



GeneXus by Globant

**GeneXus**<sup>™</sup>  
by Globant

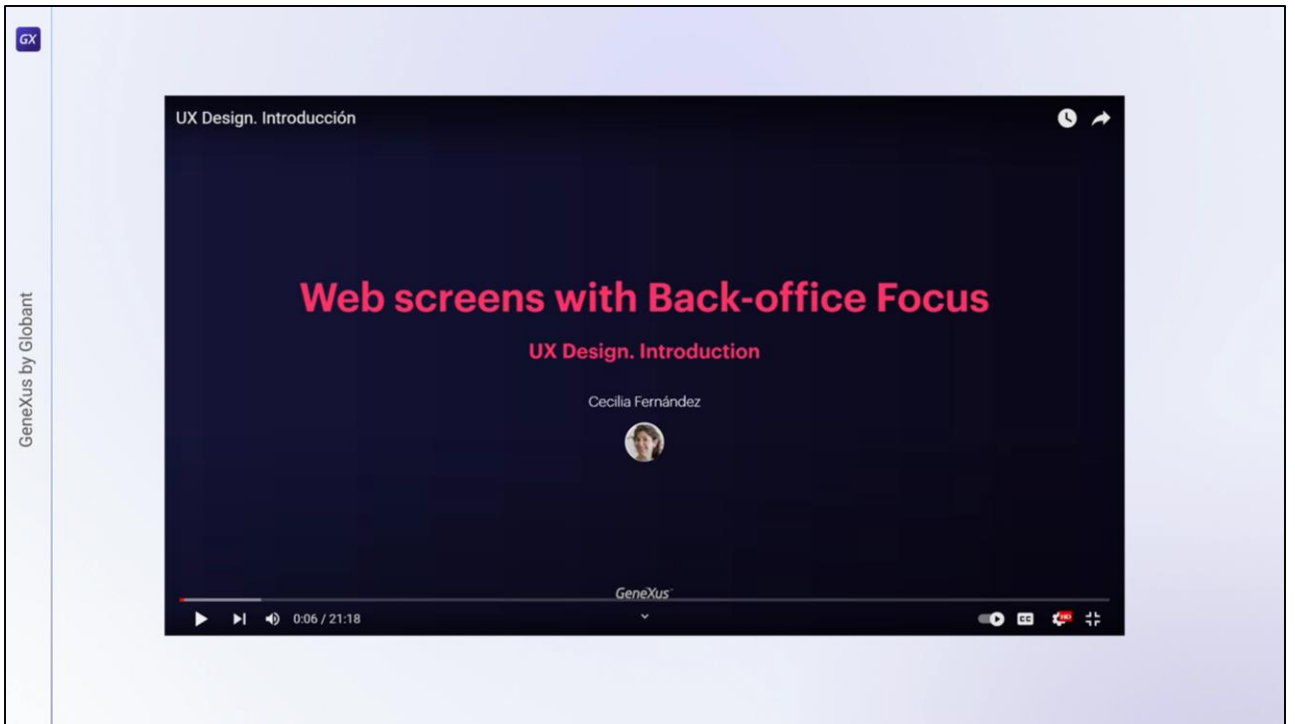
[training.genexus.com](https://training.genexus.com)

# UI customization

## Introduction

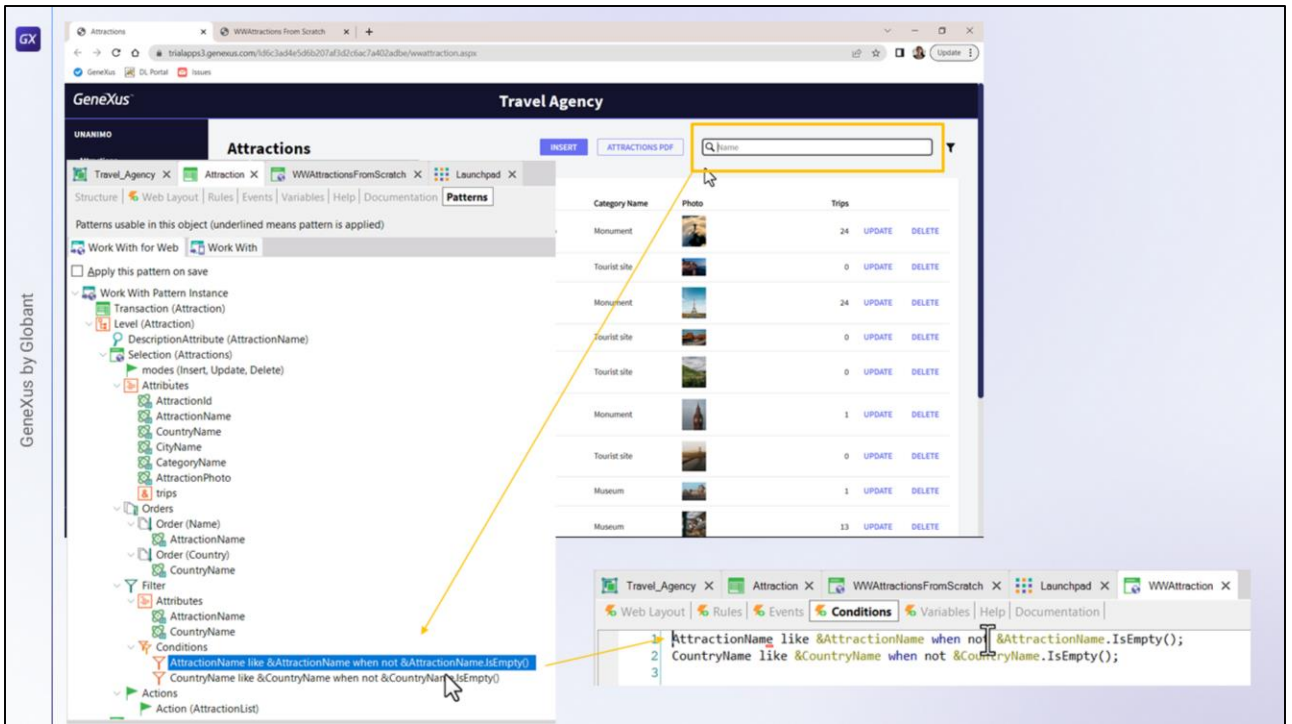


Cecilia Fernández



In this other video we had started to customize the default Design System that was created with the KB and used Unanimoweb.

Here we will delete what we had done and make more interesting customizations.



We will start by looking at the differences between the WorkWith panel that GeneXus created and our first imitation of that panel, the one we had created from scratch.

The WorkWith filtered the attractions with this field using the like operator. It didn't implement the filter in the grid conditions but in the general conditions of the Web panel.

GeneXus Travel Agency

UNANIMO

Attractions

Countries

Country Id

Attraction Name From

Attraction Name To

Grid1's Conditions

```
CountryId = &CountryId
when not &CountryId.IsEmpty();

AttractionName >= &AttractionNameFrom
when not &AttractionNameFrom.IsEmpty();

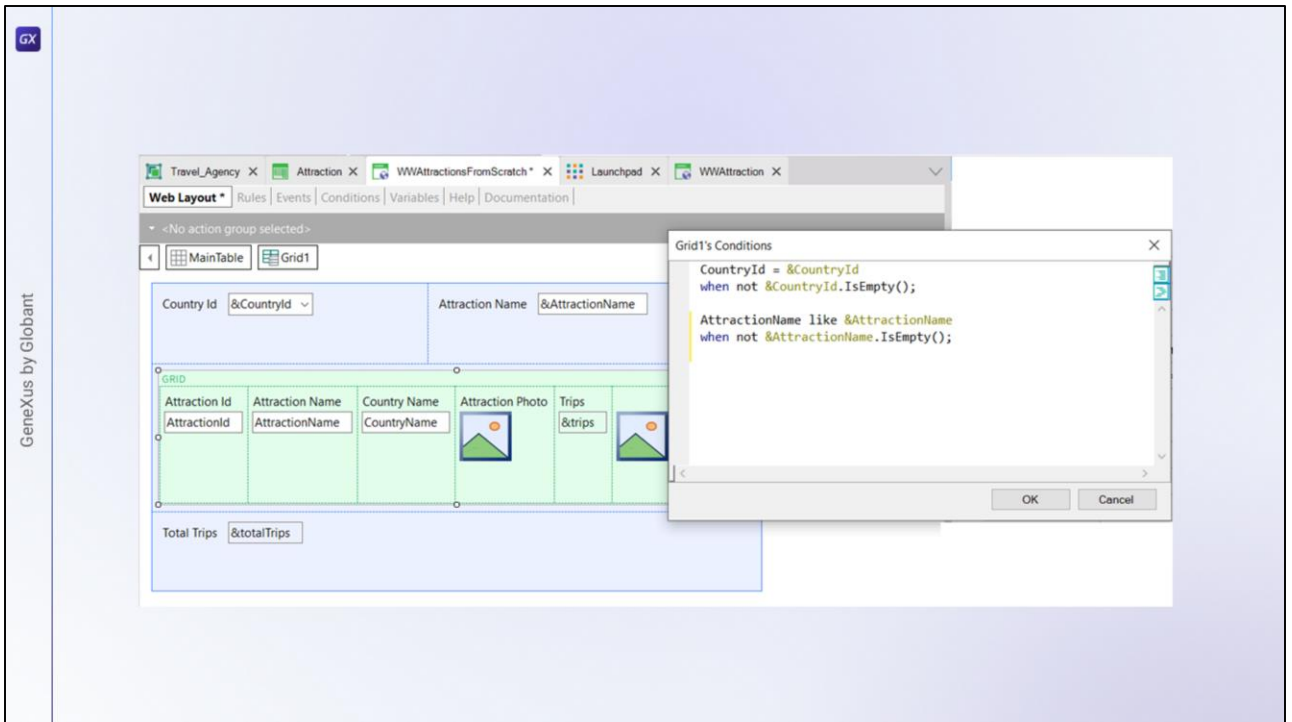
AttractionName <= &AttractionNameTo
when not &AttractionNameTo.IsEmpty();
```

Attraction Id	Attraction Name	Country Name	Attraction Photo	Trips
AttractionId	AttractionName	CountryName		&trips

Total Trips

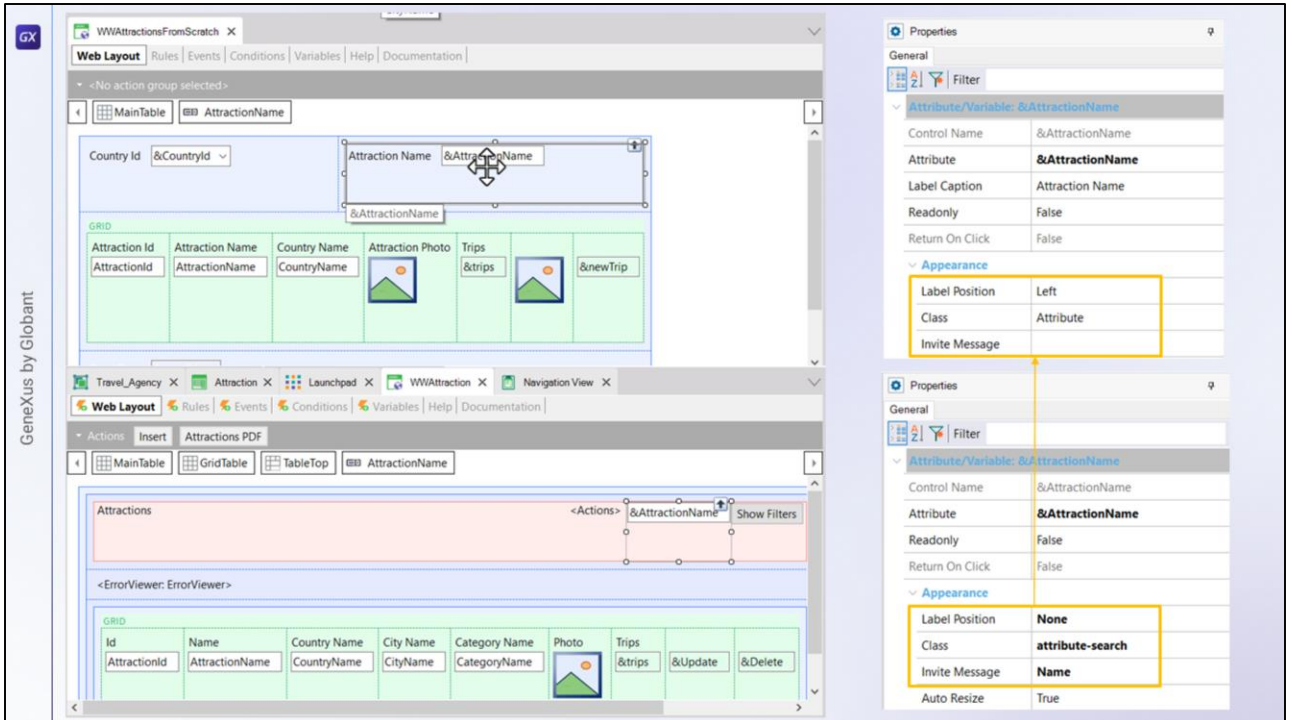
109 Musée Museum France 13 New Trip

We, on the other hand, filtered according to a range and that's why we had these two variables.



We are going to create an AttractionName variable to replace one of the range variables: we change this one for the new one; delete the other one; move this one to the right of the country filter; delete this table. Now we modify the conditions of the grid filter to use the same like operator as in the WorkWith. Let's try it.

That's fine, but our field looks less attractive than the one in the WorkWith. Why is it different?

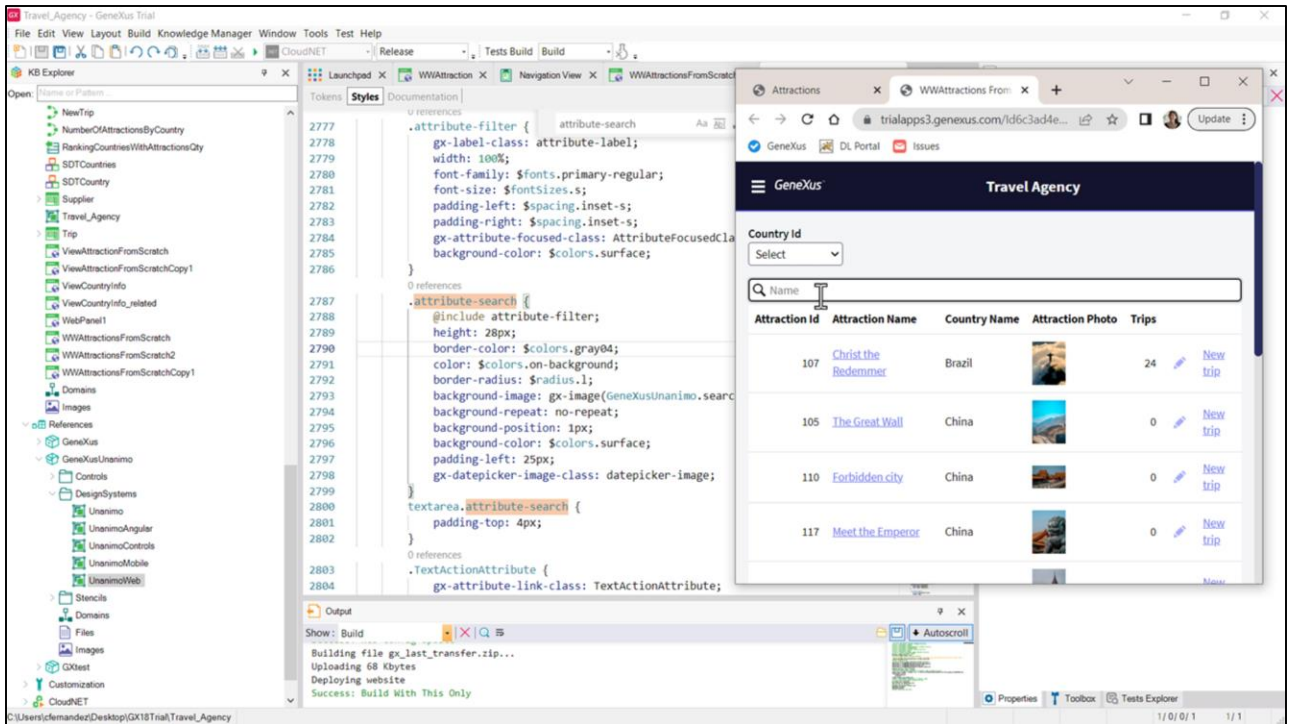


Let's compare the properties of both fields...

Here we see those of our field and here those of the WorkWith.

Let's copy them to ours... this is the text that appears in gray inside the field before the user types anything. We no longer need the label. And here is the most important one, the class. Let's associate this very specific class with it; presumably, it is the one that controls its appearance.

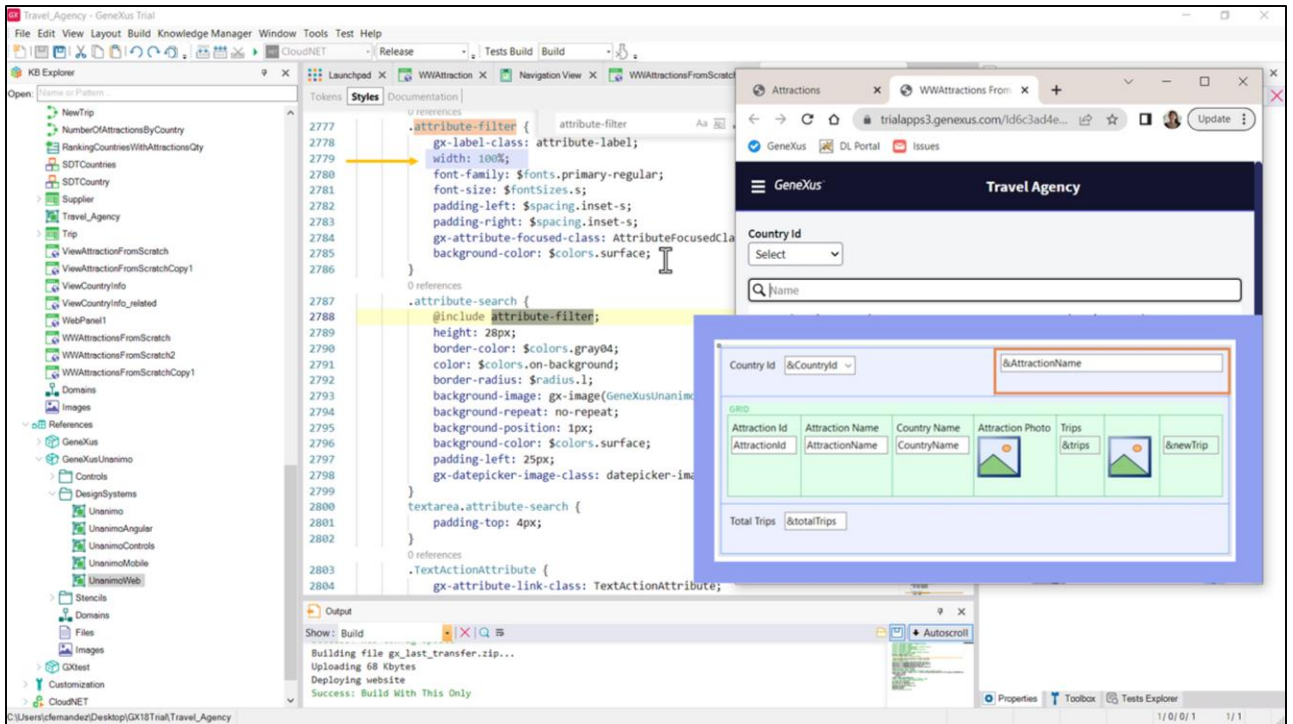
Let's try it, and look at the class properties.



The Design System Object that controls the design is this one, which is importing UnanimousWeb... and there we look for the name of the class. Here it is. It has all these properties.

For example, the one that controls the border color, the rounded tips of the border, the search image, the background color of the field... but also note that it includes another class, that is, its properties.





Among others, we can see those related to the font (its family and size) and, for example, the field width, which is 100% in relation to its container, which in this case is the cell of the table where our field is located.

