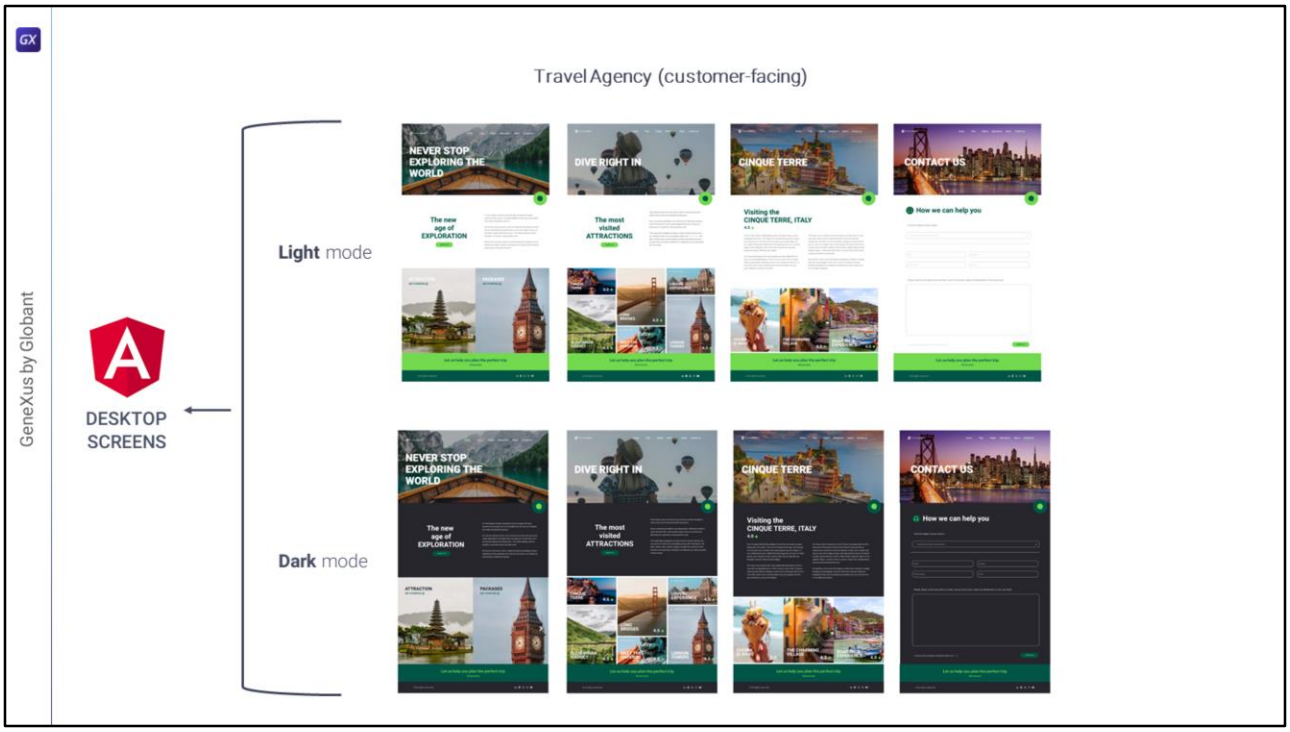
GeneXus by Globant

Multiexperience. Summary and Final Notes

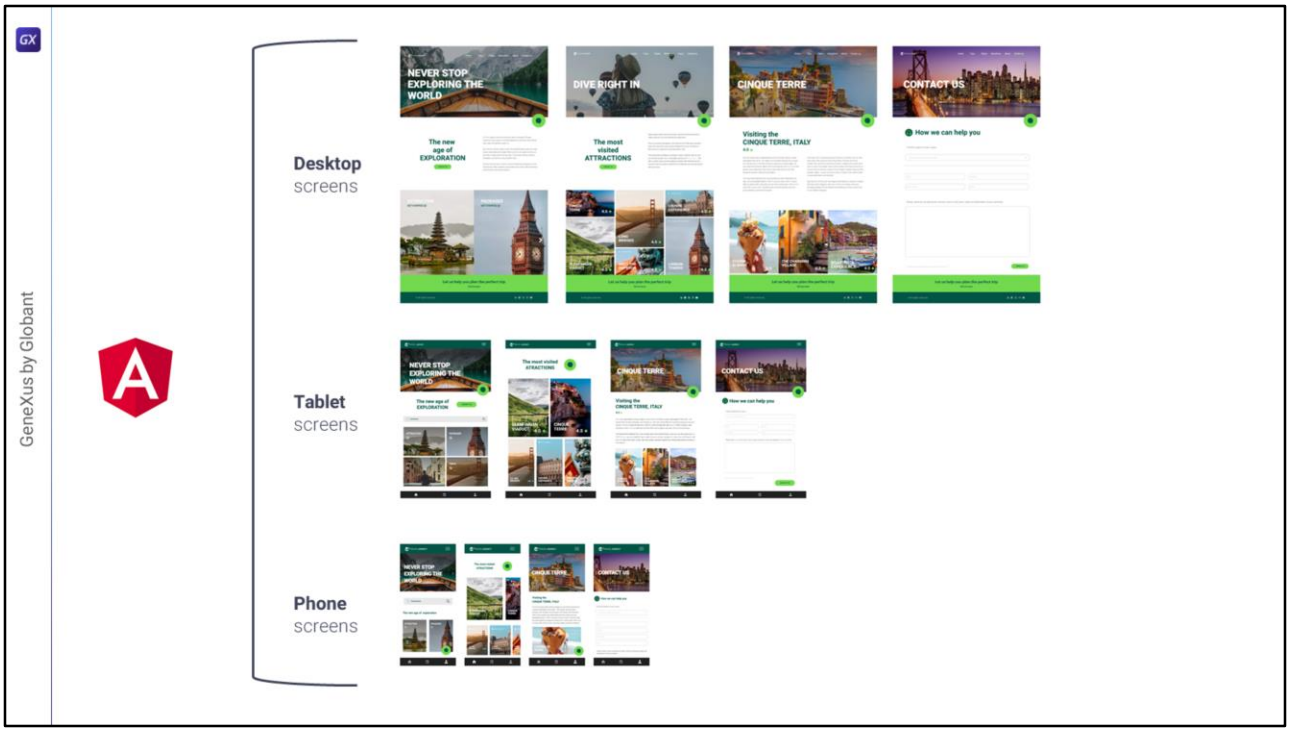
Cecilia Fernández



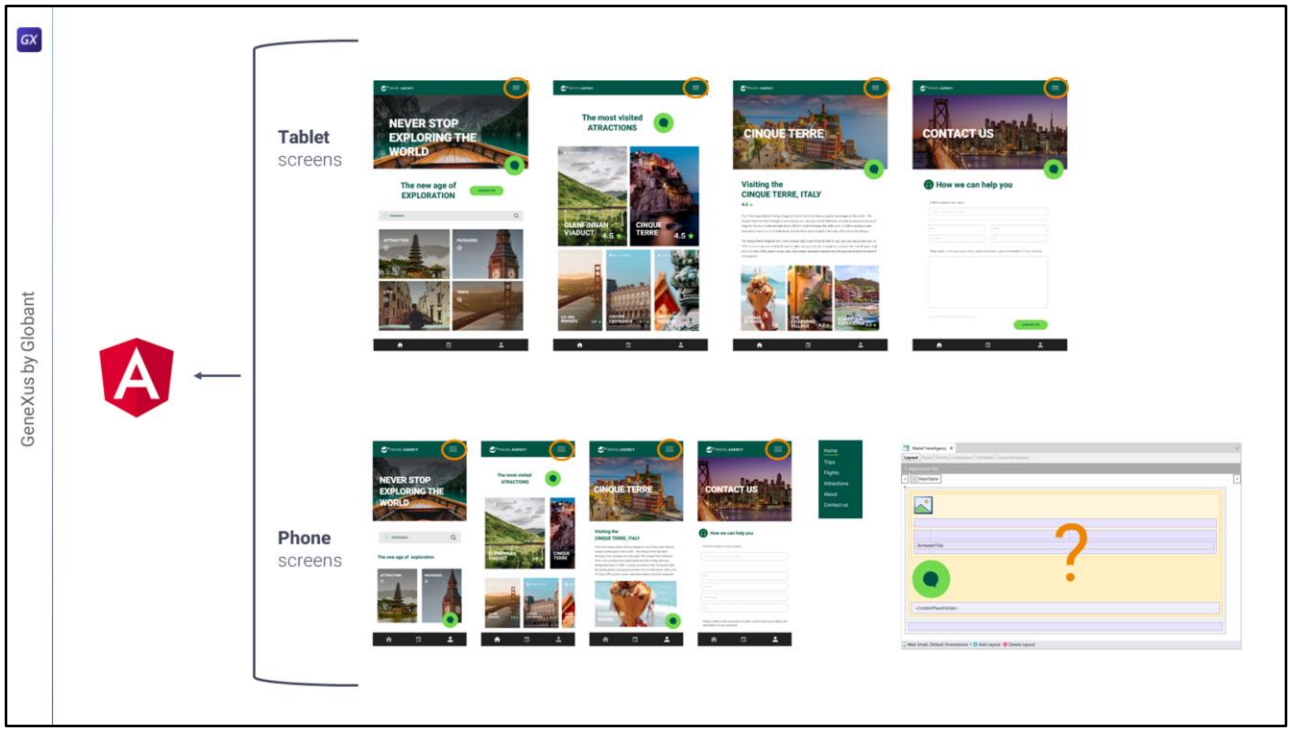
In the previous modules, we focused on how to implement an Angular application from scratch for desktop screen sizes (both regular and big screen) and in light and dark modes.

GeneXus by Globant

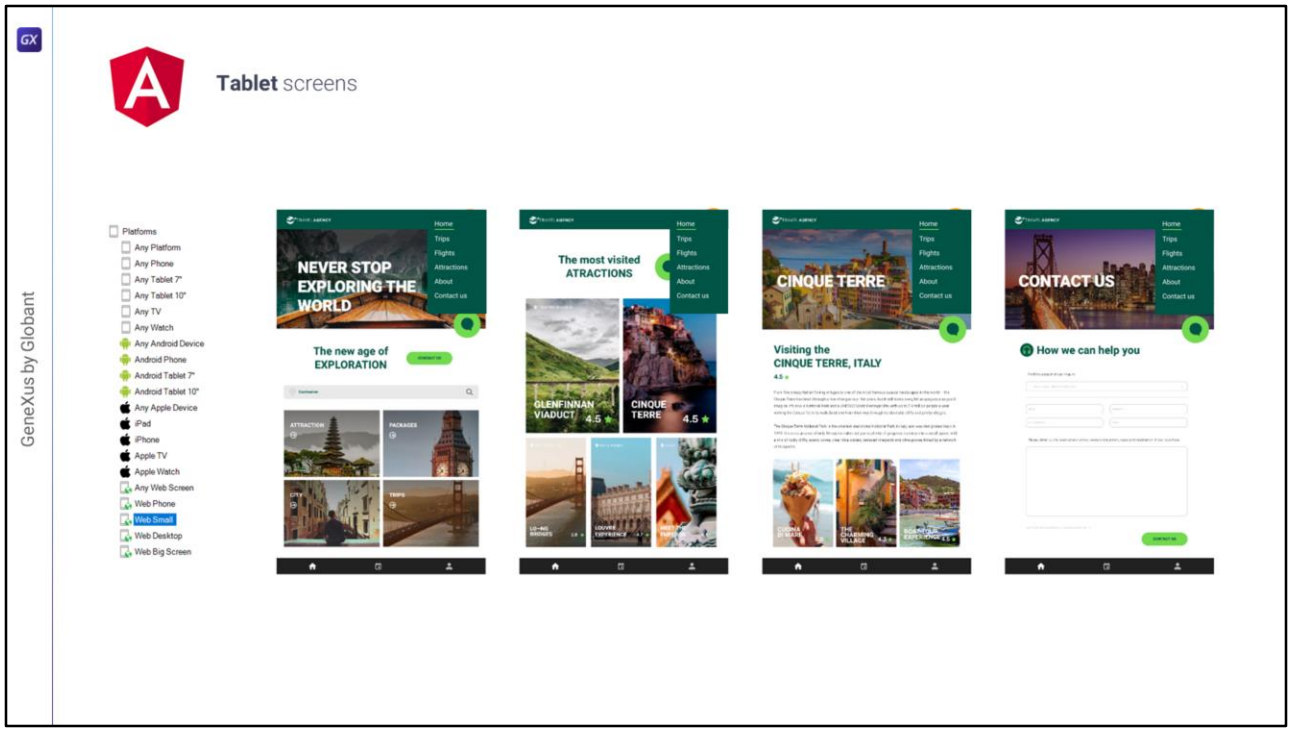
Angular



Considering what we saw there (and without getting into the native world for now) it is easy to think of the solution for the Tablet and Phone sizes, following exactly the same logic that we have been using.

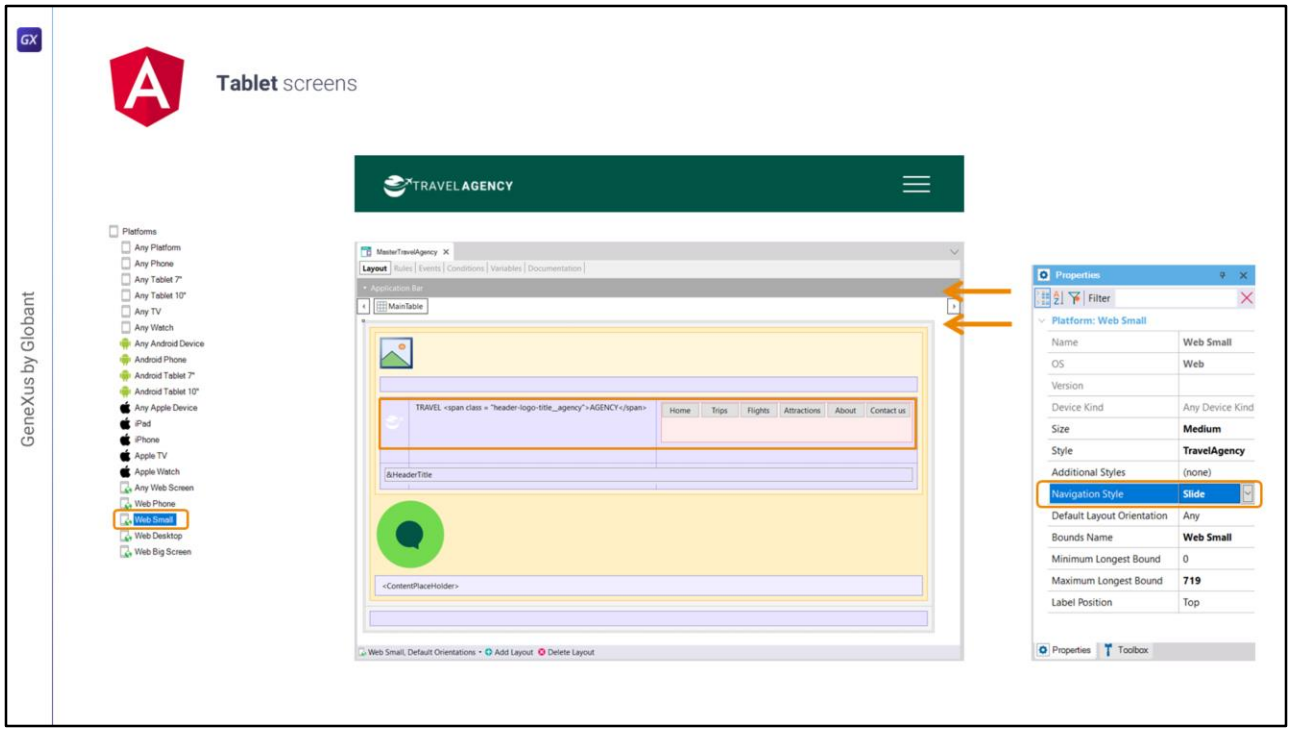


In that case, it would have been necessary to think about how to implement the hamburger menu within the Master Panel.



For example, for the Tablet size...

The tabs menu probably won't work (we'd need to check with the designer, but most likely that menu won't be suitable for the web)... the only thing, then, that we didn't cover in modules 1 to 4 is how to implement the hamburger menu in the Master Panel.

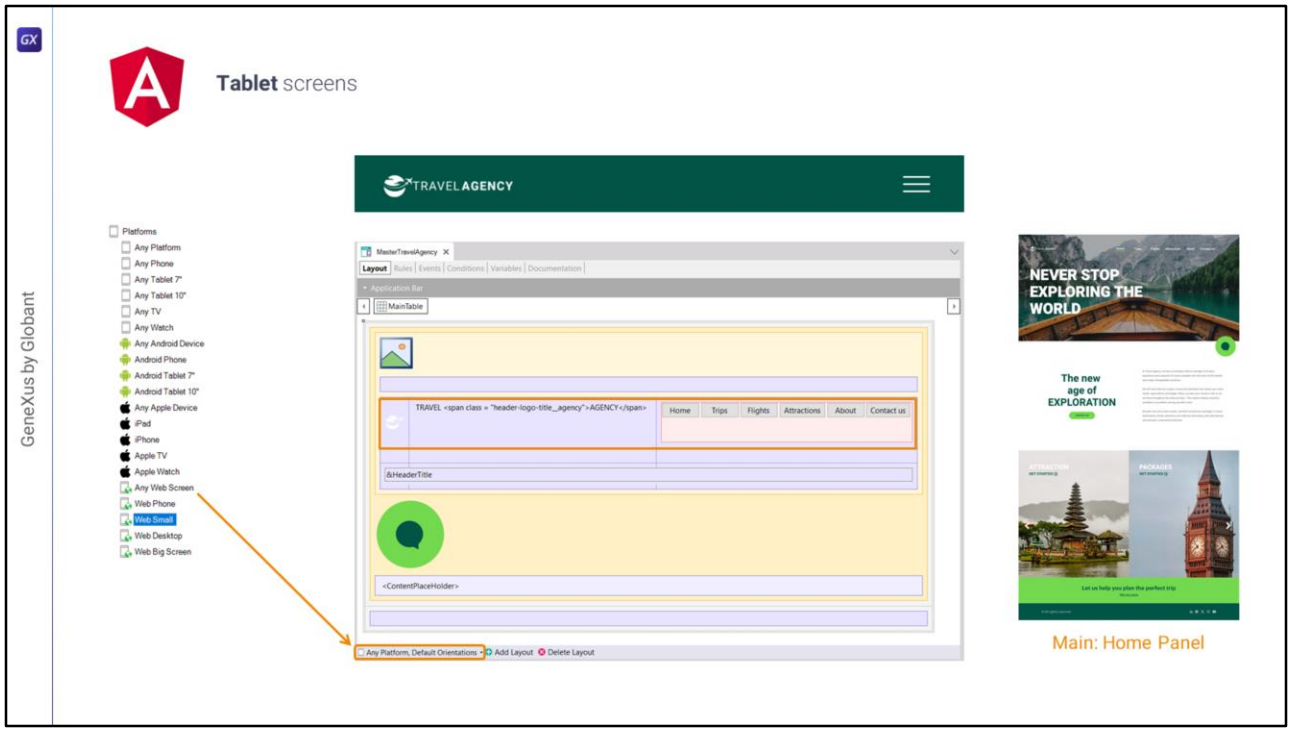


Clearly we would have to remove this row from the table, as that will now be in the Application Bar, or in row 1 of the Main table, depending on how we implement it.

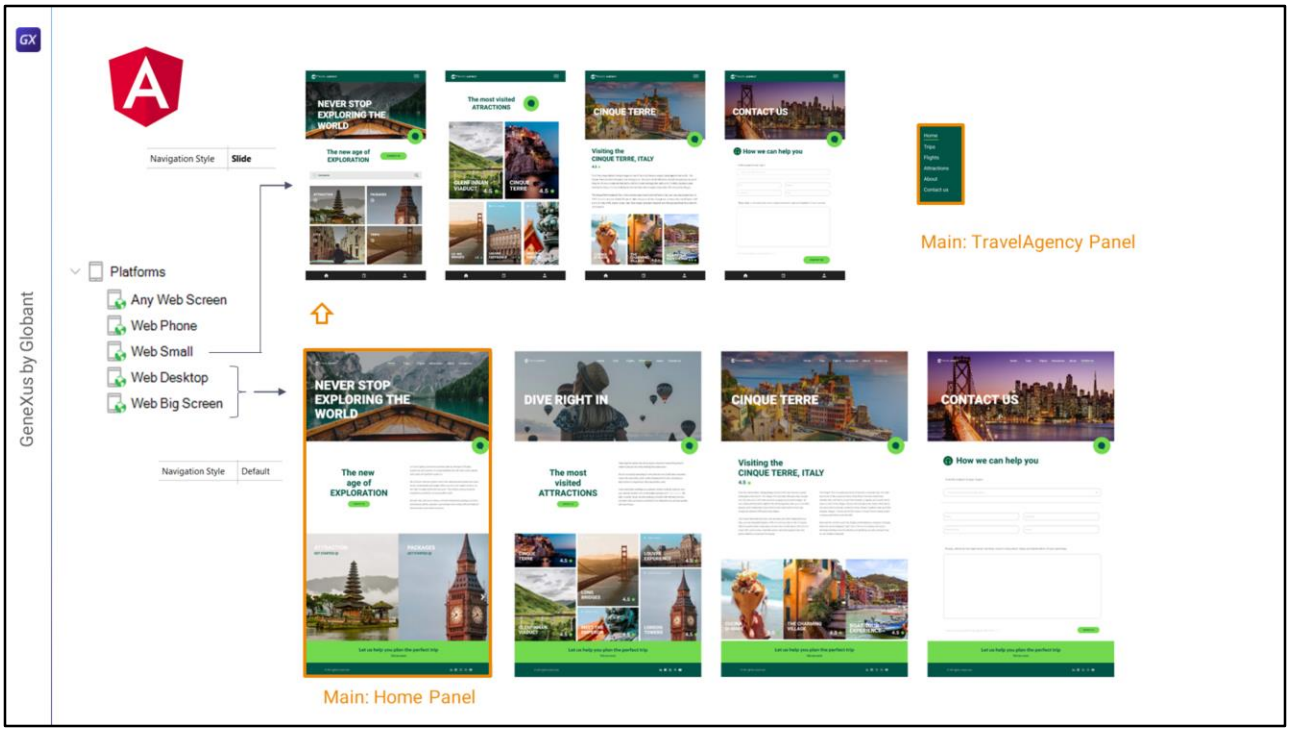
We could think of 2 implementation alternatives: an automatic and a manual one.

The **automatic** alternative would be analogous to the one we saw for Android; that is, the one provided by the Slide navigation style. What was seen there applies to Angular—what we saw in the previous videos.

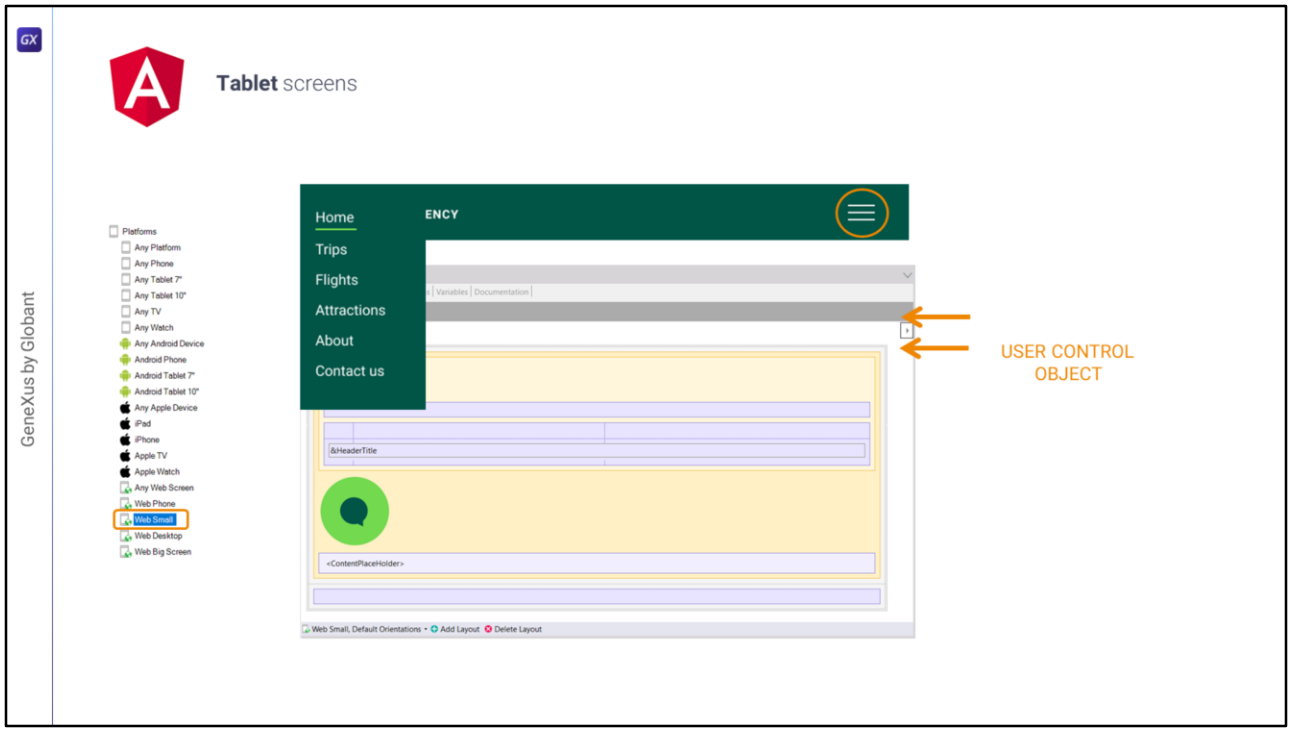
But it won't help us, because in this case we would have to dynamically change the main object of the application, which is not possible.



