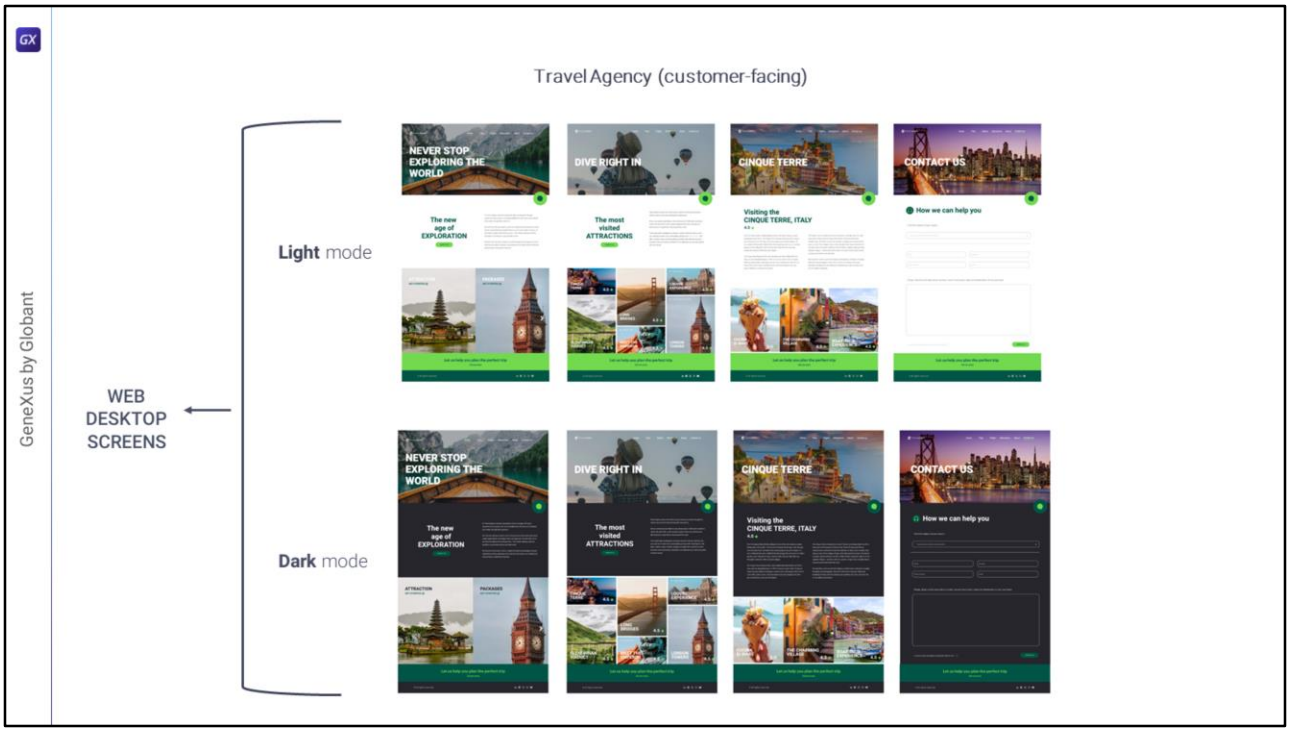
GeneXus by Globant

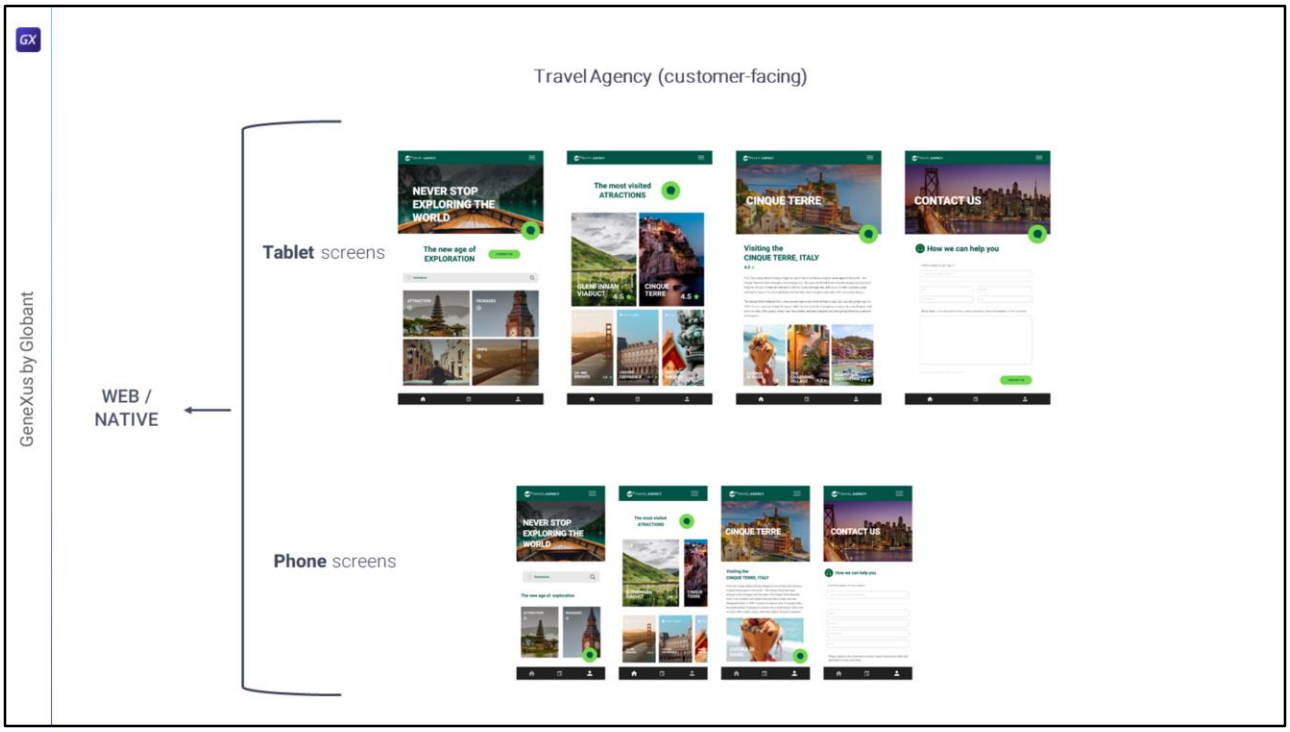
Multiexperience Approach

Cecilia Fernández

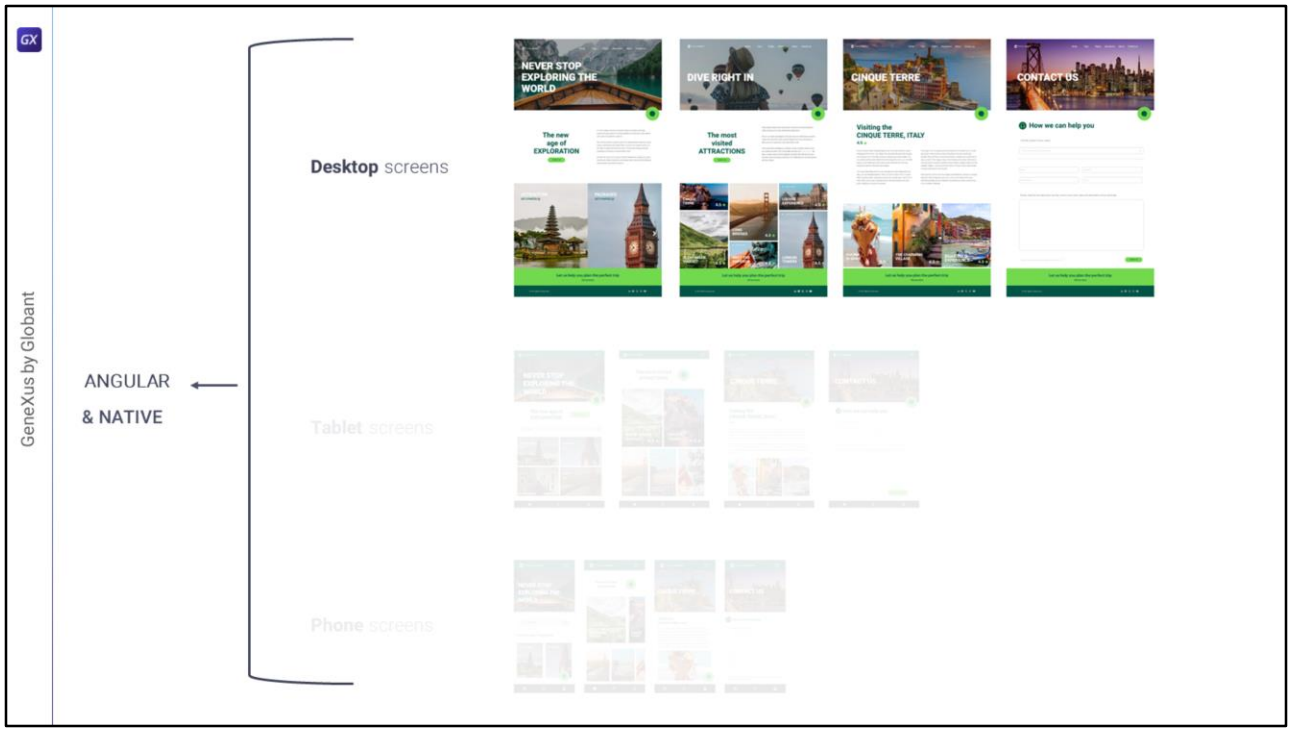


Throughout the four previous modules