Attractions | WWAttractions From Scratch

trialapps3.genevus.com/d6c3ad4e5d6b207af3d2c6ac7a402adbe/wwattractionsfromscratch.aspx

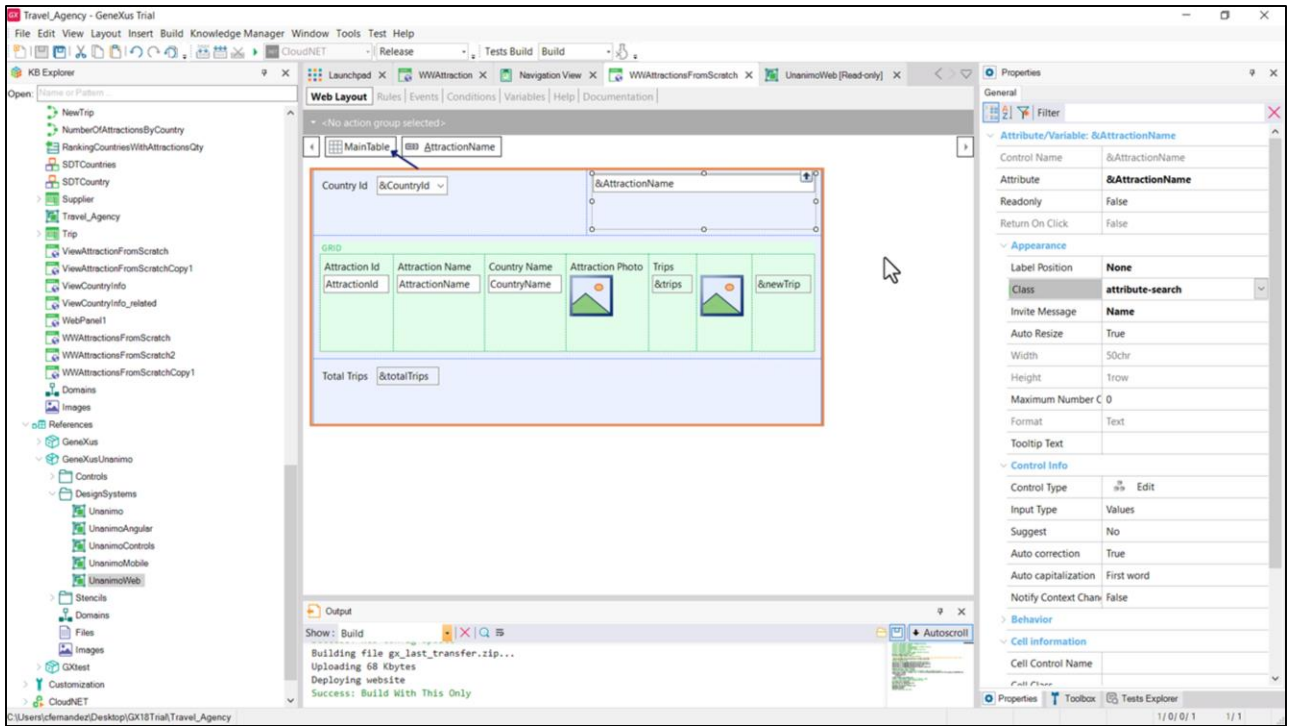
GeneXus | DL Portal | Issues

# GeneXus Travel Agency

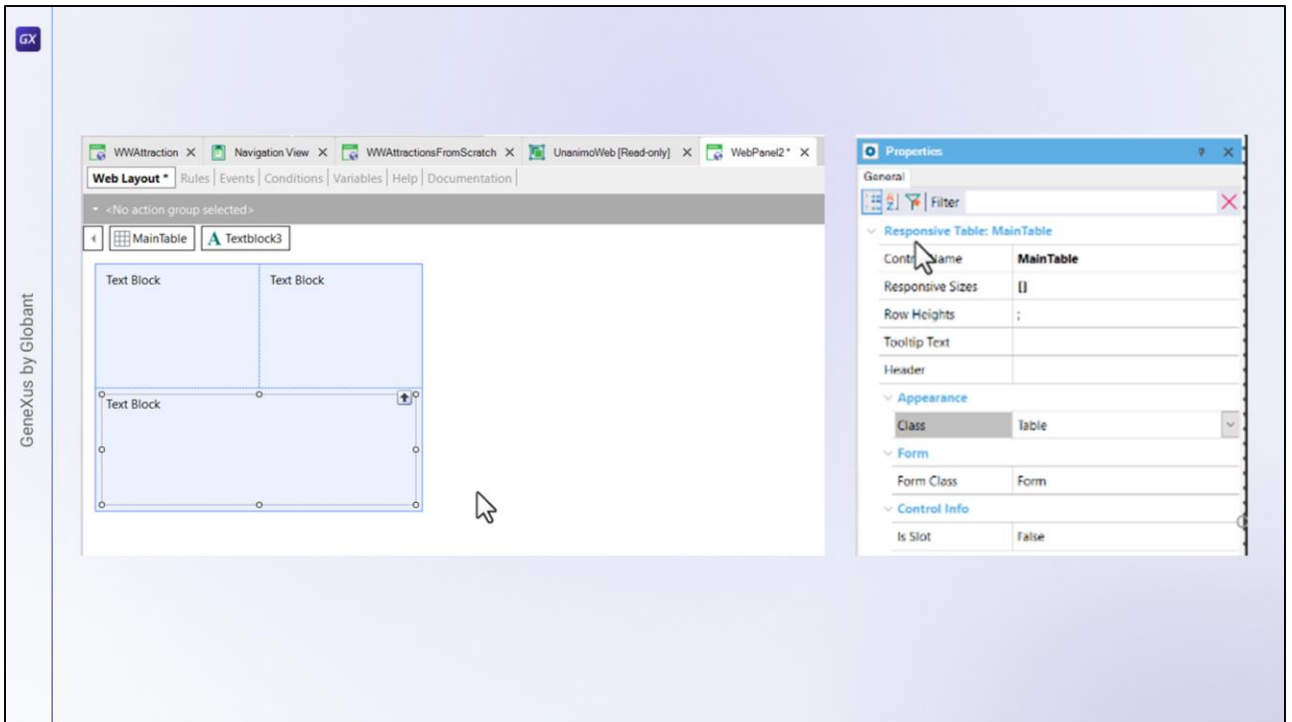
Country Id:

Attraction Id	Attraction Name	Country Name	Attraction Photo	Trips
107	<a href="#">Christ the Redemmer</a>	Brazil		24 <a href="#">New trip</a>
105	<a href="#">The Great Wall</a>	China		0 <a href="#">New trip</a>
110	<a href="#">Forbidden city</a>	China		0 <a href="#">New trip</a>
117	<a href="#">Meet the Emperor</a>	China		0 <a href="#">New trip</a>
113	<a href="#">London Towers</a>	England		1 <a href="#">New trip</a>
104	<a href="#">Louvre Museum</a>	France		1 <a href="#">New trip</a>
106	<a href="#">Eiffel Tower</a>	France		24 <a href="#">New trip</a>
109	<a href="#">Matisse Museum</a>	France		13 <a href="#">New trip</a>
111	<a href="#">Cinque Terre</a>	Italy		0 <a href="#">New trip</a>

Why is it here below this other control, and when the screen is enlarged it appears to the right, seemingly taking up 50% of the screen?



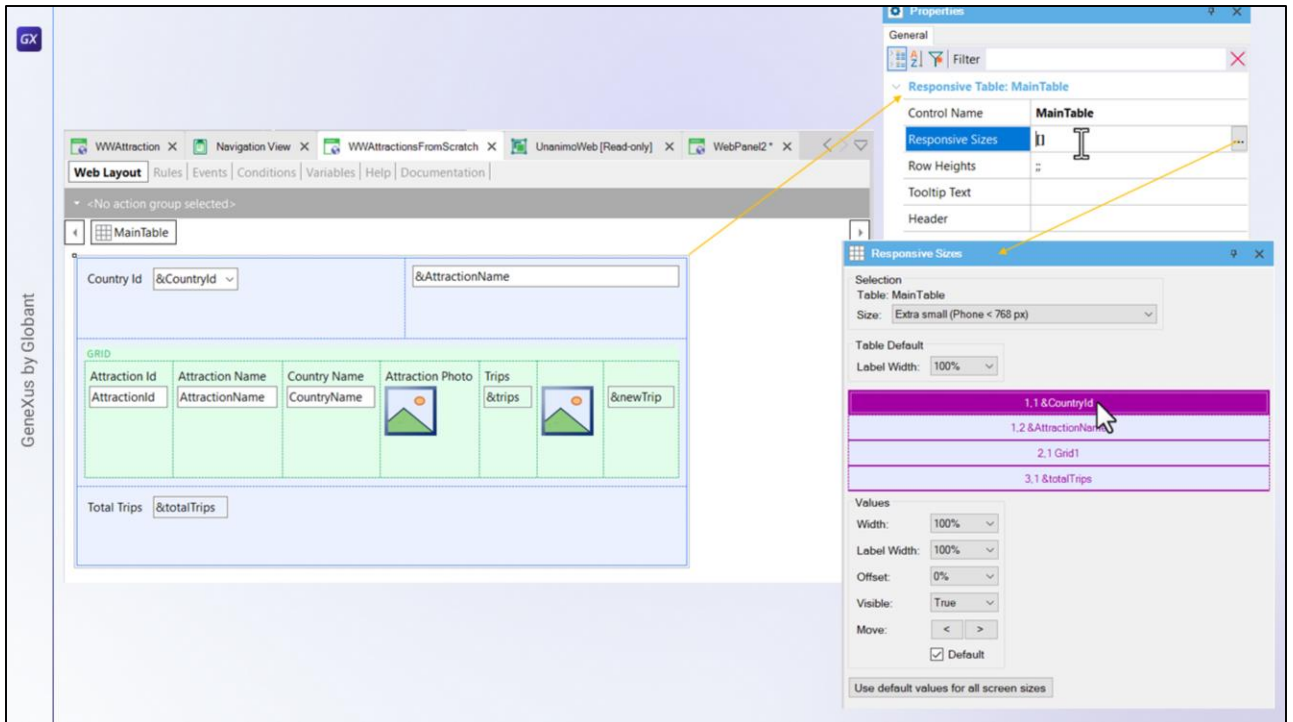
The control is located in the right cell of this row, of this table, the MainTable.



When a new Web panel is created, its layout is always initialized with a table, the MainTable.

Every control that is inserted will be inside that table, and any other control that is inserted will automatically create another cell of the table. Here the table ended up with two cells and if we insert another control, we will have one more cell.

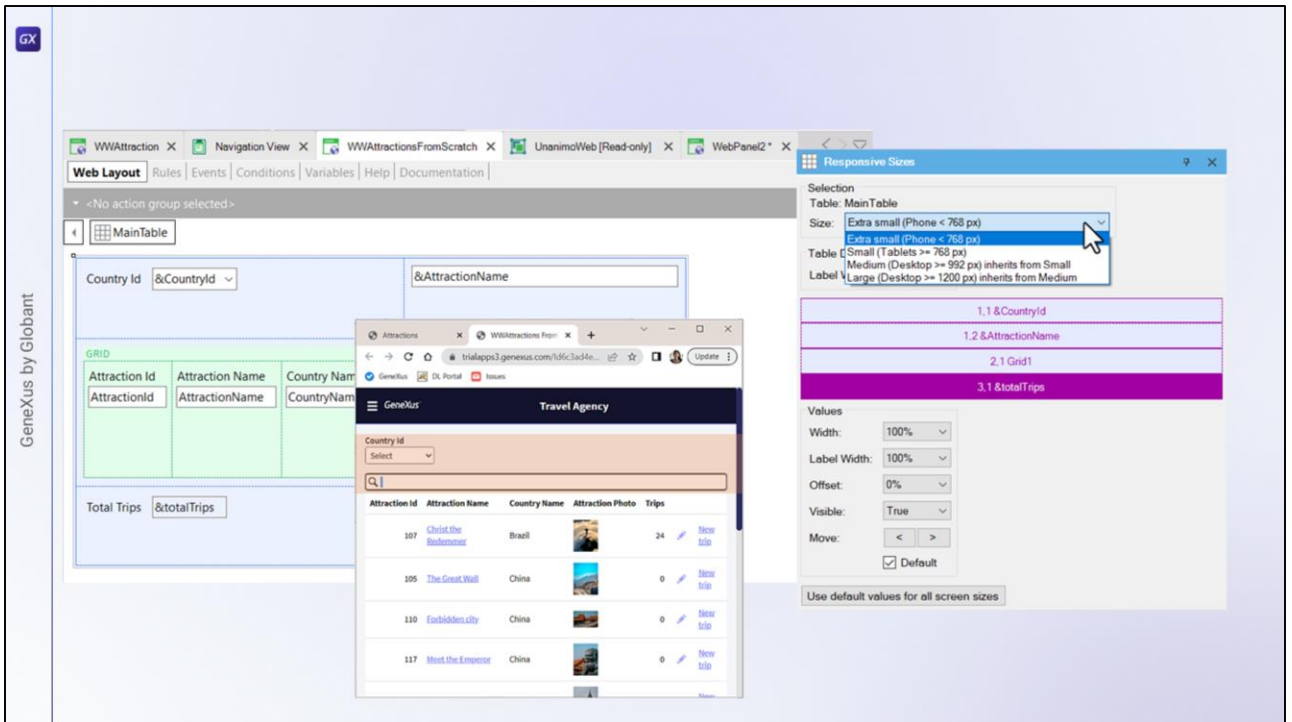
This is a special table known as a responsive table.



Here we see that the table containing all our controls is responsive and has this property, Responsive Sizes.

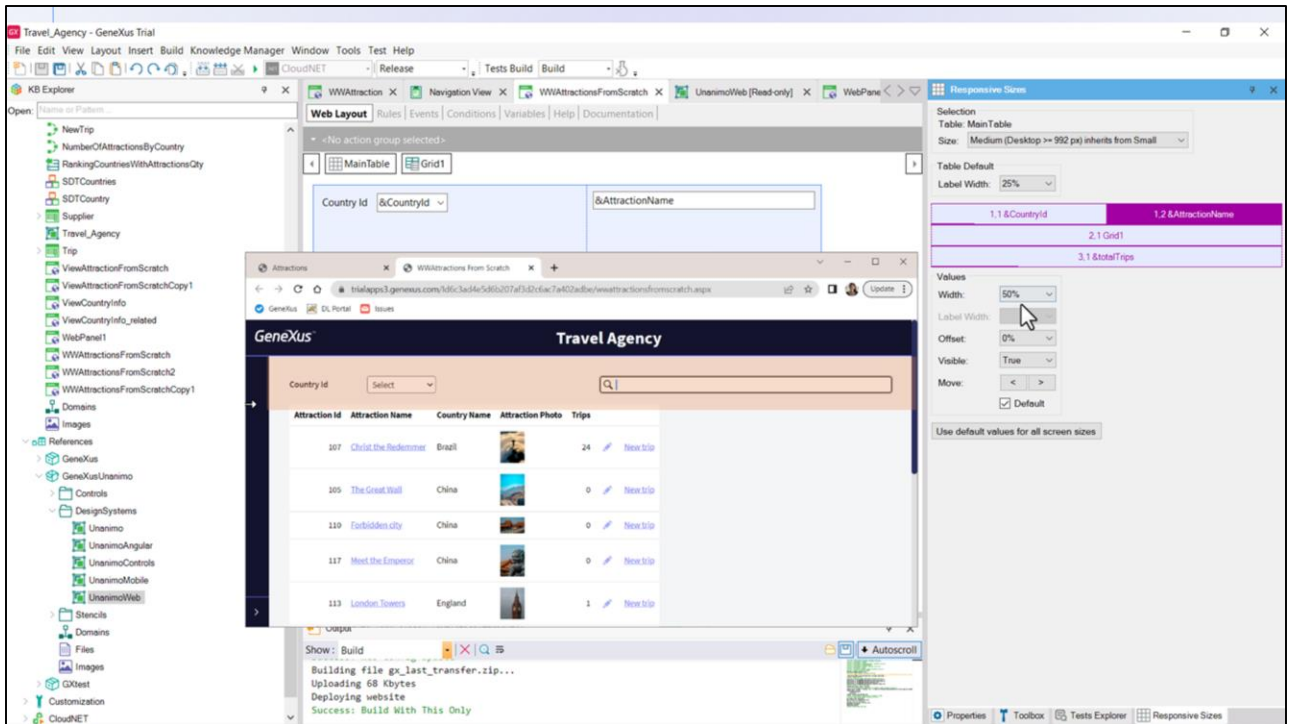
What does it indicate? The four cells of the table, with the controls that go in each one. For example, here is this control, here this one, then the grid, and finally this variable. They take 100% of the available width and are displayed one on top of the other, as if it were a table with a single column.

If we look at this combo, Size, it says that this is valid for the Extra small screen size, which is the same as that of a phone.

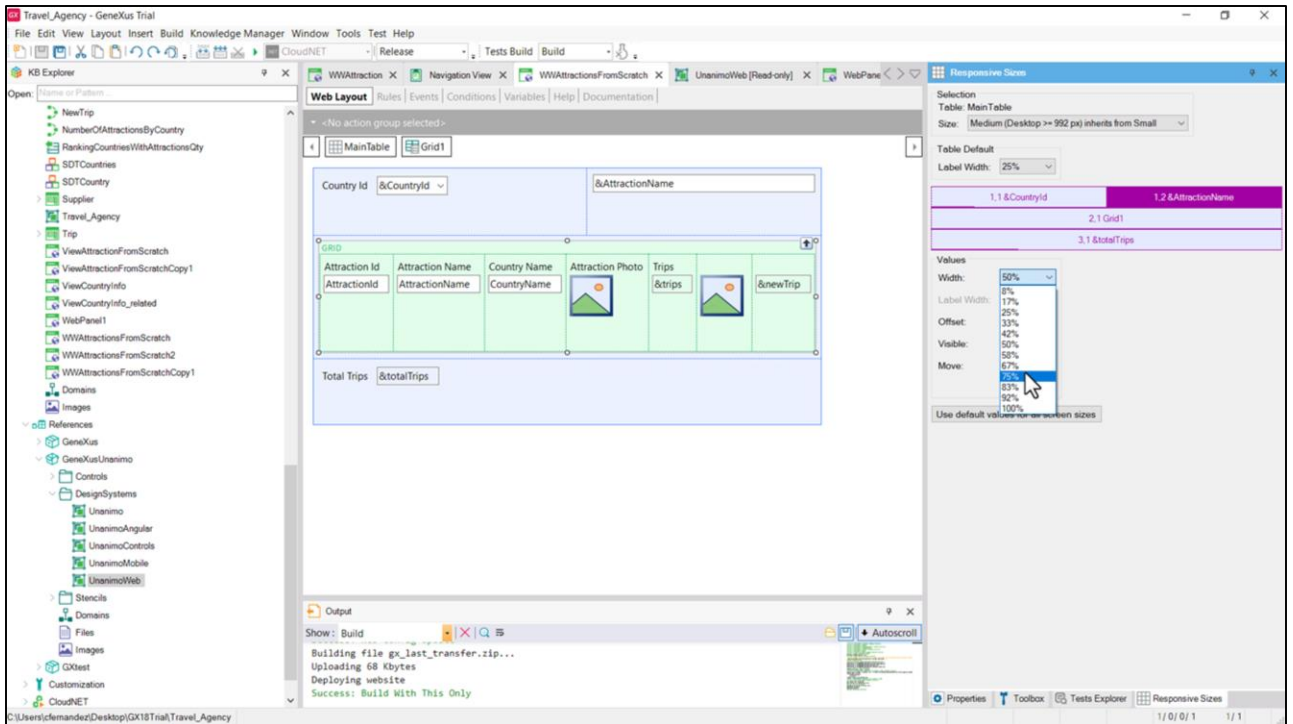


However, note that there are 3 other possible screen sizes. For the Extra small size, the two cells of the first row shown in the layout will be displayed one on top of the other, each one taking 100% of the table width.

That's why at runtime the variables for that size were shown like that.