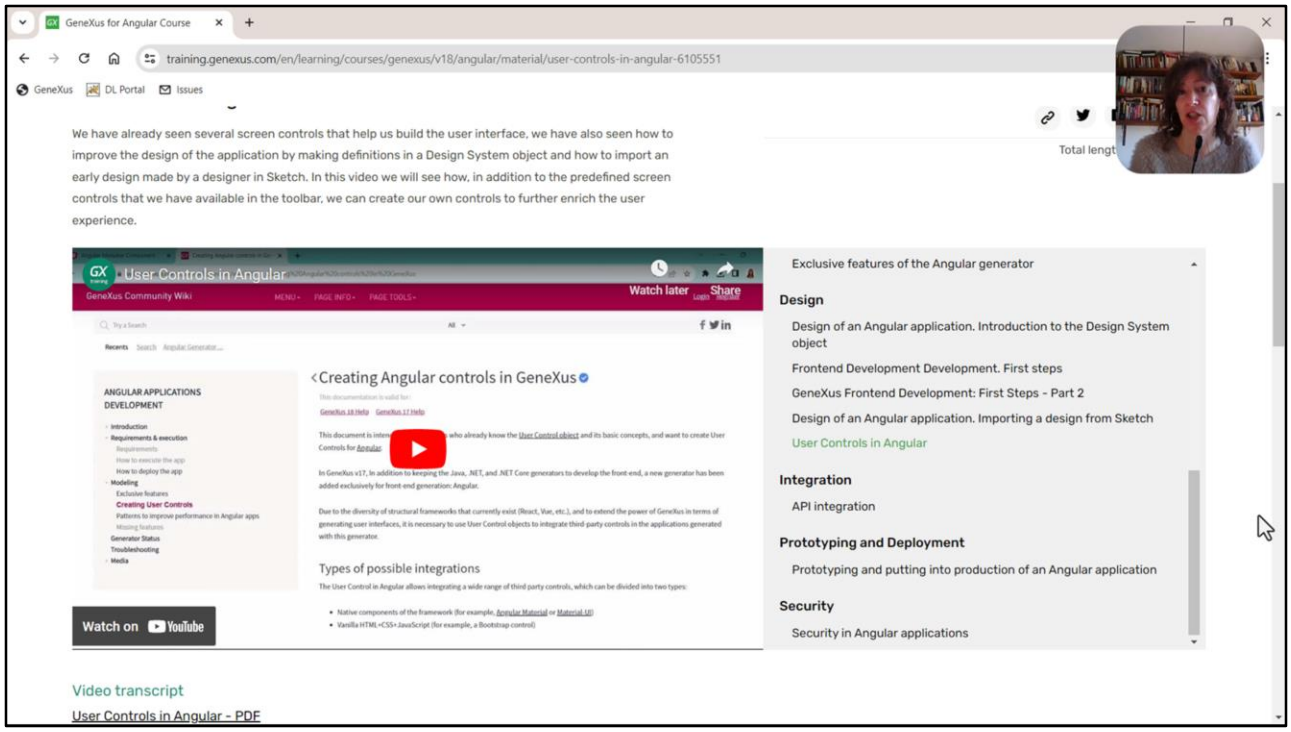
Remember that for desktop size (or larger), the menu was integrated into the Master Panel as an indistinguishable part of the rest, so the main object was the Home object.



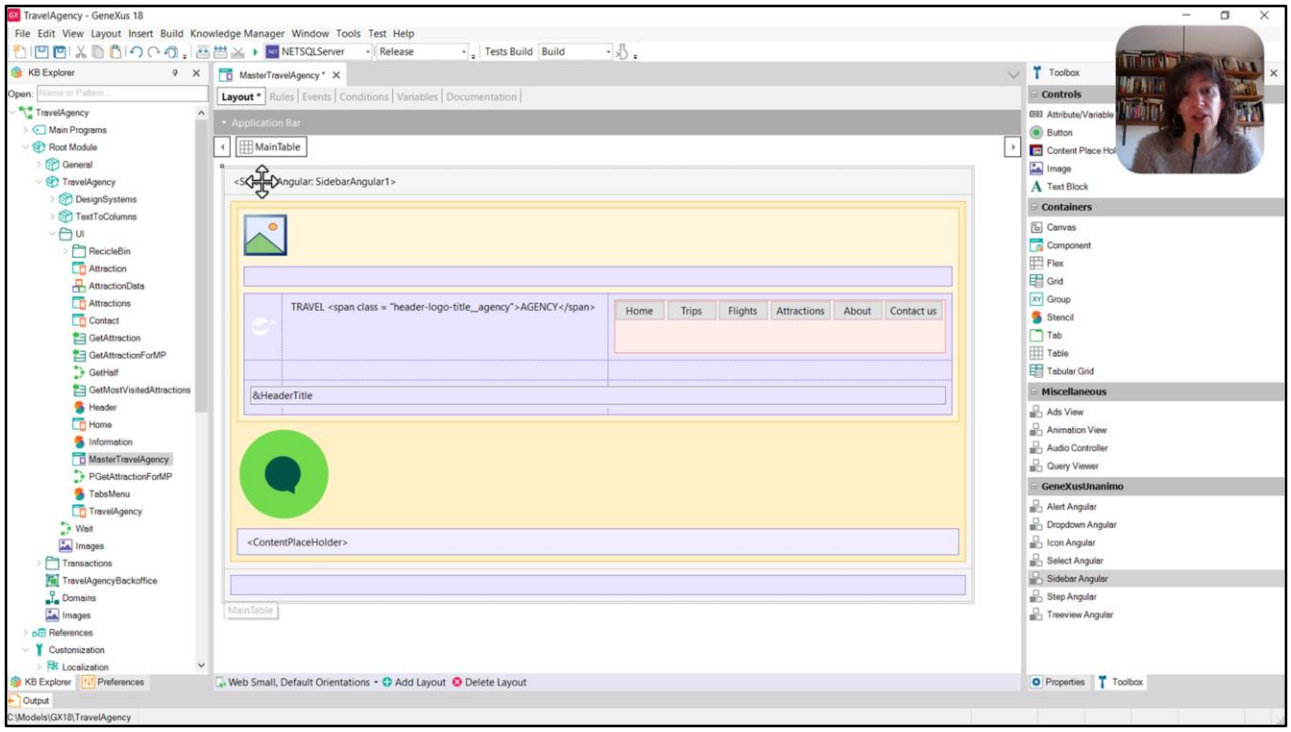
Think about what would happen when running the same Angular application on a tablet or laptop of that size. We would have to change the navigation style, and the menu would also have to be the main object.
And this isn't possible, because the application is the same.



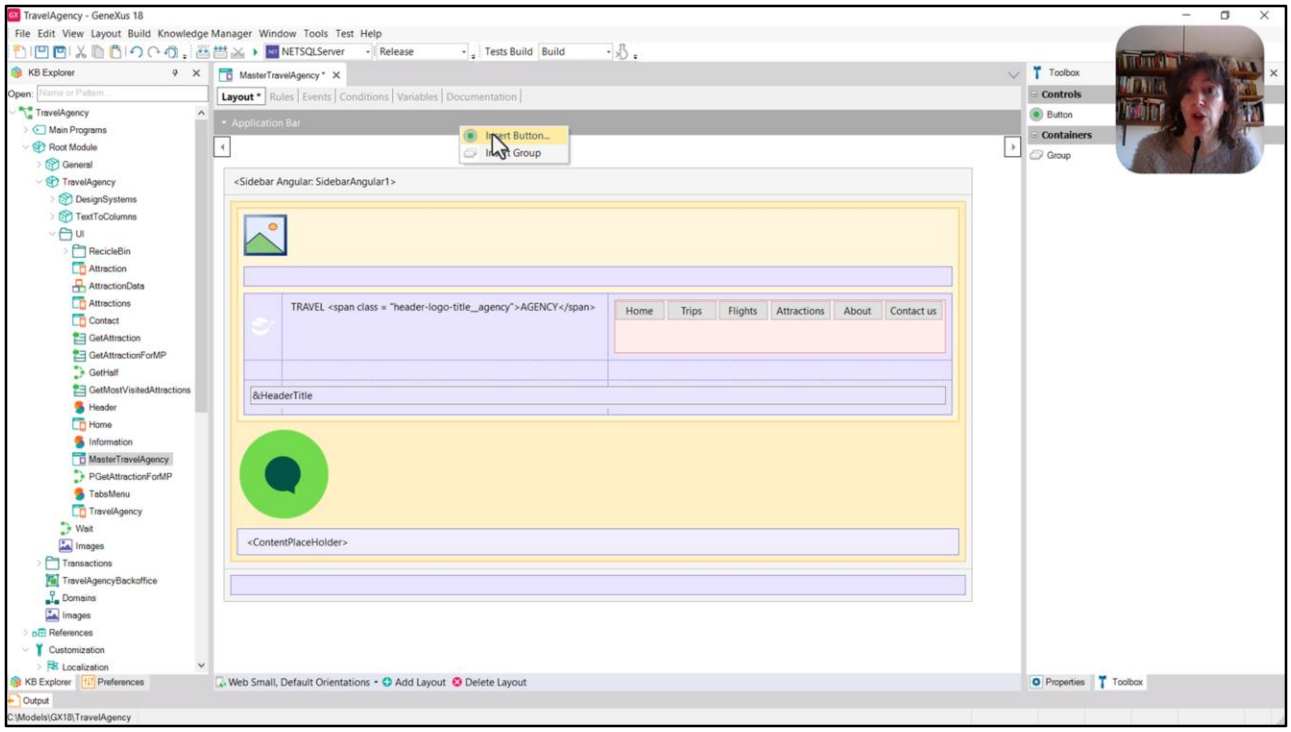
So, we are left with the **manual** solution. And there we have several alternatives. In addition to those we saw for Android, here it is also possible to implement the hamburger menu through a User Control, either developed by us or by a third party…



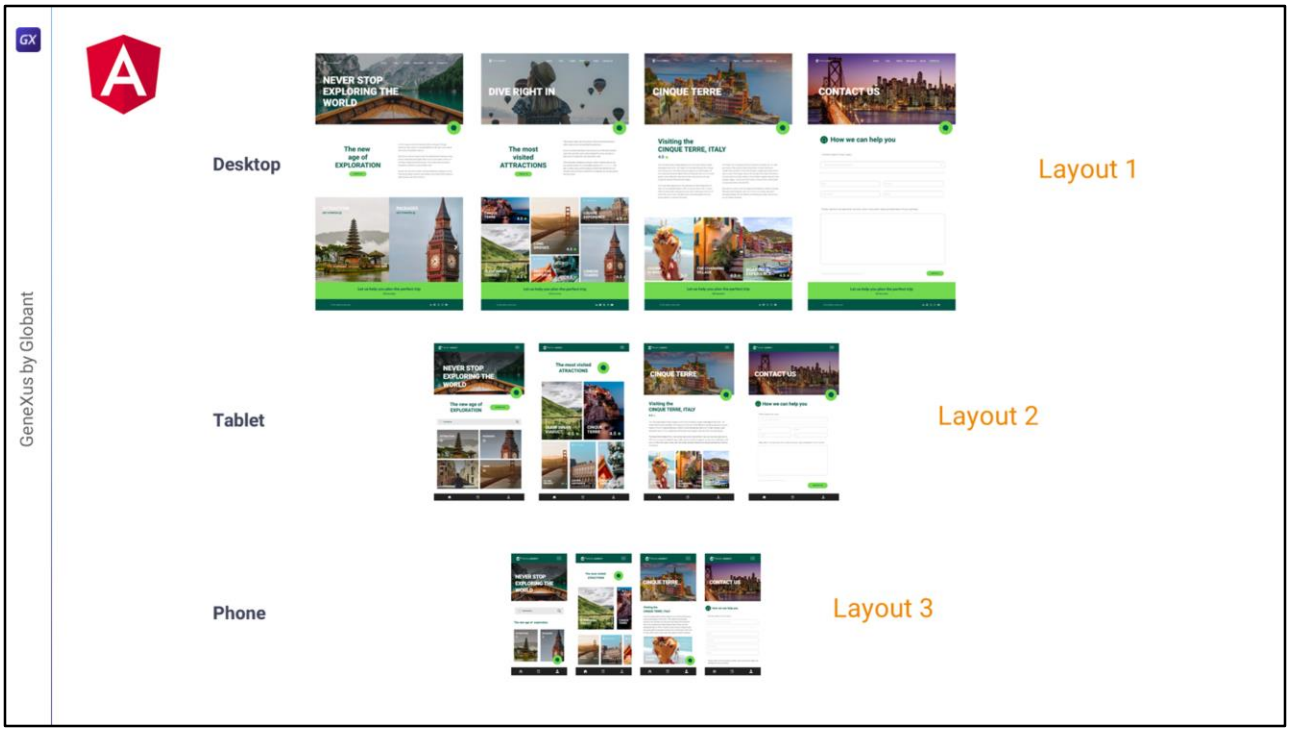
...for that we'll need to copy the HTML code from that development, place it inside the User Control object, and make a couple of modifications; then we'll be able to use it inside our KB.



...or we could even try to use the one already provided by the Unanimo Design System, which is available in the toolbox.

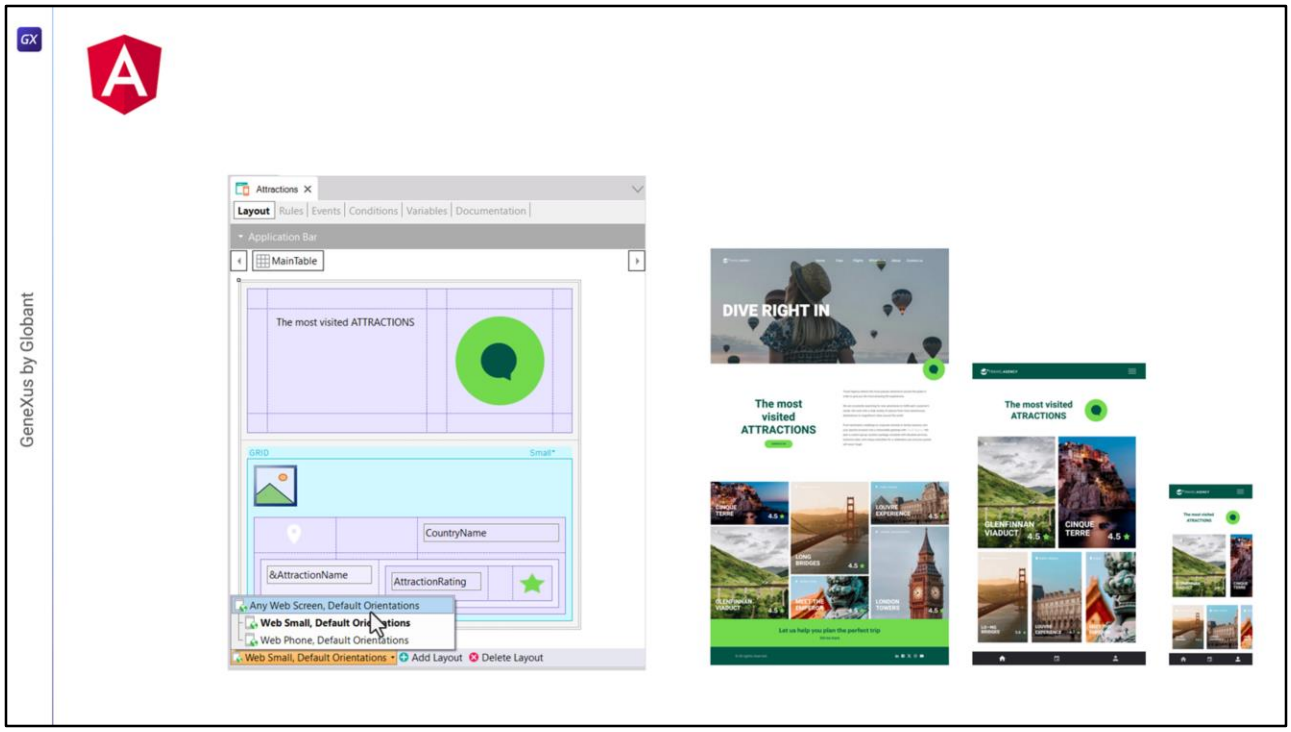


For these cases, we will have to emulate the Application Bar in row 1 of the Main Table, because in the Application Bar we can only place controls like buttons or Action Groups. We can't use anything else, particularly not a User Control.



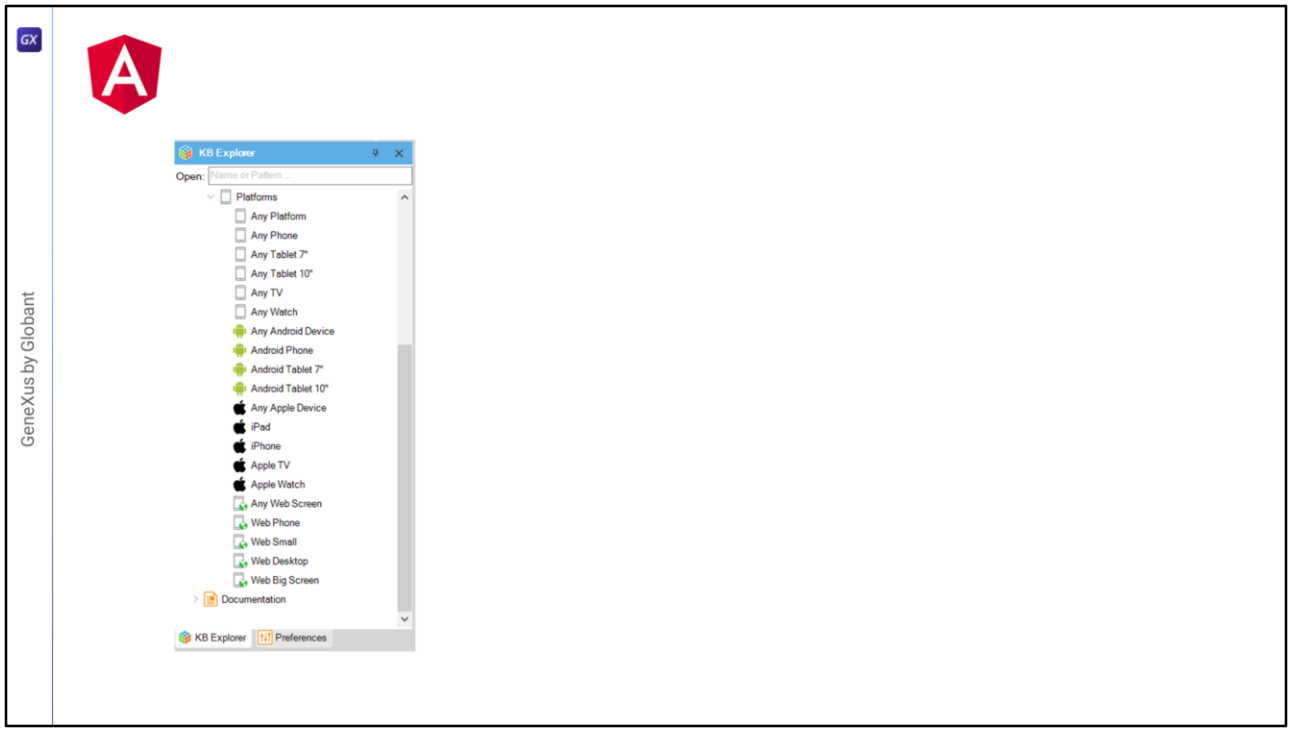
In short, to have the complete Angular solution, we would do the following:

Having the screens for Desktop, we will add a layout to each panel (and Master Panel) for the two remaining breakpoints: Tablet and Phone.

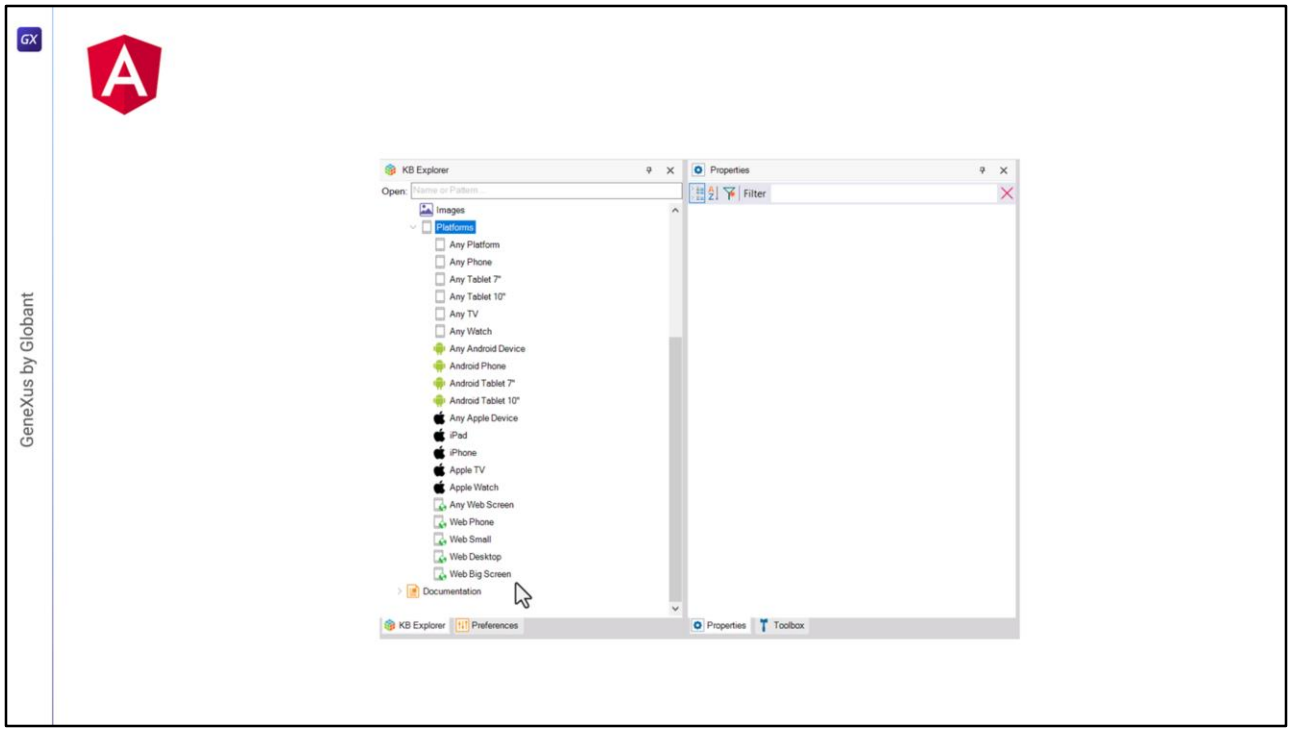


For example, for the Attractions panel we will have 3 layouts: the one for Phone size, the one for Tablet size that corresponds to the Web Small platform, and another one for any Web screen that is not one of the other two, so it will be used for Desktop and Big Screen sizes.

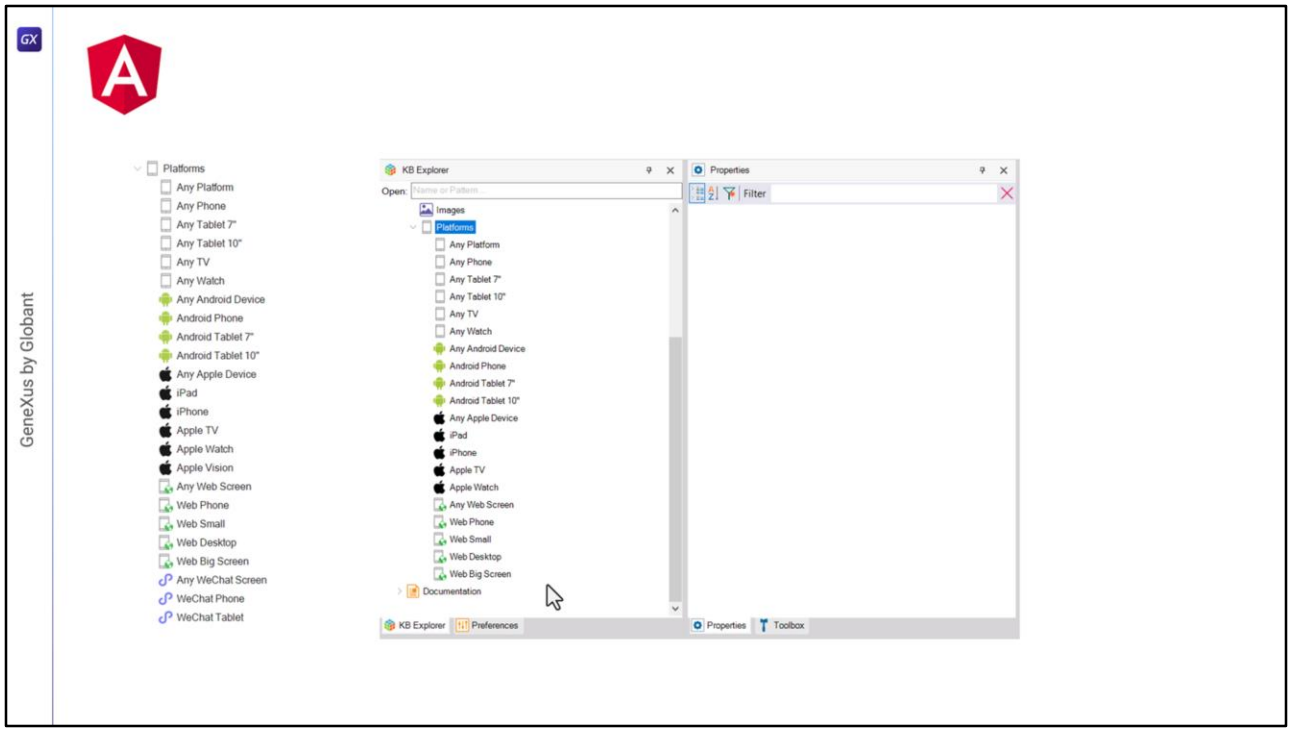
But this is not the only way to model the same 3 layouts.



