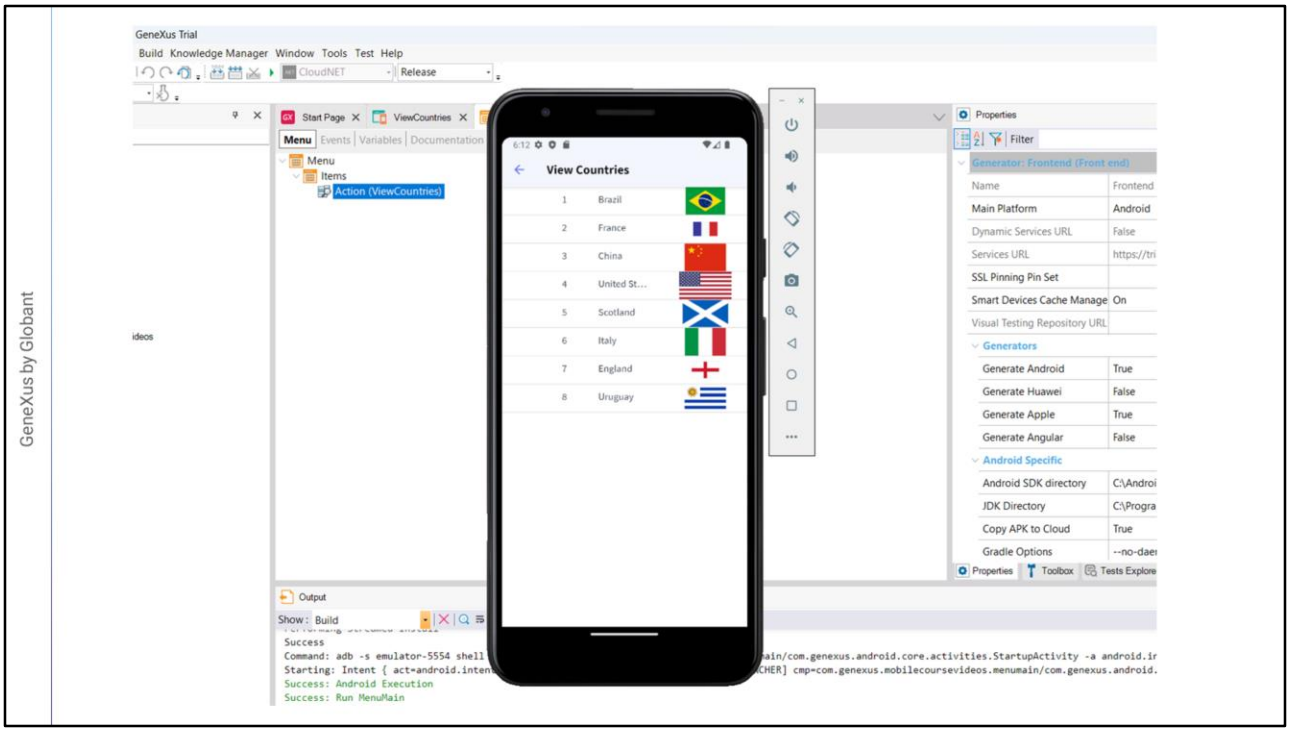


After a few seconds, the emulator opens and looks identical to a real device.



We return to GeneXus. Since we had already set the MainMenu object as a startup object, it will be the object run in the emulator, so we press F5.

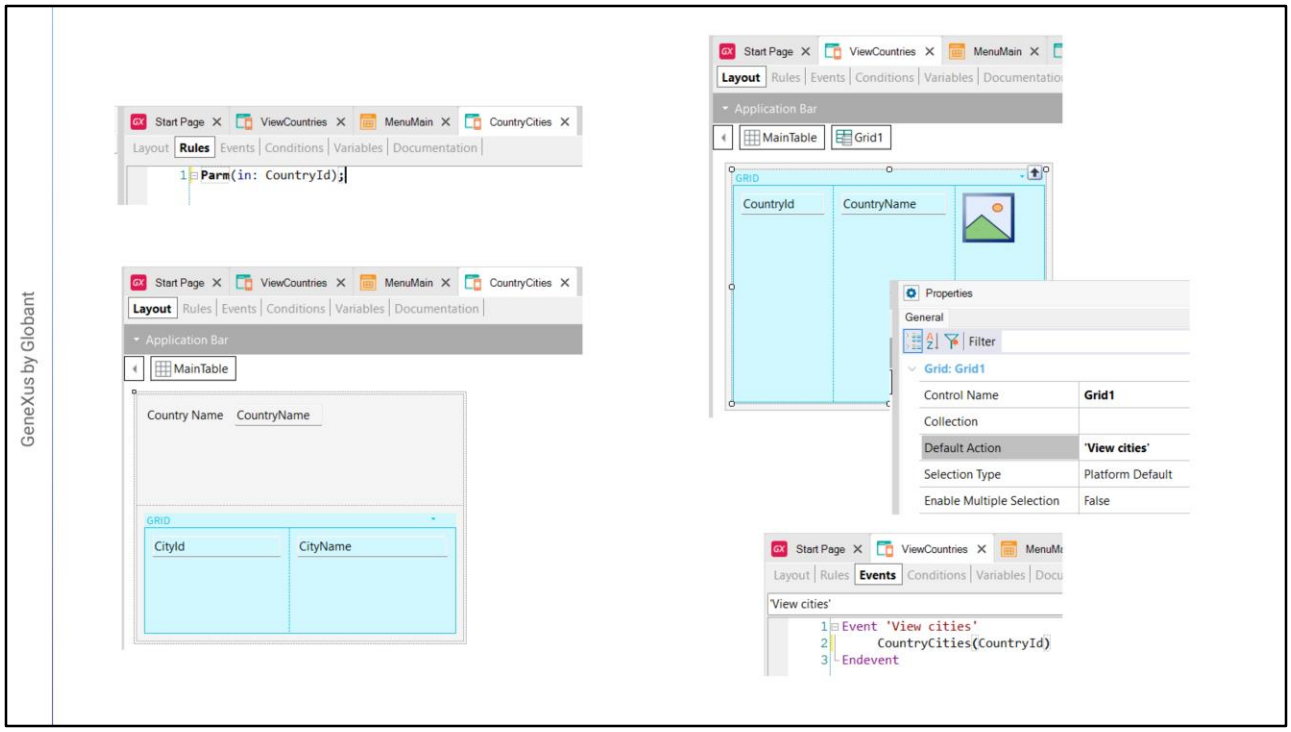
In the Output window we see that the MainMenu object was executed and the emulator displays an icon named View Countries.