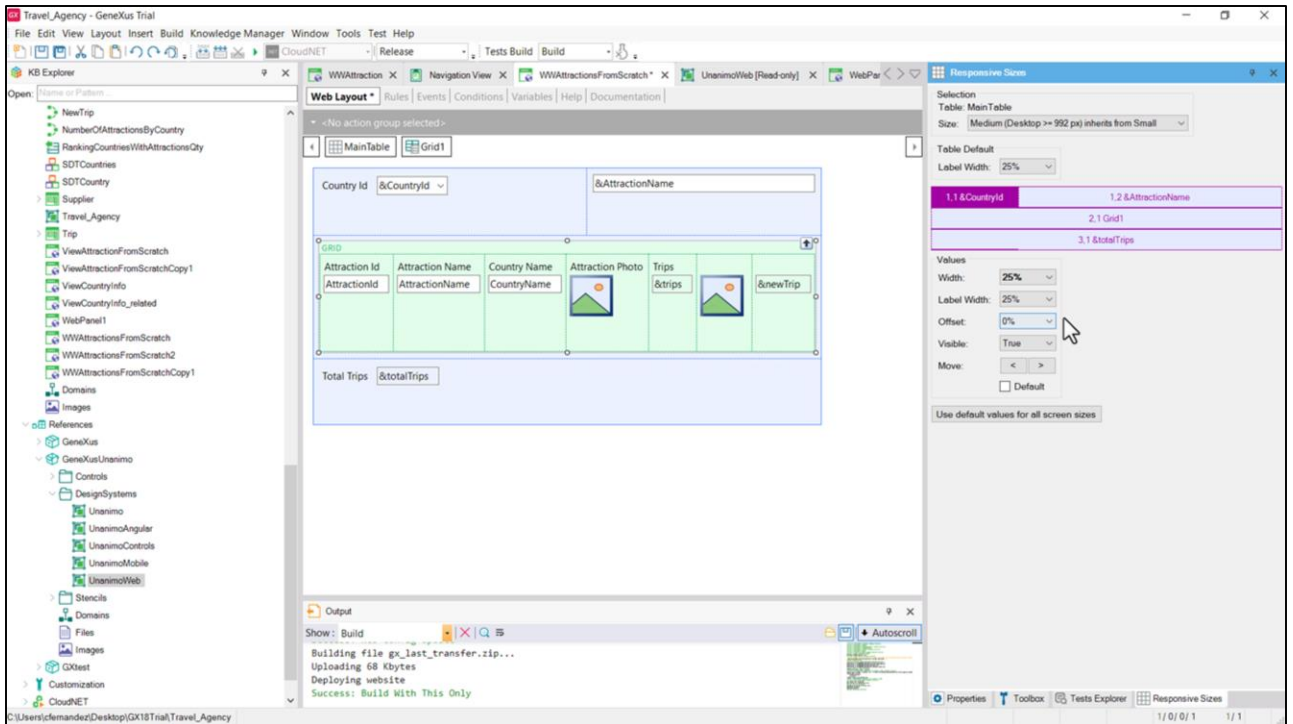


If now we select the Medium size, which is the desktop size, row 1 of the table will be effectively displayed in one row, where each cell will take up 50% of the table width, as we saw at runtime.

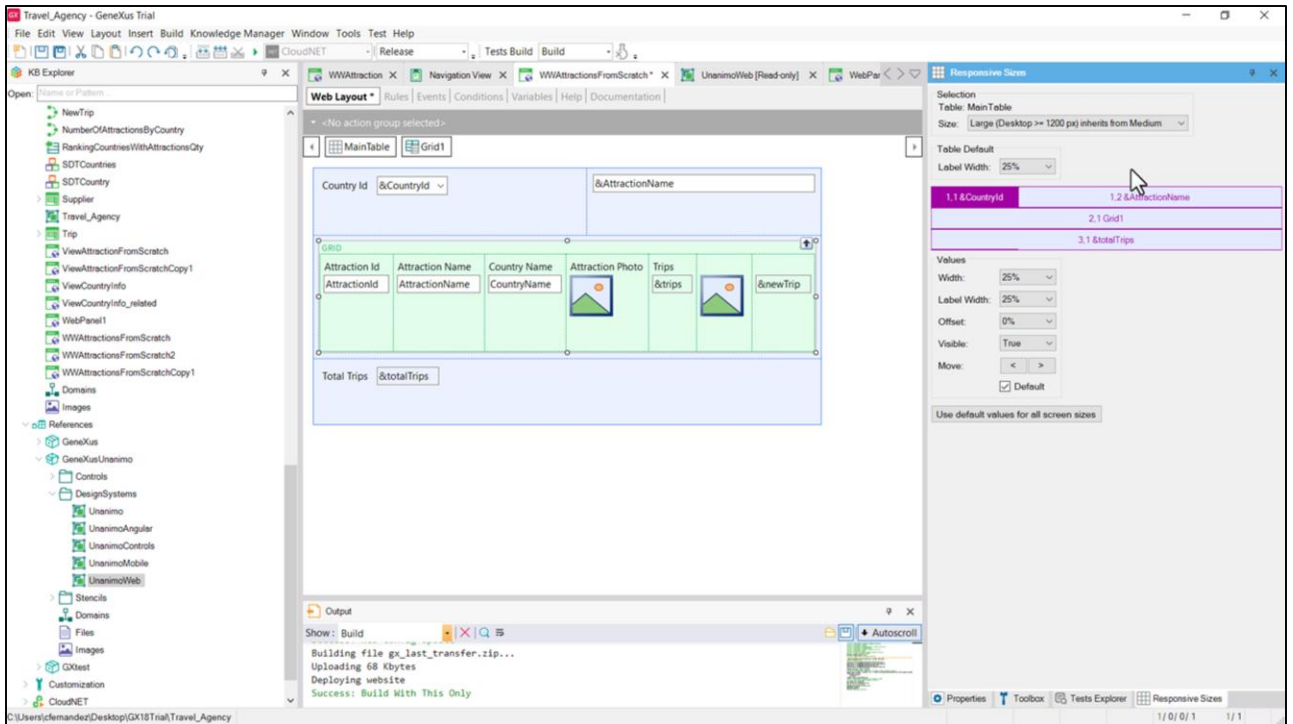


We can modify that width here, according to these 12 options. And so, for example, set AttractionName to take up 75% of the table width.

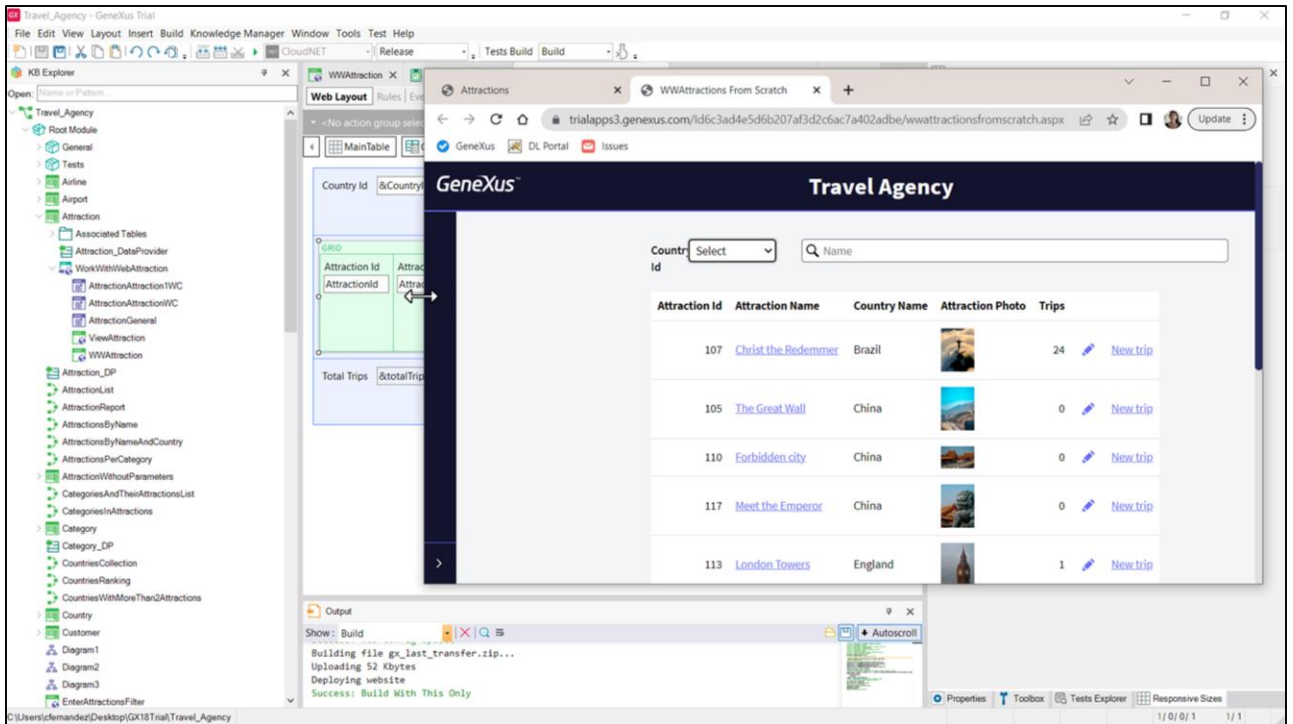




But for both fields to be in the same row, we will also have to change the width of this other one so that it takes up 25%.

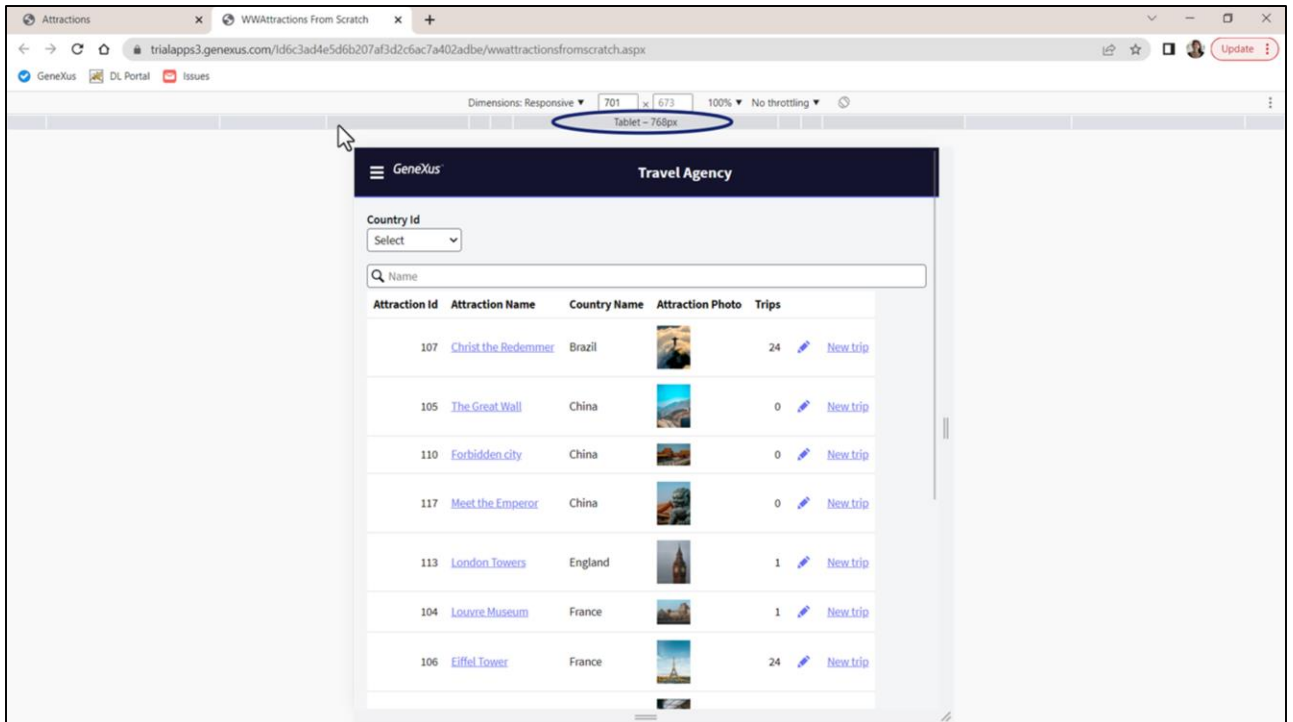


The Large size that follows it inherits the values of the Medium size we have just set and therefore looks the same.



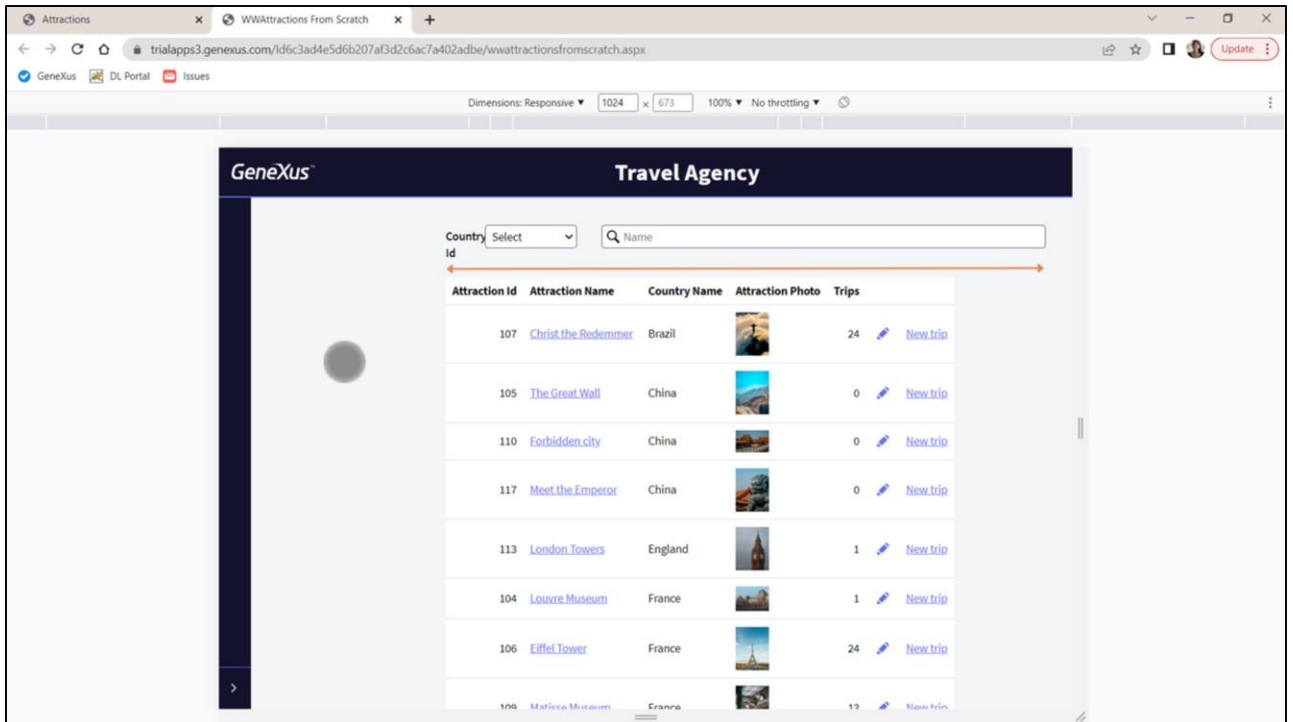
Let's test this change at runtime. We refresh and here we see that change in the widths of each field: 25% and 75% of the table width.

These percentages continue to be kept as we reduce the screen size, going through the Small size that we haven't modified, and up to the Extra small screen size where the width of each control in row 1 was 100%.



Variations by size can also be seen by pressing F12 in the browser. Here it allows us to vary in a discrete way... and here in a continuous way.

What we have seen so far is that we can vary the width and place of the controls of a responsive table according to 4 screen widths (Extra small, Small, Medium, and Large).

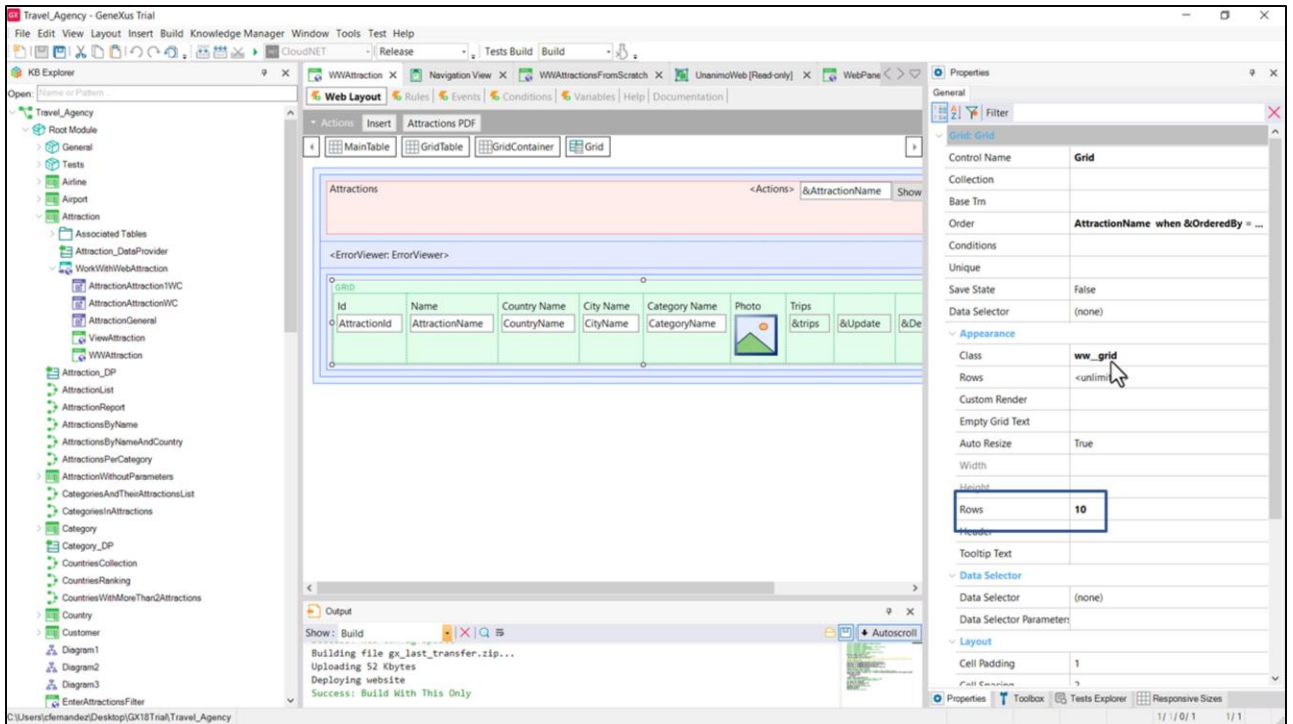


Now let's take a look at our grid and compare it with that of the WorkWith.

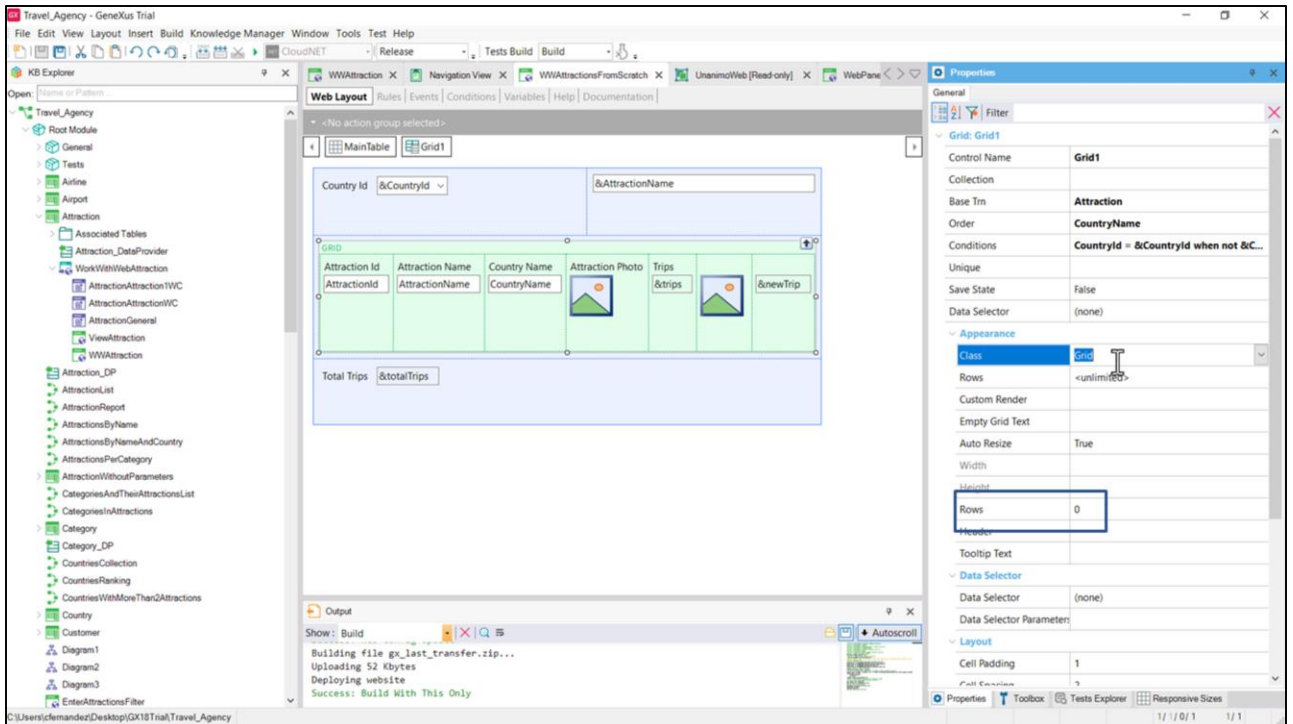
First of all, our grid has a white background, it doesn't take 100% of the table width, and it will load all the tourist attractions...

id	name	Country name	City name	Category name	Photo	trips
107	<a href="#">Christ the Redemmer</a>	<a href="#">Brazil</a>	Rio De Janeiro	Monument		24 <a href="#">UPDATE</a> <a href="#">DELETE</a>
111	<a href="#">Cinque Terre</a>	<a href="#">Italy</a>	Liguria	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
106	<a href="#">Eiffel Tower</a>	<a href="#">France</a>	Paris	Monument		24 <a href="#">UPDATE</a> <a href="#">DELETE</a>
110	<a href="#">Forbidden city</a>	<a href="#">China</a>	Beijing	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
112	<a href="#">Glenfinnan Viaduct</a>	<a href="#">Scotland</a>	Glenfinnan	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
113	<a href="#">London Towers</a>	<a href="#">England</a>	London	Monument		1 <a href="#">UPDATE</a> <a href="#">DELETE</a>
116	<a href="#">Long Bridges</a>	<a href="#">United States</a>	San Francisco	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
104	<a href="#">Louvre Museum</a>	<a href="#">France</a>	Paris	Museum		1 <a href="#">UPDATE</a> <a href="#">DELETE</a>
109	<a href="#">Matisse Museum</a>	<a href="#">France</a>	Nice	Museum		13 <a href="#">UPDATE</a> <a href="#">DELETE</a>
117	<a href="#">Meet the Emperor</a>	<a href="#">China</a>	Beijing	Monument		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>

... meanwhile, the WorkWith grid also has a white background, but it does take up the entire width, and allows paging, loading 10 at a time.