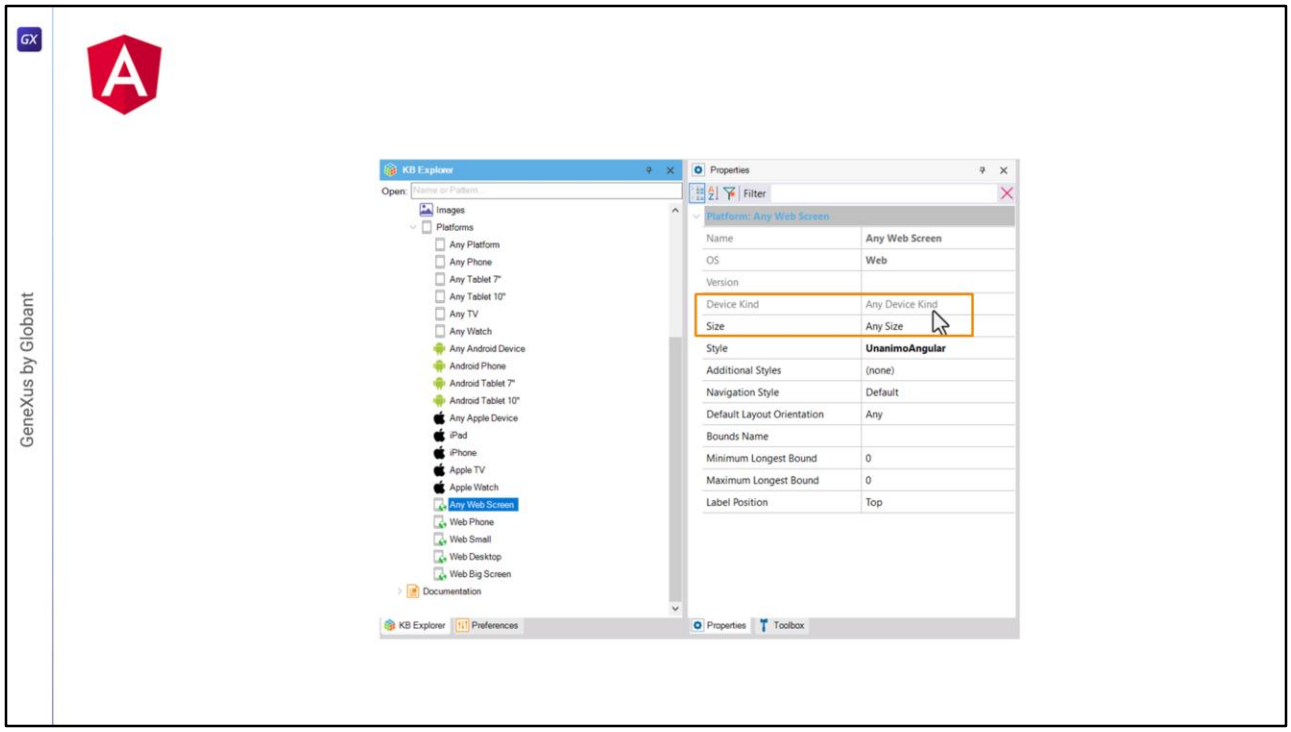
It is important to mention that this definition of the platform universe is the default and establishes the variants that are usually needed in a development. But they can be modified.



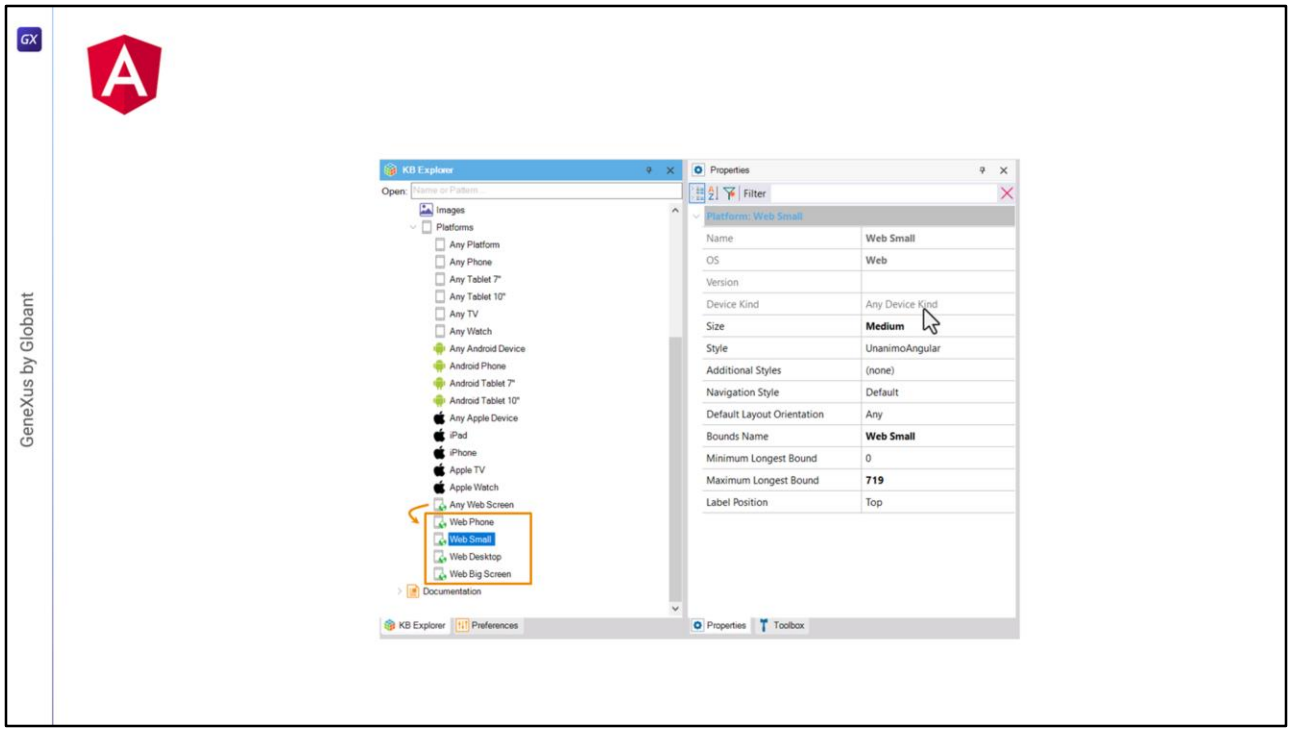
Here, I'm showing the default definitions when creating a new KB (with upgrade 8...



...then more platforms appeared, such as WeChat, but for what I want to show you now, they don't matter).

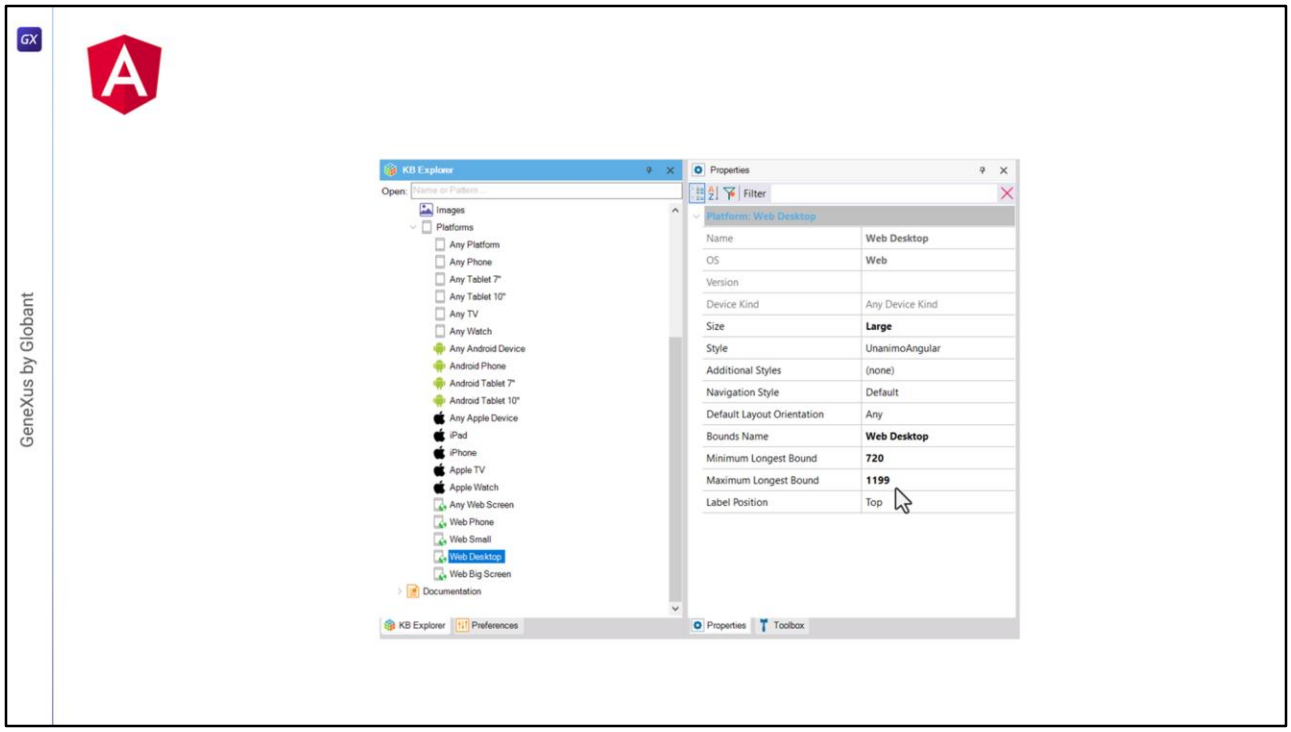


As we can see, we have a default platform that will work for all cases of Web execution: and that is why we see here “Any Device Kind” and here “Any Size”.

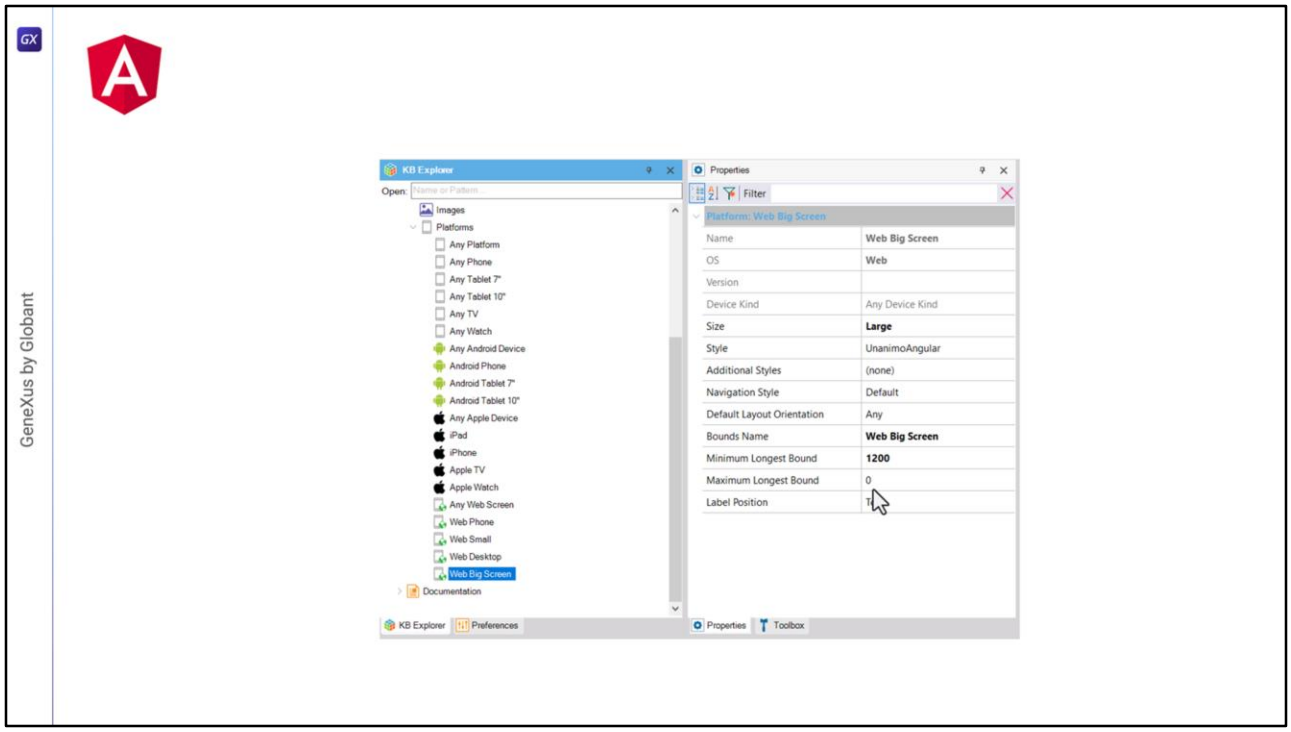


But we can also specialize it using these 4 options, which divide the Web platform universe according to: device type and screen size.

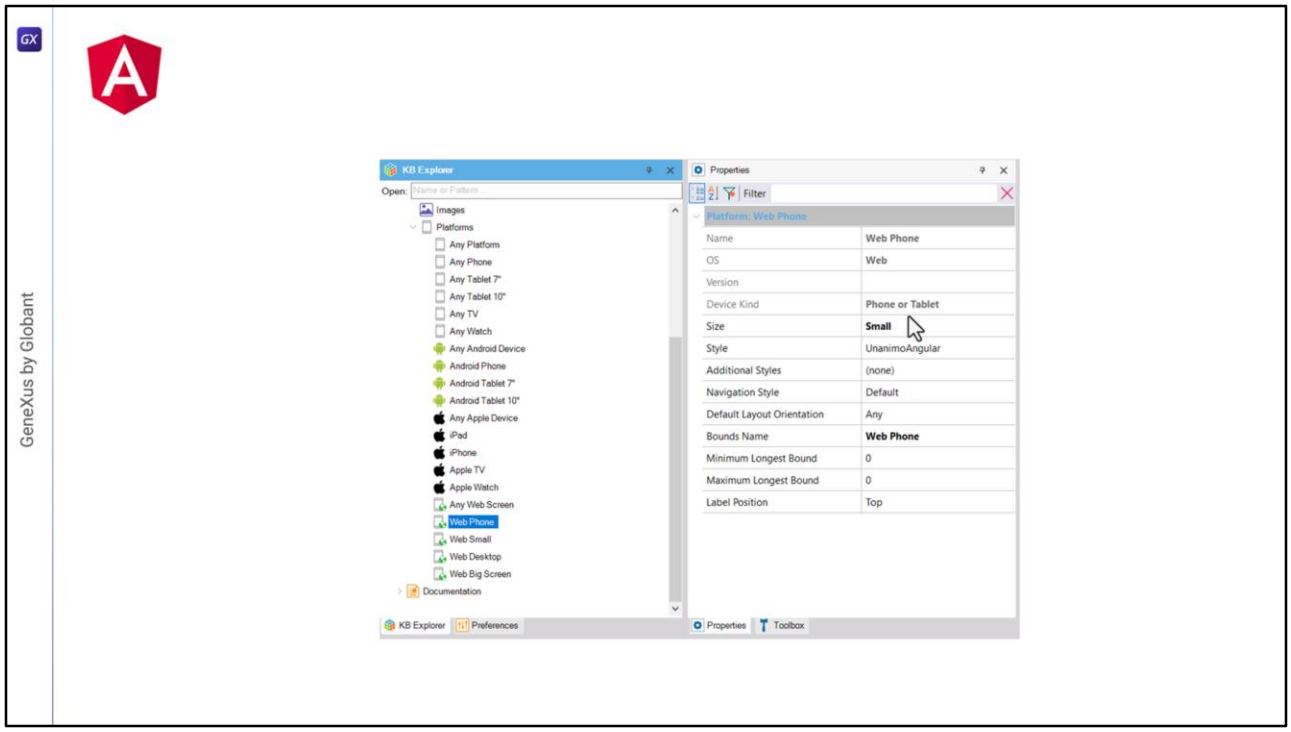
So, we can see that **Web Small** will correspond to any device (that is, Phone, Tablet, PC or laptop) whose width is **up to** 719 dips. When a bound has 0 value, it means that it is not taken into account, as if it said Any.



For any device of Large size with a width between 720 and 1199, this other platform will be used.

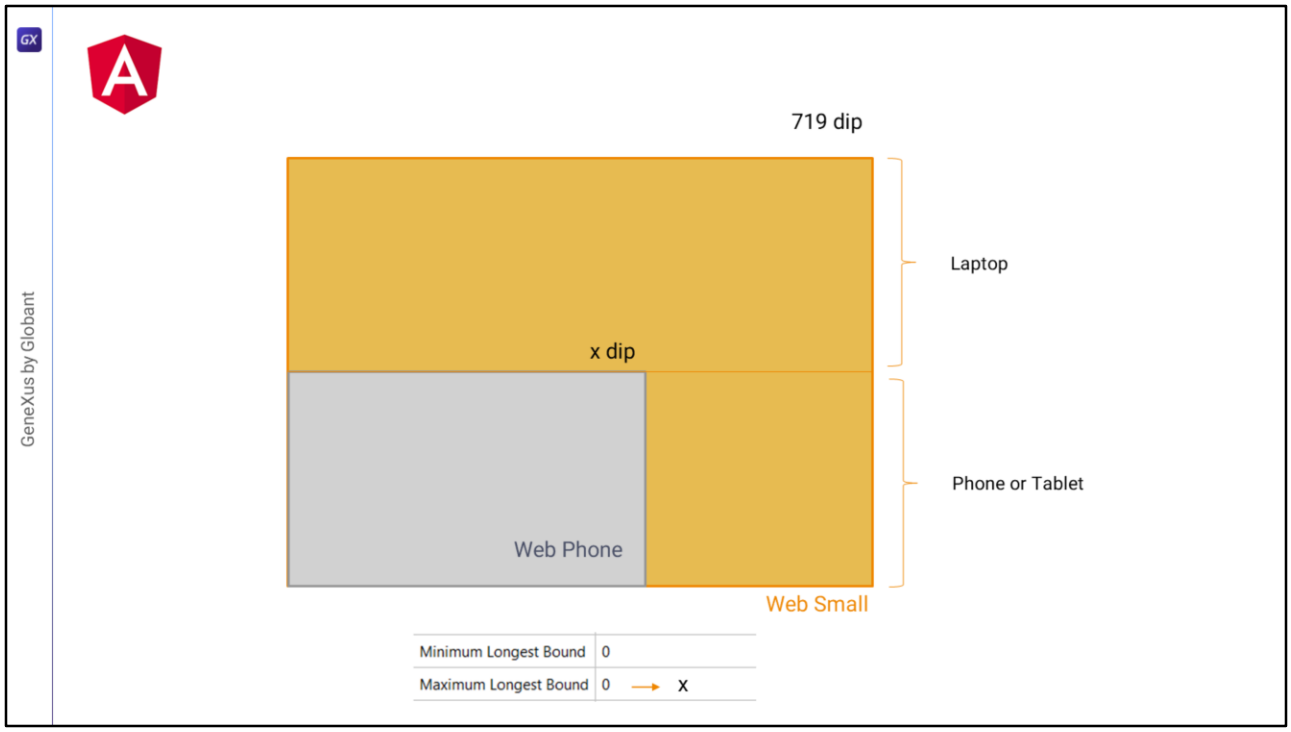


And for those wider than that, this other one.



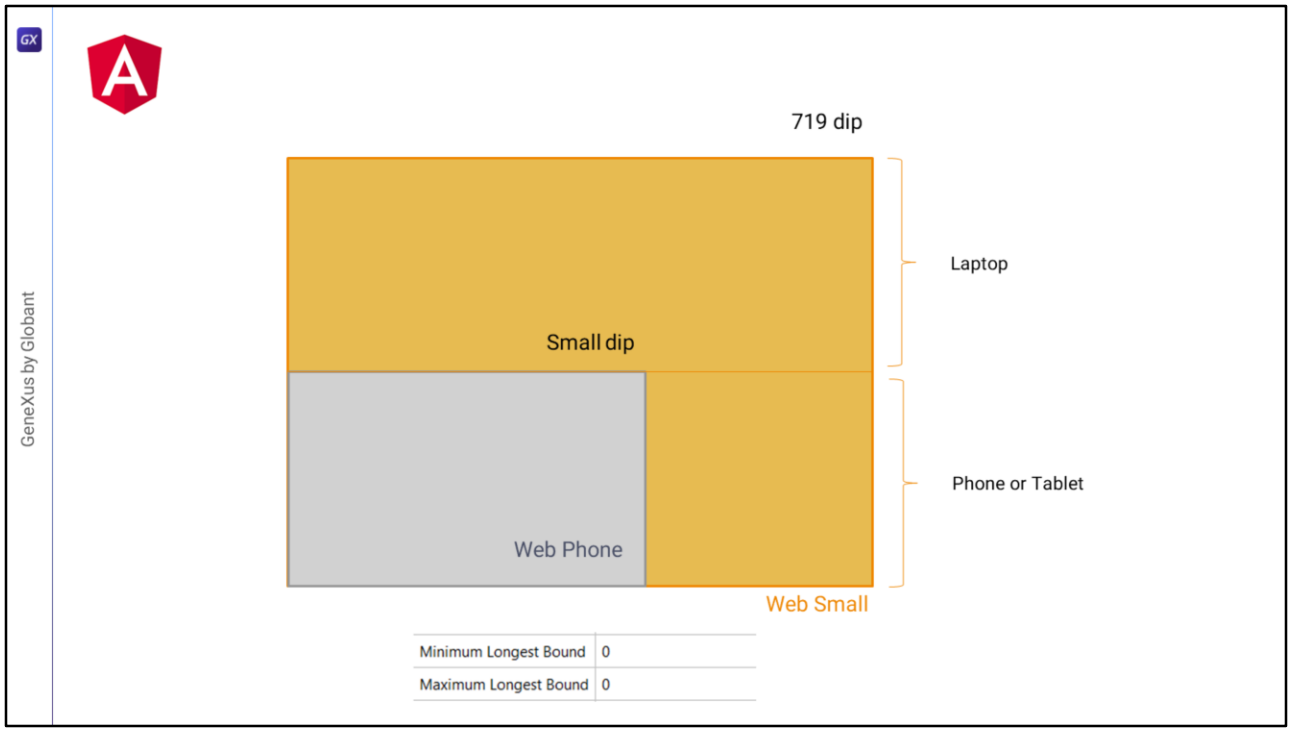
What about this other one? It is more specific, because it is only valid for small size mobile devices (not for laptops or PCs). Note that it doesn't have defined bounds, so it's like saying “any”, and the size is determined by the Size property, which in this case is Small.

But... here all this can become confusing, since the universe of the **Web Small** platform includes the **Web Phone**. So, to which cases does one and the other apply?

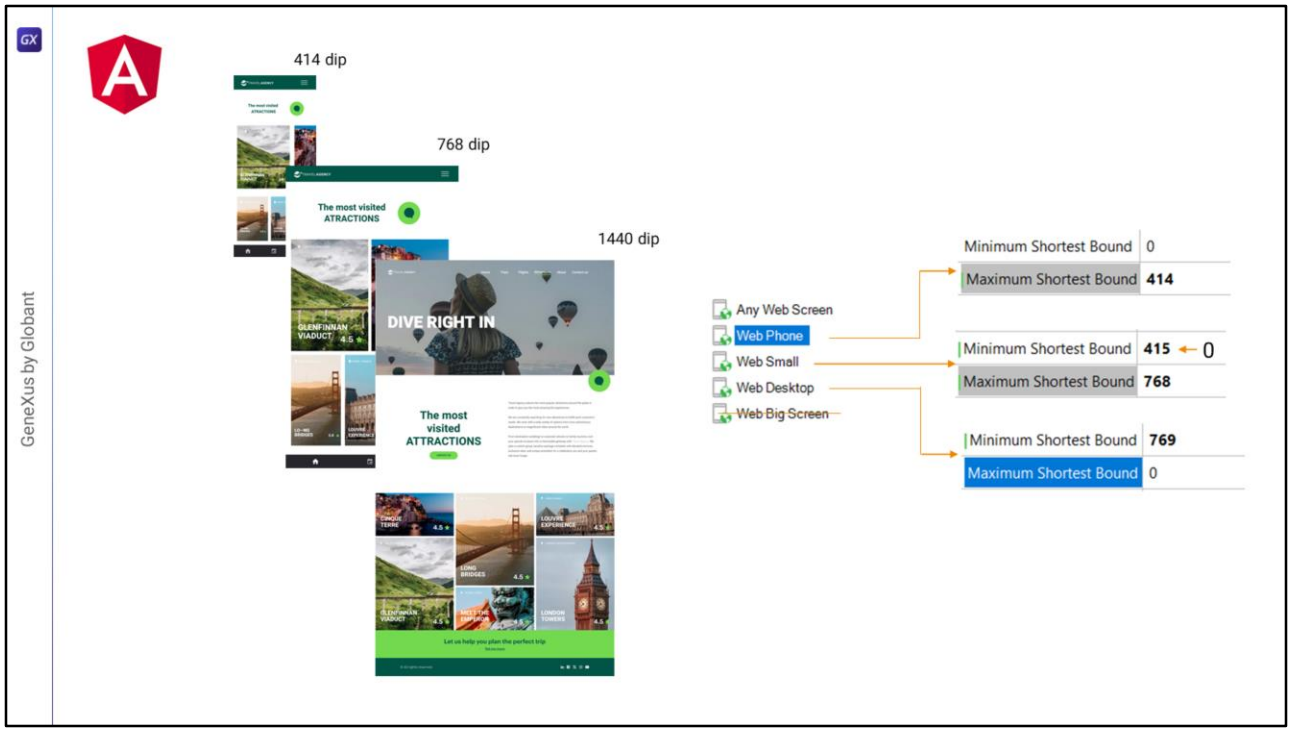


If a particular execution case falls within the universe represented by more than one platform, it will always match the most specific one, the one that most closely matches its characteristics.

