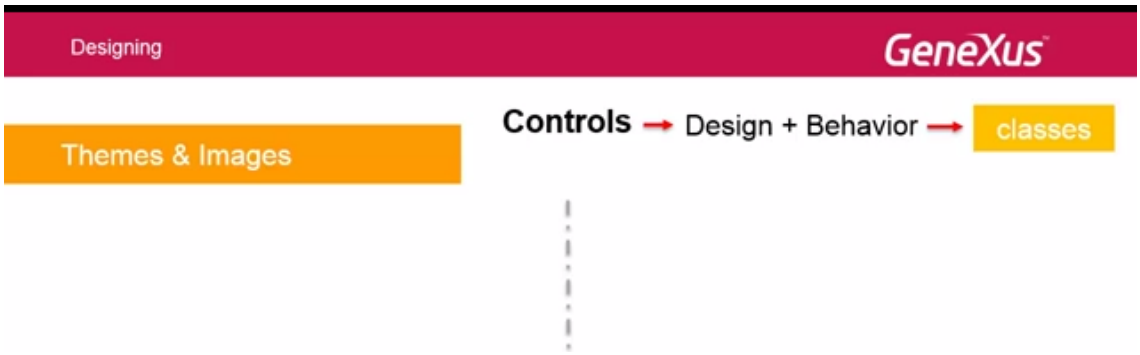
Designing: Themes and Images

Let's start from the first item. In Smart Device applications, the theme object will become essential. In this context, most of the specifications that will have to be defined for the controls displayed on screen, both in terms of their **design** and certain **behavior**, must be defined through classes.

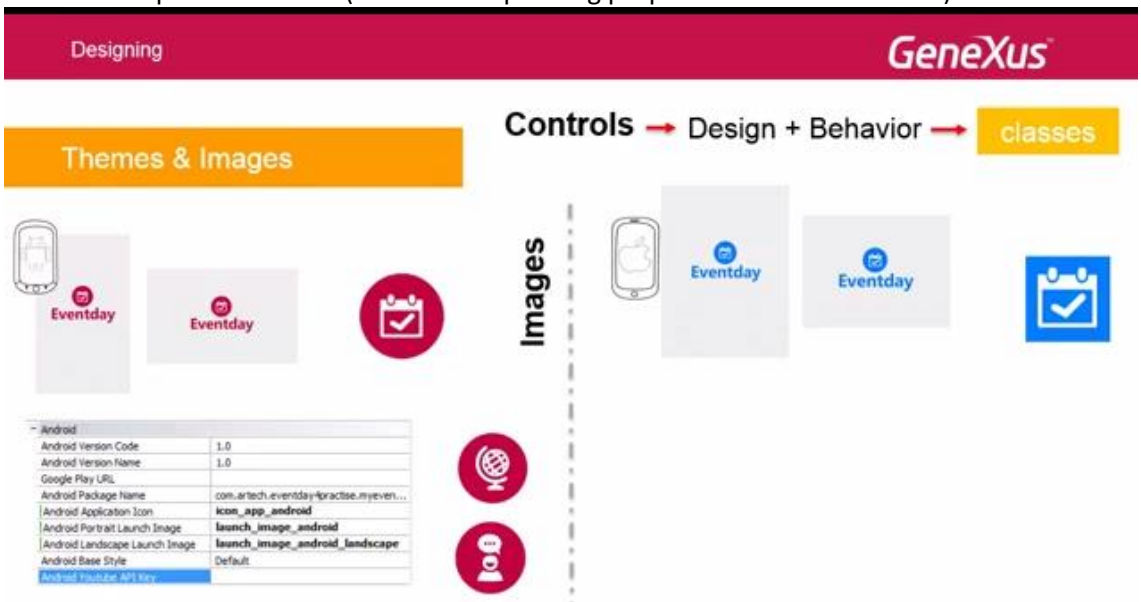


Images are another essential element in this type of applications; screens can be very small so most of the information and the way we interact with it will be based on graphics. Each platform will require a certain type and number of images, of certain sizes, for the applications to be accepted in the various stores.

Therefore, we must start by paying careful attention to these issues, working closely with graphic designers. They will define the overall look of the application that we will translate into the themes of our KB.

In addition, we will need the **launch** images (for portrait and landscape orientation) and the **application icon**, which are configured in the main object properties (probably a dashboard, as in our application). We will also need the images of the **dashboard** items (for instance, the countries and speakers icon), and the **tabs** we use, all for the various platforms.

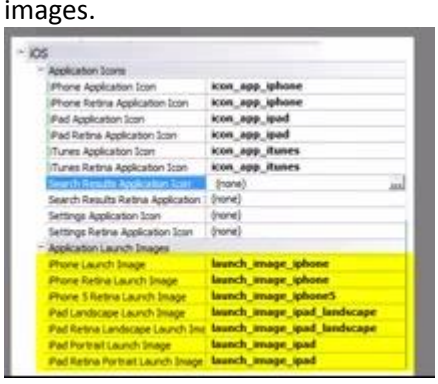
For example, on the right we see some of the launch images and one of the application icons that will be specified for iOS (in the corresponding properties of the dashboard).