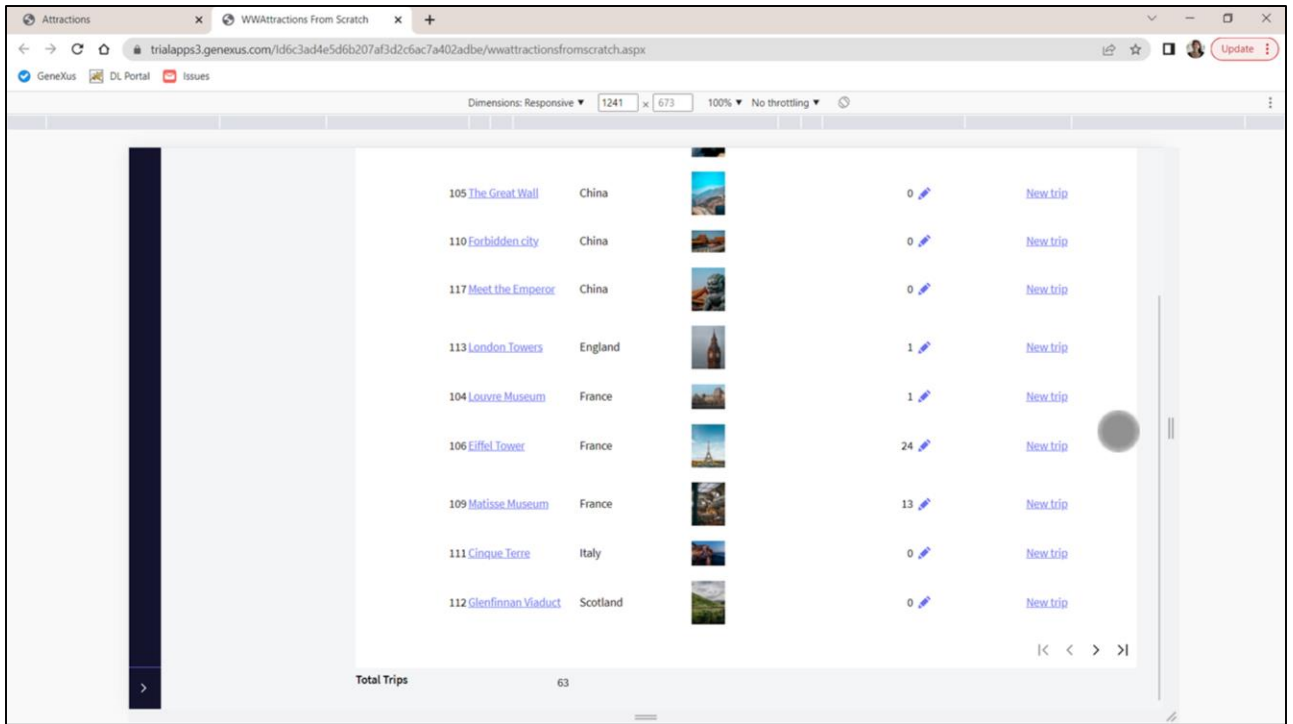


We see that the WorkWith grid has this associated class and the number of rows per page is set to 10.

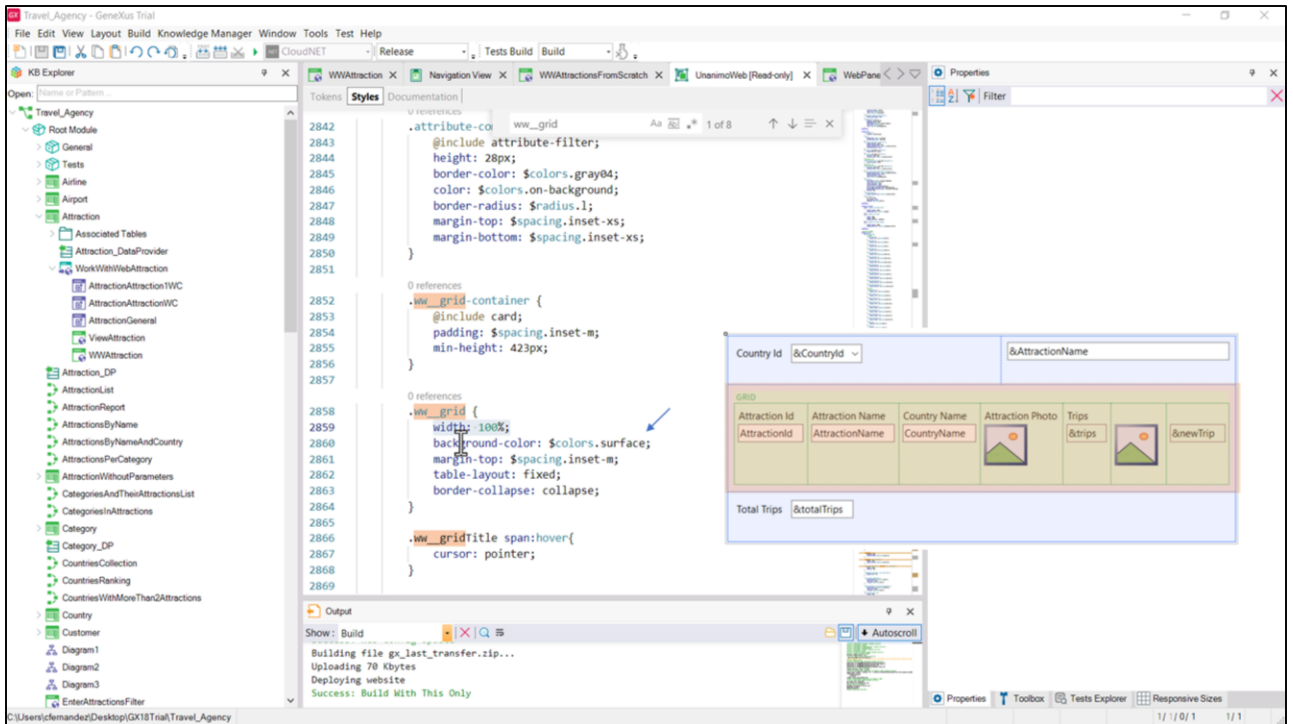


In this case, the grid class is the default one. We will change it for that of the WorkWith and set the number of rows per page to 10.





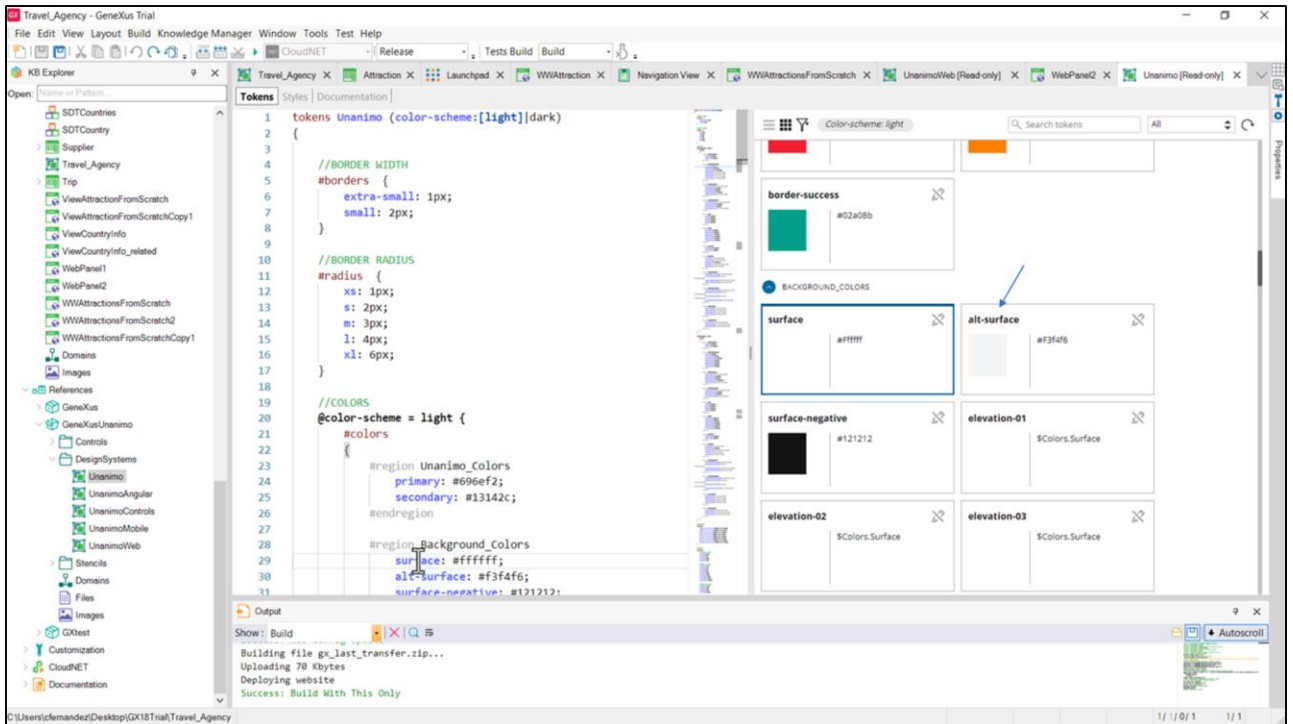
Now it takes the entire width and does paging.



Let's inspect the class properties...

Here we see why it now takes 100% of its container's width (which in this case is this cell on the second row of the Main table, so it matches the table's width).

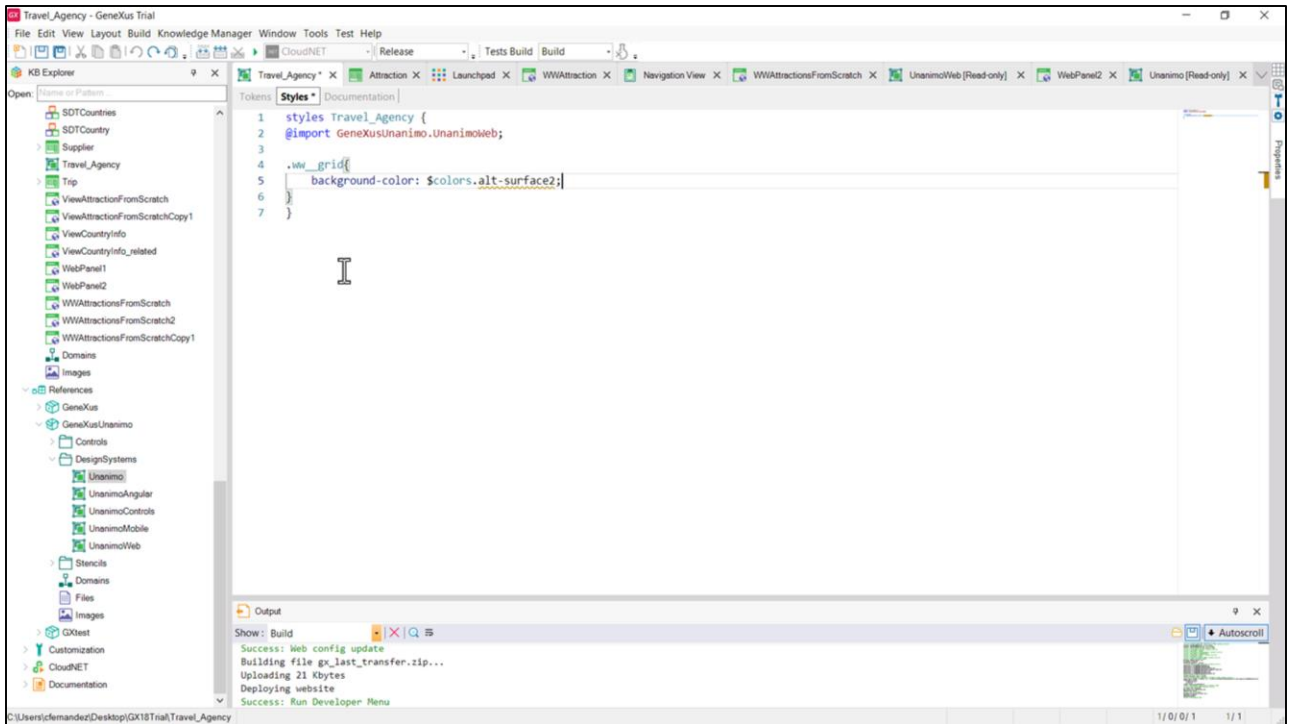
Also, the background color configured is the one that comes from this color token, which, from what we saw at runtime, is white.



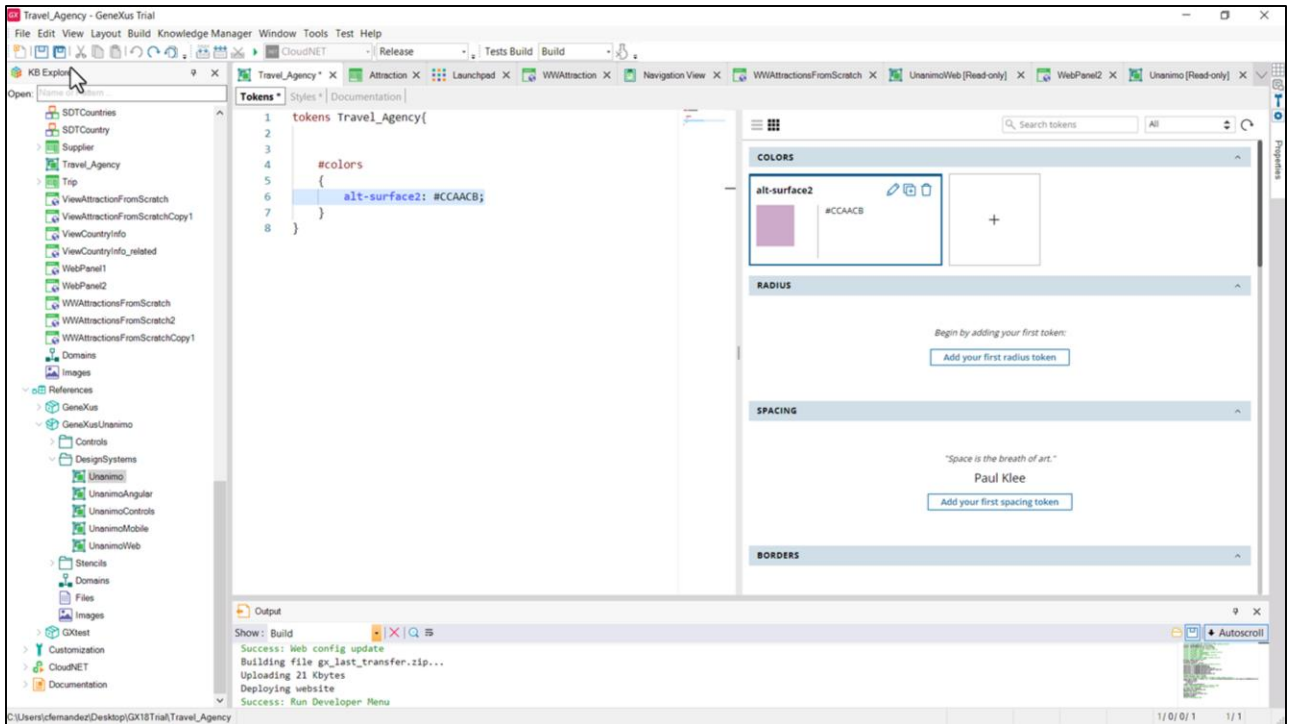
We do not have it defined in this DSO, but we do have it in this other one that is imported by UnanimoWeb.

Here we see the white color.

If we want to change the background color for all the grids that have this class defined, for example, for this one or for a new alternative color that we will have to define...



Then in our application's DSO, the one that we can customize, we select the class that we want to customize, and overwrite only the value of the background-color property, assigning it now the value of the alternative token, a new one, that we will have to define here because it doesn't exist in the tree of imported DSOs.



We add it as a color token, so we will be able to reuse this color token as an alternative background for all the classes we want.

Let's try it.

Attractions x WWAttractions From Scratch x

















trialapps3.genevus.com/ld6c3ad4e5d6b207af3d2c6ac7a402adbe/wwattractionsfromscratch.aspx

Genevus DL Portal Issues

Dimensions: Responsive 1241 x 673 100% No throttling

### Genevus Travel Agency

Country Id

Attraction Id	Attraction Name	Country Name	Attraction Photo	Trips
107	<a href="#">Christ the Redeemer</a>	Brazil		24  <a href="#">New trip</a>
105	<a href="#">The Great Wall</a>	China		0  <a href="#">New trip</a>
110	<a href="#">Forbidden city</a>	China		0  <a href="#">New trip</a>
117	<a href="#">Meet the Emperor</a>	China		0  <a href="#">New trip</a>
113	<a href="#">London Towers</a>	England		1  <a href="#">New trip</a>
104	<a href="#">Louvre Museum</a>	France		1  <a href="#">New trip</a>
106	<a href="#">Eiffel Tower</a>	France		24  <a href="#">New trip</a>
109	<a href="#">Matisse Museum</a>	France		13  <a href="#">New trip</a>

We press Control F5...