So, if this is the universe that would correspond to the Web Small platform, with the maximum width of 719 dips, and that applies to both the application running on a laptop of that width range, and a mobile device up to 719 dips, the Web Phone will be a subset, because it leaves out everything that is not a mobile device, and we have to see what happens with the width. If we set the maximum width property to a value of x dips, then it will correspond to the application running on a mobile device up to that width. And for any other case (both mobile device and laptop) up to 719 dips will correspond to the Web Small platform.



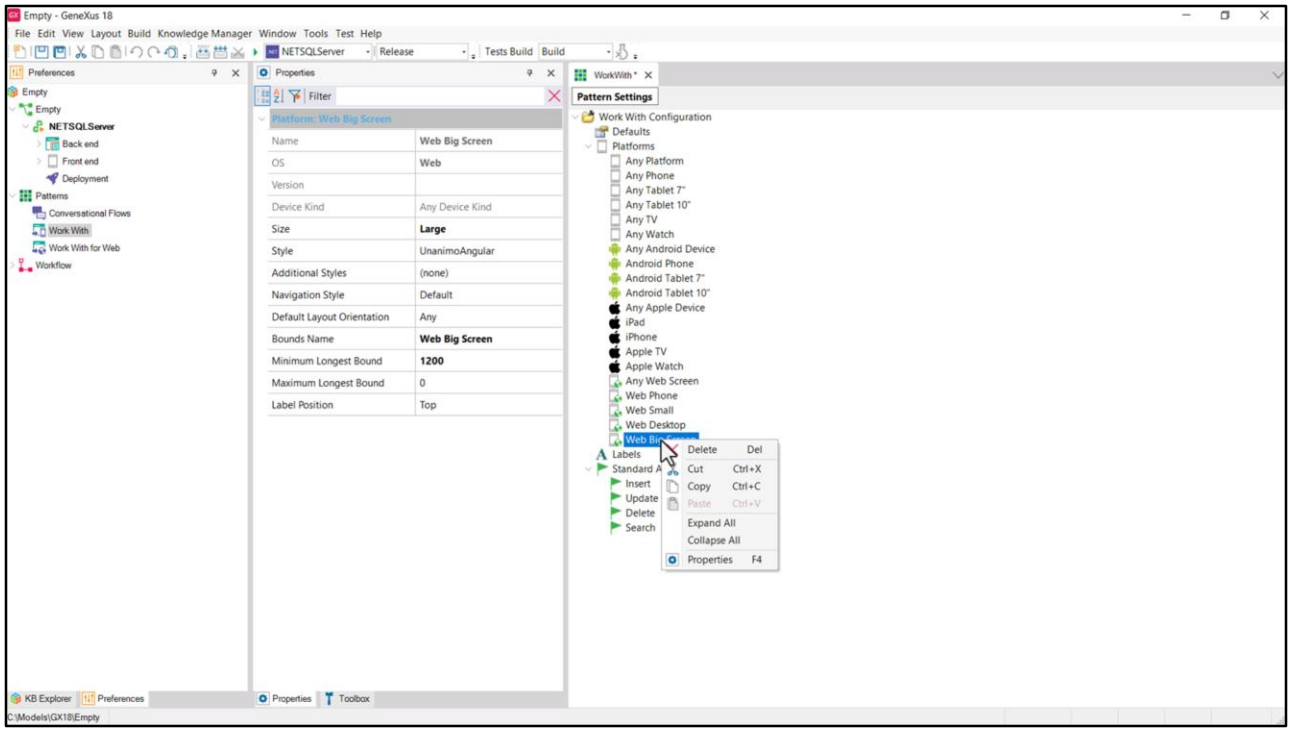
If we leave the default value of 0, then it will apply to what it understands as Small size.



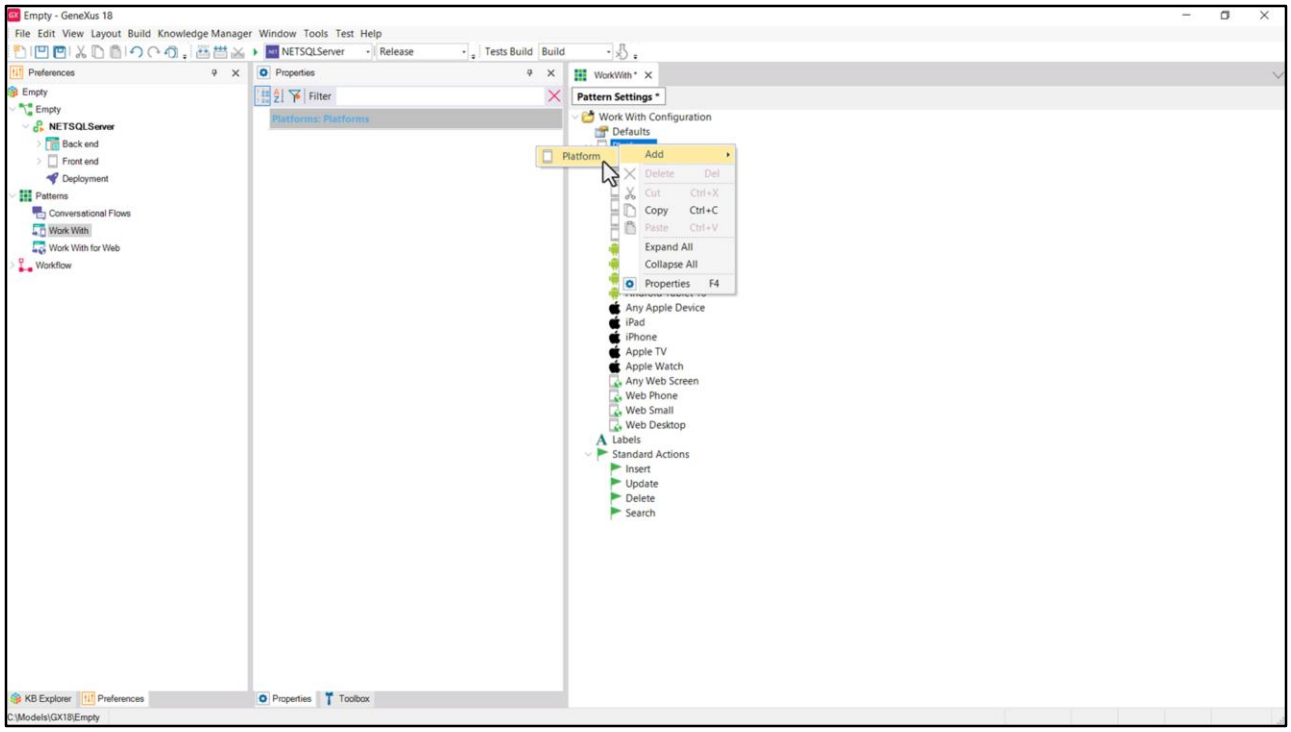
For the reality of our application, taking from Figma the widths of Chechu's designs, of the 4 we would need only 3 web platforms, where to the Web Phone we would change the maximum width to 414, to the Web Small we would define this range (although we could well leave 0 here, knowing that the Phone only applies to mobile devices), and to the Web Desktop this one.

It will not be necessary to remove Web Big Screen from the platforms node, although it would be clearer.

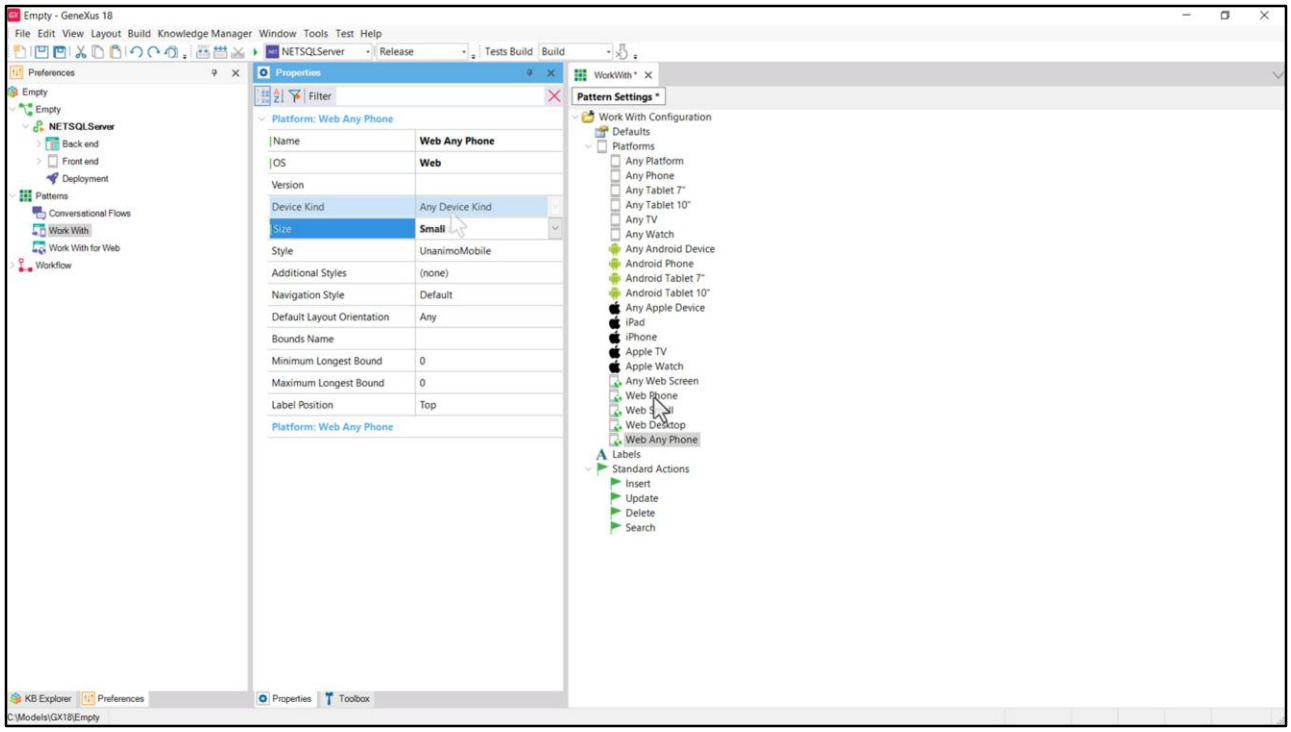
You may wonder how we would remove it, or even create some other platform if we wanted to reorganize the universe of possibilities according to other differentiations, and not these.



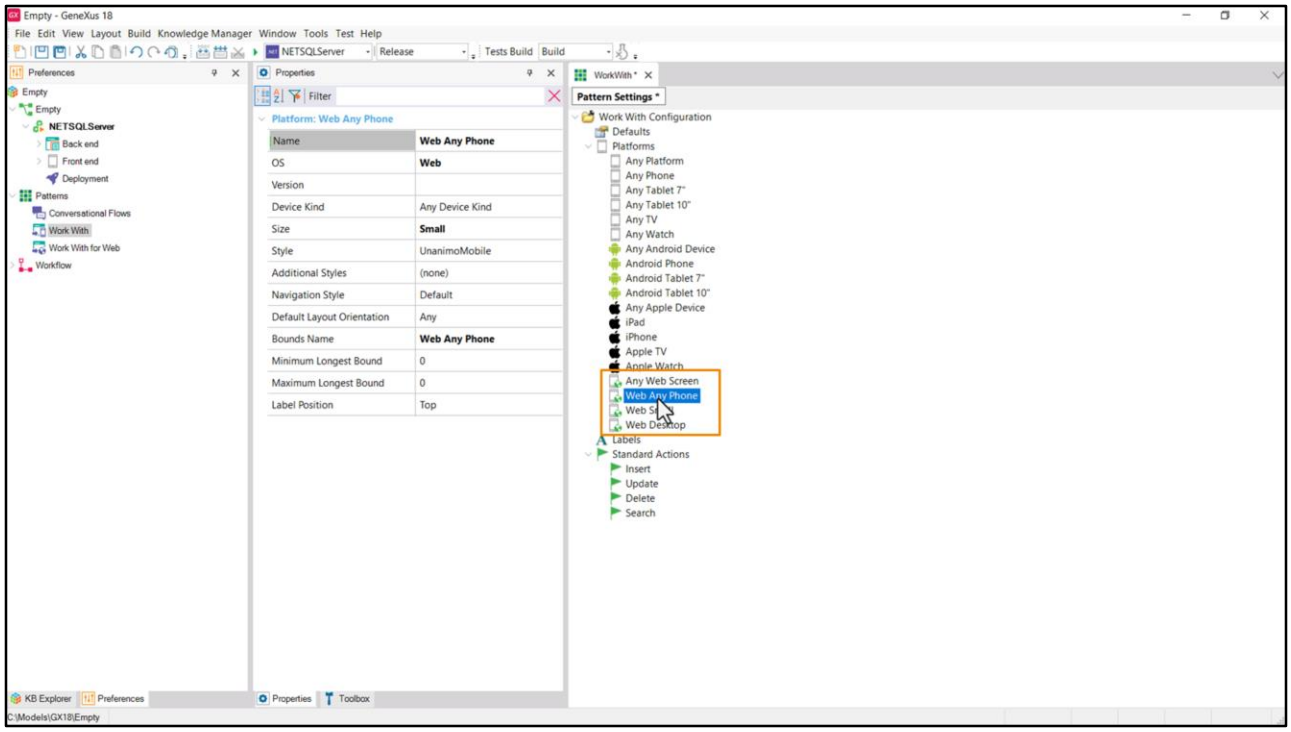
It is in the Preferences tab, WorkWith node, where we can work with the platforms. For example, deleting this one.

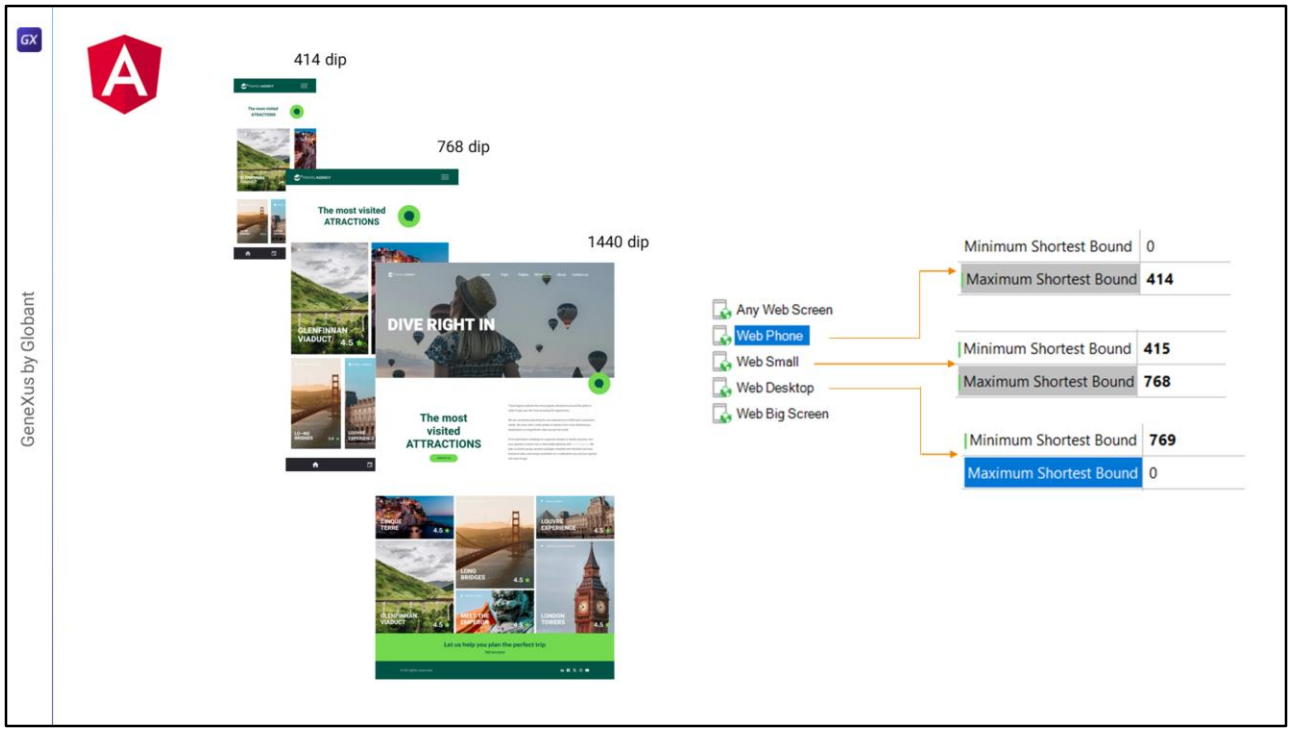


And if we want to add one to replace the Web Phone, because we will want to make it valid for Any Device Kind and not only for mobile devices... (although in theory it doesn't make sense because there are no laptops or PCs so small) this is how we get it.

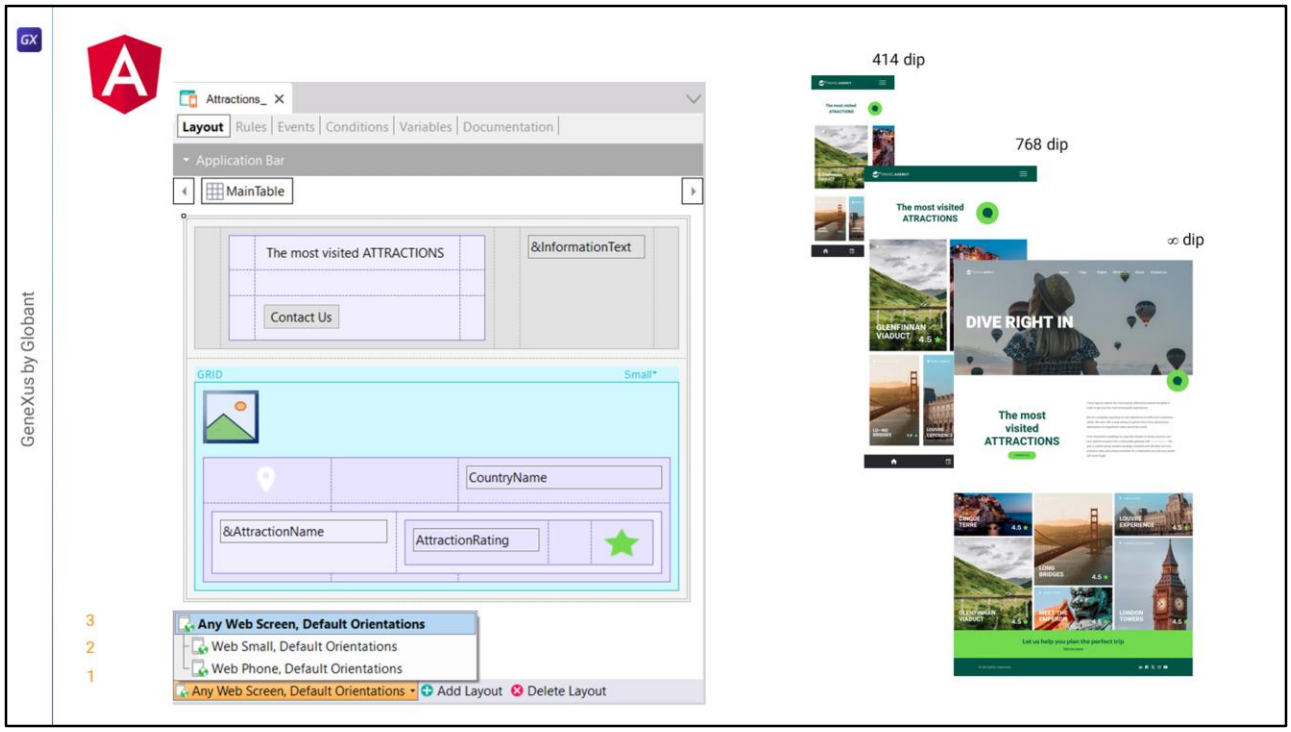


There we are adding a new platform and we have to assign a value to all its properties.


Empty - GeneXus 18
File Edit View Layout Build Knowledge Manager Window Tools Test Help
NETSQLServer
Release
Tests Build
Build
Preferences
Empty
Empty
NETSQLServer
Back end
Front end
Deployment
Patterns
Conversational Flows
Work With
Work With for Web
Workflow
Properties
Filter
Platform: Web Any Phone
Name
Web Any Phone
OS
Web
Version
Device Kind
Any Device Kind
Size
Small
Style
UnanimoMobile
Additional Styles
(none)
Navigation Style
Default
Default Layout Orientation
Any
Bounds Name
Web Any Phone
Minimum Longest Bound
0
Maximum Longest Bound
0
Label Position
Top
WorkWith *
Pattern Settings *
Work With Configuration
Defaults
Platforms
Any Platform
Any Phone
Any Tablet 7"
Any Tablet 10"
Any TV
Any Watch
Any Android Device
Android Phone
Android Tablet 7"
Android Tablet 10"
Any Apple Device
iPad
iPhone
Apple TV
Apple Watch
Any Web Screen
Web Any Phone
Web Desktop
Labels
Standard Actions
Insert
Update
Delete
Search
KB Explorer
Preferences
Properties
Toolbox
C:\Models\GX18\Empty

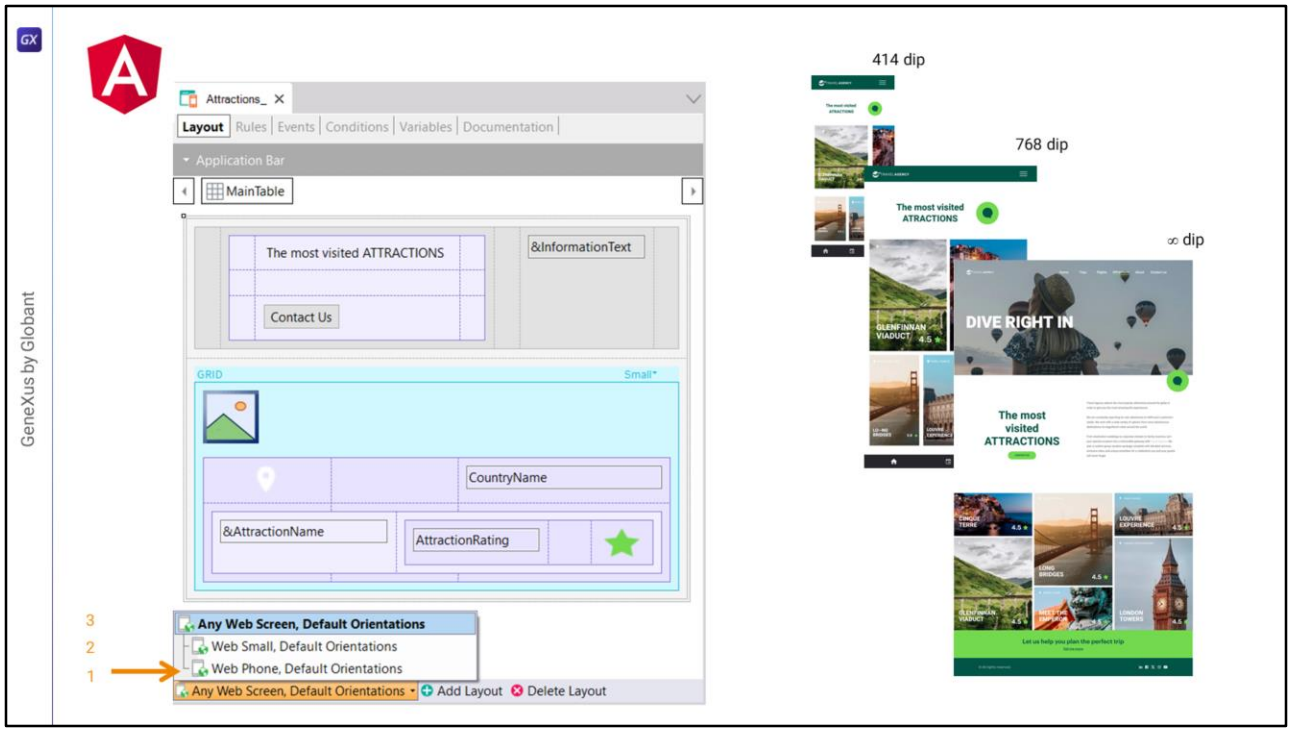


So having defined the platforms that we need, let's say that we leave these...