we mainly focused on acquiring the necessary skills to develop the Customer-facing application for Angular.

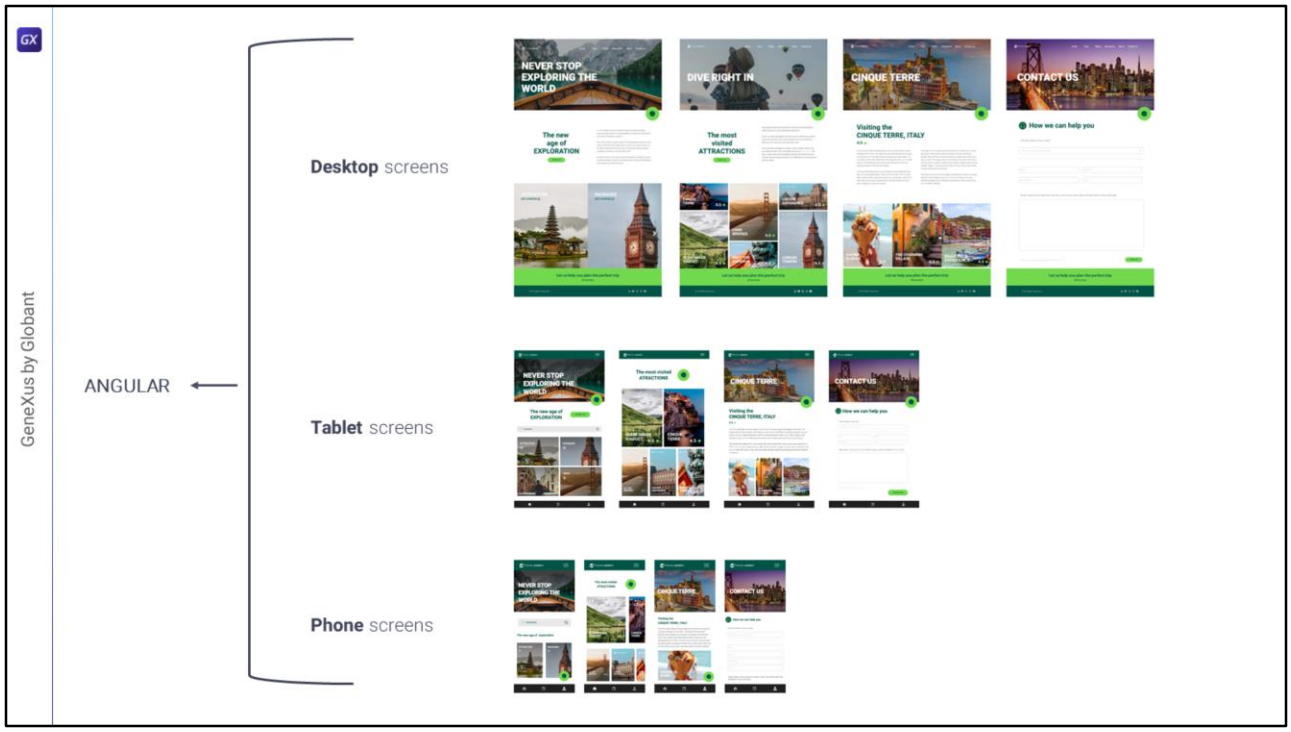
We focused almost exclusively on the development of Desktop size screens...