Attractions WWAttractions From Scratch

trialapps3.genexus.com/ld6c3ad4e5d6b207af3d2c6ac7a402adbe/wwattraction.aspx

GeneXus DL Portal Issues

# GeneXus Travel Agency










UNANIMO

- Attractions
- Countries

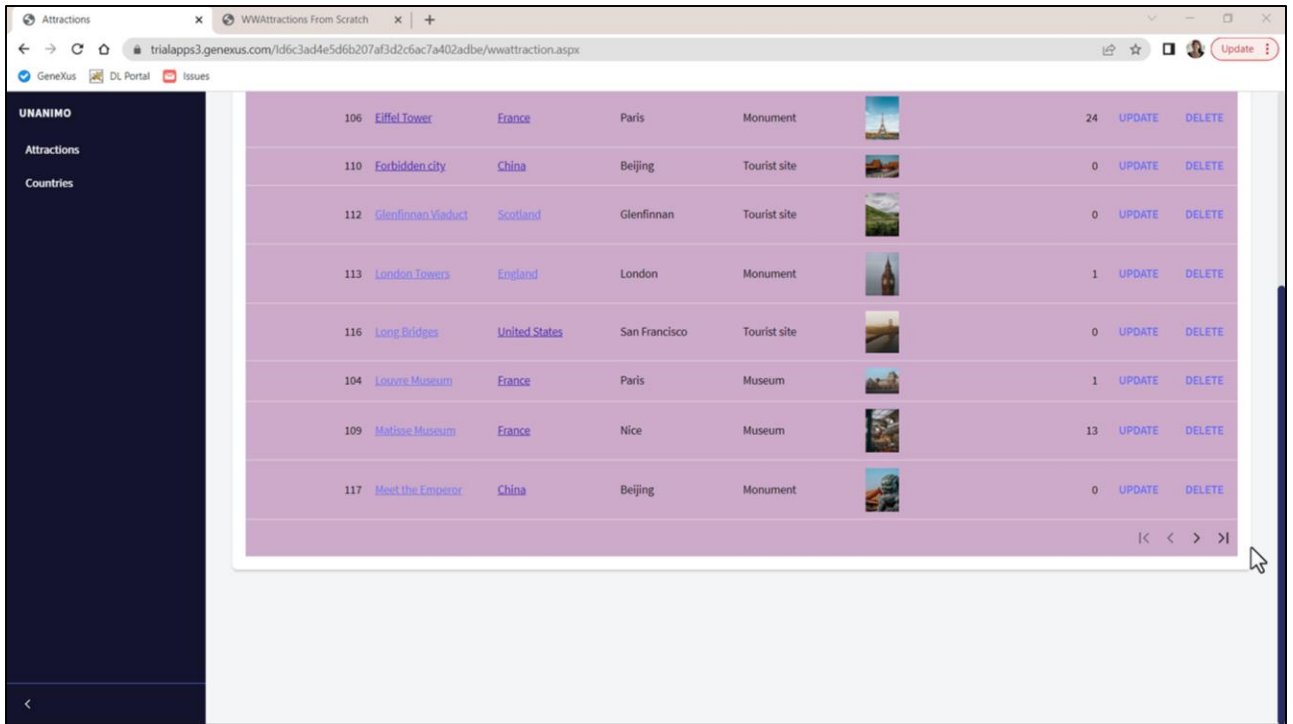
## Attractions

INSERT ATTRACTIONS PDF

 ▼

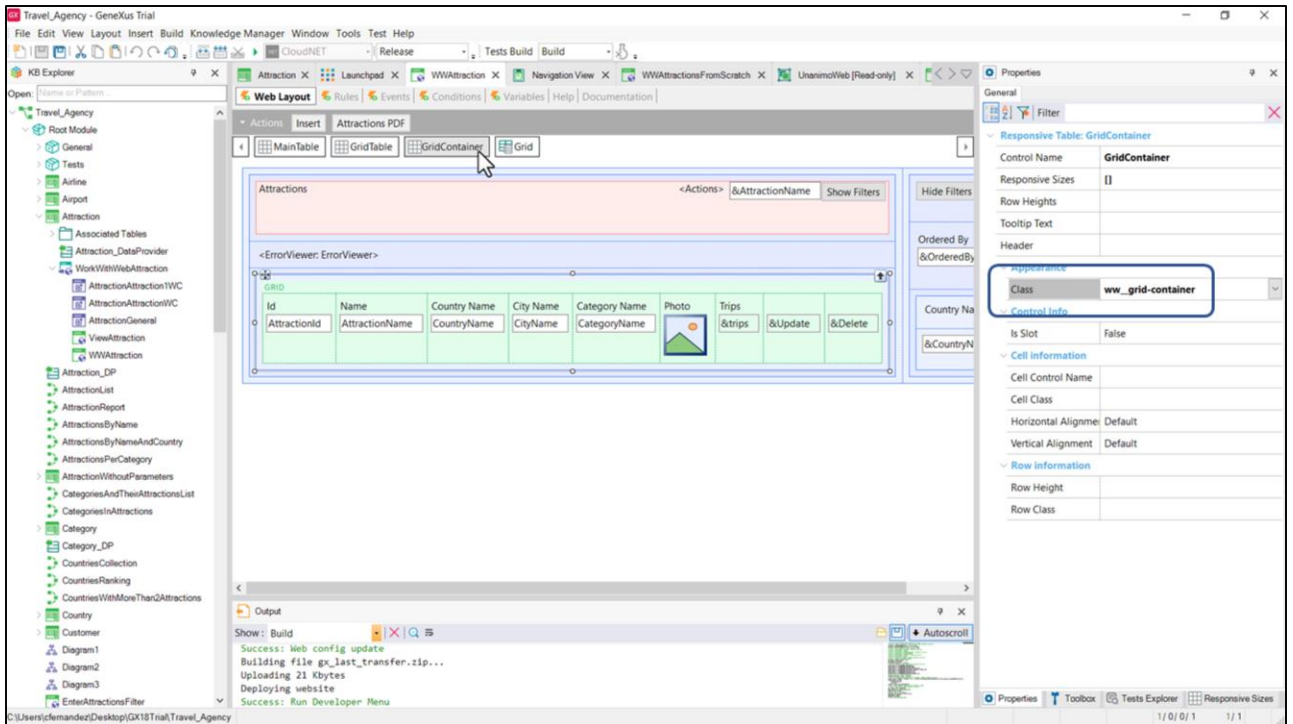
Id	Name	Country Name	City Name	Category Name	Photo	Trips
107	<a href="#">Christ the Redeemer</a>	<a href="#">Brazil</a>	Rio De Janeiro	Monument		24 <a href="#">UPDATE</a> <a href="#">DELETE</a>
111	<a href="#">Cinque Terre</a>	<a href="#">Italy</a>	Liguria	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
106	<a href="#">Eiffel Tower</a>	<a href="#">France</a>	Paris	Monument		24 <a href="#">UPDATE</a> <a href="#">DELETE</a>
110	<a href="#">Forbidden city</a>	<a href="#">China</a>	Beijing	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
112	<a href="#">Glenfinnan Viaduct</a>	<a href="#">Scotland</a>	Glenfinnan	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
113	<a href="#">London Towers</a>	<a href="#">England</a>	London	Monument		1 <a href="#">UPDATE</a> <a href="#">DELETE</a>
116	<a href="#">Long Bridges</a>	<a href="#">United States</a>	San Francisco	Tourist site		0 <a href="#">UPDATE</a> <a href="#">DELETE</a>
104	<a href="#">Louvre Museum</a>	<a href="#">France</a>	Paris	Museum		1 <a href="#">UPDATE</a> <a href="#">DELETE</a>
109	<a href="#">Matisse Museum</a>	<a href="#">France</a>	Nice	Museum		13 <a href="#">UPDATE</a> <a href="#">DELETE</a>

We see it applied to both grids.

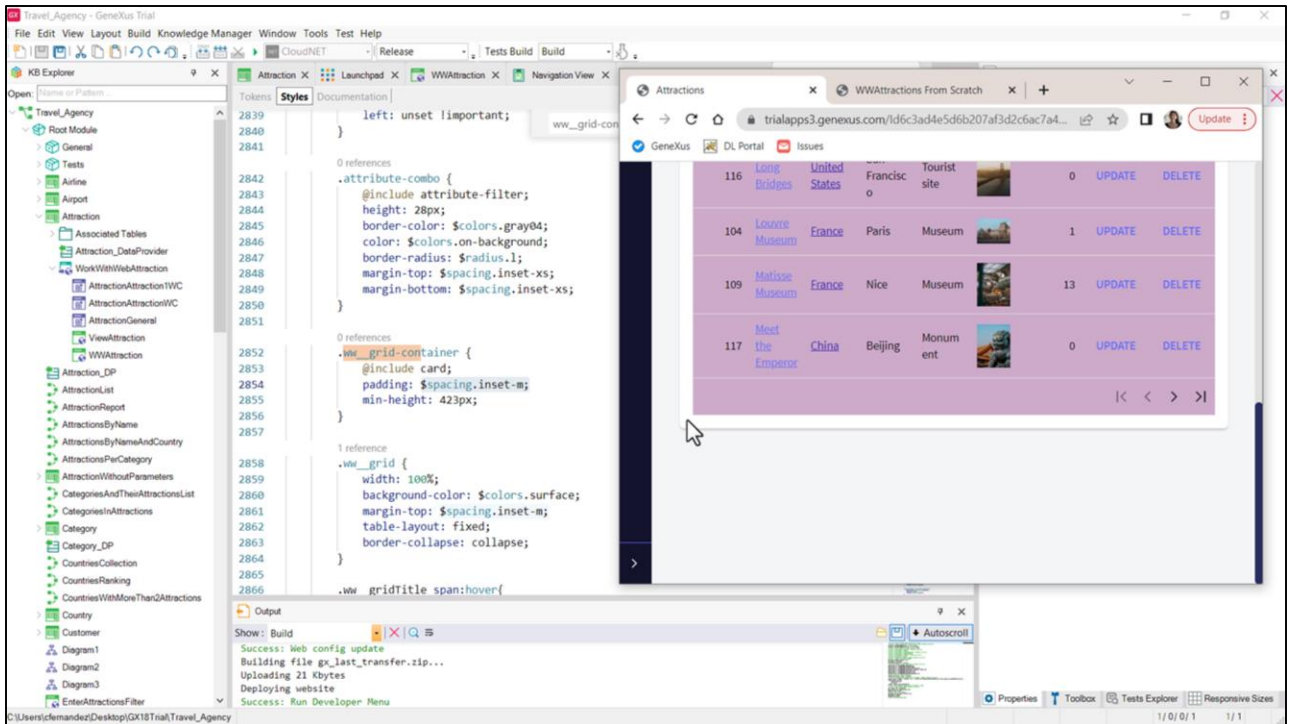


Where does this white color with rounded edges come from in the WorkWith and with this spacing in relation to the grid?





Here we see that the grid with the class we've just customized is inside a table, this one, which has this other class associated with it. This appearance must be configured here. We are going to look for it to the Unanimoweb DSO...



Here is the spacing property: padding. It indicates that the content of the control that has this class must be spaced in relation to the control's borders in this magnitude, given by this token, which will be in pixels.

The screenshot displays a development environment with a code editor on the left and a browser preview on the right. The code editor shows the following CSS styles:

```
1134 .expanded {
1135   margin-left: 57px;
1136   width: calc(100% - 74px);
1137   transition: margin-left .3s, width .2s ease;
1138 }
1139
1140 0 references
1141 .table-login {
1142   width: 360px;
1143   padding: 45px 50px 45px 50px;
1144   background: $colors.surface;
1145   border-radius: $radius.xl;
1146   box-shadow: $shadows.xxs;
1147 }
1148 0 references
1149 .card {
1150   background-color: $colors.surface;
1151   border-radius: $radius.xl;
1152   box-shadow: $shadows.xxs;
1153 }
1154 0 references
1155 .card-body {
1156   padding: $spacing.inset-m;
1157 }
1158 0 references
1159 .card-heading {
1160   box-shadow: 0px 1px 1px $colors.gray01;
1161   padding: $spacing.inset-m;
1162   font-family: $fonts.primary-bold;
1163   font-size: $fontSizes.s;
1164   letter-spacing: $spacing.letter-spacing-s;
1165 }
```

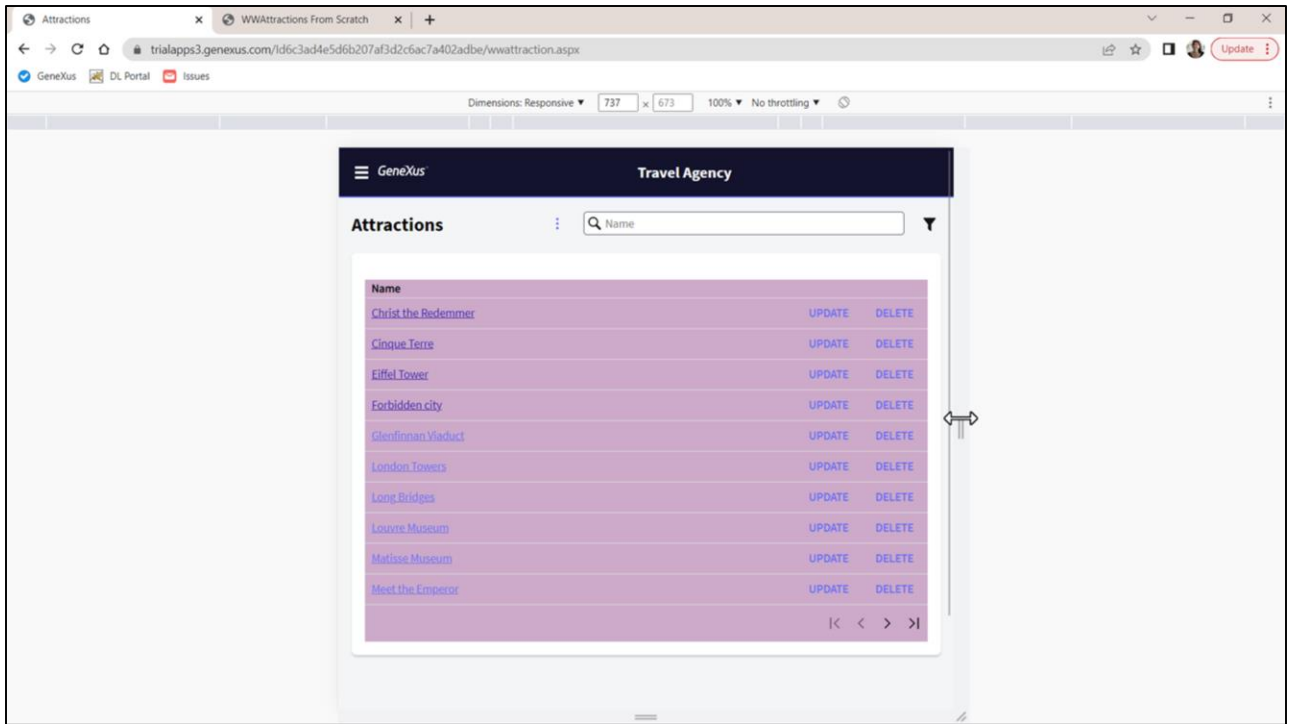
The browser preview shows a table of attractions with the following data:

116	Long Bridges	United States	Francisco	Tourist site		0	UPDATE	DELETE
104	Louvre Museum	France	Paris	Museum		1	UPDATE	DELETE
109	Matisse Museum	France	Nice	Museum		13	UPDATE	DELETE
117	Meet The Emperor	China	Beijing	Monument		0	UPDATE	DELETE

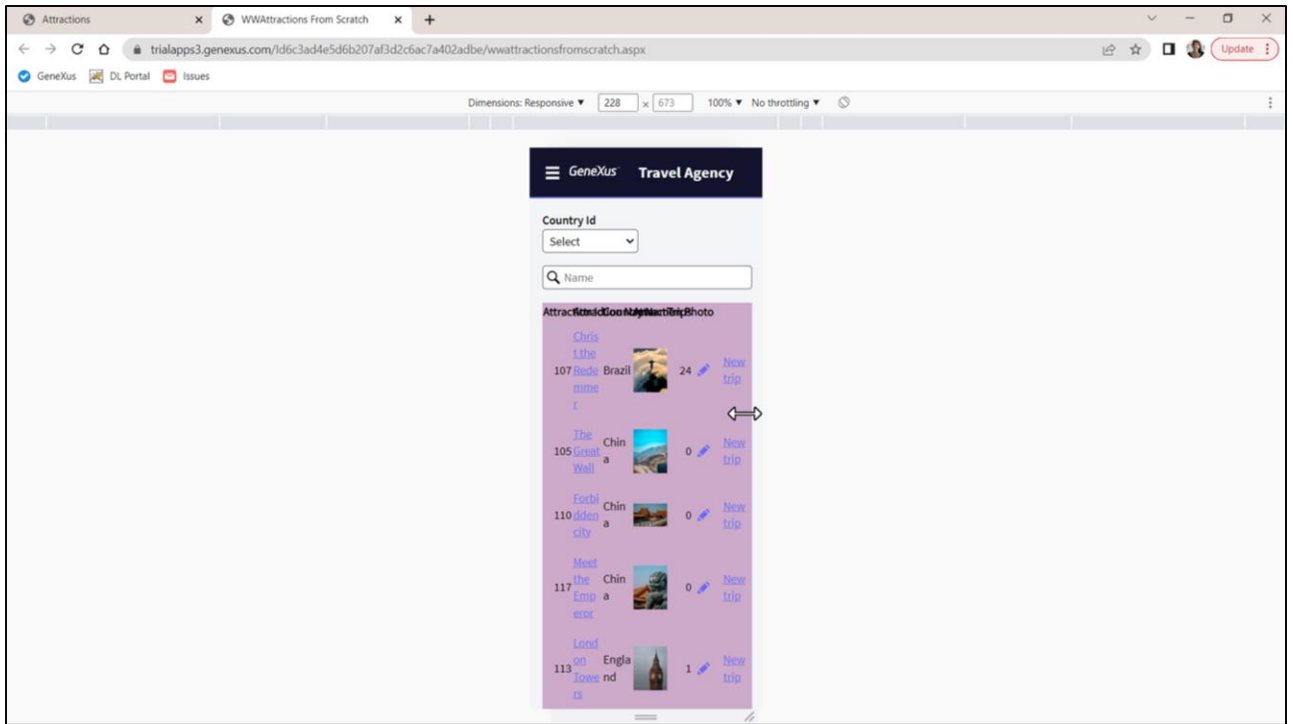
The code editor also shows a build output window with the following messages:

```
Show: Build
Success: web config update
Building file gx_last_transfer.zip...
Uploading 21 kbytes
Deploying website
Success: Run Developer Menu
```

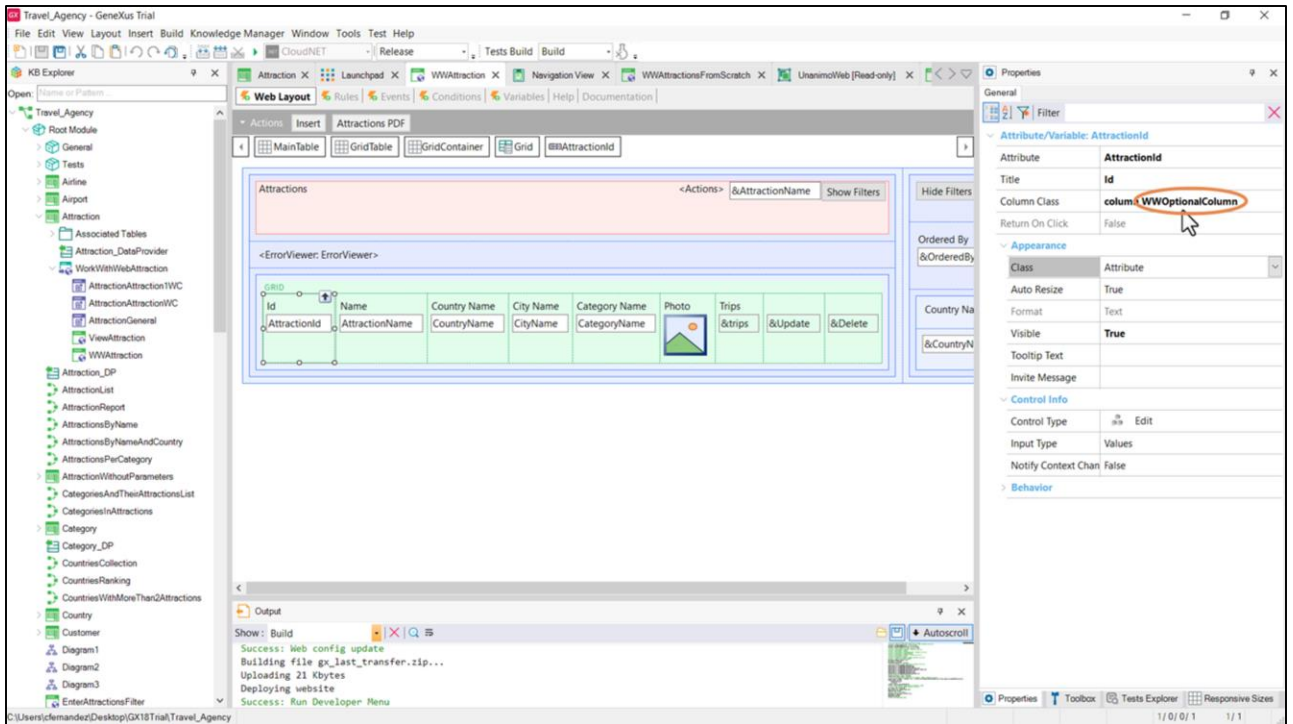
In addition, this class includes this other one, that is to say, its properties. Here we see the background color (that of the Surface token which was white) and both the property for the rounded edges and for this shadow that you see.



Now note another interesting thing: while for the WorkWith grid (we press F12) there are columns that are no longer displayed when the screen width decreases and only the attraction name and the actions remain, this is not the case for our grid: all of them are displayed even if they don't fit well.