Note that while in Android we need to specify a single image for the application icon,

in iOS we will have to enter 6 images:

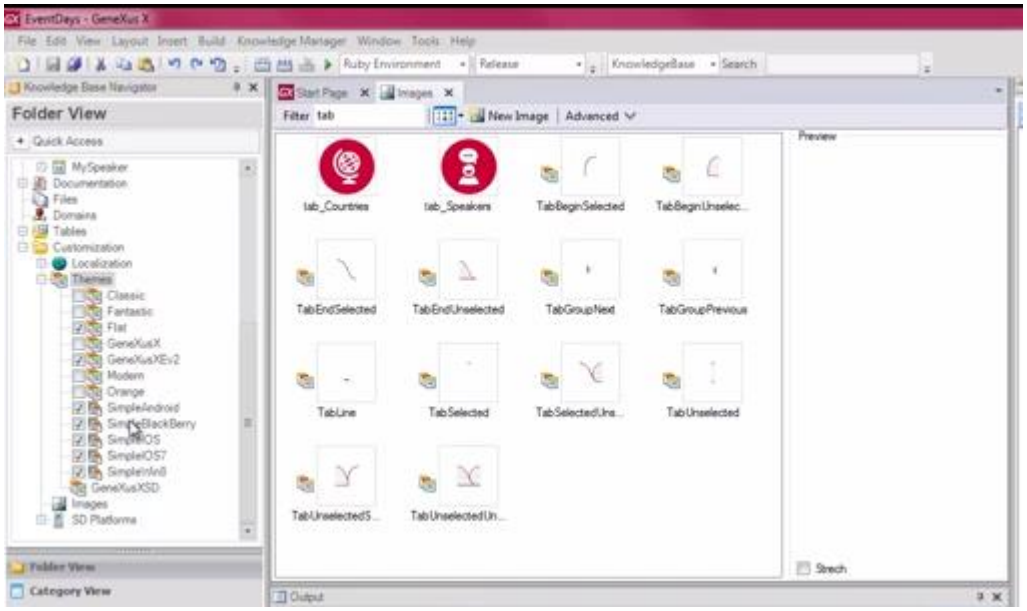
for the iPhone and iPhone Retina (Retina is an Apple trademark for screens with such high resolution that the human eye can't see individual pixels from a normal distance), for the iPad and iPad Retina, and for iTunes and iTunes Retina. The same will happen with the launch images.



In addition, we also have iOS versions of images corresponding to dashboard items.

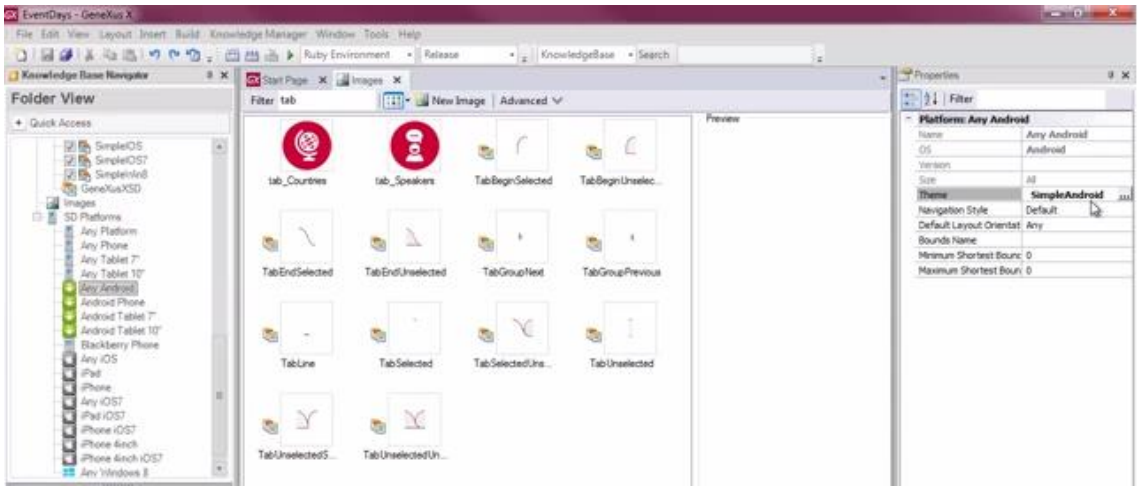
By default, when we add the smart device generated to the KB, we import the predefined **themes** we see.

In GeneXus,…we have them under the themes node.



How can each platform know which Theme to use?

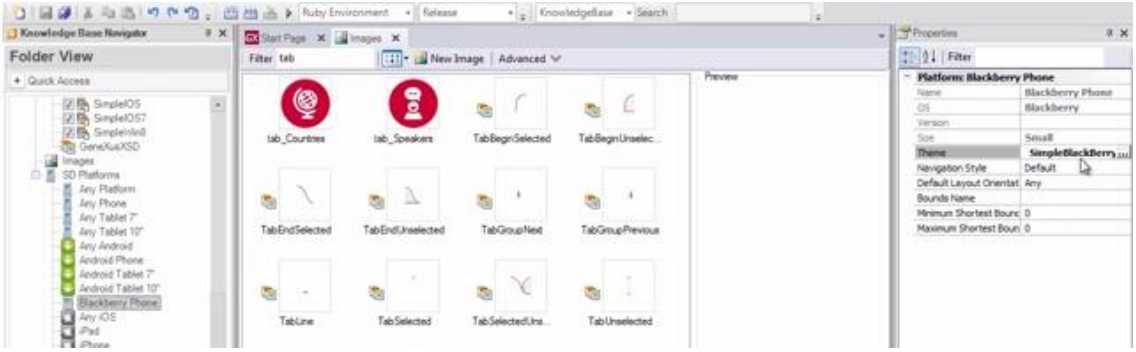
That is defined in the SD Platforms node, for each platform. For example, we can see that the SimpleAndroid theme is defined for any Android.



If the Android is a phone… it inherits the same theme as the more general.