...But we saw that our designer had also designed variants for Tablet and Phone sizes.

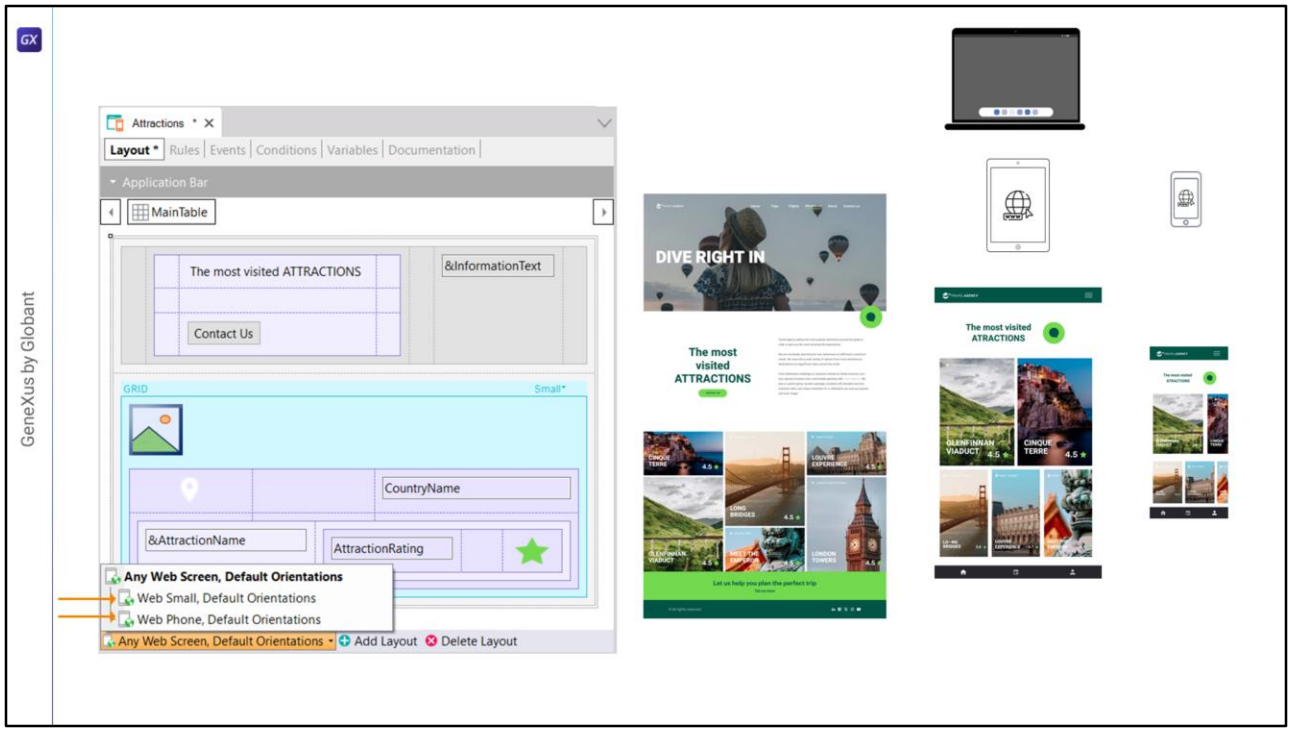


And here this becomes more complex, because while for Desktop size we will only have the Angular application, for Tablet and Phone sizes we will find the now well-known multiexperience: that is, we will need to implement both the Angular application and the native application for Android and Apple.



Let's think for a moment only about the Angular application. The same application should work by adapting to the different breakpoints, according to the adaptive design that governs this platform.