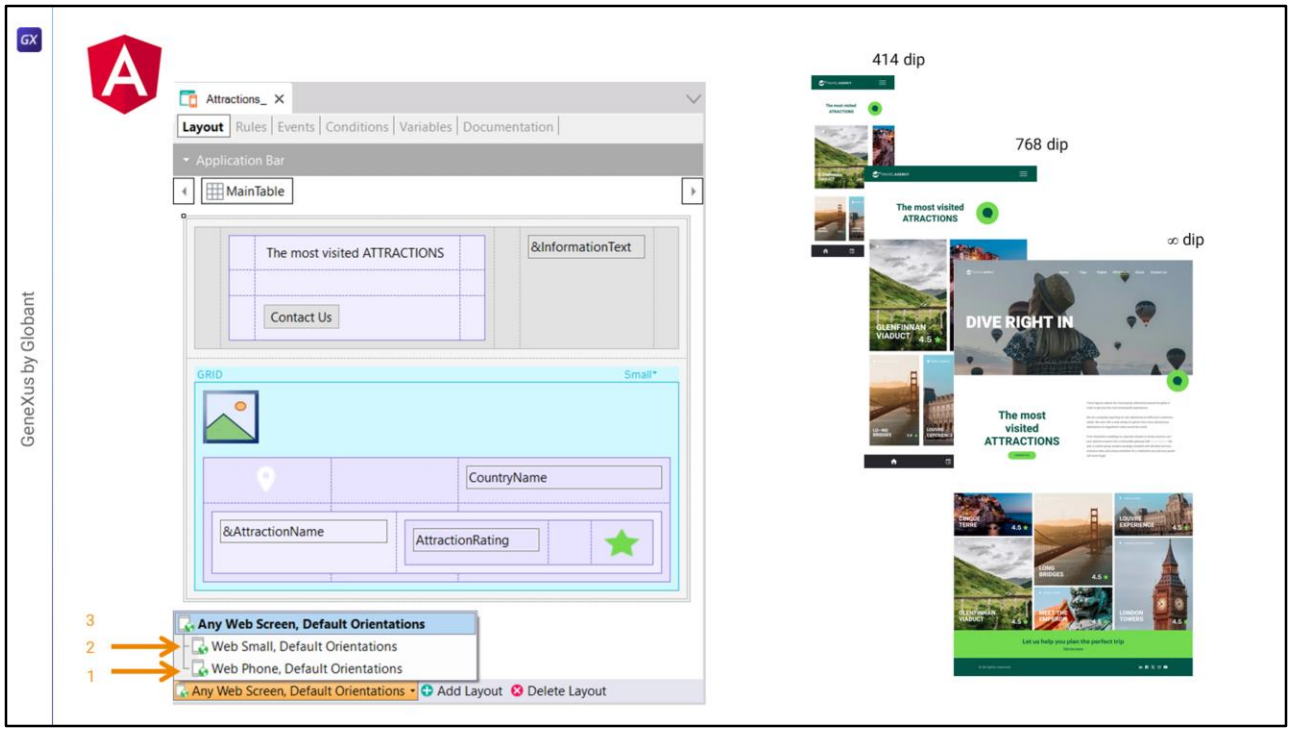


For every object with a UI, in principle we should create as many layouts as different designs are needed, being careful to define those layouts in such a way that they are the right ones for each platform where we need the application.

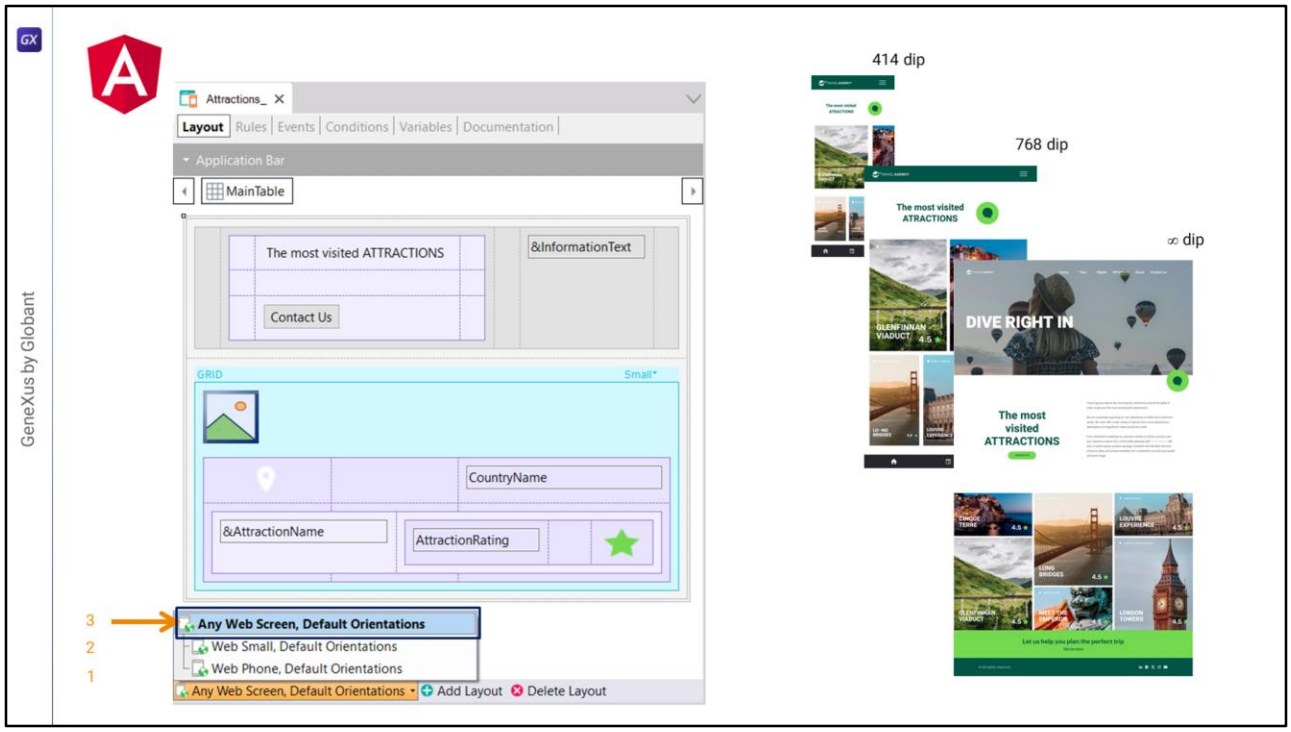
For example, if we have these 3 layouts defined for the Attractions panel, what is done internally to know which one to choose for each case is: first, extract the parameters of the execution platform; second, make an ordered list of the defined layouts, from the most specific to the most general; and third, from that ordered list, the first layout that matches the execution parameters will be the chosen one.



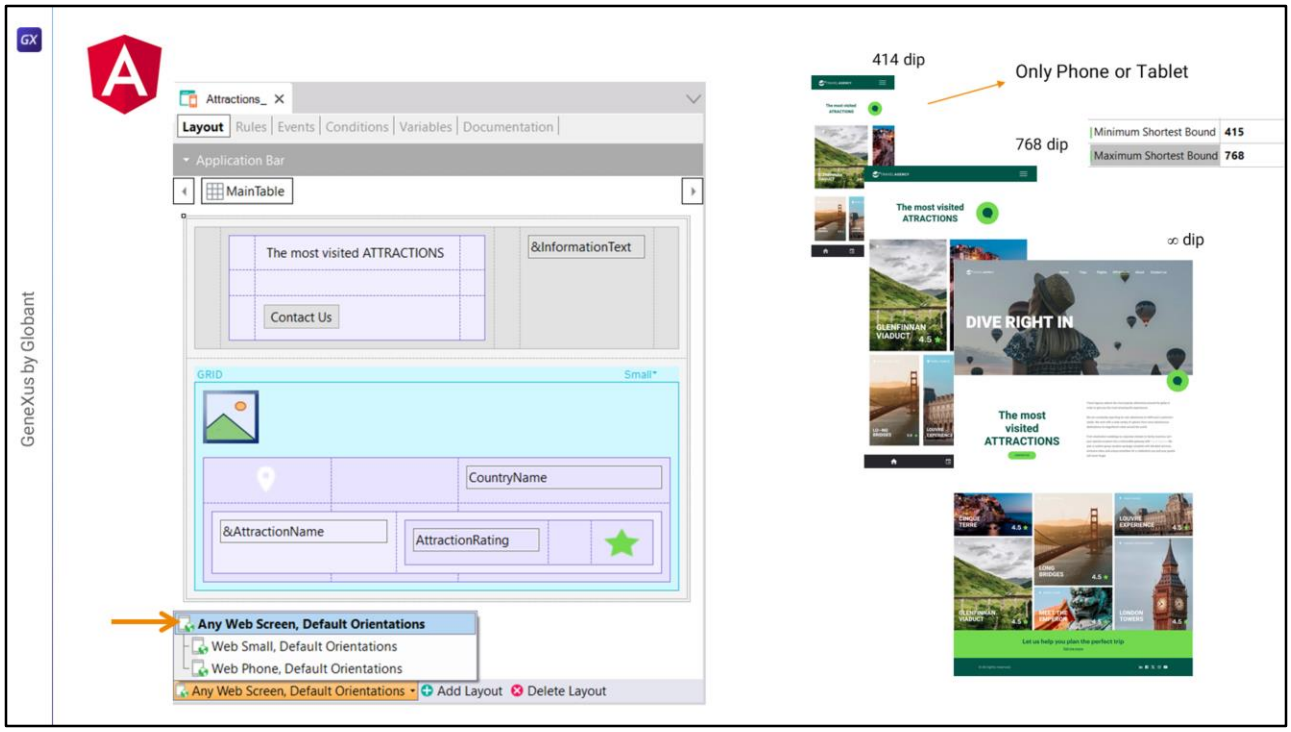
So in this case, if we are going to run it on a Phone up to 414 dips this one will be chosen.



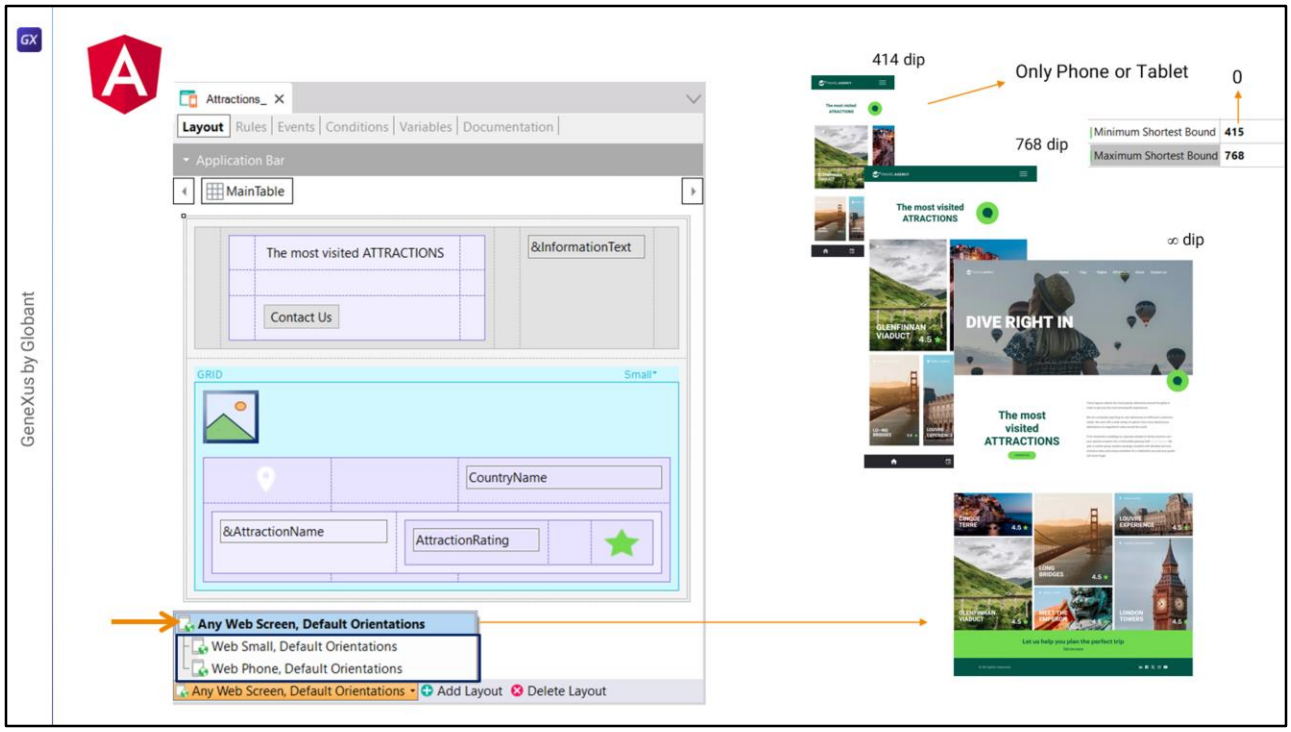
If we will do it on a Tablet between 415 and 768, or on a laptop with a screen width of up to 768, it will choose this other one.



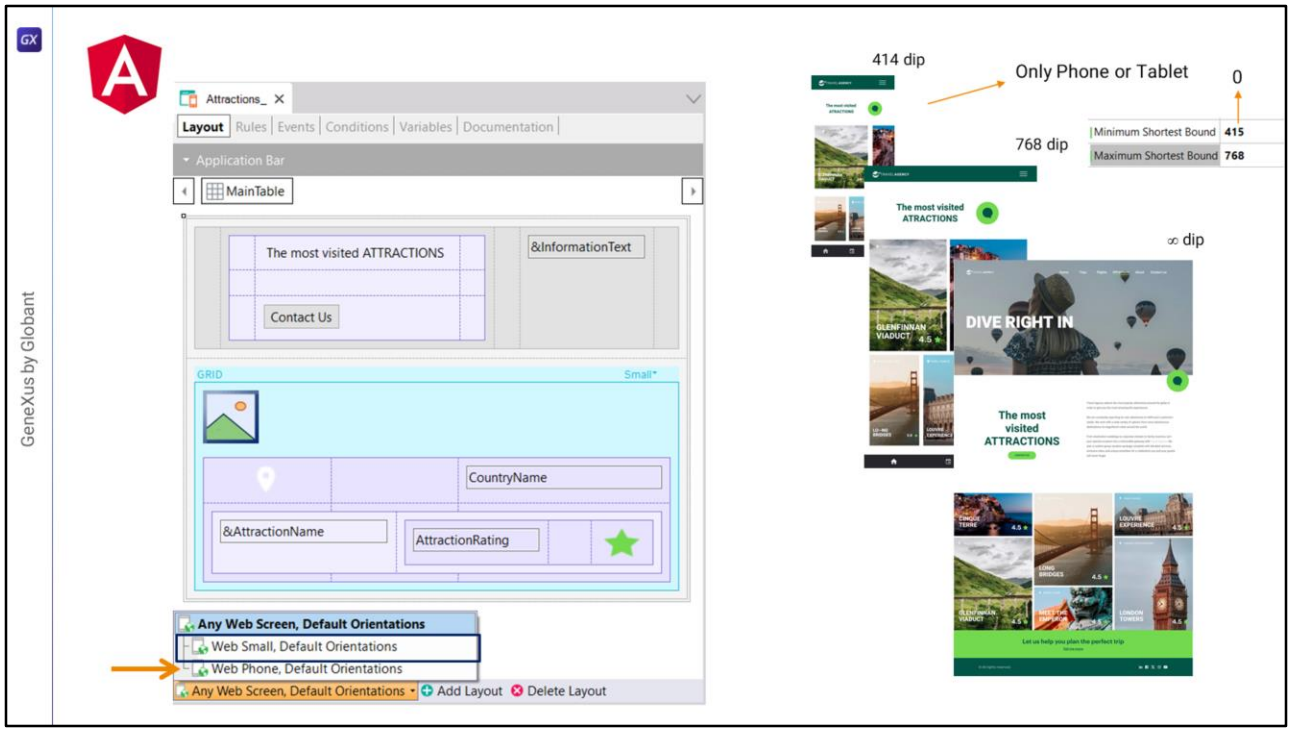
And in any other web case, it will choose this other one. So in the browser of any device of screen size greater than 768 dips it will choose this one. Note that we didn't exactly indicate a layout for Web Desktop. We could have done it. We will have to consider carefully which options are left out.



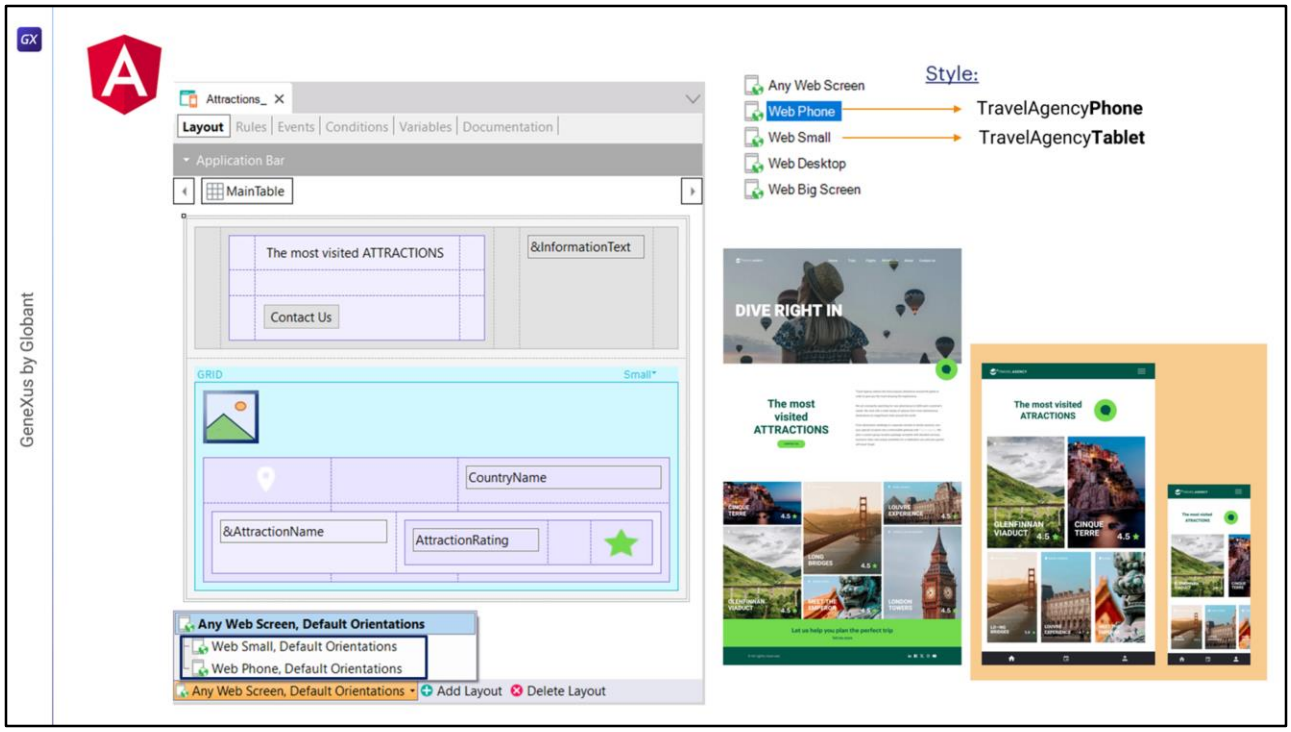
For example, as we place this 415 here, if there were a laptop smaller than 415, it would have to choose this layout, because this is for Phone or Tablet only.



To avoid this case we would set this bound to 0 and so what is left out of these two possibilities is only for any device larger than 768 dips.

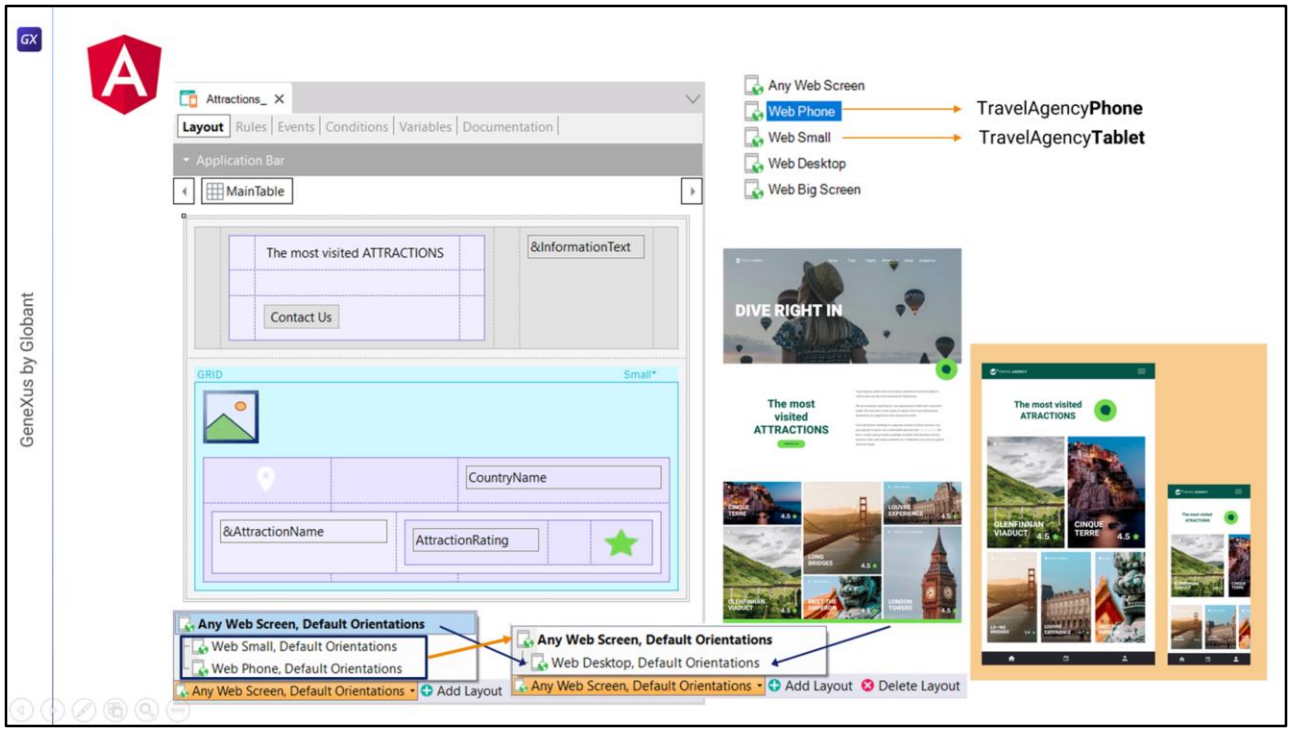


Of course, with this definition of platforms, if there was a laptop of size smaller than 415 dips, for that case it would choose this layout. There we would need to do what I showed you in working with Platforms, so that it chooses this other one.

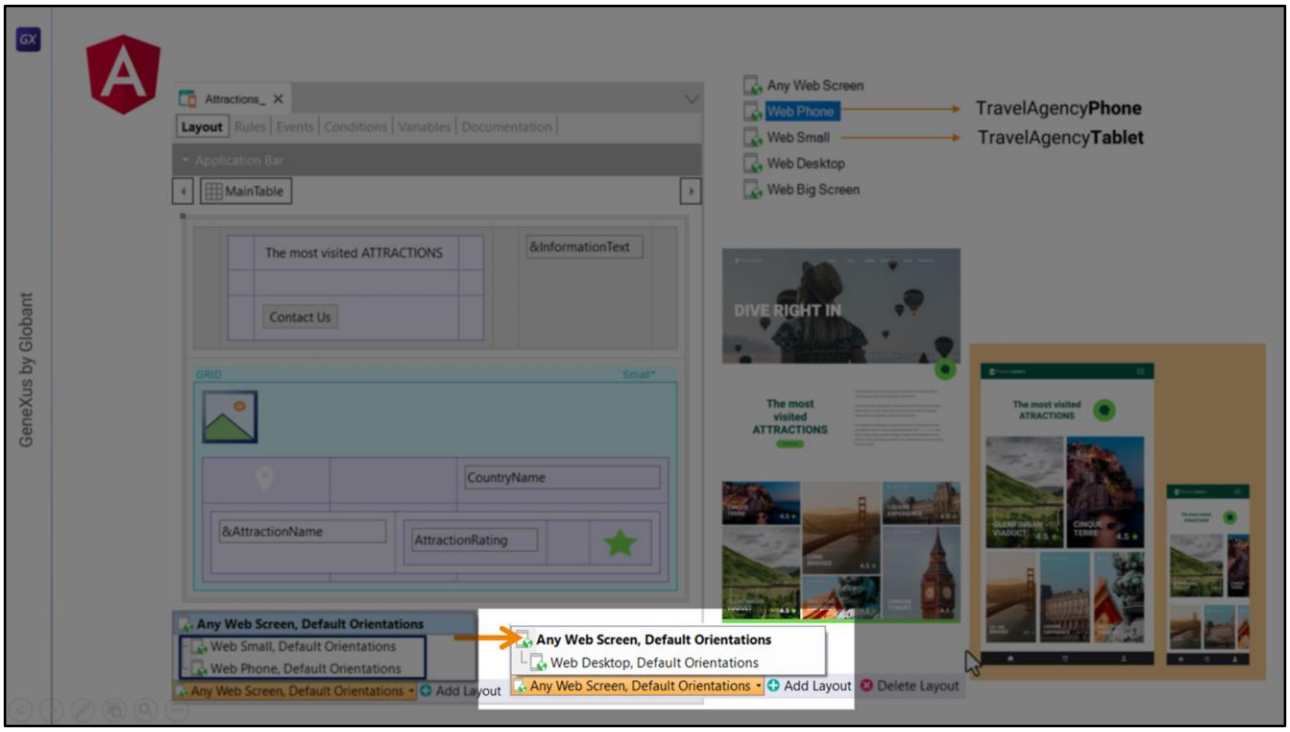


On the other hand, we don't have to define the layouts in the same way for all panels. This is panel by panel.