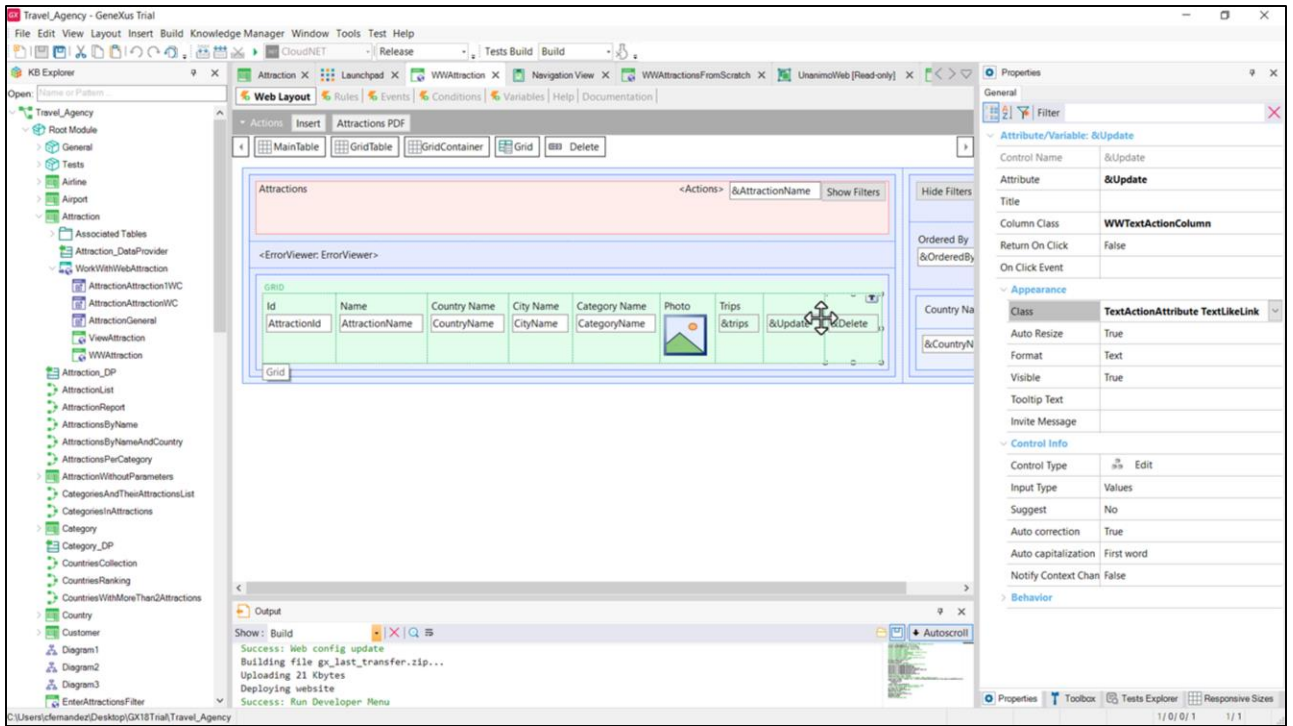


Why?

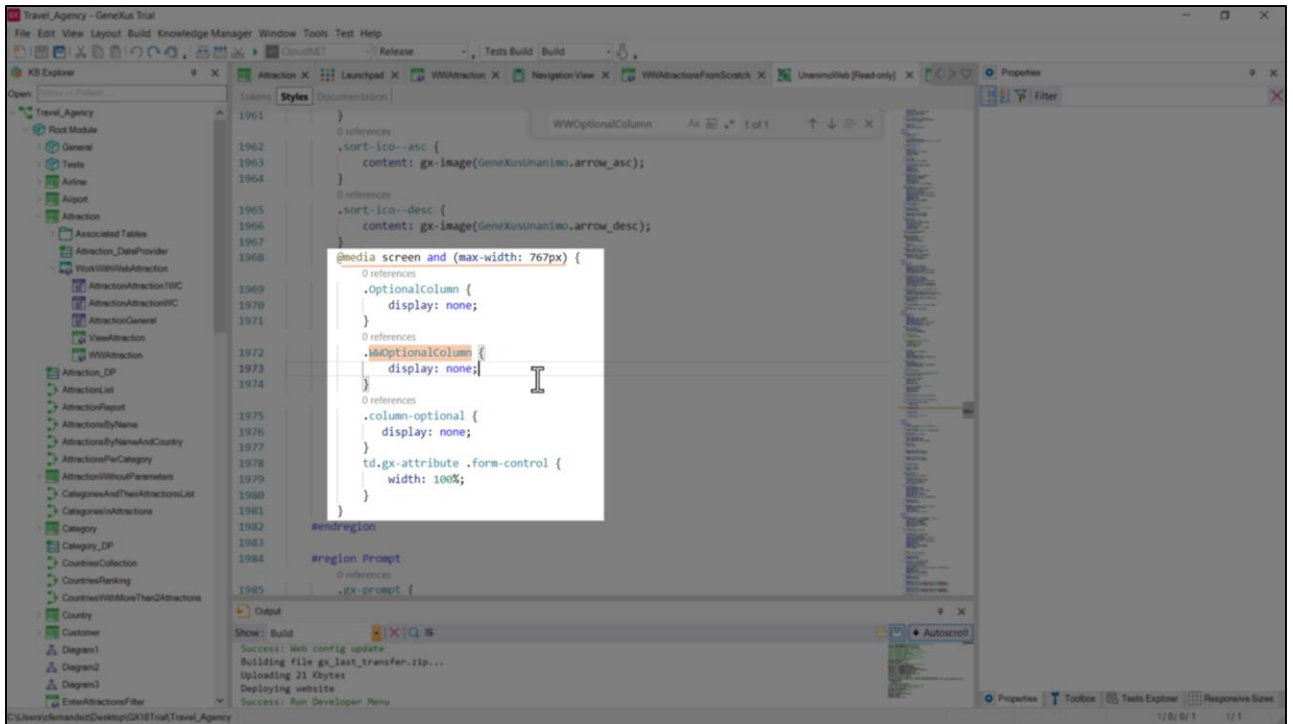


The reason is that these columns have associated classes that determine whether or not they are displayed according to the screen size.

Thus, the Id column has two classes (so the properties of both will be applied): the one named "column" which is also in the attraction name column, but it also has this other one... which is repeated for all the columns that were hidden for the small screen size.

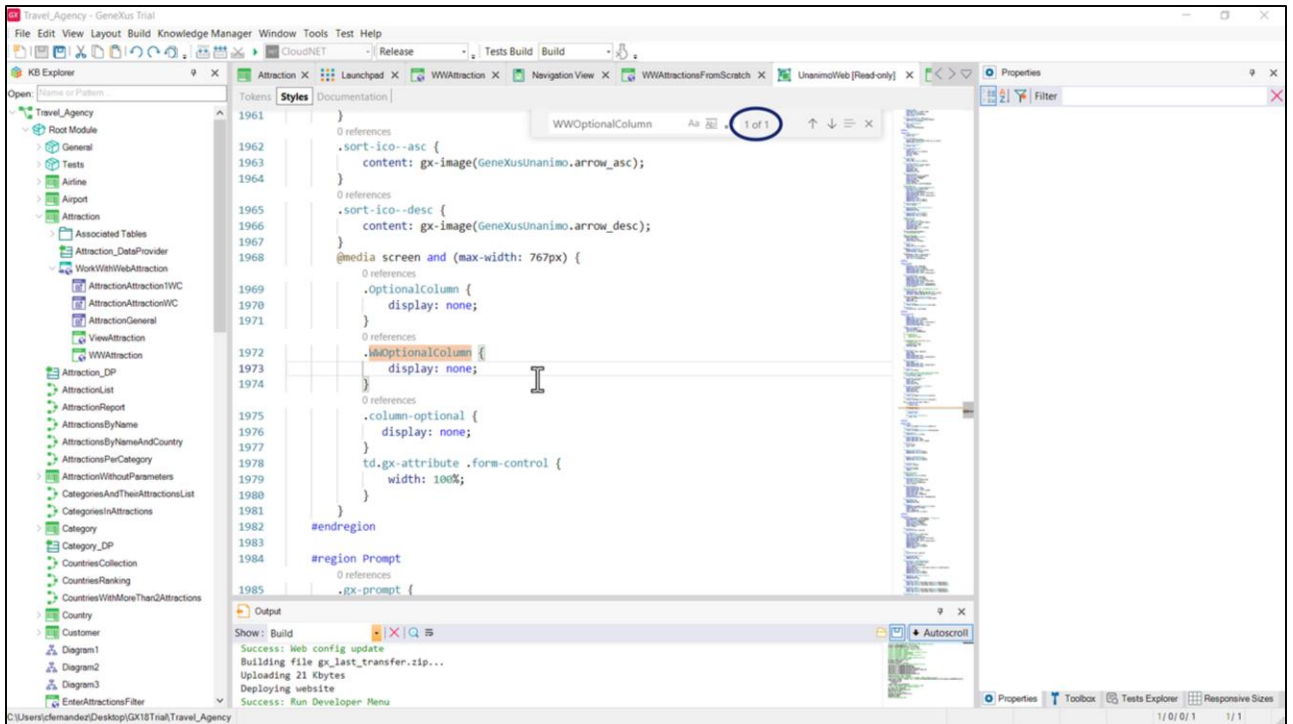


Those of the actions, not coincidentally, do not have that class.

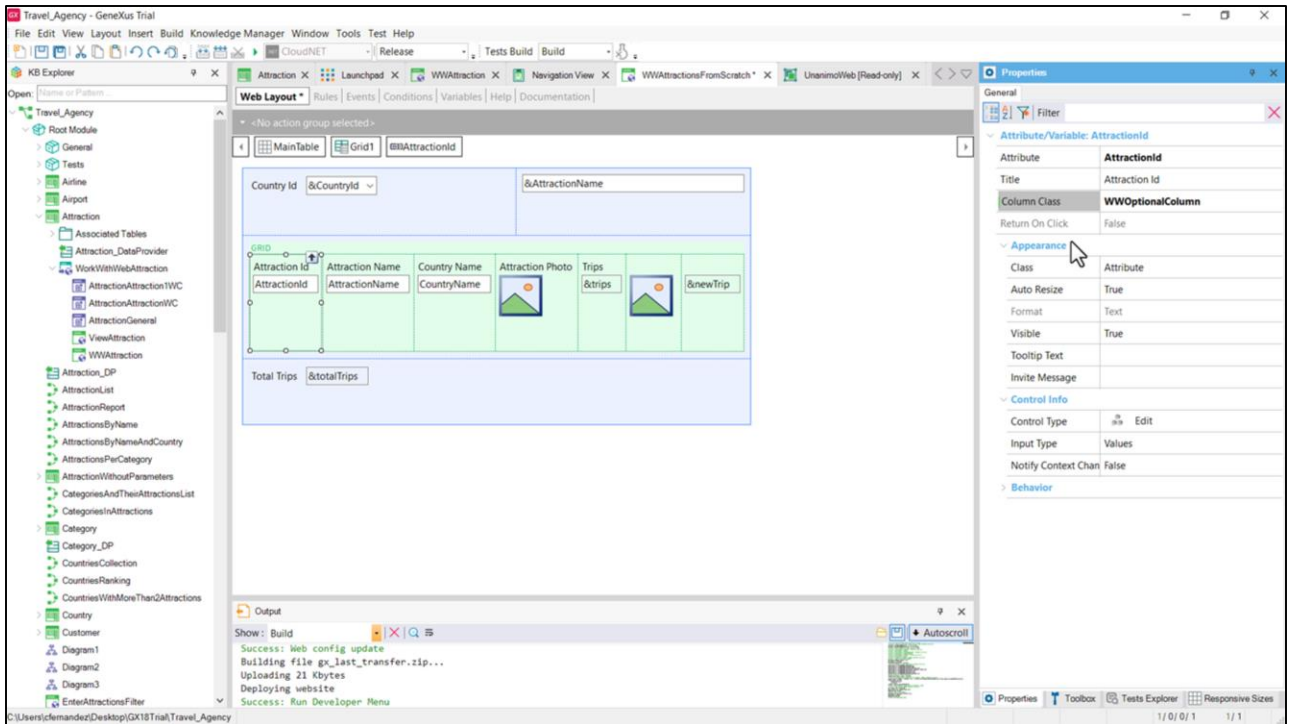


Let's look for the class... Here we find it, with the display property set to none. But if we look closely, this class is being defined in this way only when this rule applies; it is the one that allows conditioning the output to a maximum screen width of 767 pixels. If the screen size is bigger, this definition is not valid.



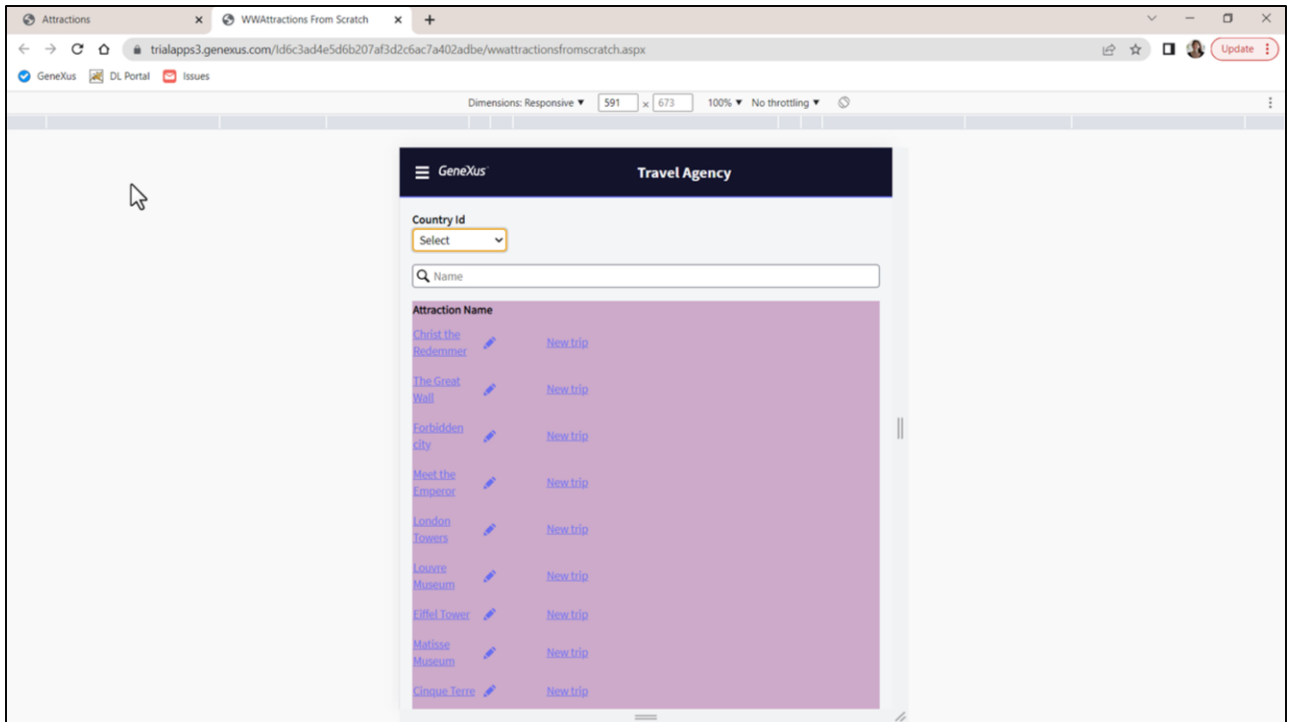


Actually, since this is the only occurrence of this class in this Styles tab, it means that for any other screen size there are no properties for the class; that is why in that case we see the columns displayed on the screen.



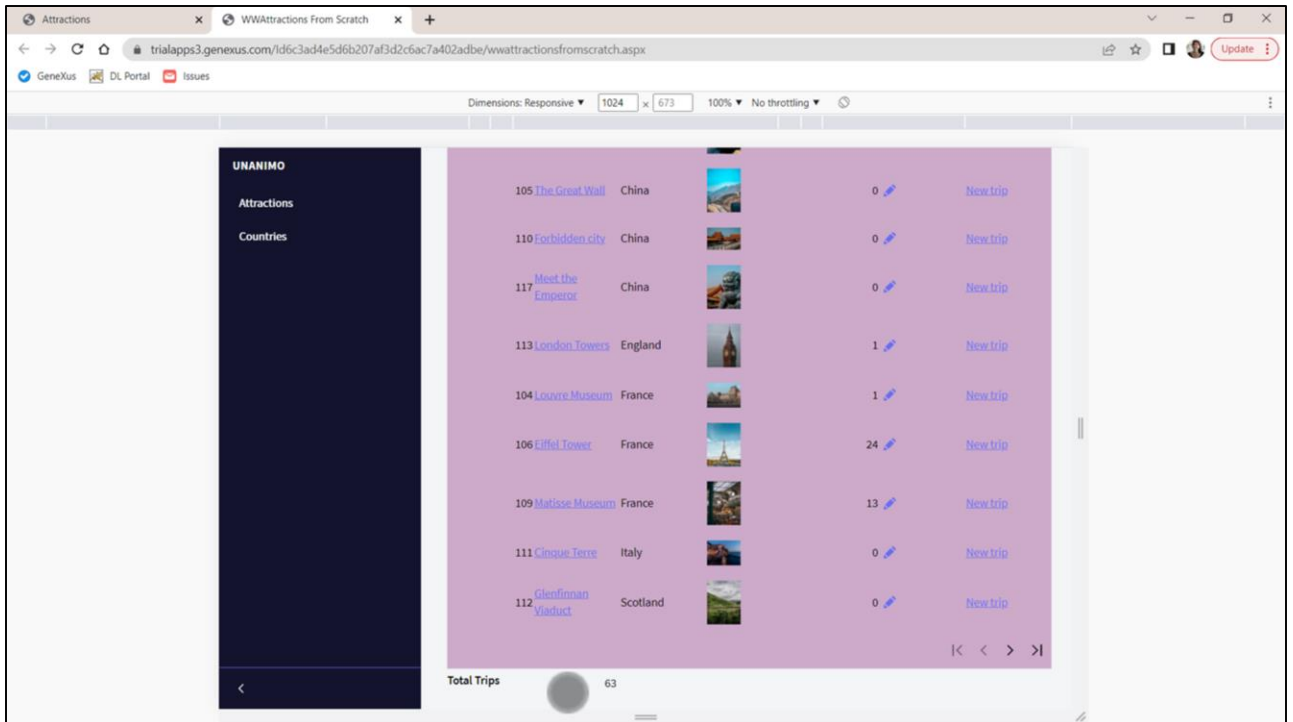
In our grid, let's add this property to the columns that we don't want to be displayed in the small size.

We generate...



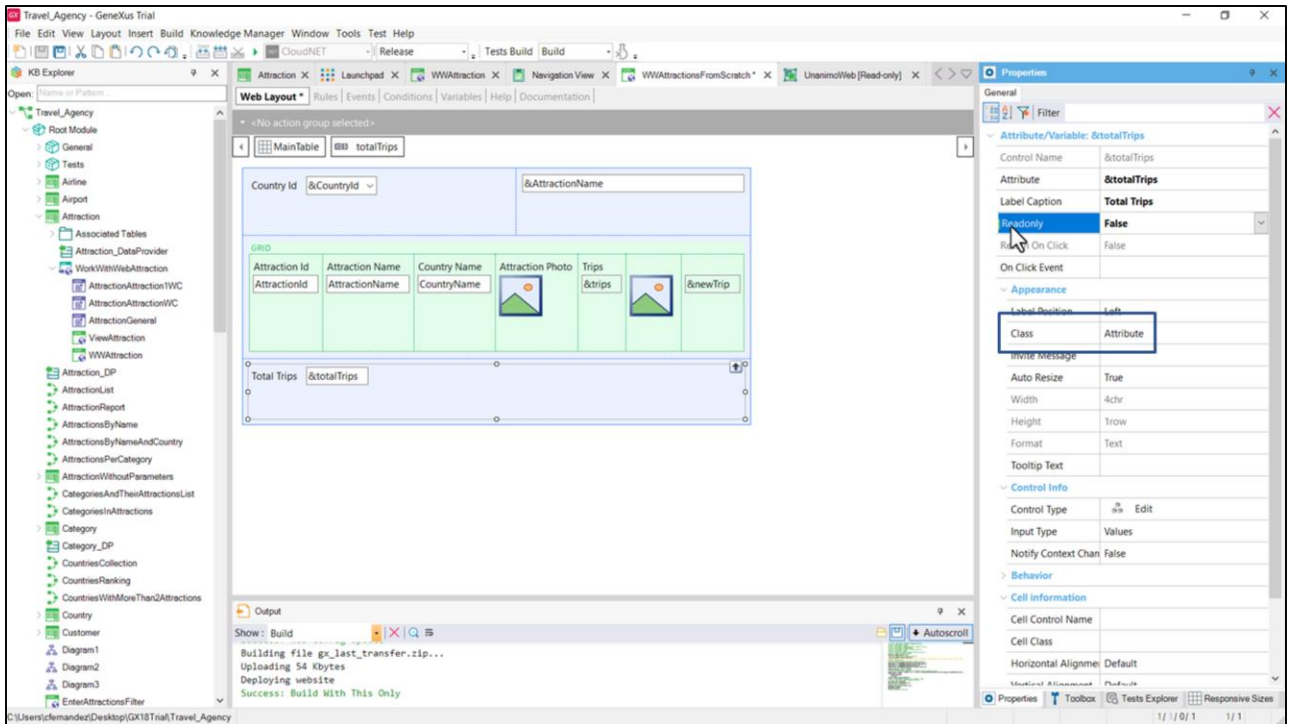
Before refreshing, all the columns are shown. On the other hand, once we refresh... the ones we have just worked with are only displayed when the screen width is greater than 767 pixels.