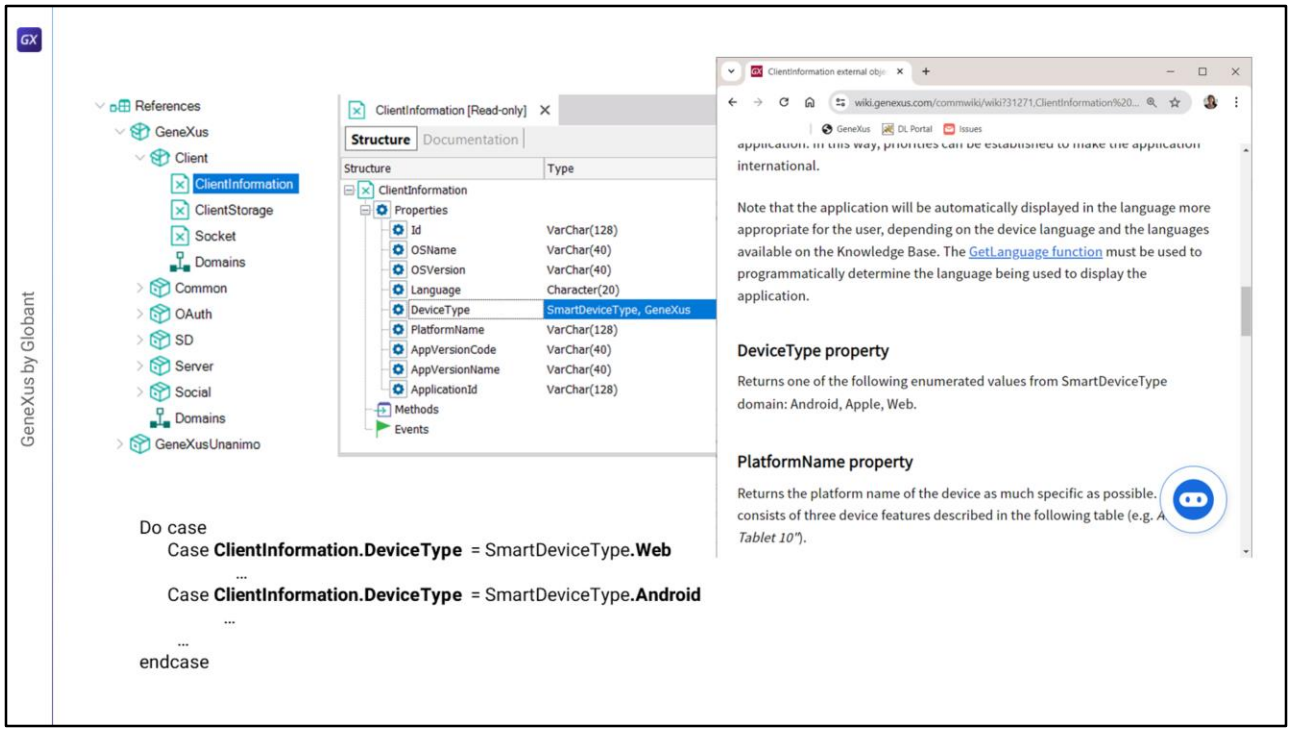
This is done automatically if we define these two types of layouts for each object, in addition to the one we already had.
That is, in total we will have, for this case, 3 layouts per object.



So, for this panel we will have the default layout (which we could also specialize to indicate that it is Any Web Screen, which would leave out everything that is not Web). I was saying say then, that we will have this default layout that will correspond to this design... and a layout for Tablet and another for Phone.

The Web Small layout will be the one chosen whether you are running the application on a mobile device that has the size corresponding to that platform, or on a laptop of that screen size. On the other hand, the Web Phone layout will always correspond to mobile devices of the size indicated by its platform.

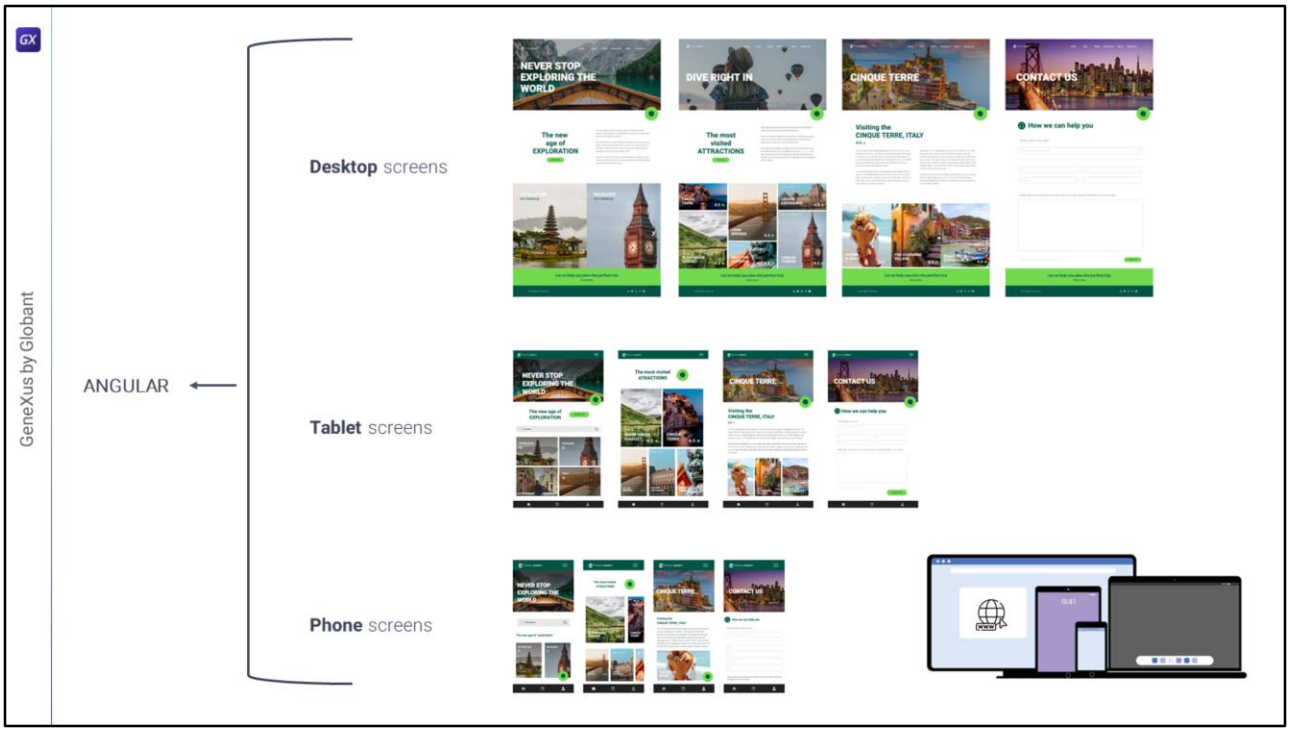
It is also important to mention that the Attractions panel object will be the same, regardless of the number of layouts we define. In a single object we implement all the variations, although we will be able to set these different versions only for the Layout section. The other sections will be the only ones for all variants. That is, there will be no versions there. The events specified here will apply to all layouts...