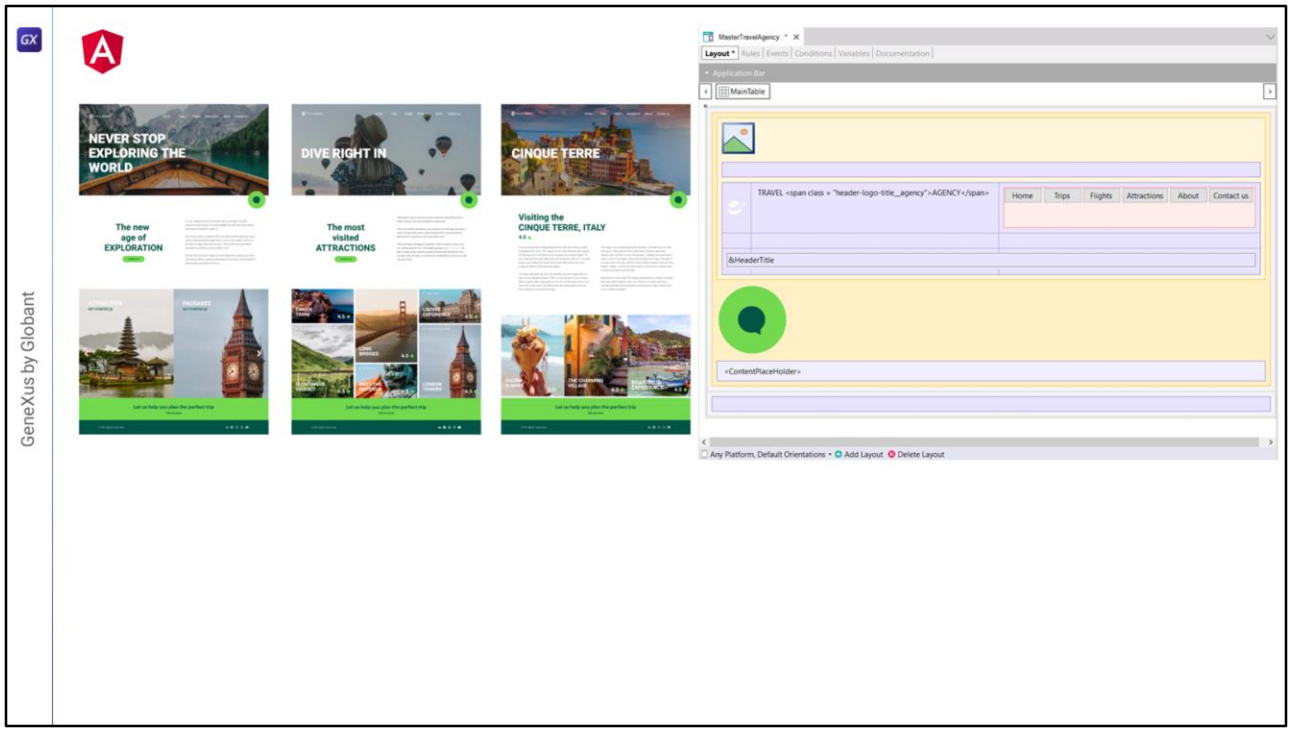
For example, note that between these two layouts, really the only difference seems to be in the font sizes and heights and widths, and nothing else. If those sizes were defined at the DSO level and not at the level of the controls in the layout, then both layouts could be the same, so we could avoid defining them twice. We could group these two into one, and have the difference determined only by the DSO associated with each platform.



But for that, we would have to indicate differently these layouts here. We should specify that this one is for Web Desktop and not for Any Web Screen; and leave the Any Web Screen to join these two. Is it clear?

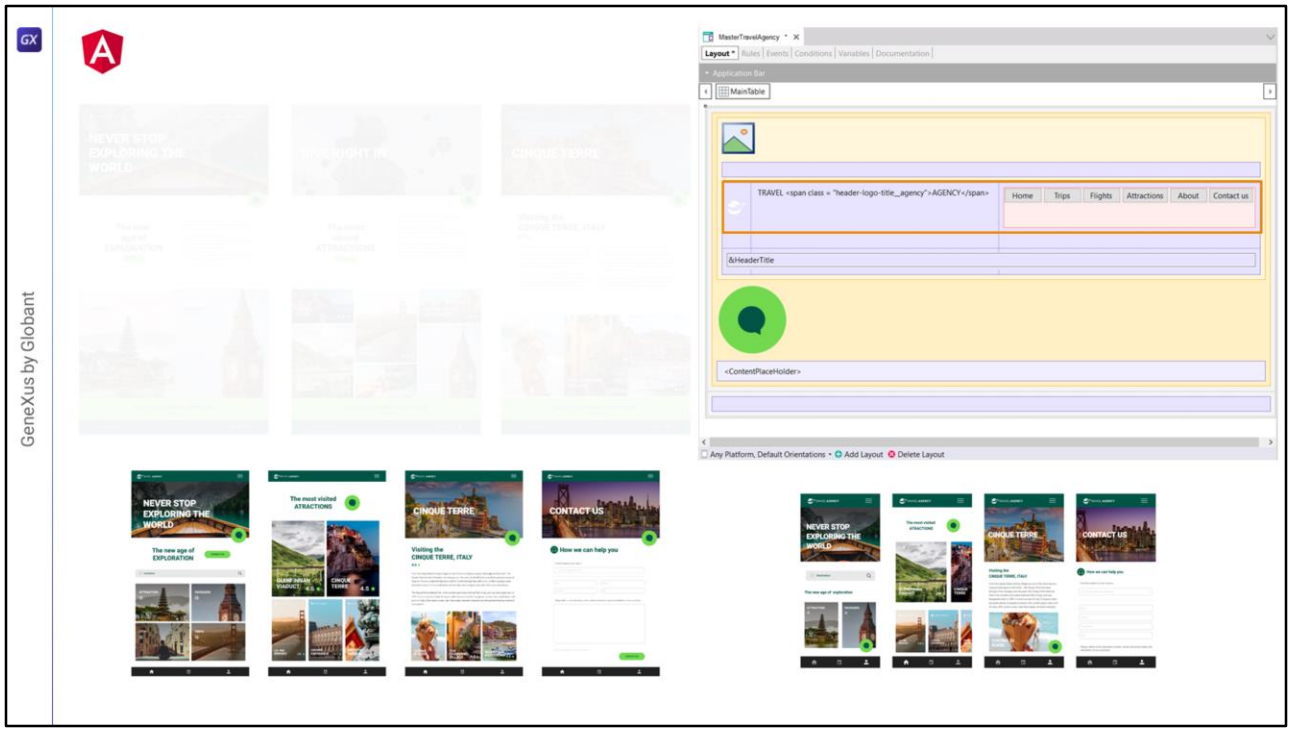


Because in that case, if we are running in the browser of a phone it will not match that of the Web Desktop, which is the most specific one defined, and then it will have to keep this other one. And the same will happen for Laptop or Tablet size up to 768 dips.

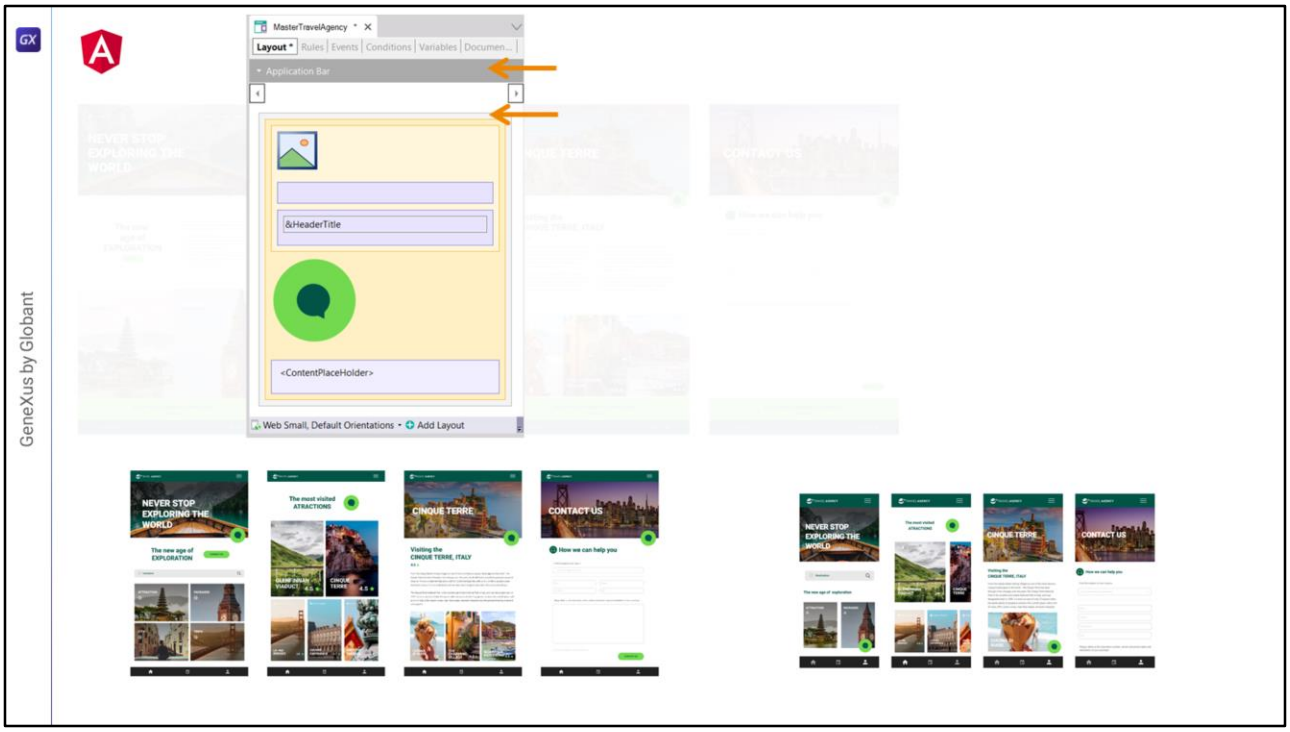


Well, now that we know all this...

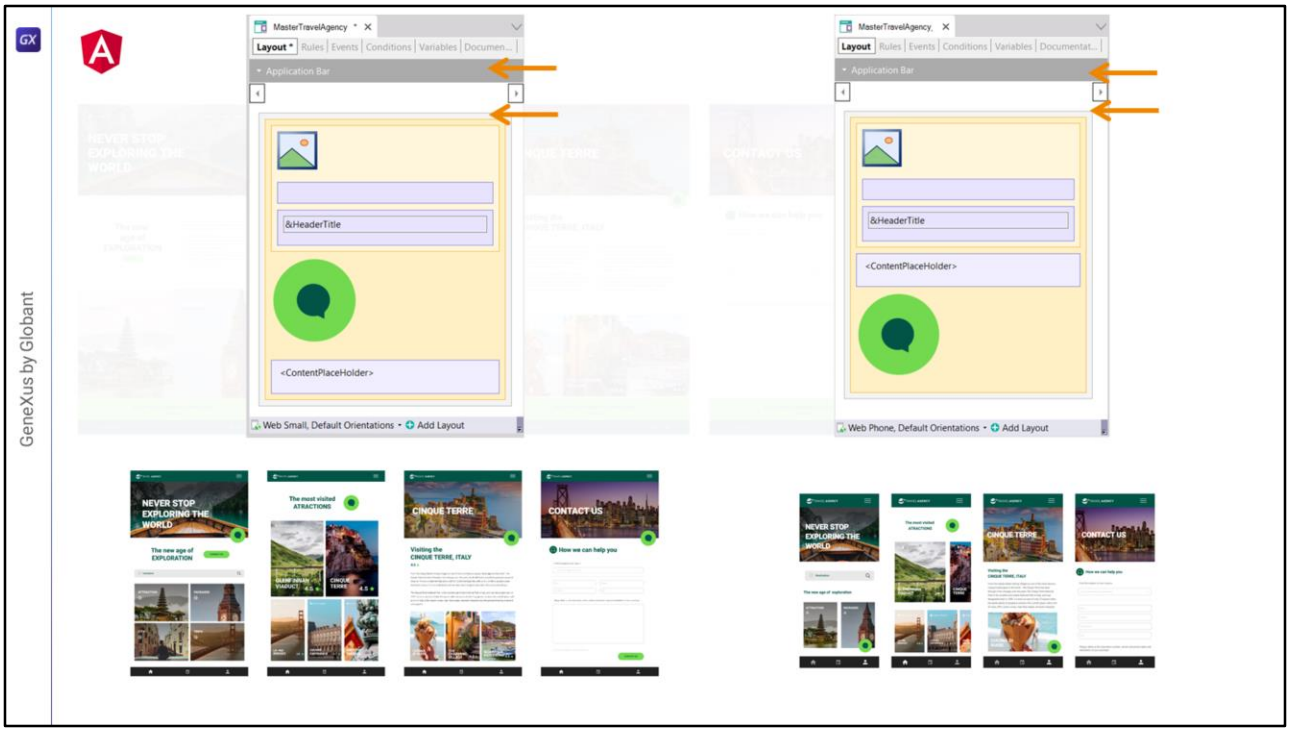
As far as the Master Panel is concerned, if here we had the Desktop implementation (only the footer is missing)...



The breakpoints for Tablet and Phone size should remove all this and implement the hamburger menu functionality.



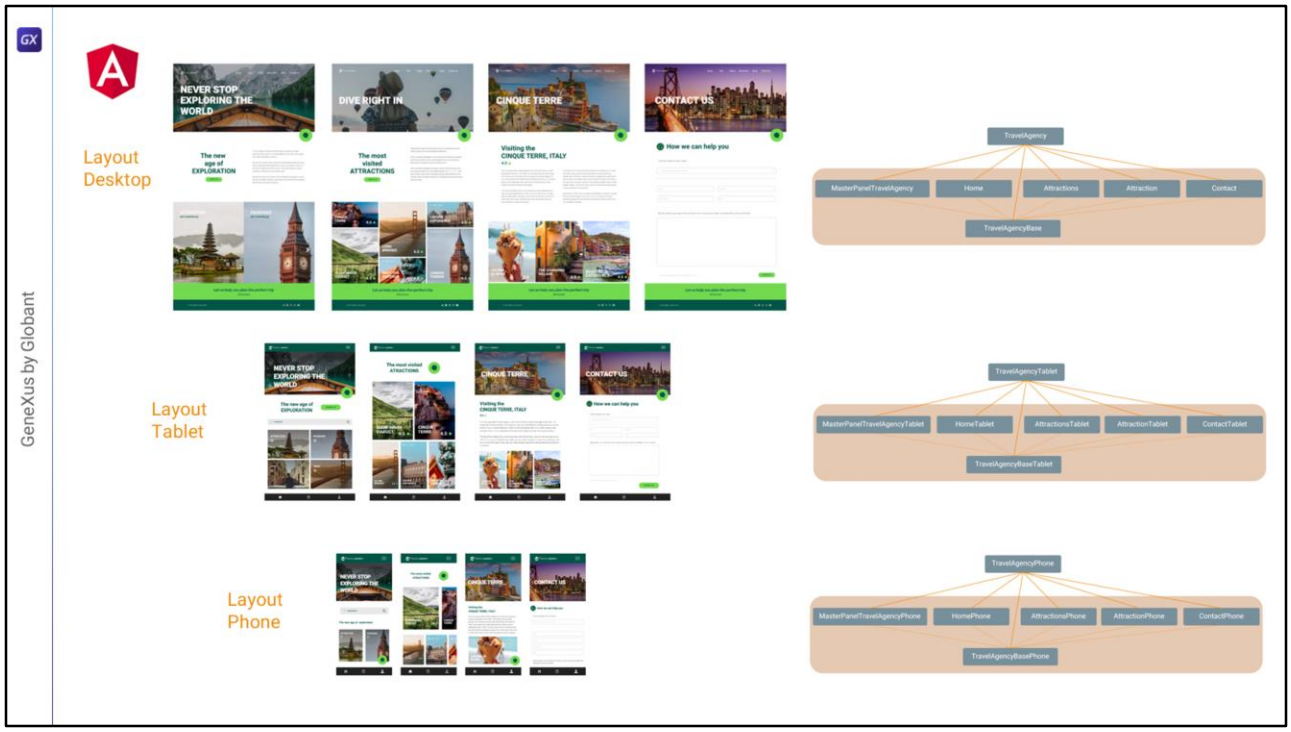
Either here or here, but also...



And we will also need to modify the Master Panel for Phone, since there the Chatbot image goes at the bottom, but only in two of the screens.

So one option would be through a particular, different layout.

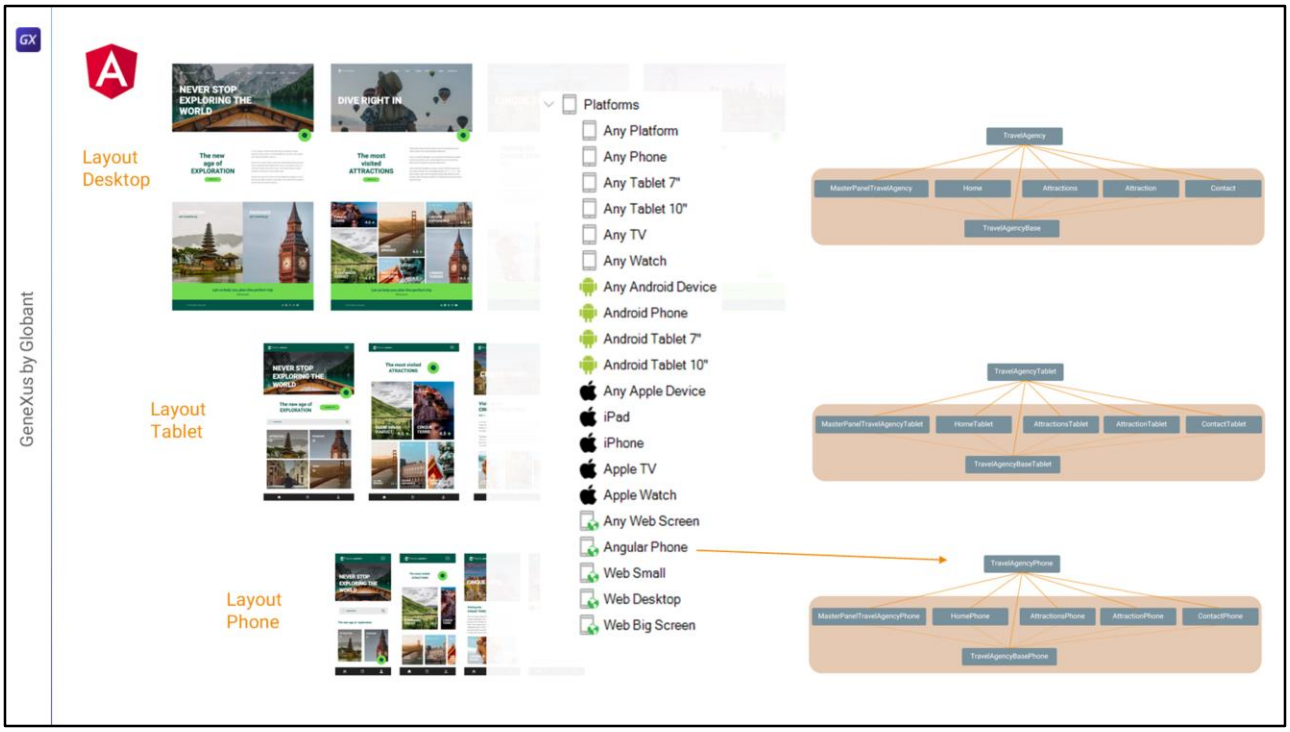
I leave as a task for you to think how to hide the Header in the Attractions panel for the Tablet size; and for the Phone size, to hide the chatbot at the bottom for the Attractions panel and for the Contact panel.



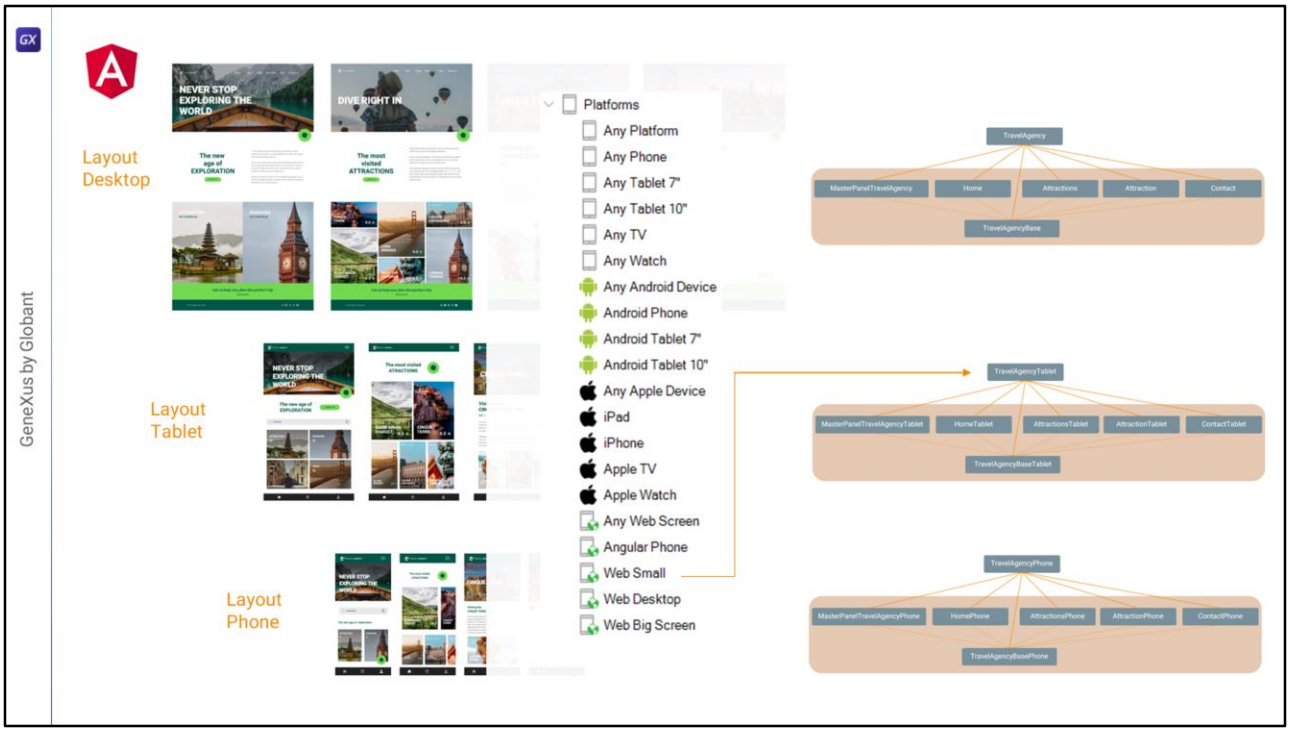
Well, let's summarize what we have to do to solve the adaptive Angular application: create layouts for Tablet and Phone sizes for the cases in which they vary, and the other thing is to **specialize** the DSO tree that we had developed for Desktop, to modify only what changes, in that new screen size, such as font sizes, spacing and some other details. This, in fact, we had started early on when we started working with the typographic classes for Desktop. Remember that there, for example, we had seen that while for Desktop the classes for card texts were identical between large and small cards, and between those of Attractions and Attraction, for Tablet and Phone, on the other hand, they varied, and therefore we had specified, already in the project preparation module, variations for these classes.

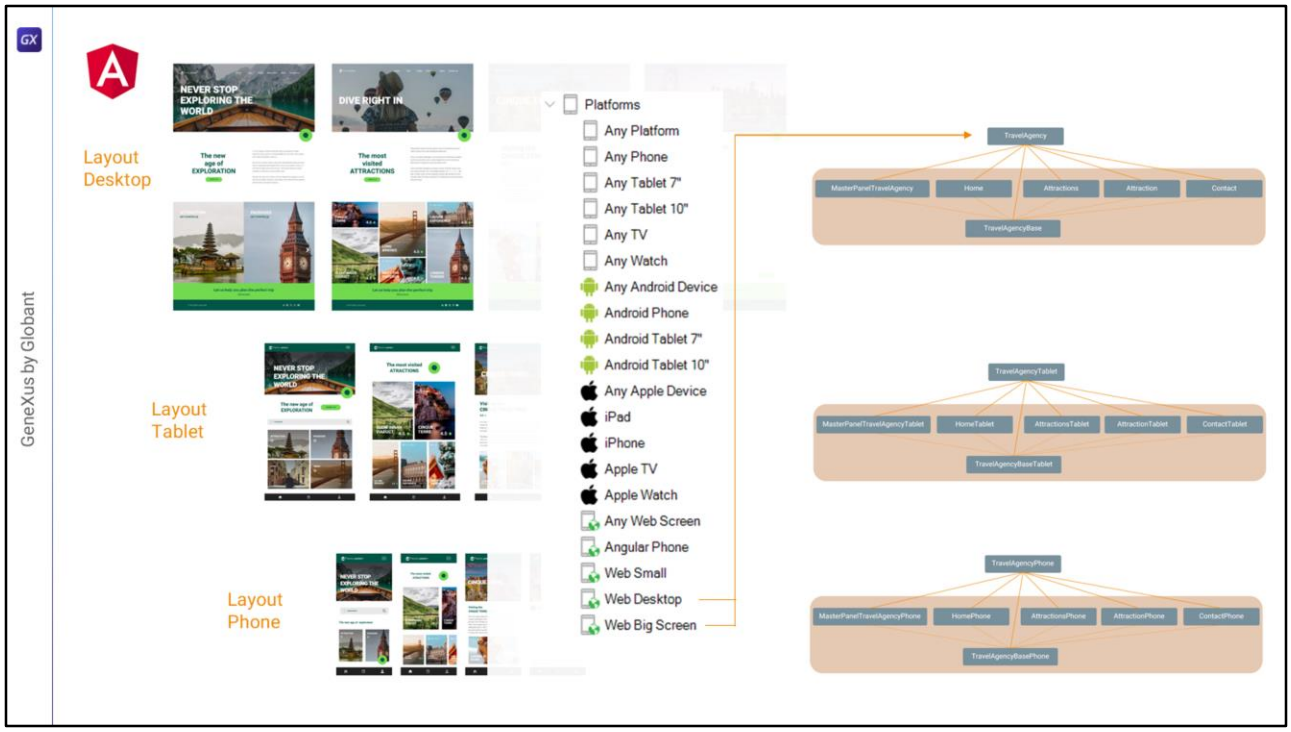
It is possible, as in this case, that we also have to modify the implementation of some other part, as in the case of the hamburger menu.

All in all, the adaptive design of the Angular solution will not be costly. We will have to adapt to the breakpoints what has already been developed for the screens of the initial size, and this work is much less cumbersome than designing each breakpoint from scratch.