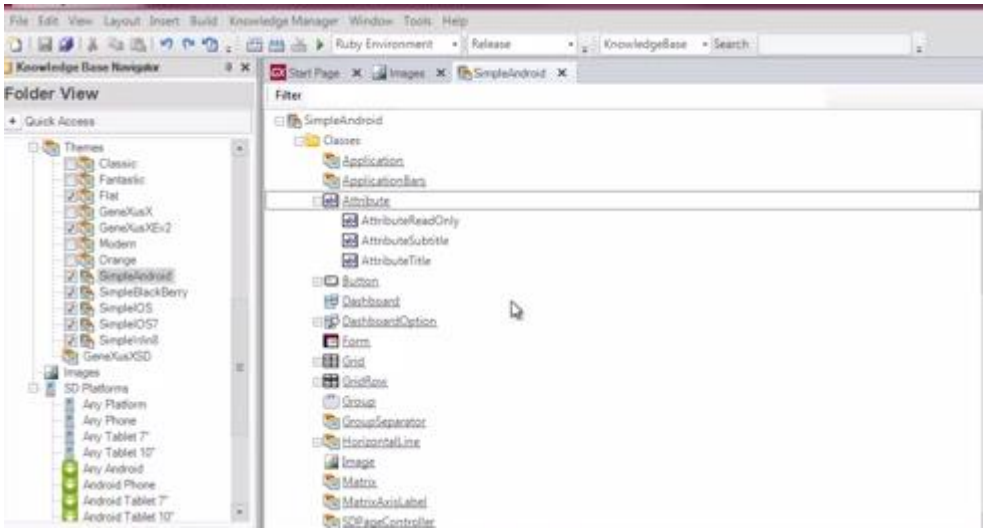
And the same happens if it is a 7inch tablet, … or a 10inch tablet.

In the case of Blackberryphone, the predetermined theme is SimpleBalckberry.



And the same happens with each of the platforms available.

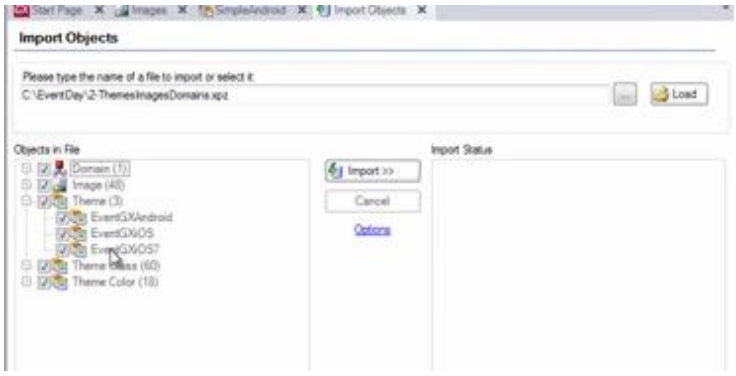
And we have the possibility of customizing these themes. Let's take, for example, SimpleAndroid… aware, from the design the graphic designer sent us, that we will have to color the Font in some attributes. We would add subclasses of the Attribute class, one for each color.



Or, for instance, to color the background of a table's rows we can add subclasses of the Table class, also, one for each color. So, every table in our Android application associated with, for example, the class of the blue table, will have a blue background defined in the class.

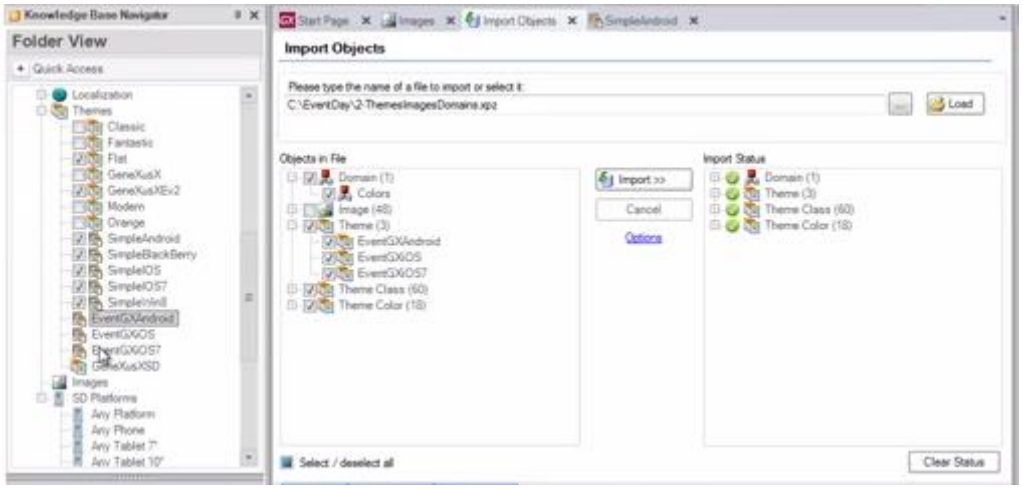
As we said, we may customize the SimpleAndroid predefined theme, and we may also create a new theme for Android, and to the same for the other platforms.

For example, we will import an xpz with three themes: one for Android, one for iOS, and another one for iOS7…

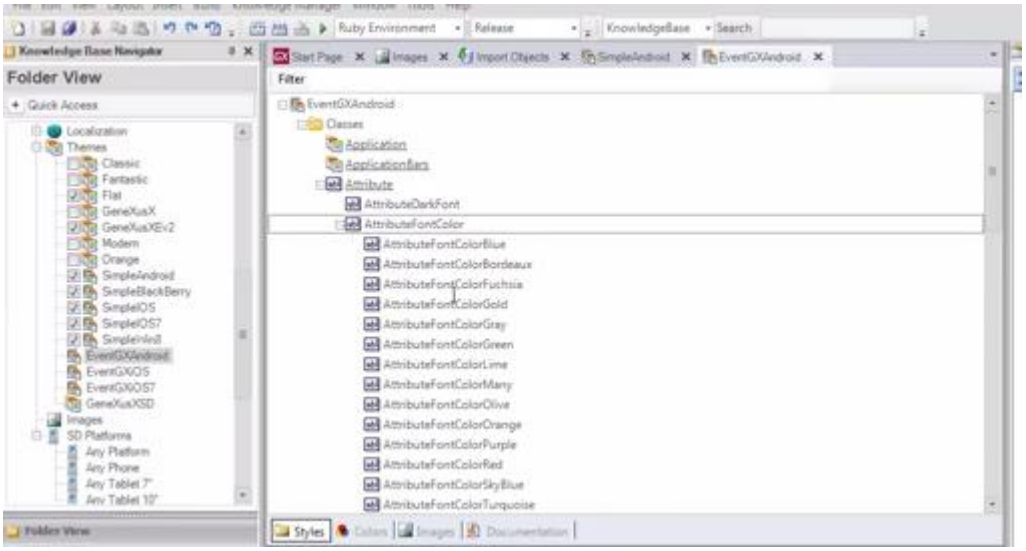


in addition to the classes and colors of the themes, a domain –Colors-, and a group of images. We will now uncheck them to import them later.