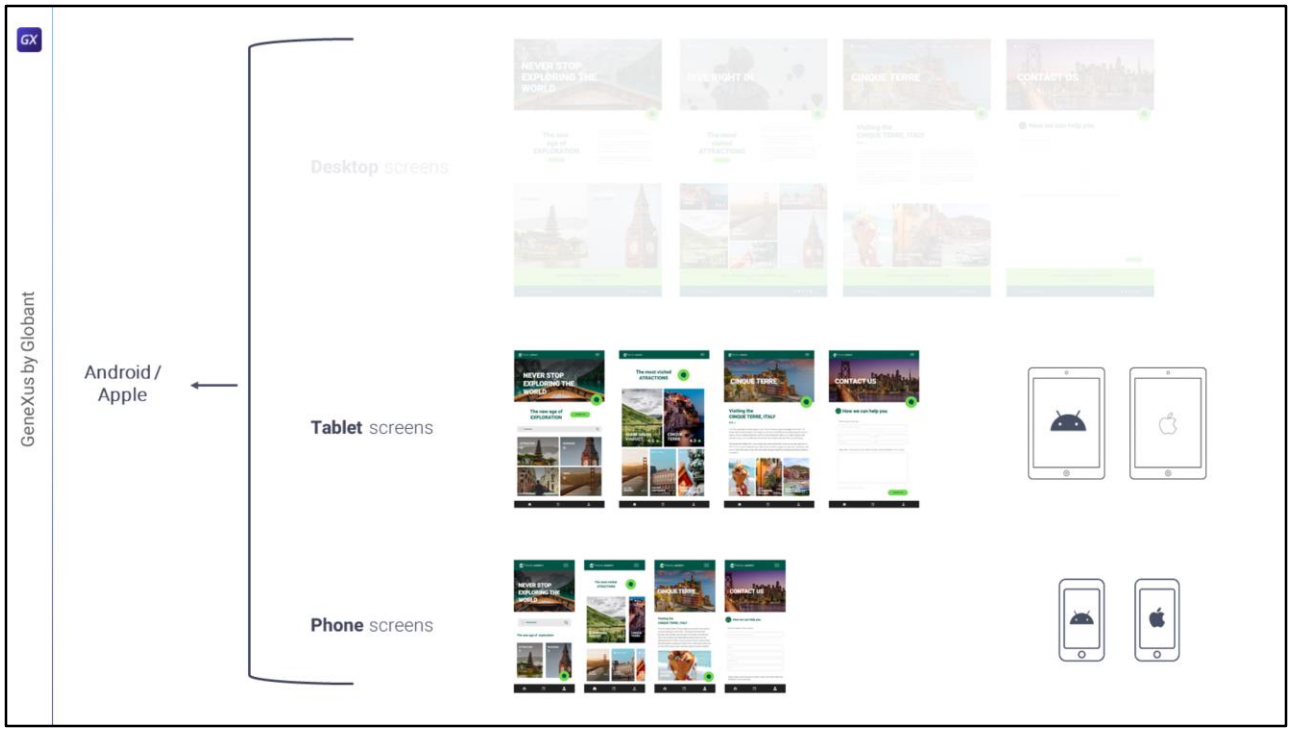
But we can execute one code or another depending on the platform. To do so, we have this external object offered in all KBs inside the GeneXus module, Client submodule.
Note all the properties offered by the object. And, in particular, note the DeviceType.

By asking for its value we can know if the type of device running is Web, Android or Apple.
This way we can program specific behavior anywhere code is supported, such as events.