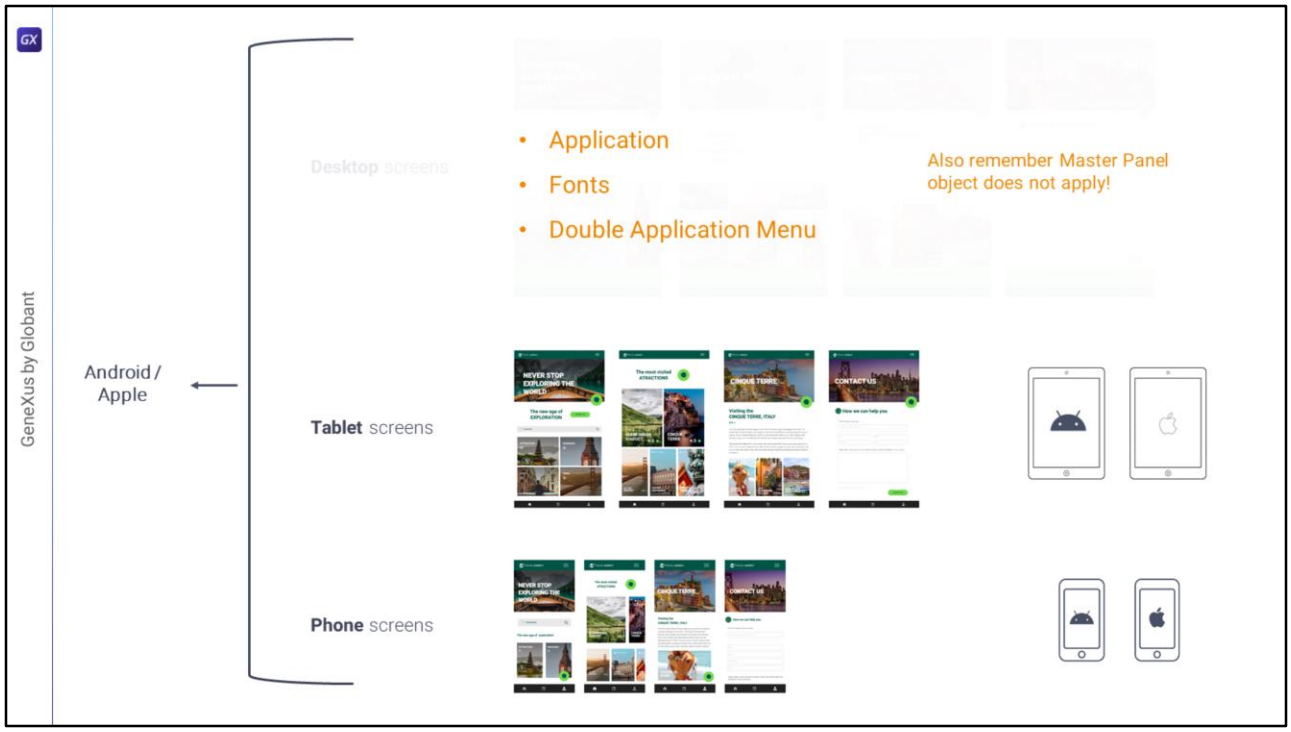
Of course, we must indicate at the Platforms level the root of the DSO tree that will correspond to each breakpoint.


GeneXus by Globant
Layout Desktop
NEVER STOP EXPLORING THE WORLD
DIVE RIGHT IN
The new age of EXPLORATION
The most visited ATTRACTIONS
Layout Tablet
Layout Phone
Platforms
Any Platform
Any Phone
Any Tablet 7"
Any Tablet 10"
Any TV
Any Watch
Any Android Device
Android Phone
Android Tablet 7"
Android Tablet 10"
Any Apple Device
iPad
iPhone
Apple TV
Apple Watch
Any Web Screen
Angular Phone
Web Small
Web Desktop
Web Big Screen
TravelAgency
MasterPanelTravelAgency
Home
Attractions
Attraction
Contact
TravelAgencyBase
TravelAgencyTablet
MasterPanelTravelAgencyTablet
HomeTablet
AttractionsTablet
AttractionTablet
ContactTablet
TravelAgencyBaseTablet
TravelAgencyPhone
MasterPanelTravelAgencyPhone
HomePhone
AttractionsPhone
AttractionPhone
ContactPhone
TravelAgencyBasePhone


GeneXus by Globant
Layout Desktop
Layout Tablet
Layout Phone
Platforms
Any Platform
Any Phone
Any Tablet 7"
Any Tablet 10"
Any TV
Any Watch
Any Android Device
Android Phone
Android Tablet 7"
Android Tablet 10"
Any Apple Device
iPad
iPhone
Apple TV
Apple Watch
Any Web Screen
Angular Phone
Web Small
Web Desktop
Web Big Screen
TravelAgency
MasterPanelTravelAgency
Home
Attractions
Attraction
Contact
TravelAgencyBase
TravelAgencyTablet
MasterPanelTravelAgencyTablet
HomeTablet
AttractionsTablet
AttractionTablet
ContactTablet
TravelAgencyBaseTablet
TravelAgencyPhone
MasterPanelTravelAgencyPhone
HomePhone
AttractionsPhone
AttractionPhone
ContactPhone
TravelAgencyBasePhone

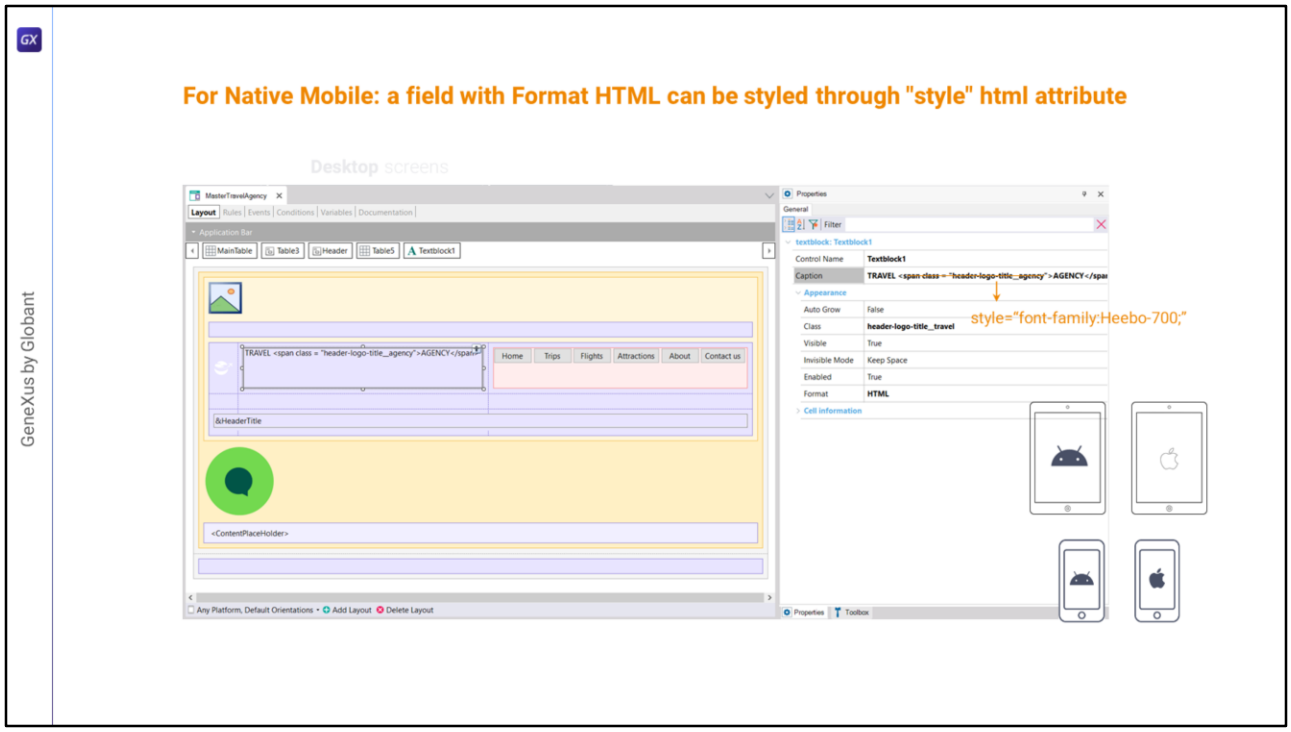
GeneXus by Globant

Native Mobile

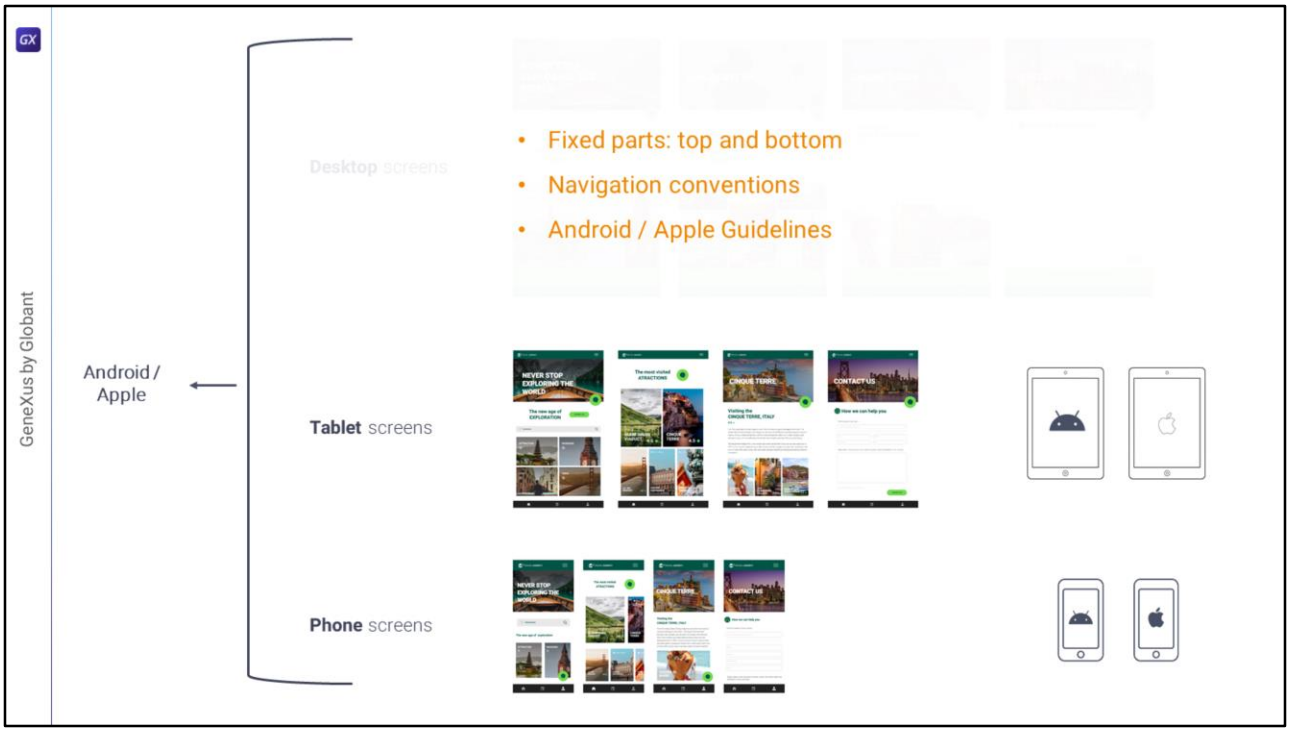


What if we only wanted the native application, and we were not interested in developing it for Angular?

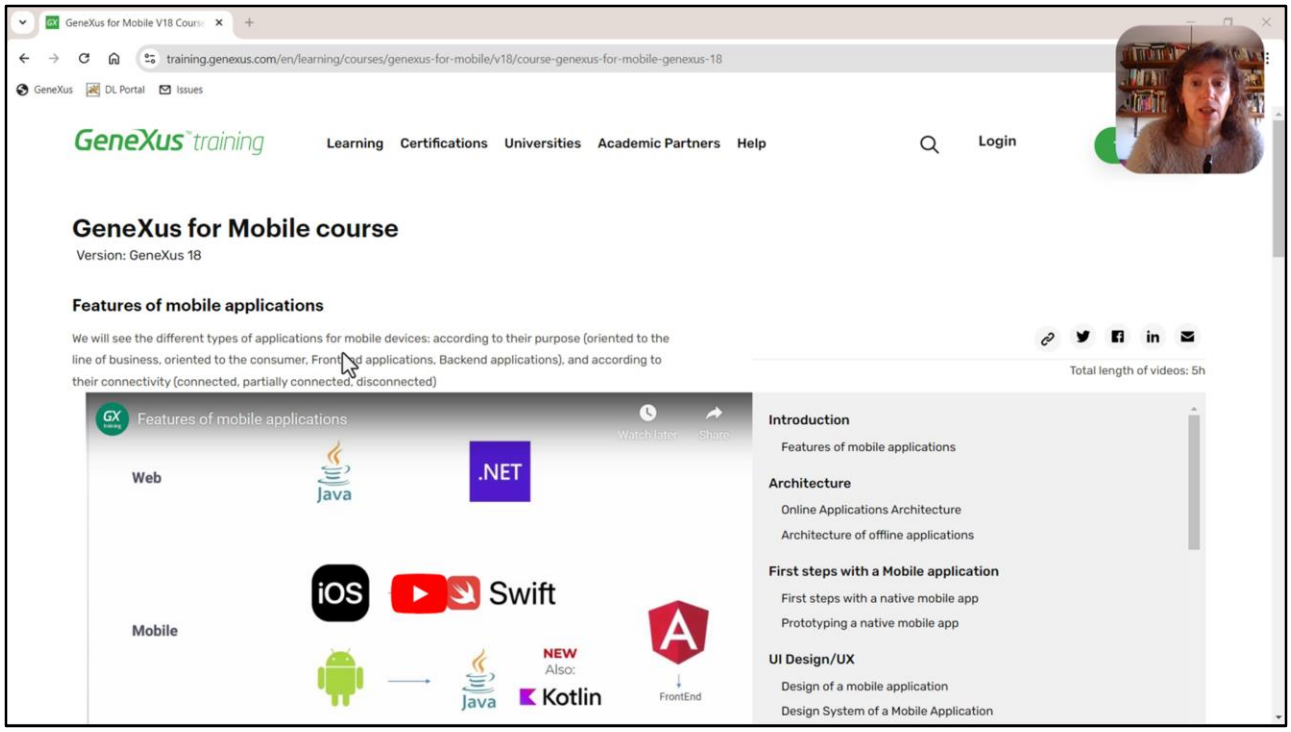
It wasn't exactly what we did in the previous videos, where I focused rather on a kind of comparison between Angular and native, although I also told you about aspects that are particular, special to native, as for example when I talked about the colors that are defined at the Application class level, or the use of fonts, or the impossibility of having a double menu for the application, at least for the moment.



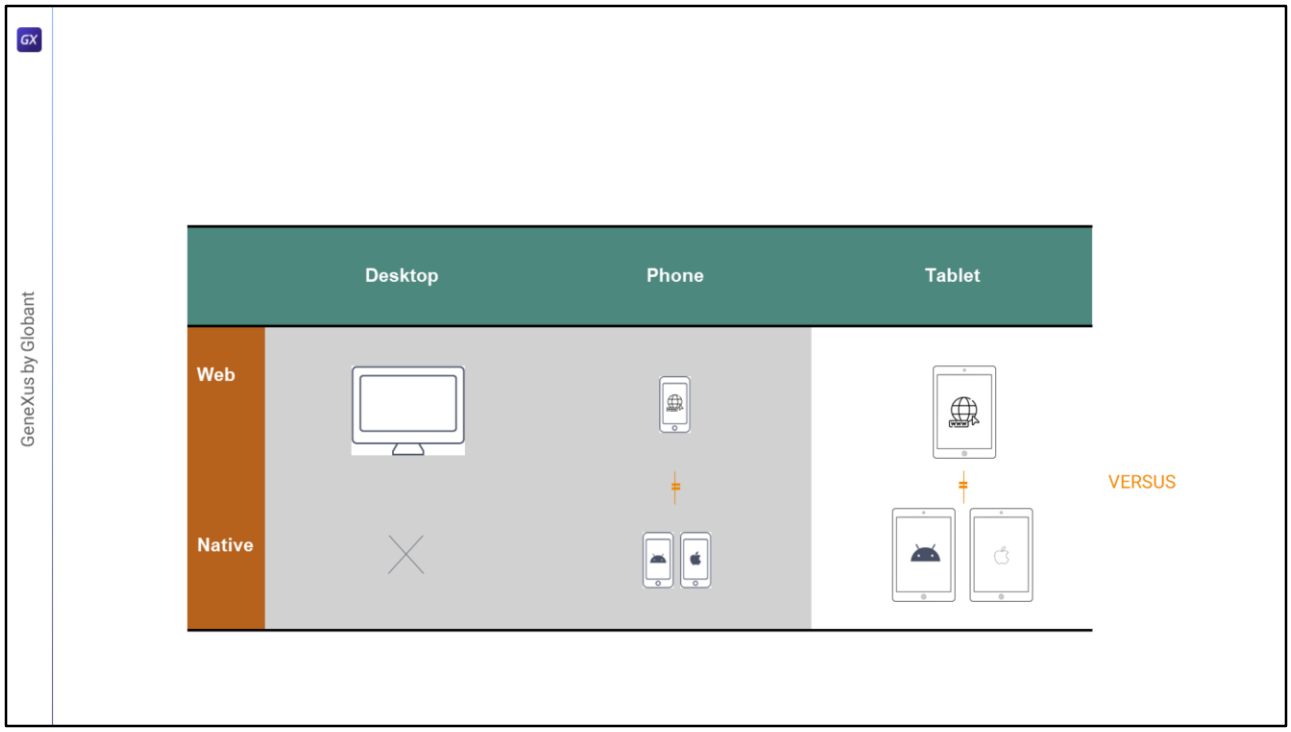
One difference, in the comparison we made, that I didn't mention there but that we saw when we changed the font weight of the word Agency for the Angular Desktop application was that we can use the inline **style** to do this same thing in Android or Apple, if we set HTML formatting for the textblock, remember? Although it doesn't seem to make sense to talk about HTML for native, while the class attribute will not be taken into account, the style will be, which allows us to do things like this.



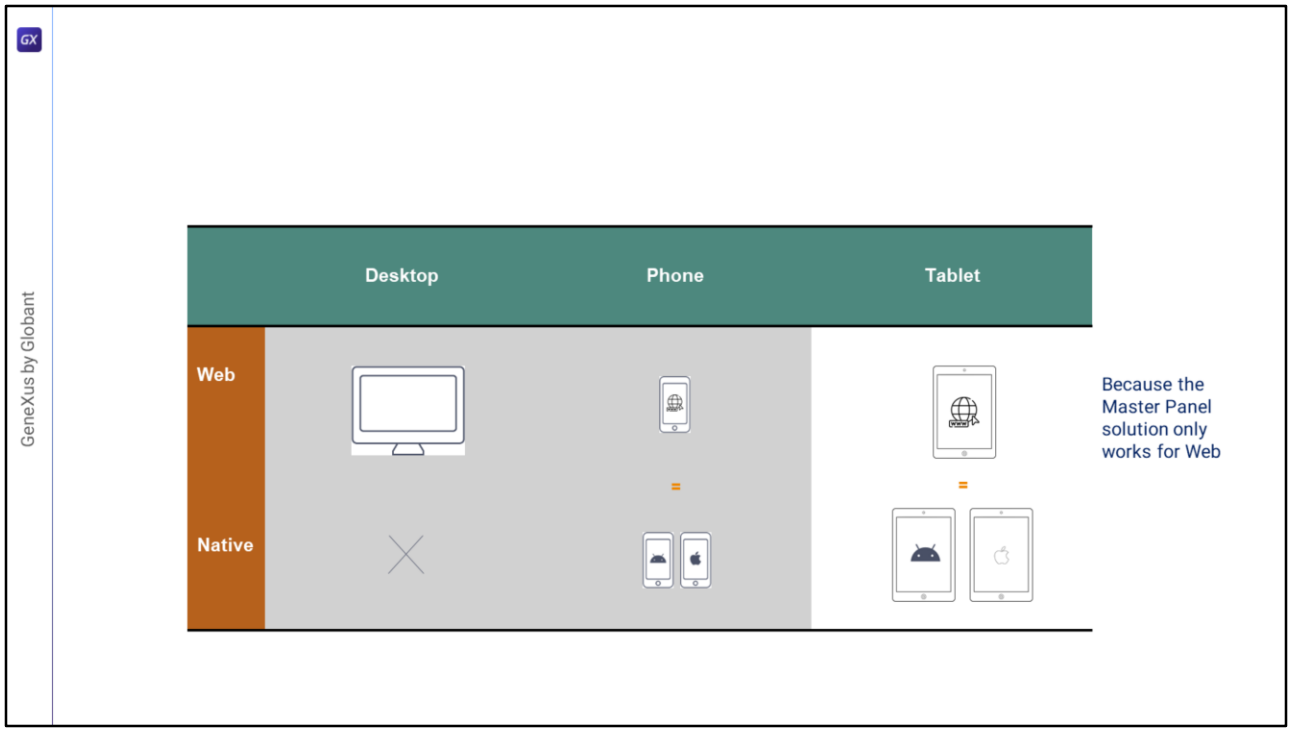
The other characteristic I mentioned about native screens is that, in general, elements are fixed at the top and bottom edges of the screen, and the scrollable area is usually the middle one. But there are also conventions regarding navigations, and ways to return to previous screens, which are specific to operating systems, and that every application must respect. Following Android and Apple design guidelines is very important.



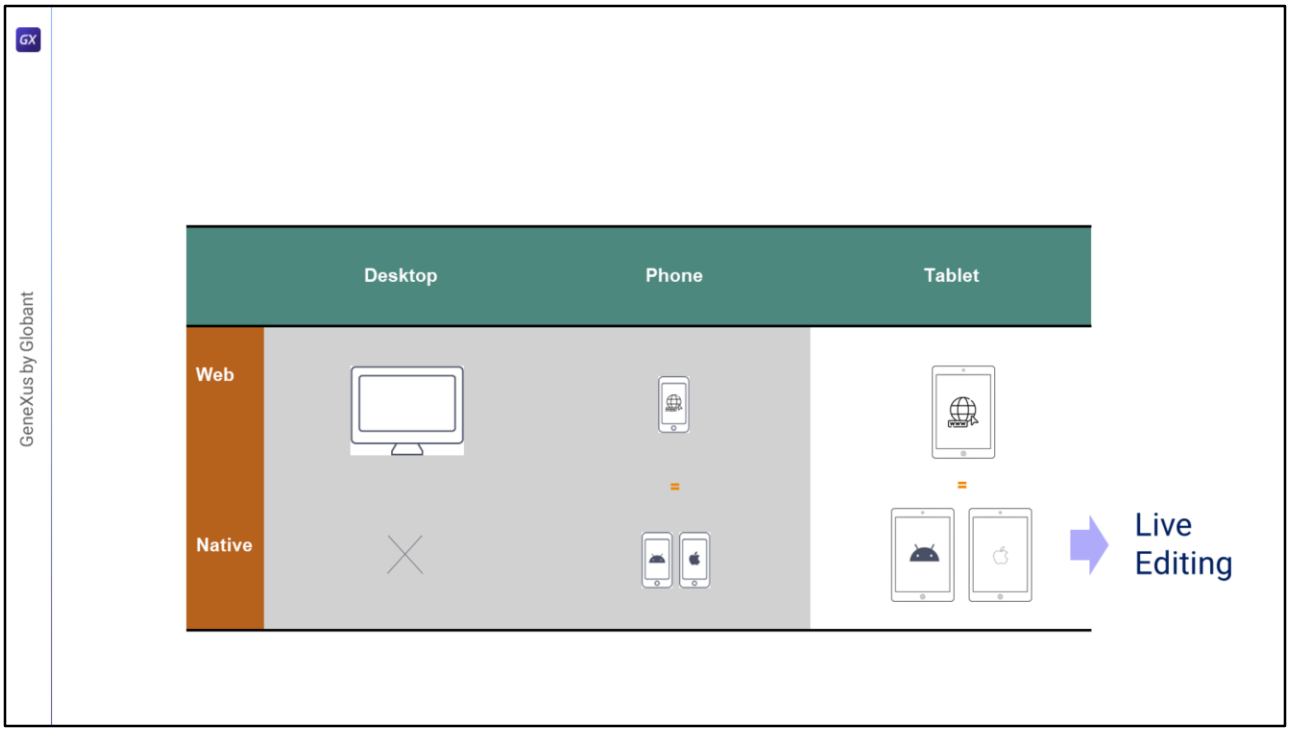
All of this would deserve an entire course, part of which already exists, and that is the Mobile course that I have already mentioned, together with additional material that you can find there.