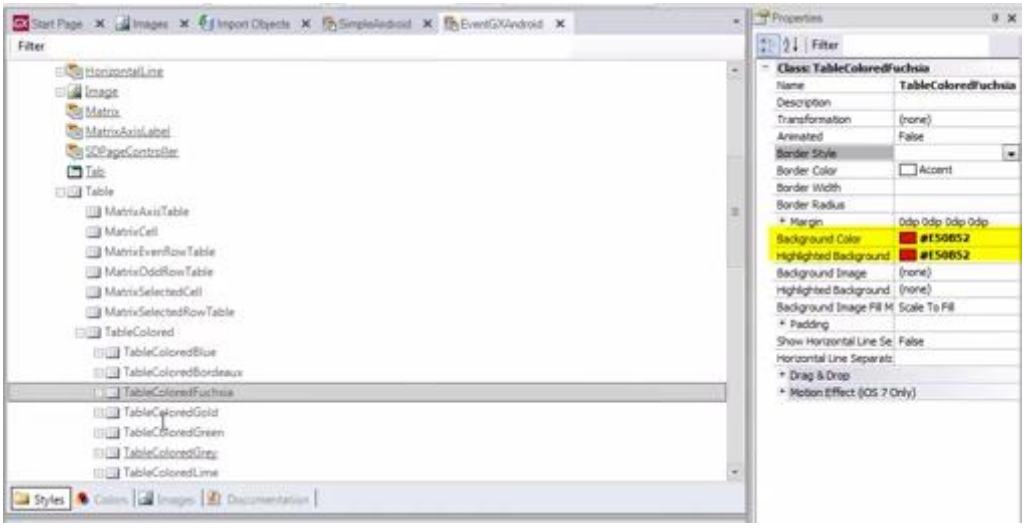
Now we import,… so we now have these three themes in our KB.



We open this one for Android, and we can see that here, the classes have been added to specify different font colors for the attributes…

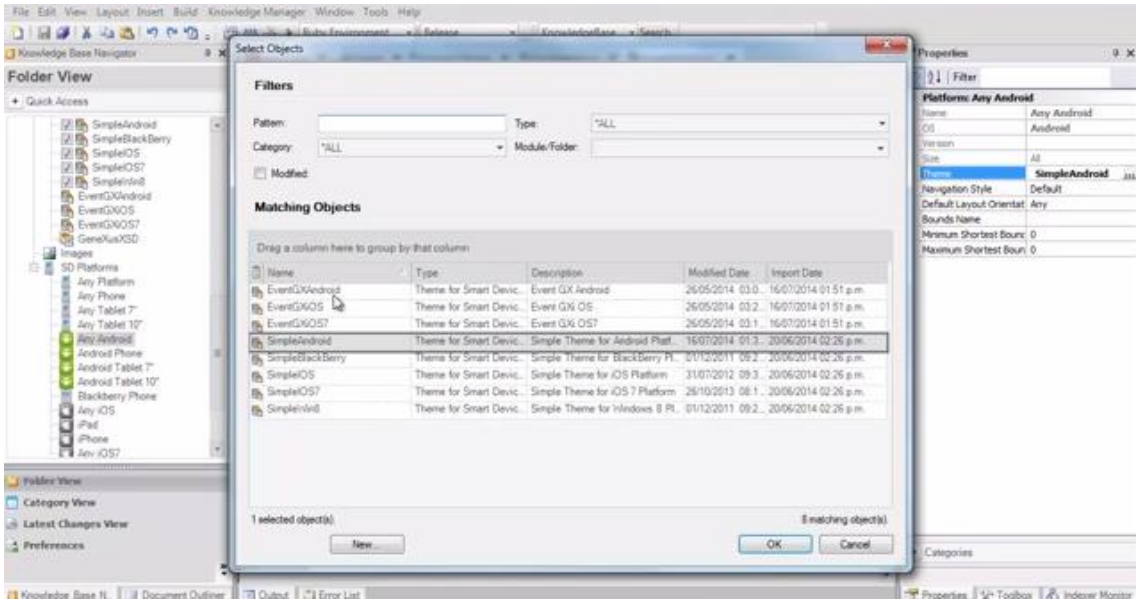


and in the Table class… we see that now we have these other ones to color the tables' backgrounds.

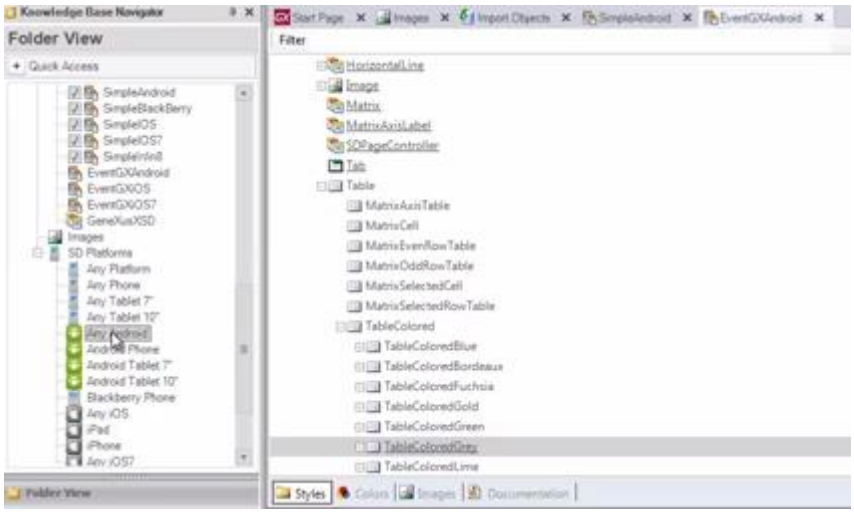


All we need now is to make this theme operative for the Android platform instead of using SimpleAndroid.