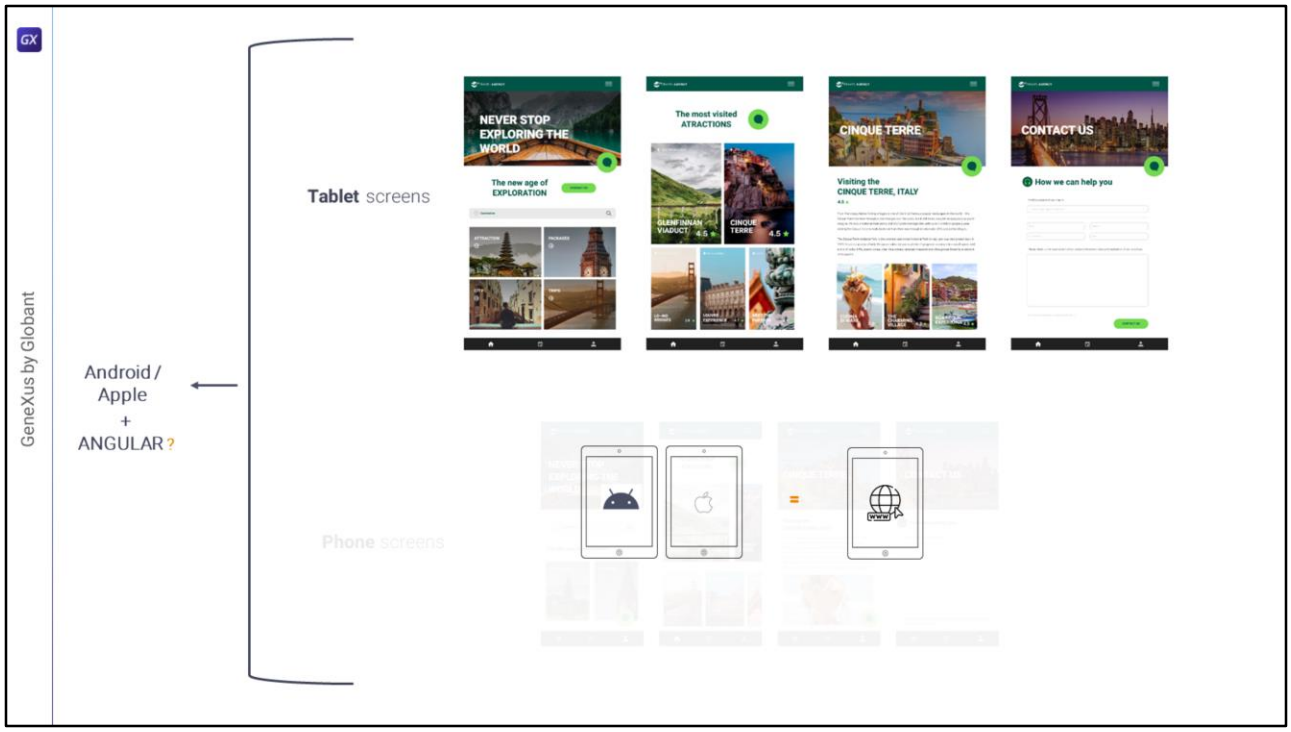
Not all these properties are valid for all platforms, but in this way we can get to know some characteristics of the environment in which our application is running, and take the actions we want.



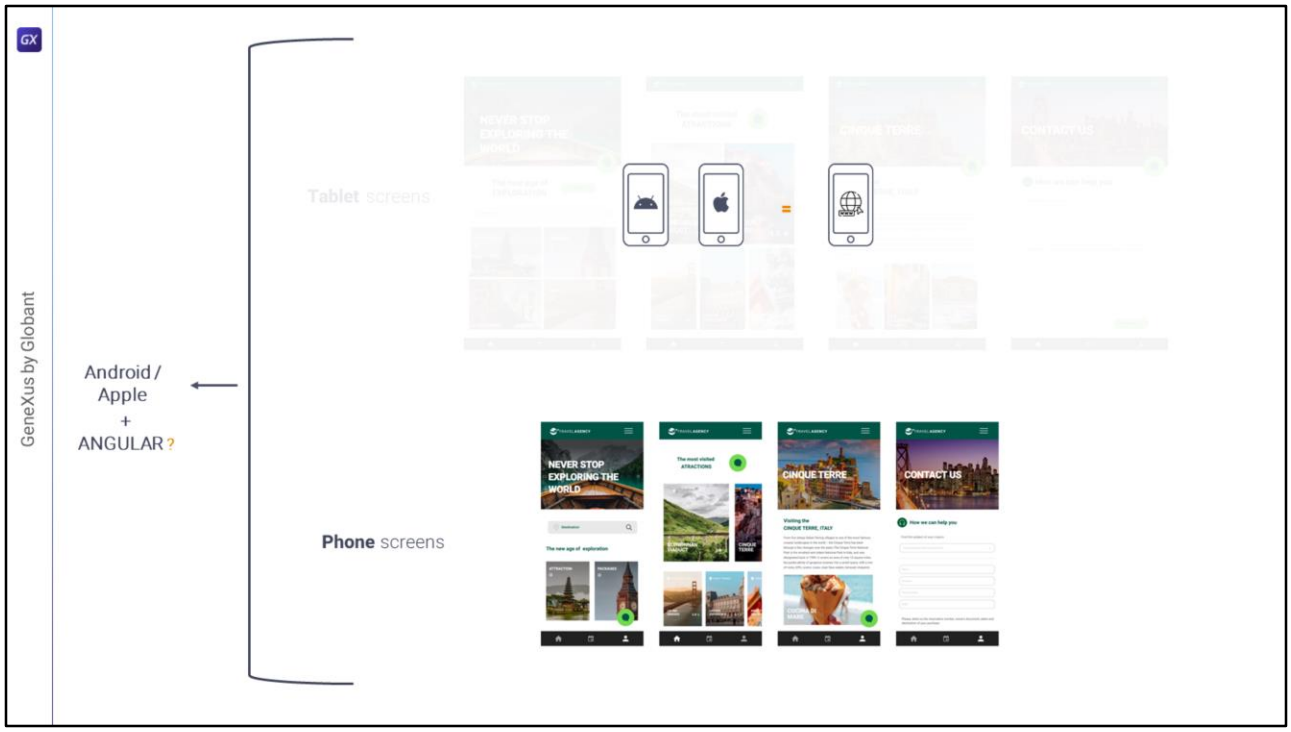
So we will have the same 4 panels for the three variants. And the main object will be the Home panel. This will be the one compiled according to its invocation tree. And its URL will be the entry point of the application from the browser of any device, both desktop and mobile.



If now we think exclusively about native applications, they will use the same 4 panels, but...