We can continue to look into the differences between our grid and the WorkWith grid to understand how to standardize the design.



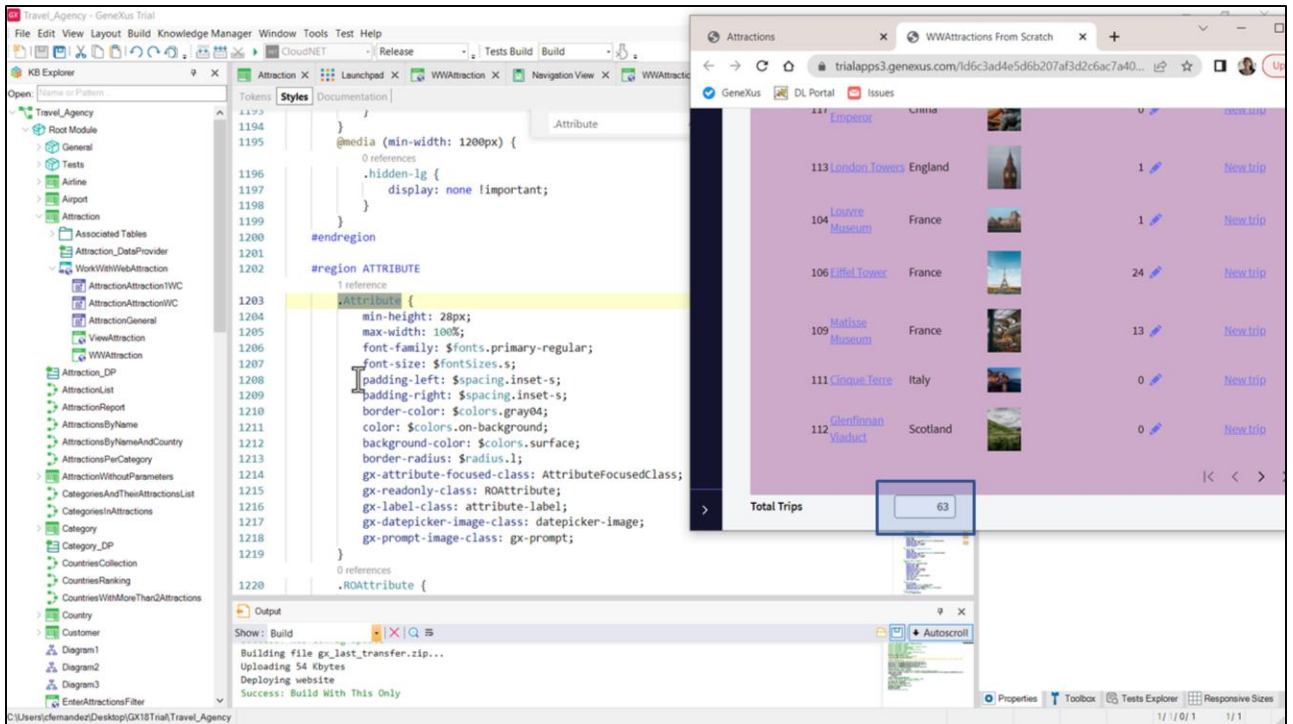
To understand better how to style our controls, let's examine the variable we have in our panel.

Suppose we want to change the color of the label, which is highlighted in black, and have the variable's contents also appear in a different color and set against a different background color.



Variables by default are input variables in Web panels. We had changed it to Readonly, but let's change it back to the default value for a moment.

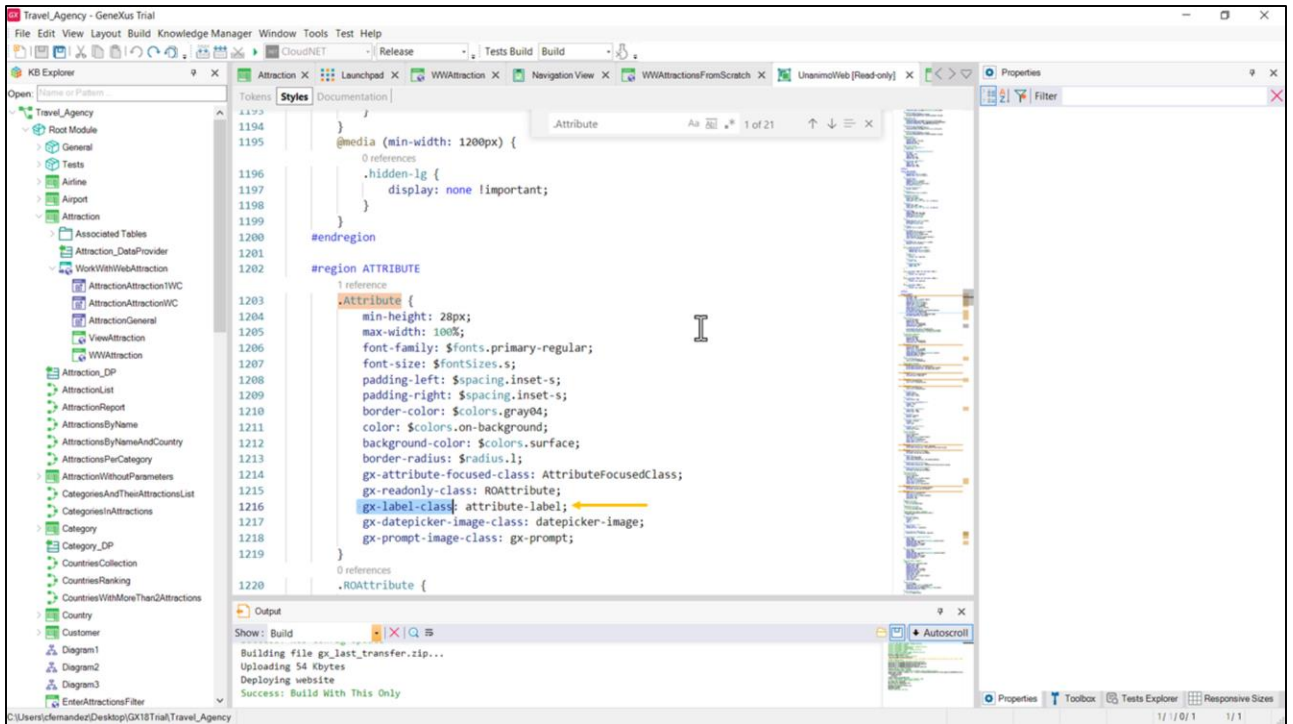
Let's run the build and, meanwhile, note it has the Attribute class associated with it.



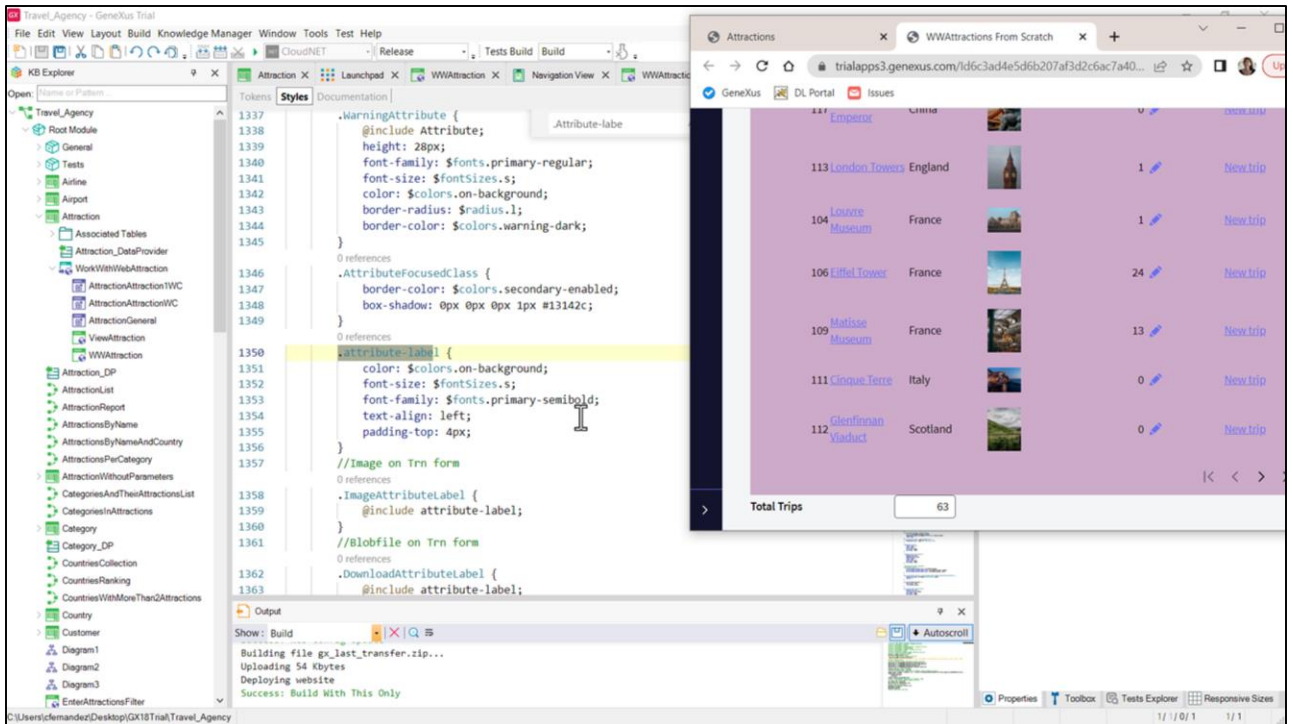
If we look for it, we see all these properties configured. We run... refresh...

Font-family, font-size, padding left and right, border-color, color, background-color, border-radius... all this is being configured, the variable field.

And where is everything related to the label configured? In another class, one that sets the label's style. And how do we tell this class that the one that controls the design of the label is that other one?

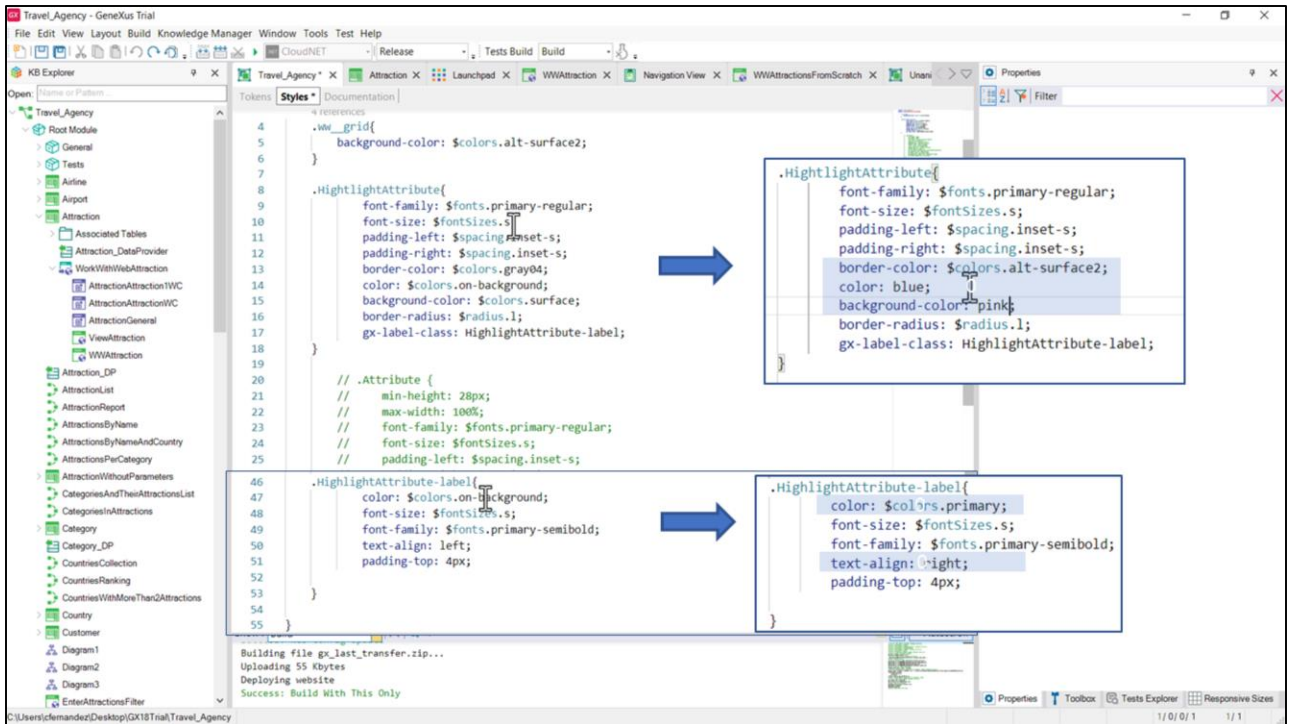


Through this GeneXus property... Here we are indicating that the class named attribute-label is the one that sets the style of the label. Let's look for it...



For example, here we see the font in semibold, the text aligned to the left, and with padding of 4 pixels in relation to the upper border of the cell where the control is located...



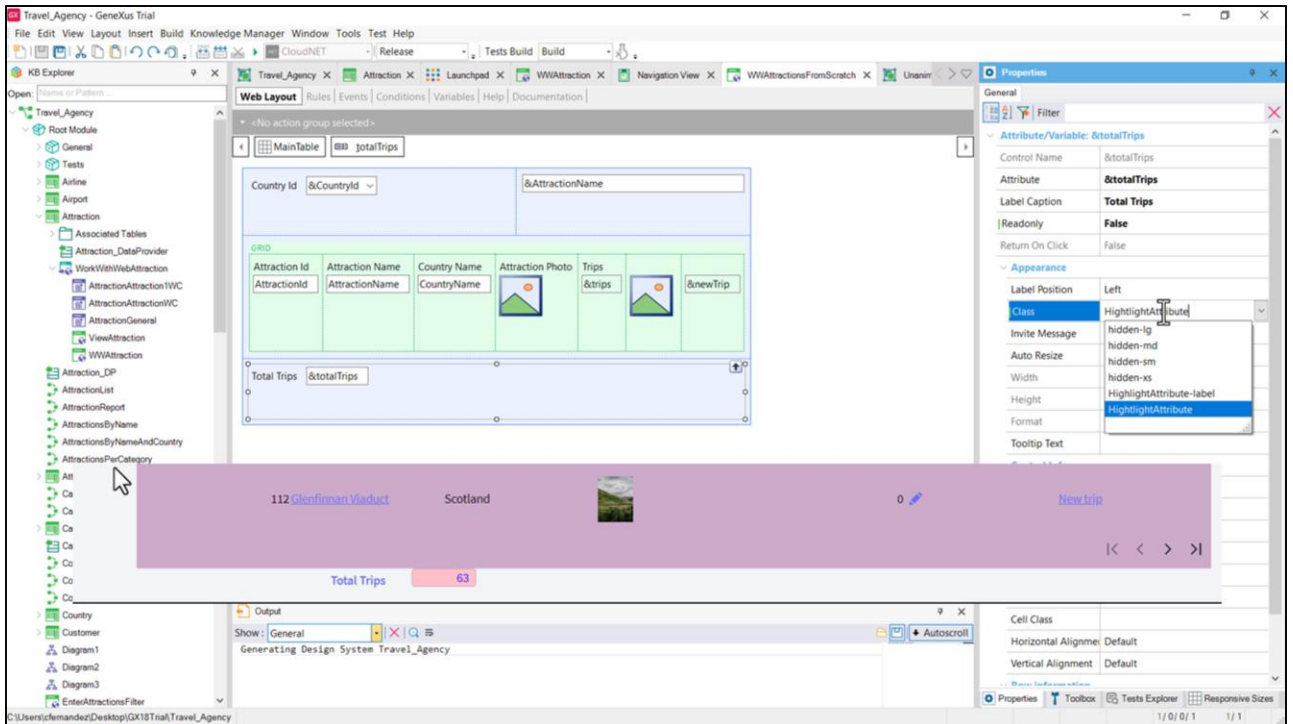


Let's copy both classes to our DSO and make changes.

We could modify these here, but that would affect all the controls that have them as classes. It's better to create special classes to customize our special use case that will correspond to a highlighted attribute or variable. Let's create, then, a `HighlightAttribute` class.

Let's copy all these properties of the `Attribute`, and that of the label class. We will call it in this way. Let's comment out the `Attribute` class that we won't modify at all, and select the new one to style the label. In addition, let's copy the properties of this other one.

Now let's make changes. For example, to the colors... and align the text of the label to the right. Let's save the DSO.



Now we change our variable's class to the new one and run it.

We see all the changes.

GeneXus by Globant

<No action group selected>

MainTable totalTrips

Country Id &CountryId &AttractionName

GRID

Attraction Id	Attraction Name	Country Name	Attraction Photo	Trips	&newTrip
AttractionId	AttractionName	CountryName		&trips	