And so, … we change themes, with the new one.

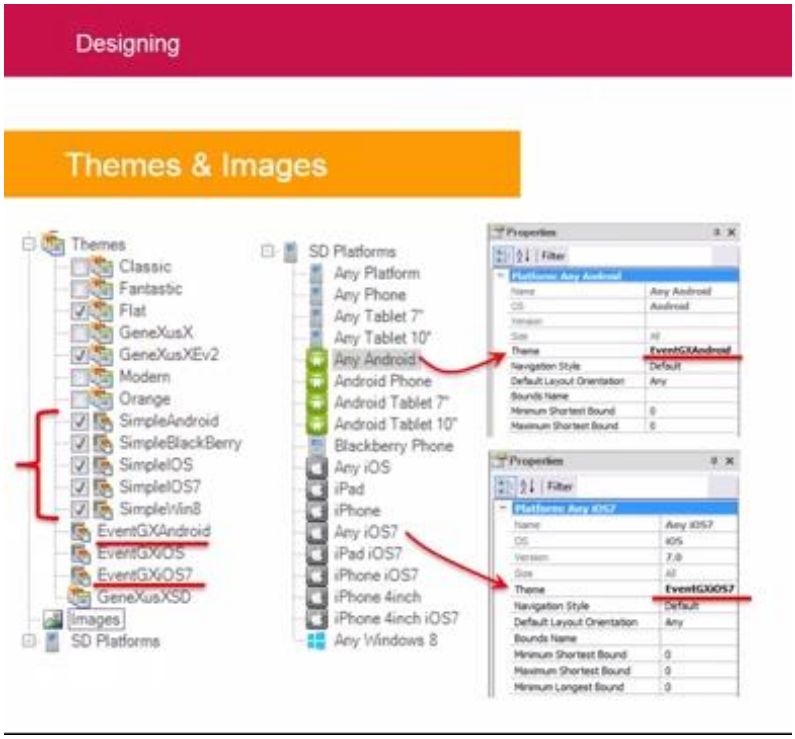


We can see that, if the platform is a phone or a 7inch tablet or a 10inch tablet, it automatically inherits the same theme. We could change it and set a different theme for 10inch tablets, for the new theme to be used only when the application is executed on that type of tablets. Otherwise, the more general theme will be used.

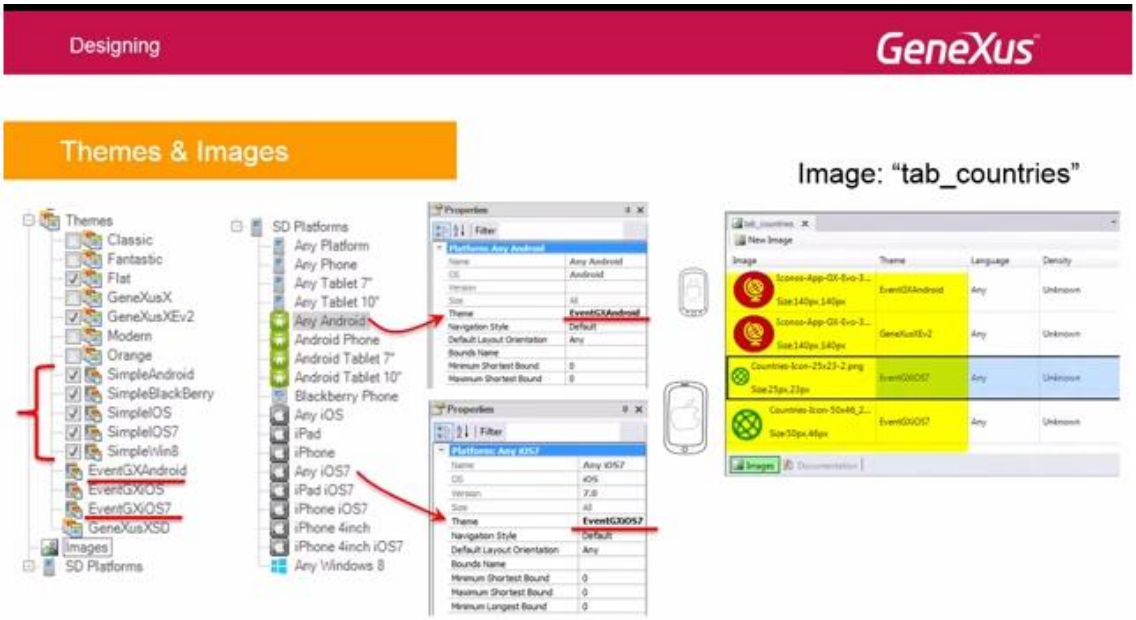


And we would do the same for AnyIos… and for AnyiOS7…

Here is what we have just done.