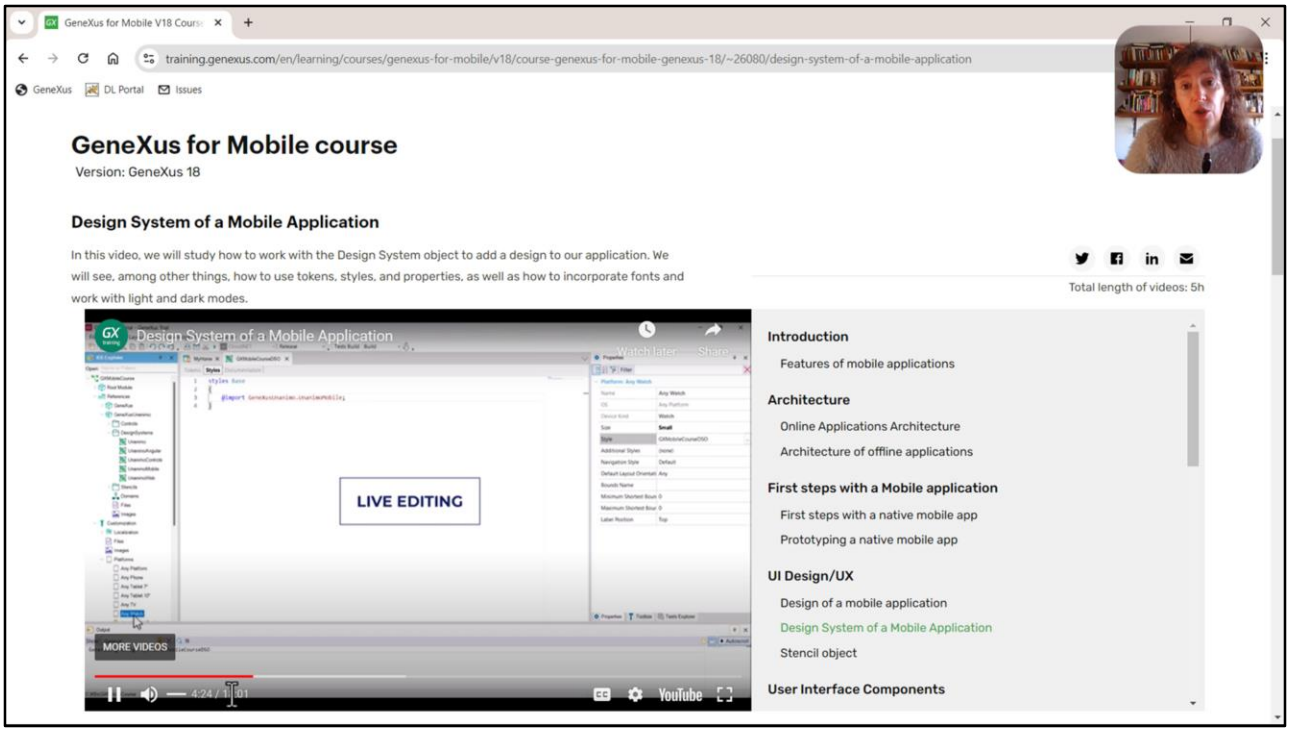
In the previous videos of this module that is coming to an end I focused, as I said, on showing the differences, but I feel that I didn't adequately emphasize everything that is the same, which is a lot. For example, the controls are basically the same, with their properties and their values, and also many of the properties that we associate with the classes are going to be the same (the properties and their values).



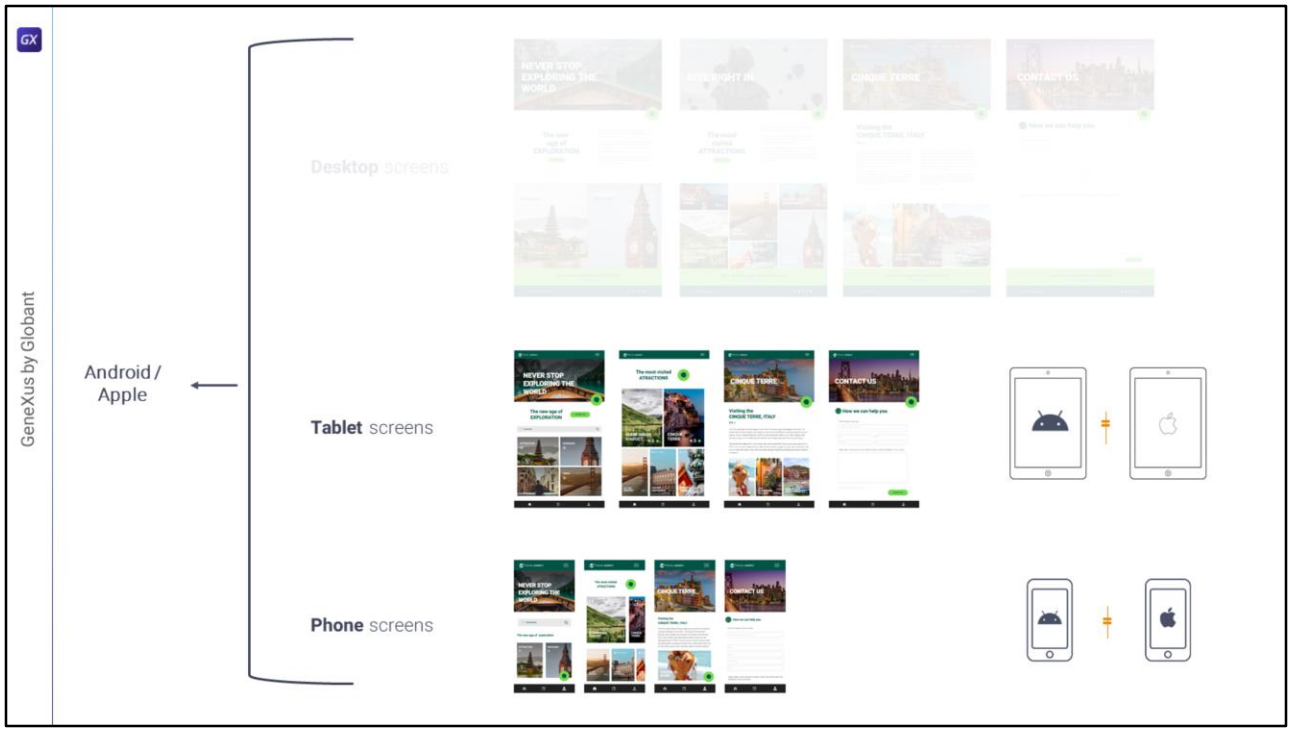
We could have started the course the other way around. We could have started by designing the native application for phone, and then study everything we studied in modules 1 to 4, but for this application. For phone and for tablet. And then move on to the Web. We would have gone through more or less the same stations, except for the global events and the Master Panel.



We would have also seen the Live Editing tool, which in the previous videos I didn't bother to show you, because we weren't prototyping for native. But well, the Live Editing tool is very important for making changes and seeing them instantly without having to compile and run; and not even save.



If you take the Mobile for GeneXus course you will see it clearly.

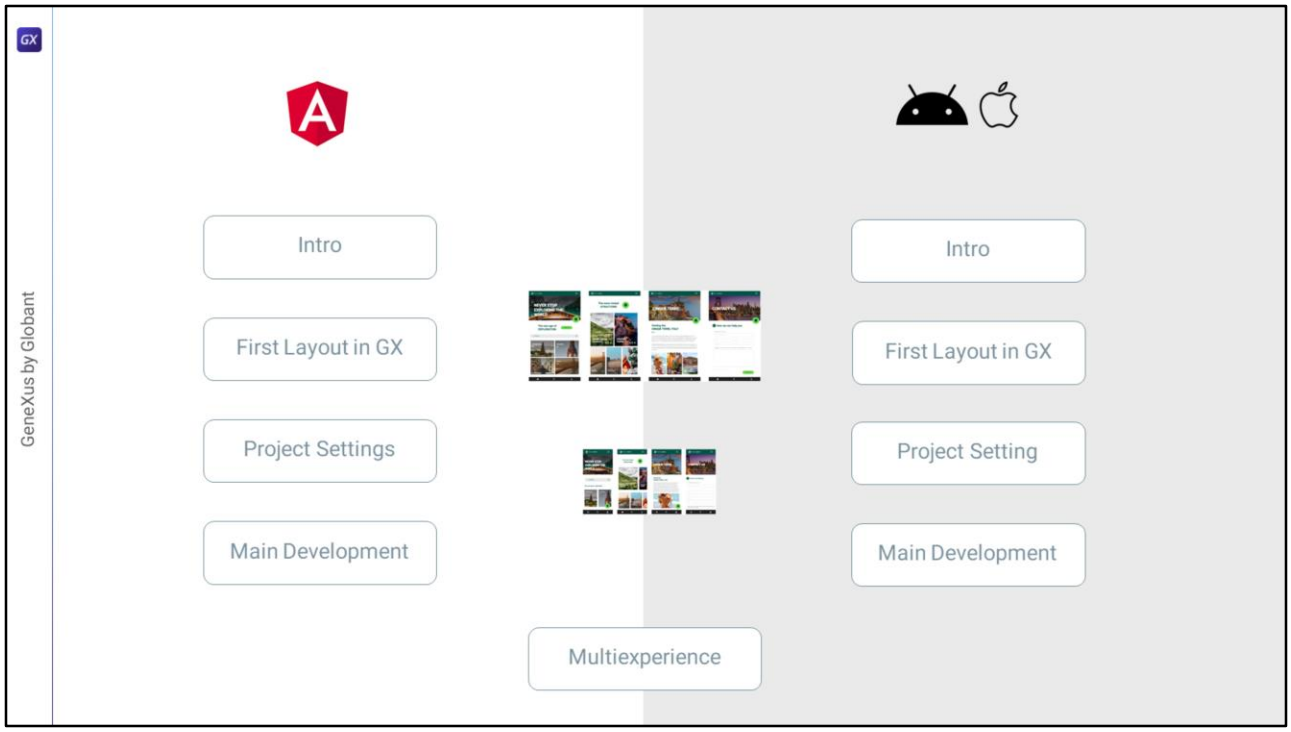


Moreover, in the first video of this module I wasn't very precise, let's say, in matching the Android application and the Apple application. Why is that? To keep it simple, really, because they have more things in common than differences. But clearly there may be specific aspects that we must consider, that belong to each platform, or even to the design guidelines, that may justify differentiating, then, the layouts by platform.

And well, of course this is possible to achieve, we have to try to focus first on everything they have in common, and then, when we have to deal with the differences, we can differentiate precisely, in order to start from a common base.

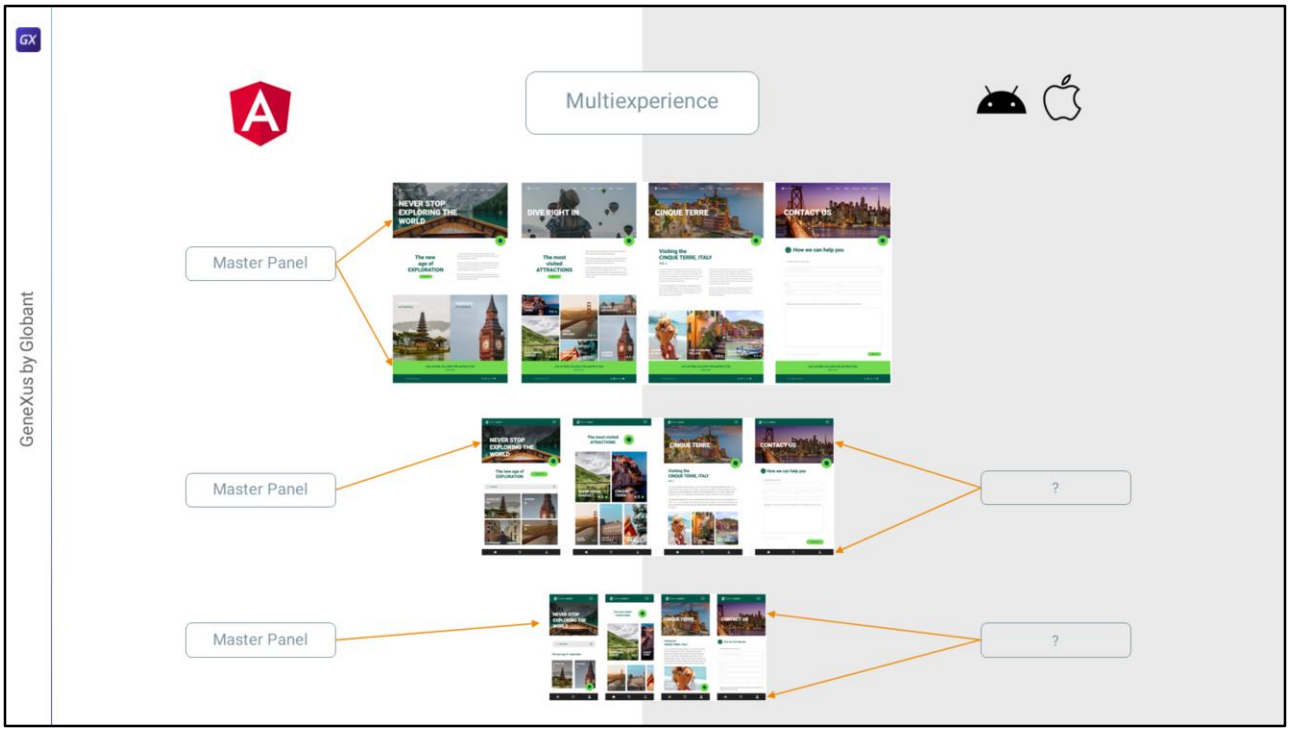
GeneXus by Globant

MUX: Angular & Native Mobile

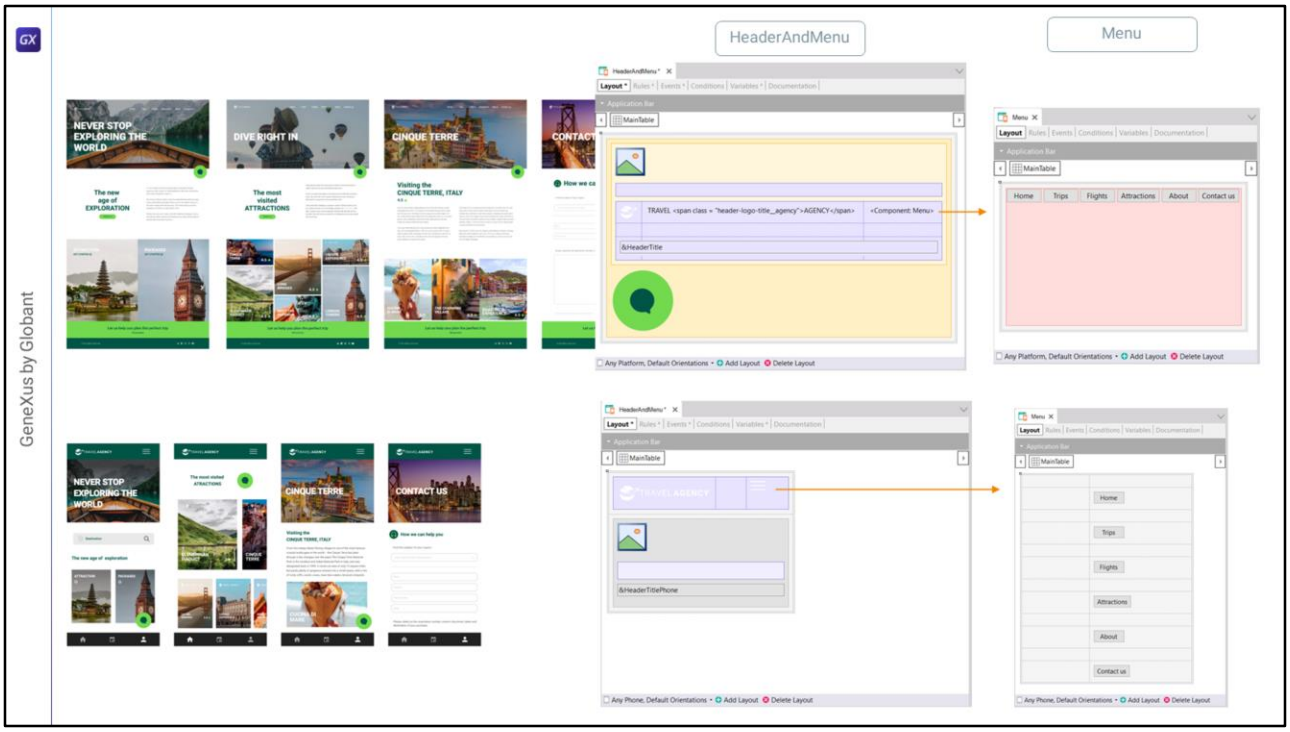


In short, we would have needed to replicate a significant part of modules 1 to 4 for native, and then add a last module that would be the synthesis of both paradigms.

So, if we started the application from scratch now, considering everything at the same time, I have some ideas of how I would organize things, and I want to share them with you, so that we can think about them beyond this course.



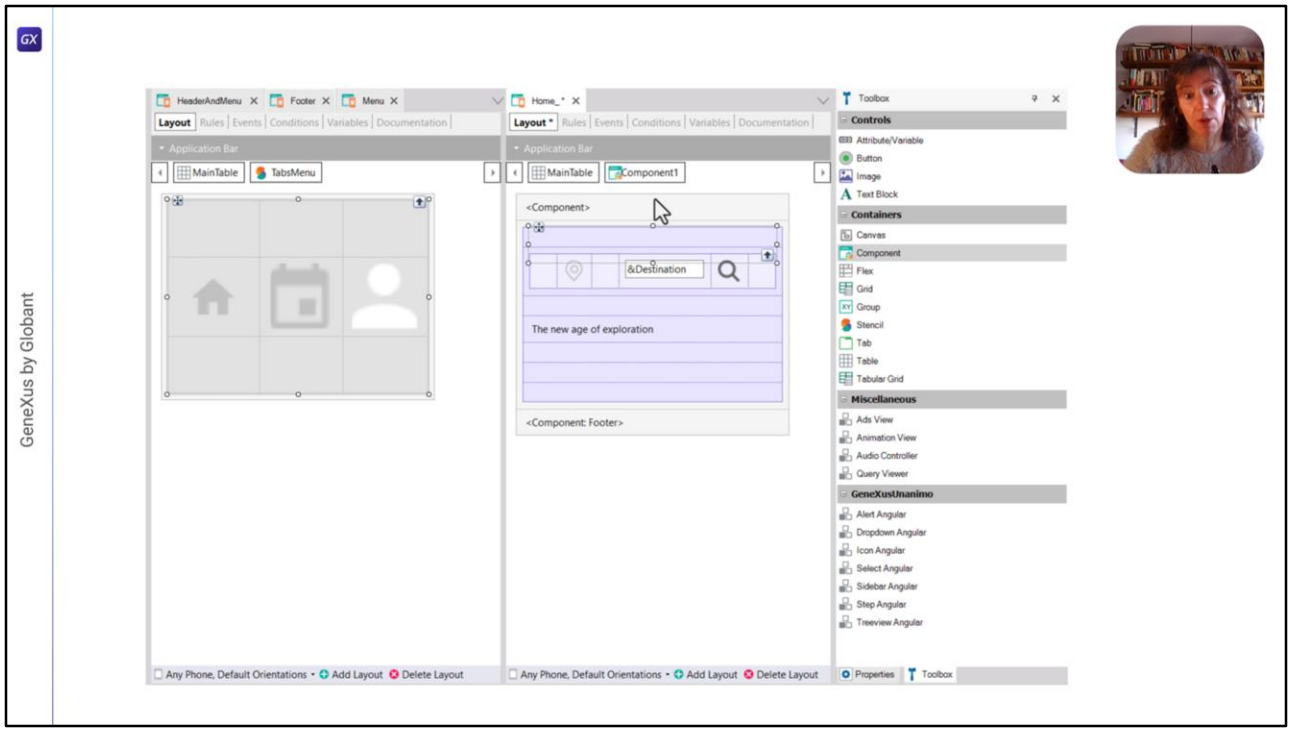
In order to use the most similar solution possible for Angular and for native, knowing that the native application will not be able to use the Master Panel, I can think of several ideas, but I'm going to share one with you.



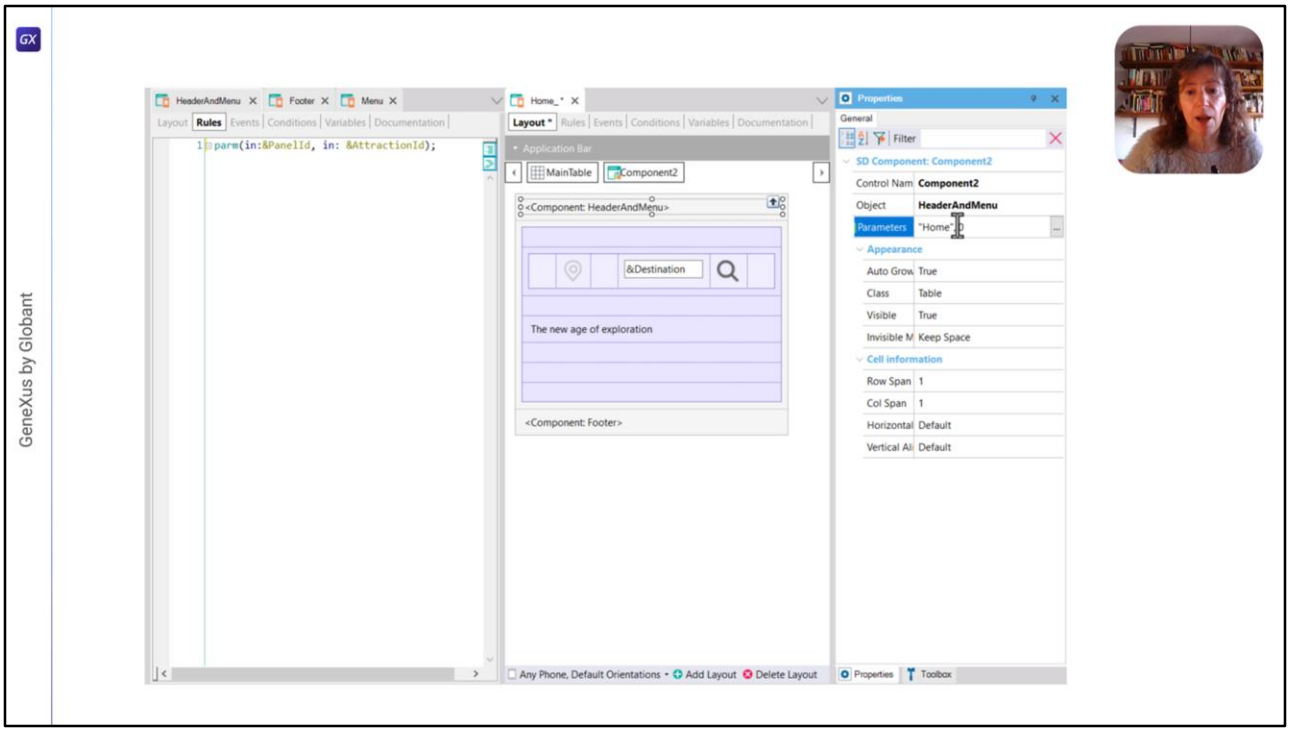
Implement the Application Bar and Header in a separate panel...

...and insert it as a **component** control in the 4 panels. Do the same with the tab bar (it will be convenient to use a component instead of a stencil to avoid having to repeat the coding of the tap events of each image).

All this is to use exactly the same solution for the native application as for the Angular application.

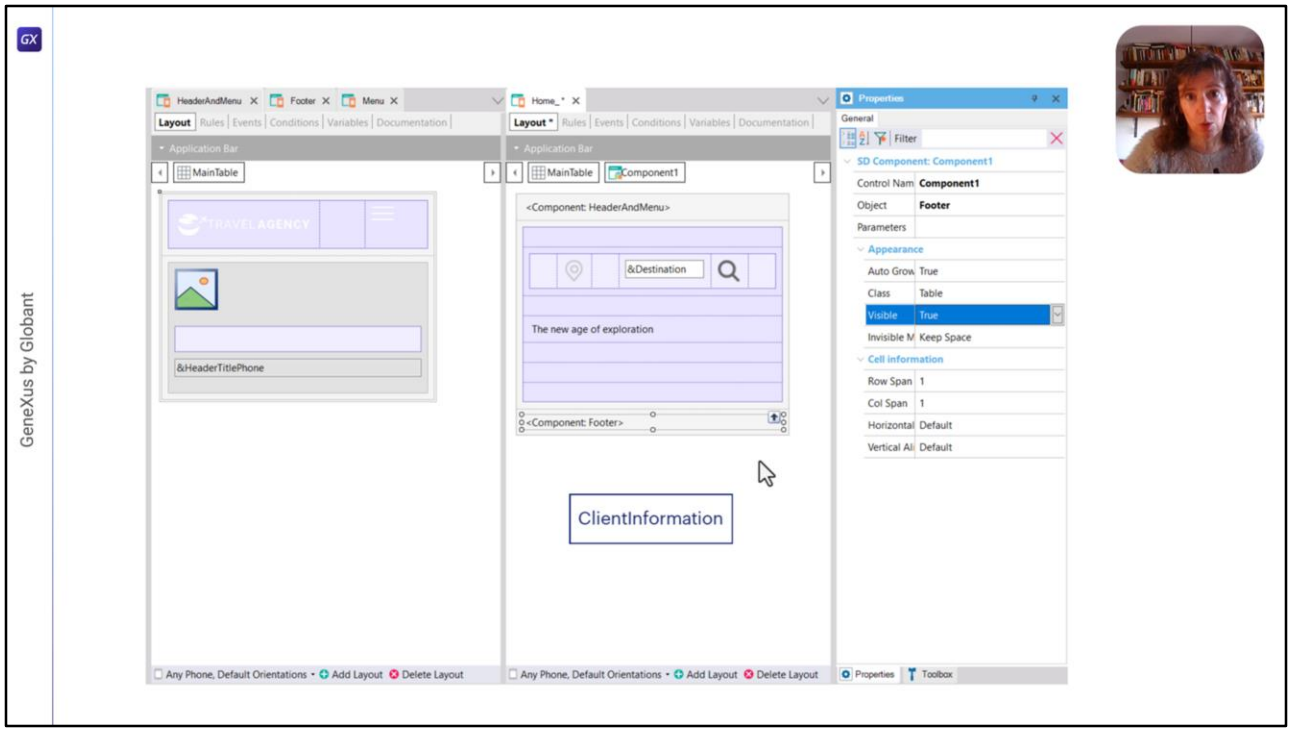


We didn't see it in this course, but when inserting a component control...



...we can pass parameters to it (which we couldn't do with the Master Panel and that's why we had to use the global events). So for this solution we won't need the global events, which are valid, both for native and Angular, on the other hand.

Each panel that is loading will dynamically load the HeaderAndMenu component, passing it an identifier of itself. And so the component will know what it should load for the image and for the title. And in the case of Attractions, it will not even have to load them.



And as for the tabs menu, we can load the component only if the application is running on an Android or Apple device (using, as we already know, the ClientInformation external object to know, precisely, on which platform the application is running).

It seems simple and that it would work. We would have to test this solution to check if any obstacles appear, but... unfortunately this course has come to an end. Goodbye.


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