Total Trips &totalTrips

Properties

General

Filter

Attribute/Variable: &totalTrips

Control Name: &totalTrips

Attribute: &totalTrips

Label Caption: Total Trips

Readonly: True

Return On Click: False

Appearance

Label Position: Left

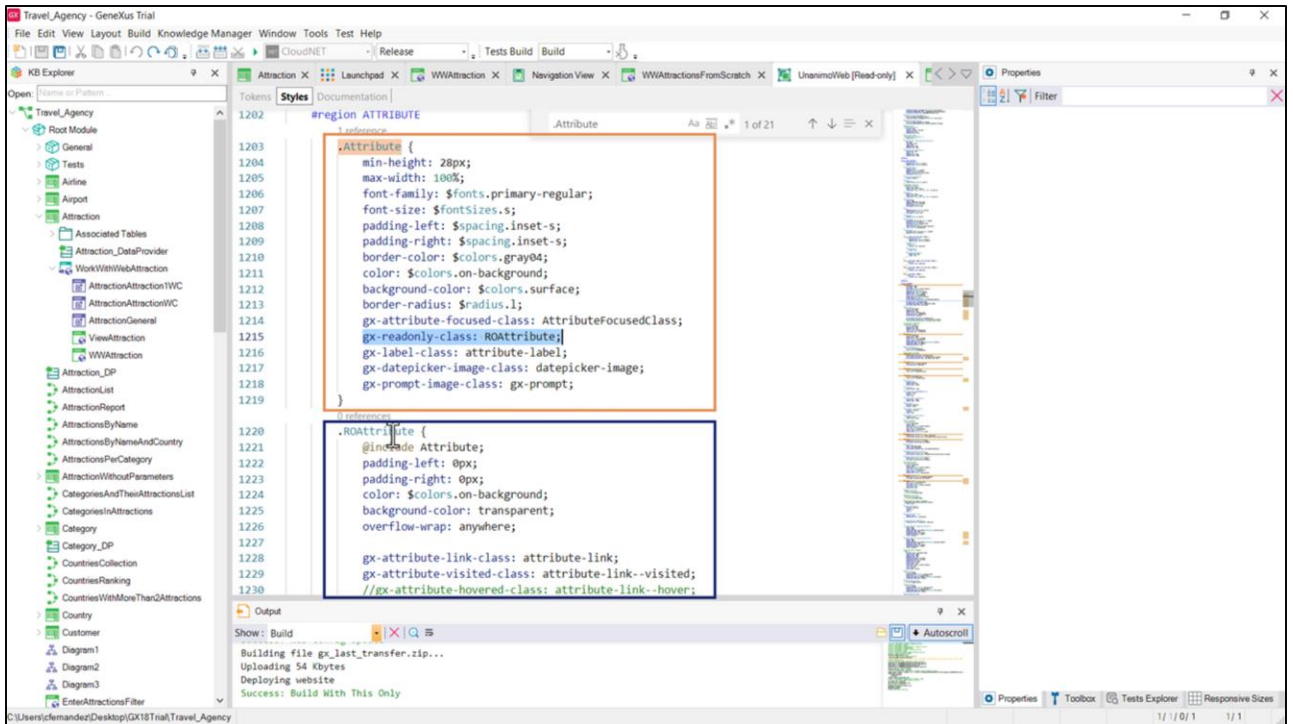
Class: HighlightAttribute

109	Matisse Museum	France		13	New trip
111	Cinque Terre	Italy		0	New trip
112	Glenfinnan Viaduct	Scotland		0	New trip

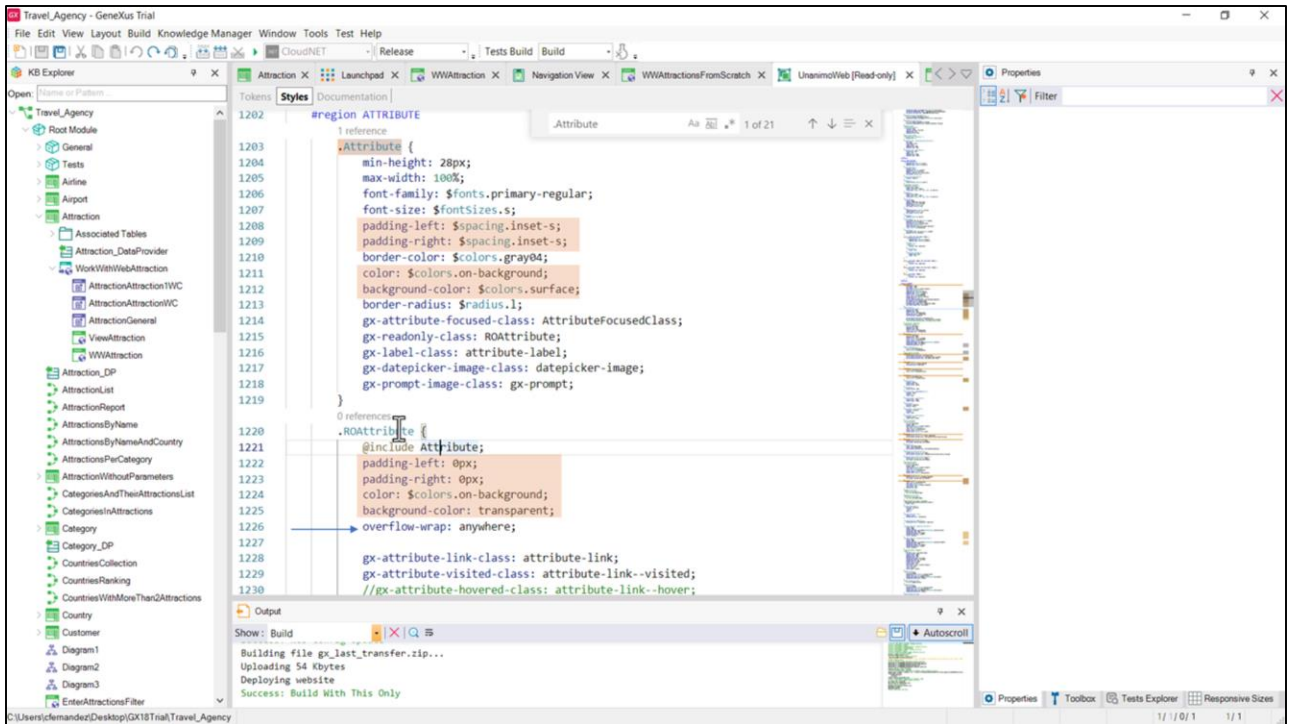
Total Trips 63

Total Trips 63

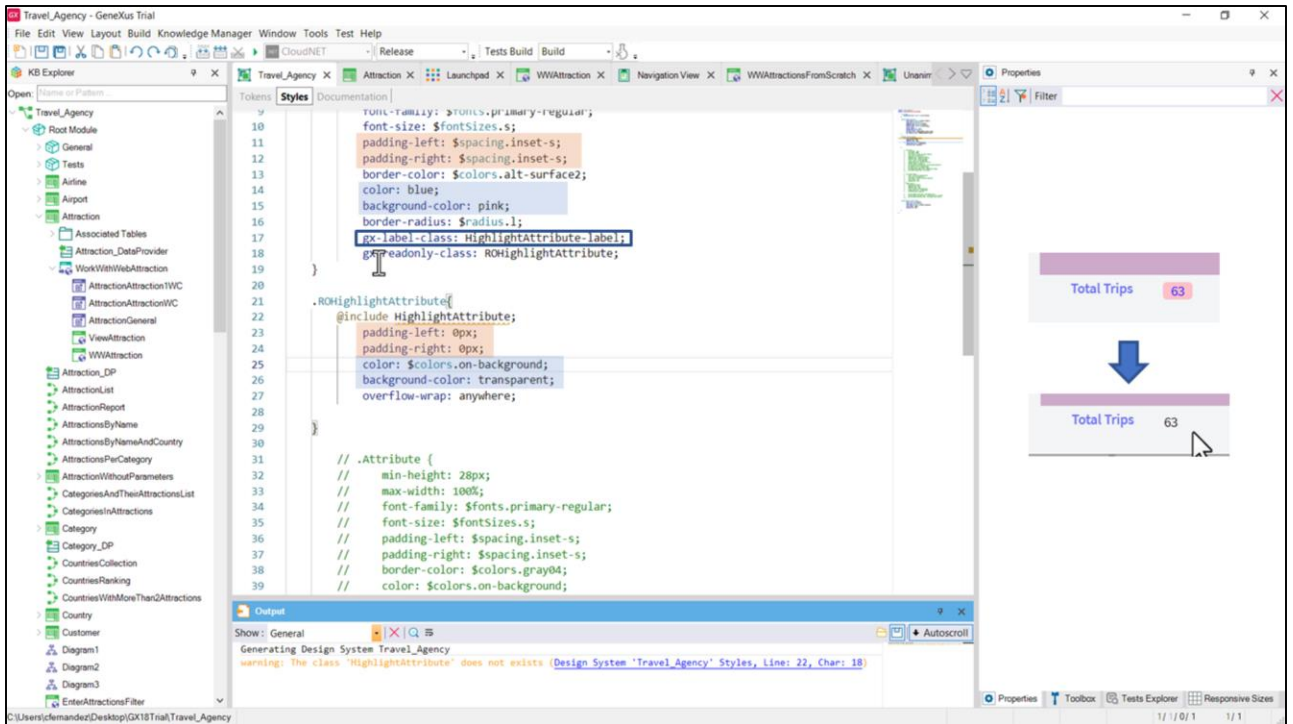
What will the variable look like if we now change it to Readonly?



In the properties of the predefined Attribute class, we can see it has this property: `gx-readonly-class`, which is defining `ROAttribute` part of the style when this Attribute class is applied to an attribute or variable that is readonly. That is to say, if it is not readonly, this property will not have any effect on the control, but if it is, then to the properties of the Attribute class the properties of this other one will be added to determine the style, overwriting those that are repeated.



Here we see that everything in the Attribute class is included but these properties are overwritten, and this other one is added.



Let's copy the class to our DSO so we can see it.

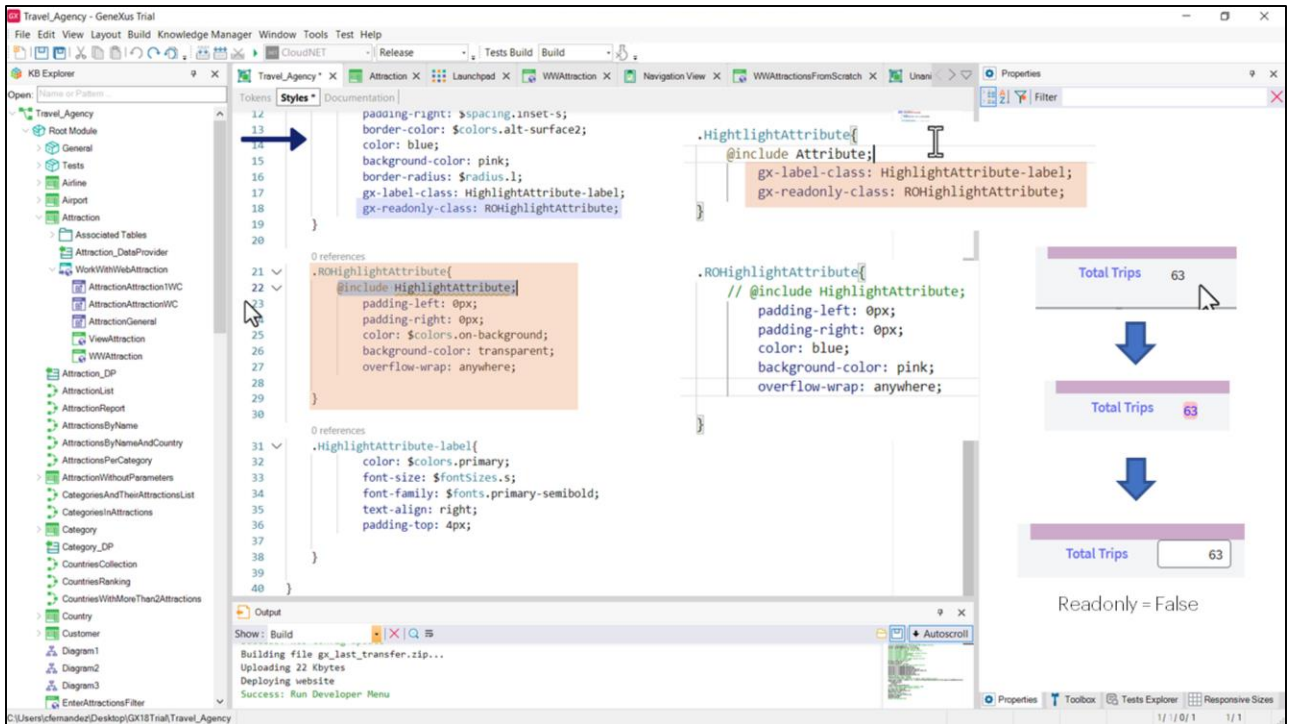
Let's add to our HighlightAttribute class the property that will indicate the class when the attribute or variable control is readonly. We give it this name.

And here we define it. We can include the HighlightAttribute class to secure the shared properties, and modify and/or add the ones we want. For example let's copy these...

The padding left and right is removed; we leave it at 0, and overwrite the color and background-color.

Let's try... We refresh... And see how these properties are now being applied.

We clean the DSO.



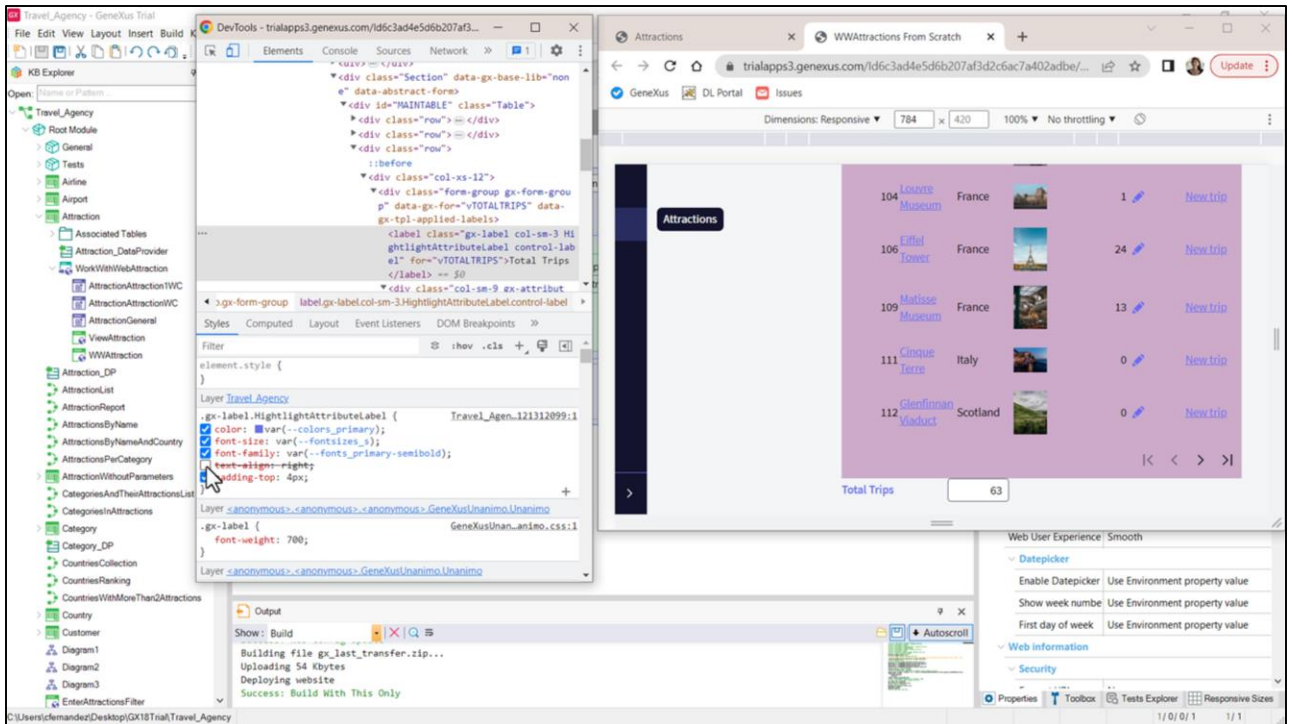
In short: the HighlightAttribute class will always apply to the variable, but when it is readonly this other one also applies (so it was not really necessary to include the first class in the second one).

Now let's make the following change: let's set the blue color and pink background when the control is readonly, and use instead what comes predefined in the Attribute class in Unanimoweb for the class in general, overwriting these two properties because we want to apply our two classes in those cases.

Let's try it at runtime. Our variable is readonly, so now we see it in pink and blue.

And if we now make it non-readonly... it takes the style of the Attribute class but with the style of our class label.

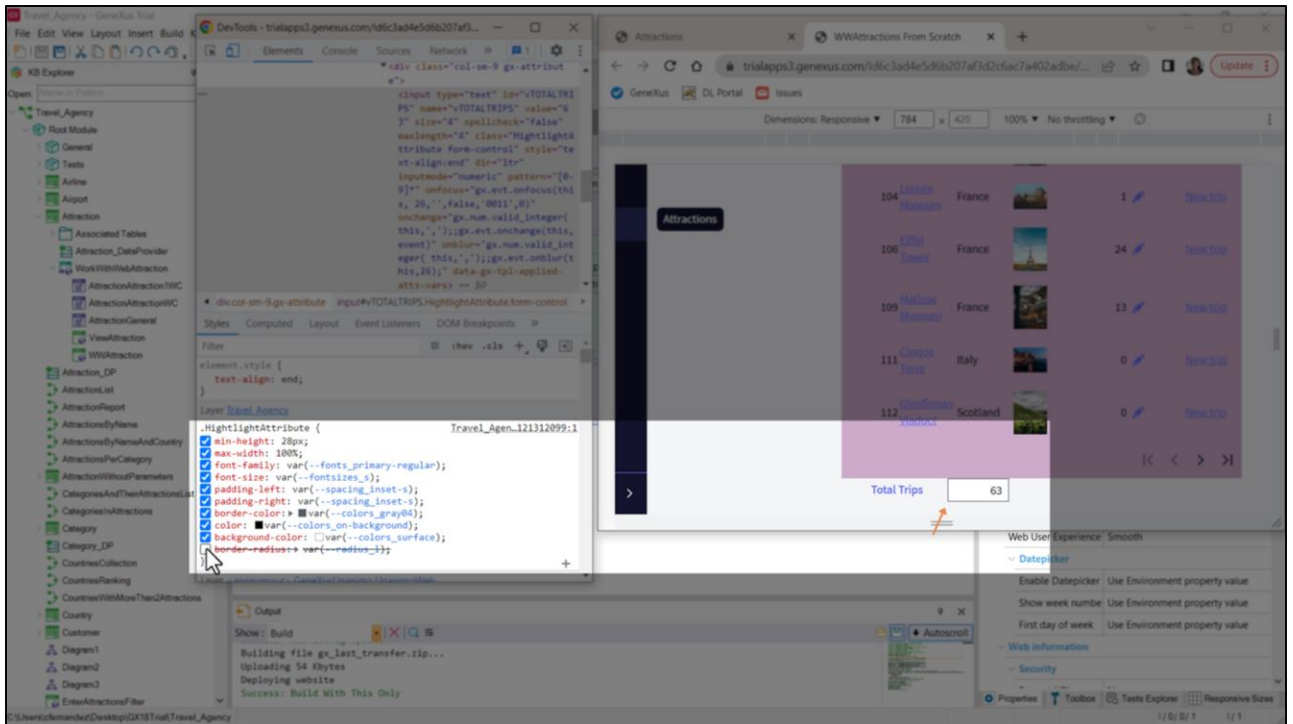




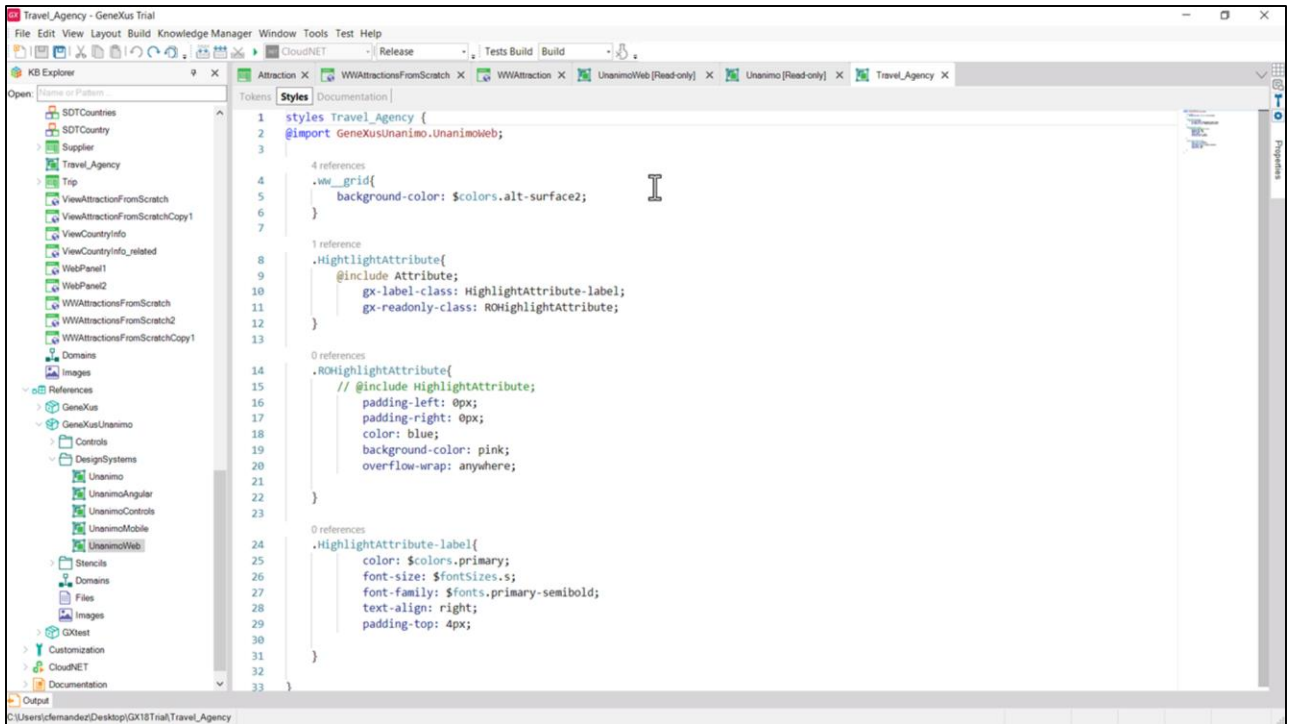
One last thing: to see what properties are being applied in the CSS that controls the appearance of this HTML, since we had pressed F12 we can inspect the elements of the HTML and its appearance.

Here we have the element corresponding to the label... and see this class, with the properties... We can remove this property and see how the label stops aligning to the right... The same with padding-top. Note how it changes depending on whether or not we have it. The same happens with the color.





And if we now look for the input control itself, it has the HighlightAttribute class with all these properties. Let's try to remove the padding-right... and the border-radius...



In this video, we wanted to complete a small sample of the technical aspects related to adding a style to the User Interface, using as an example the customization of the Design System of Unanimo for web, a predefined Design System. But, of course, we could create a Design System completely from scratch.

As you may have noticed, most of the properties of the classes we are working with are identical to the CSS properties. In addition, there are some that are specific to GeneXus.

There is an entire world to discover that only requires you to know the GeneXus controls and which properties to use in the Design System.

We encourage you to continue exploring.



GeneXus by Globant

**GeneXus**<sup>™</sup>  
by Globant

[training.genexus.com](https://training.genexus.com)