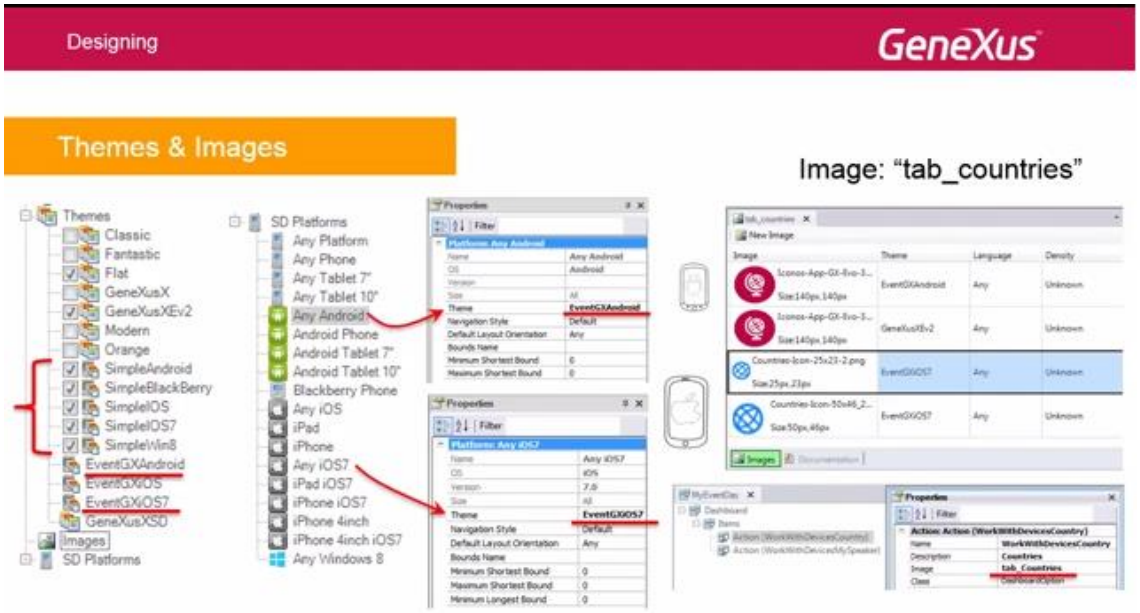


Another interesting fact is that the **images** we import in our KB could vary from one platform to another. So, for the image of countries in the dashboard, we need not create an image different from that icon for each platform: the tab_countries image we show has different jpg's for each theme



(note that for iOS7 we will need two, one for Retina, and we have also defined, the corresponding jpg for the web theme; and we have not defined images for other themes, like EventGXiOS, so, in the case of using the iOS platform –not iOS7-, the image will be empty. To avoid this we can add a jpg valid for Any, as we will see now).

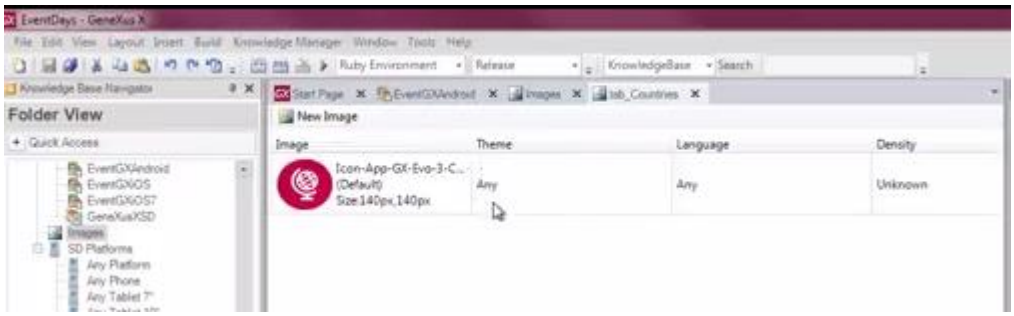
If we vary the image content by platform, then in the dashboard we will only associate the image with the item,



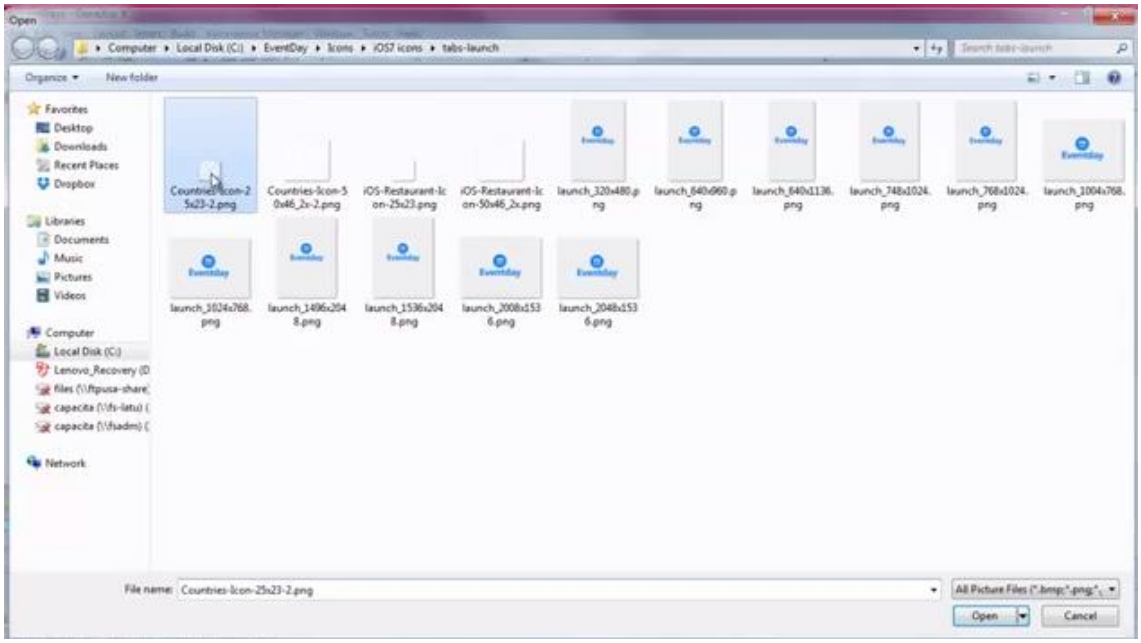
and depending on the platform used every time, the jpg will be one or the other.