We tap on the icon and the data of the countries stored in the database is displayed.

Our first native mobile application is working! We will deal with the design later.

To view the information of a country's cities, we must create a panel that is run when we tap on a grid row.

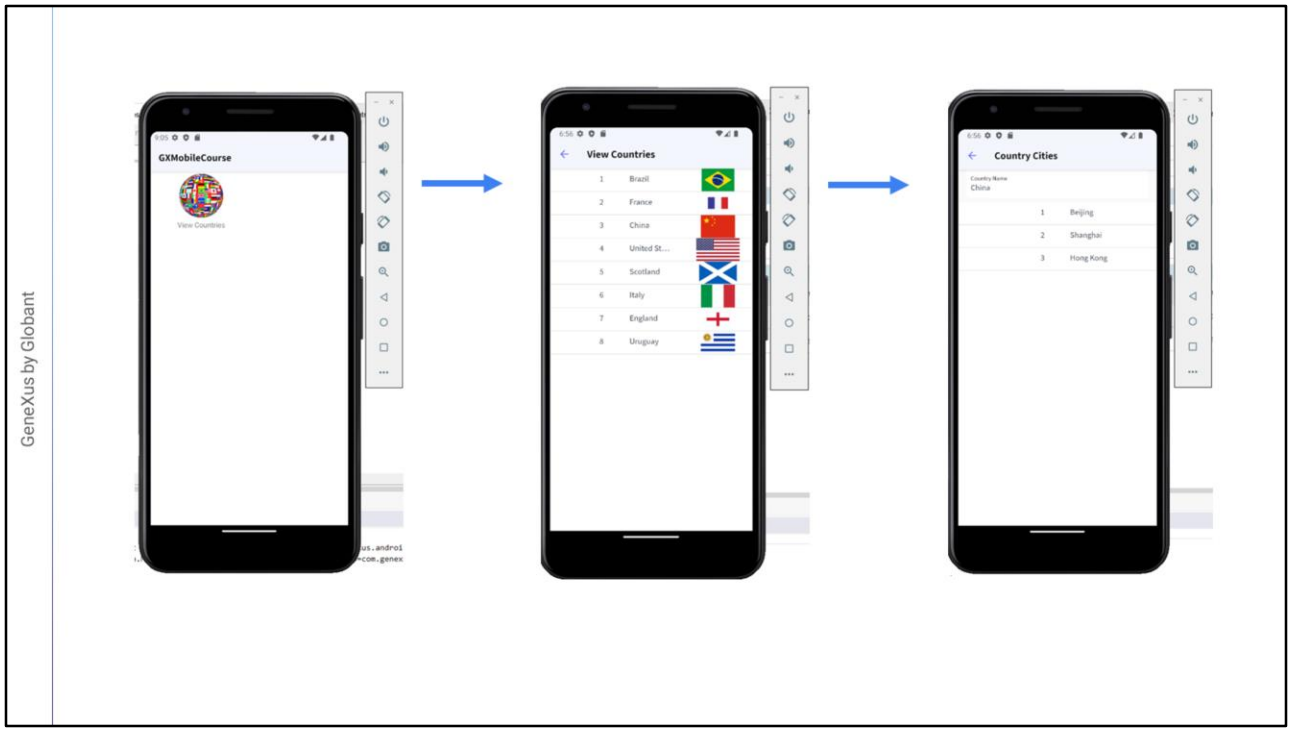


Therefore, we create the CountryCities panel and add a Parm rule to receive the CountryId attribute as a parameter. This ensures that all the information in the panel will be automatically filtered by this value, and even if we access the table where all the cities are stored, we will see only the cities of the country we are interested in.

Then we drag to the layout the attribute CountryName and a grid, which we select to contain the attributes CityId and CityName. We save and return to the ViewCountries panel.

We select the grid and in its Default Action property we choose the value <new> and write View cities. Now we open the events tab, and in the View Cities event we invoke the cities panel by typing on CountryCities and sending the country identifier in brackets. When we tap on a country, this will cause the panel to open with its city data

Let's test this by pressing F5.



The emulator screen is refreshed and we see the View Countries icon. We tap to show the list of countries and now we tap on China.

The cities of China are shown: Beijing, Shanghai, and Hong Kong. Again, we could improve how this data is displayed.

To return to the previous screen, an arrow appears at the top left. Although the Android device provides a specific button for that, not all devices provide a particular button.

Later on, we will go into this and other functionality considerations defined in the design guides of each platform.


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