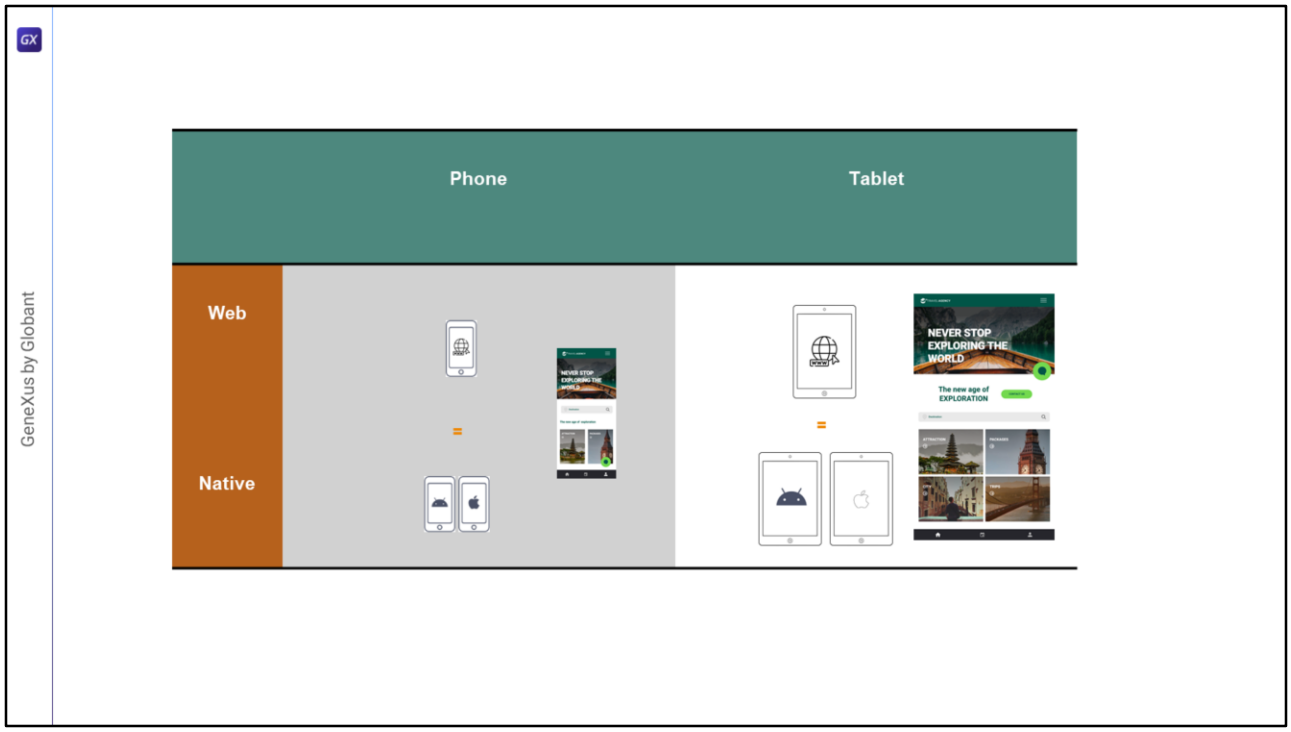


...will those for Tablet use exactly the same layouts as its Web double for Tablet...?



... And will the Phone ones have the same layouts as the Phone Web? The implementation in GeneXus could be exactly the same?

It would be great, for streamlining and reducing duplications and the disadvantages they cause. But, will it be possible?

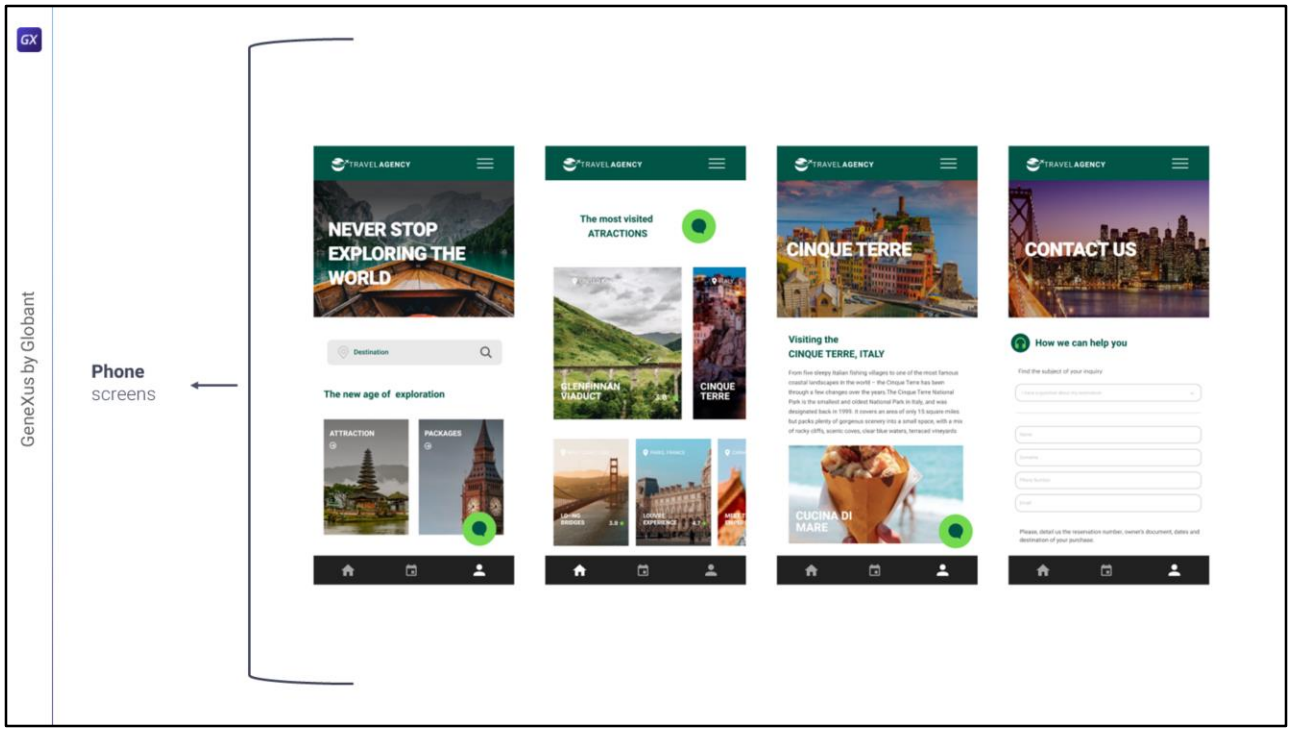


The answer is yes and no.