Let's see it in GeneXus.

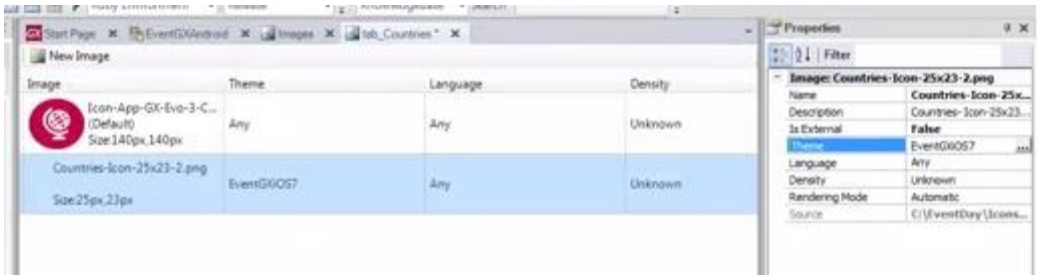
When we defined the tab-countries image in the initial video of this course, we did not vary it by theme:



it was the same for all themes. If we now want to have a different icon for the theme corresponding to iOS7, such as this one here, which has a white border,



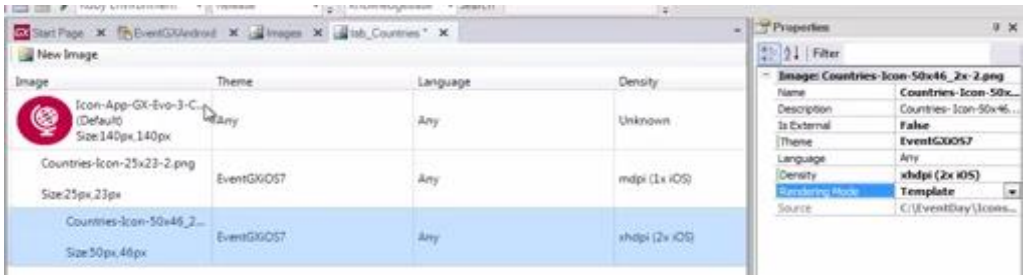
… we will have to specify the theme to which this icon will apply…,



and also the density… to define whether it is Retina or not… And in rendering mode we will use template for it to assume the color corresponding to the device, which, as we saw in the initial demo, is blue,… and that's why the designer sent us the image border white.

Then we would add the icon for the Retina mode, also specifying the theme for which it will be valid, and the density corresponding to Retina…

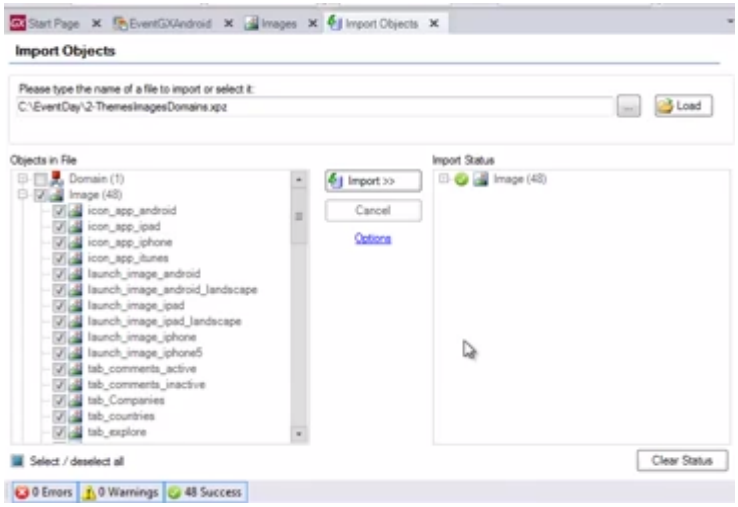
And so on.



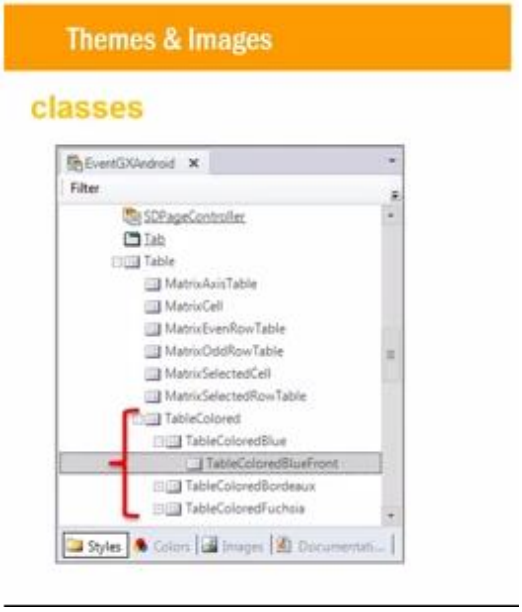
With the image so defined, then, if the application is executed on a platform with iOS7 theme -this one here-, it will use one of these images, depending on whether it is Retina or not. Otherwise, it will use this icon.

Let's now import the images we did not import in the last step.

Here, for each dashboard icon we will use we already have these images varying by theme.

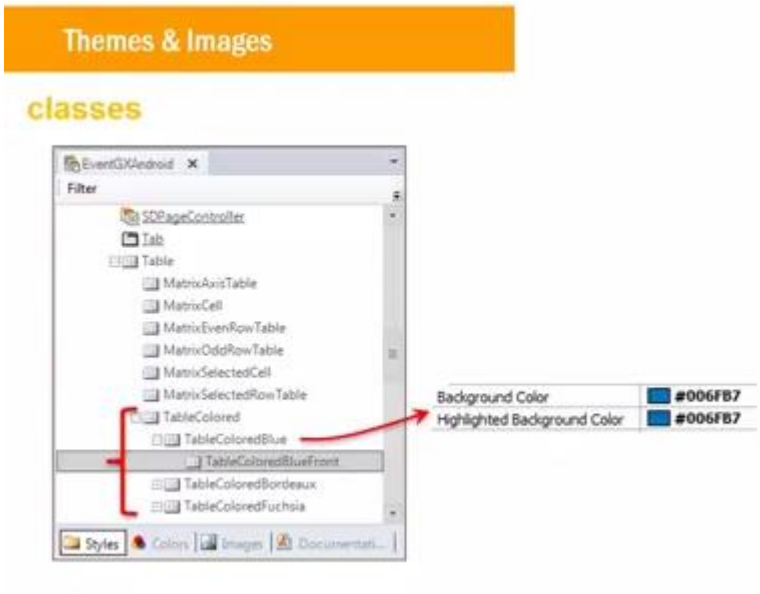


So, we can see the definition –we overwrote the one we had- and we will have defined all the images corresponding to the tabs.

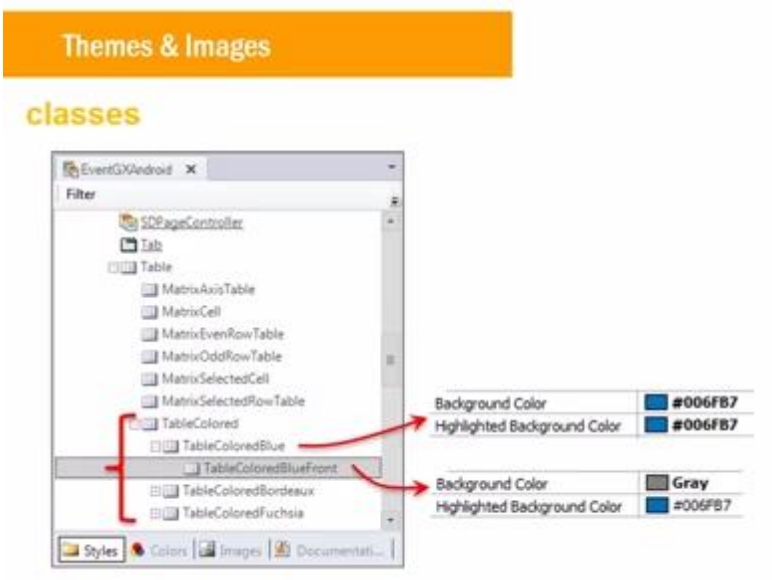


To sum up then, we have added this new theme for Android, with the classes indicated.

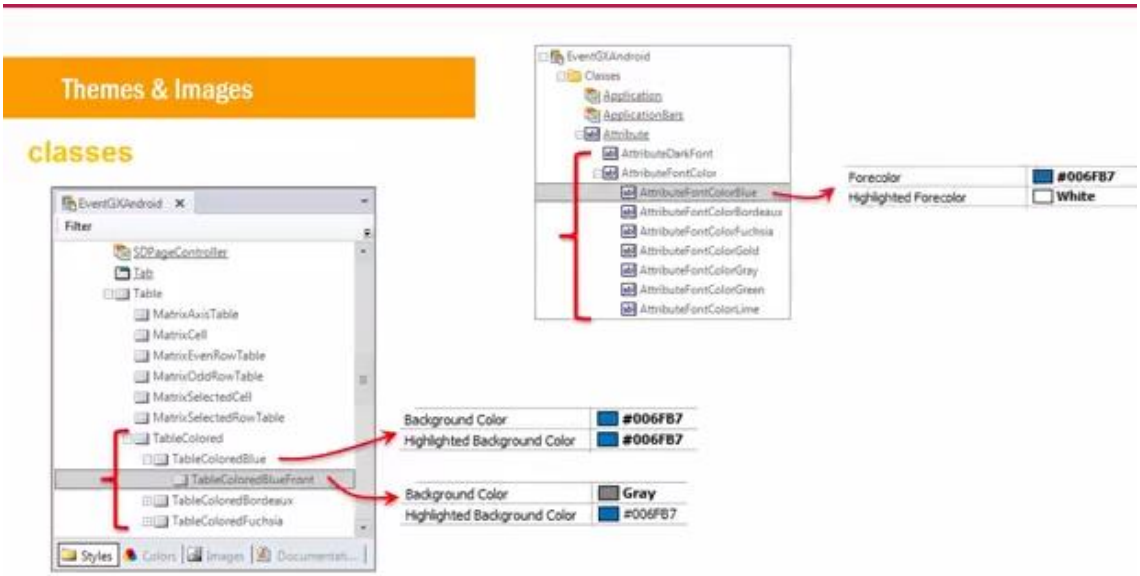
For example, classes for coloring the background of a table control: always blue (TableColoredBlue class),



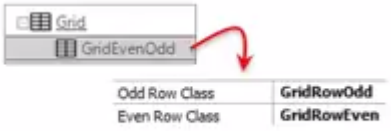
or grey when not selected and blue when selected.



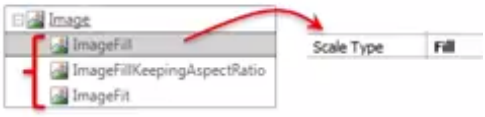
Or, classes for the font color of an attribute (such as blue when not selected, and white when selected).



Or a subclass of the Grid class, for the odd-number rows to take their values from the GridRowOdd class, and the even-number rows from the GridRowEven class.



Or subclasses of Image, in order to define how an image is scaled within the cell where it is located.



The use of classes is very important because most properties of controls are centralized here. We will not find them at the control level in the layout, but rather at the level of the class associated with the control. And we will see that… when we study controls in l