Logically, in order to reuse as much as possible, we will not be able to solve each scenario in the best way for its platform, but we will have to sacrifice some aspects that would be more natural for that platform to find a solution that will also suit the other platform. This is because the goal is to cover as much as possible while developing as little as possible.

In other words: either we choose the platform we are going to start with, develop on that platform according to its most common practices and then think about the other platform, or we think about a solution that adapts as well as possible to both platforms. This, of course, is also valid for the design.

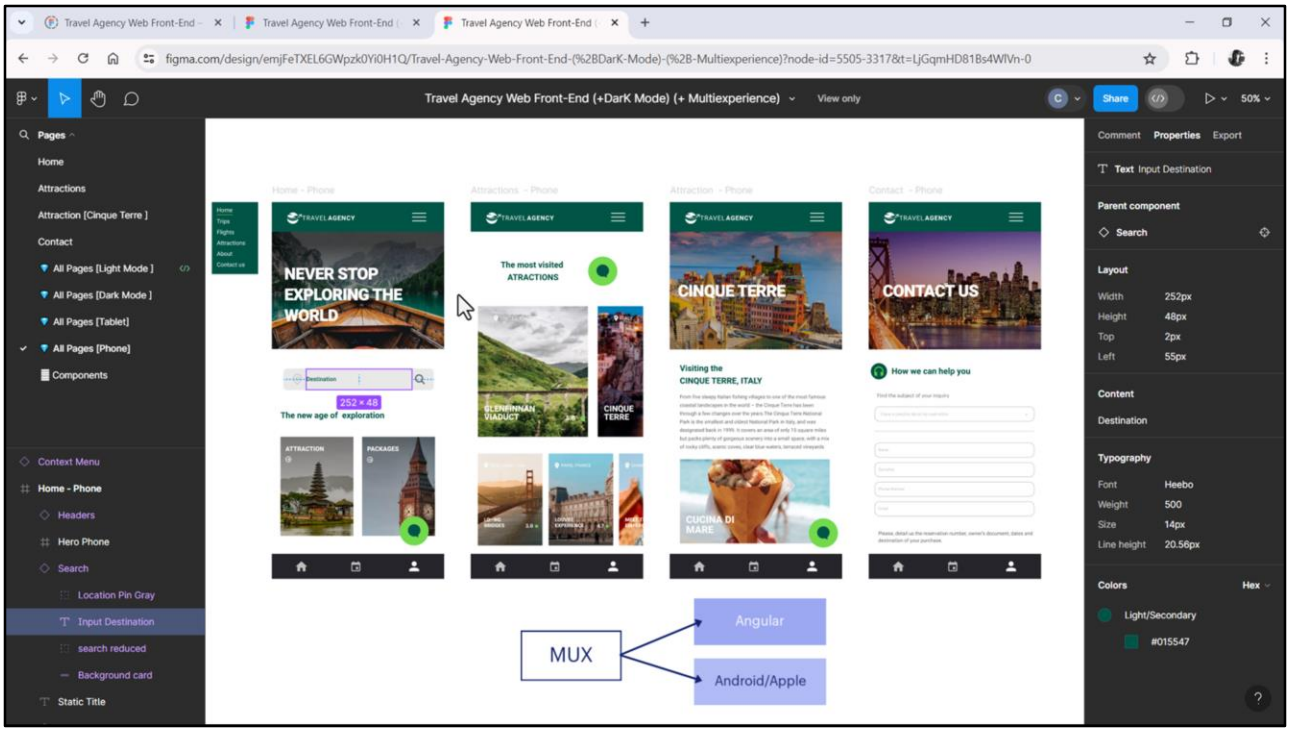
In short, it would be nice to have the same layout for the Phone size for web and for the native application, and the programming as similar as possible. And for the Tablet size as well.



Let's think about this for the Phone case, since the Tablet will be analogous.

The question is whether we should look for the most transversal solution, if we can think of one that is valid for Web and for Android and Apple.

For this we will need some knowledge about the differences between the web and native implementation.



So, for example, implementing all these parts of the pages at first looks the same. It seems that if we implemented these parts for the layout of the Phone size for Angular, the controls and their classes and the DSOs, everything would work exactly the same, for example, for the native Android application.

It would be desirable, but for now, although it is a goal for the GeneXus development team, this integration, this desired transversality, hasn't been fully realized yet. What I mean, and now I'm going to give an example, is that there will also be some differences, mainly regarding the properties of the classes, which, while for the Web world include all the CSS, for the native world there is neither HTML nor CSS. Therefore, some will be the same, others will have different names or values, and others will not exist at all. This, of course, creates a problem for us.

As I was saying, in order to decide what suits best for a multiexperience development, we need to know the particular characteristics of each world.

In the first modules of this course, we started the development getting into the Angular world, without paying any attention to the native world. And that led us to make some decisions that we will now revisit, with some specific examples.

We will continue in the next video.


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