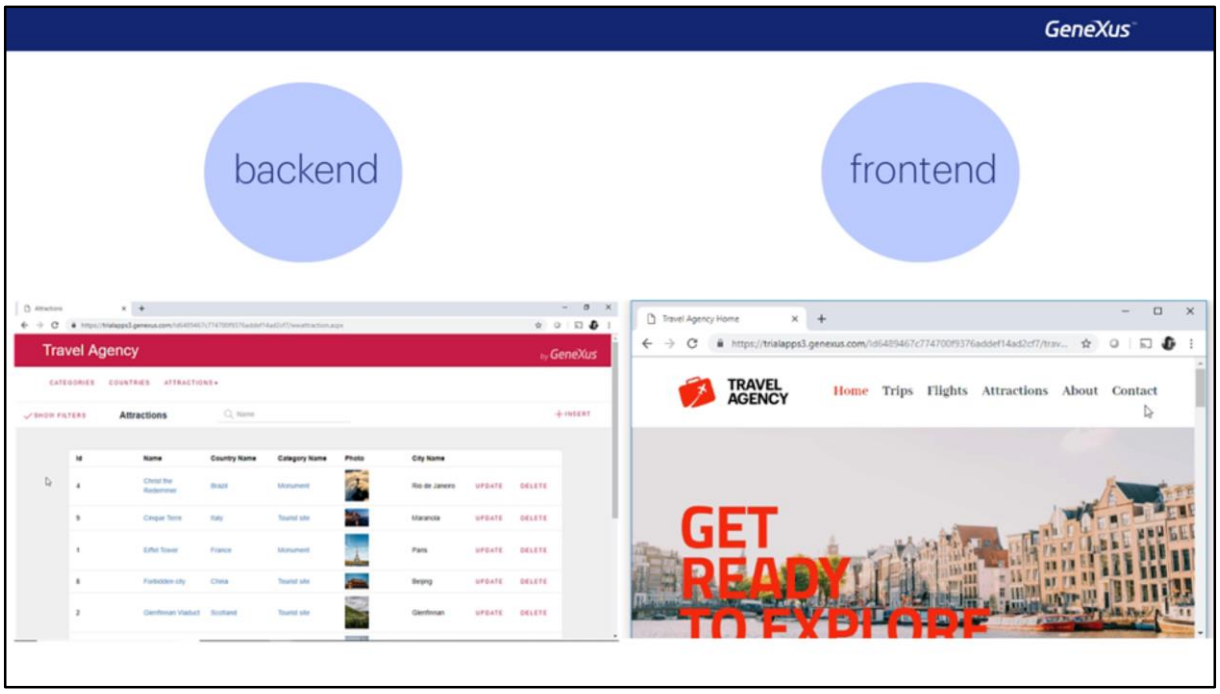
Design System in GeneXus

Default

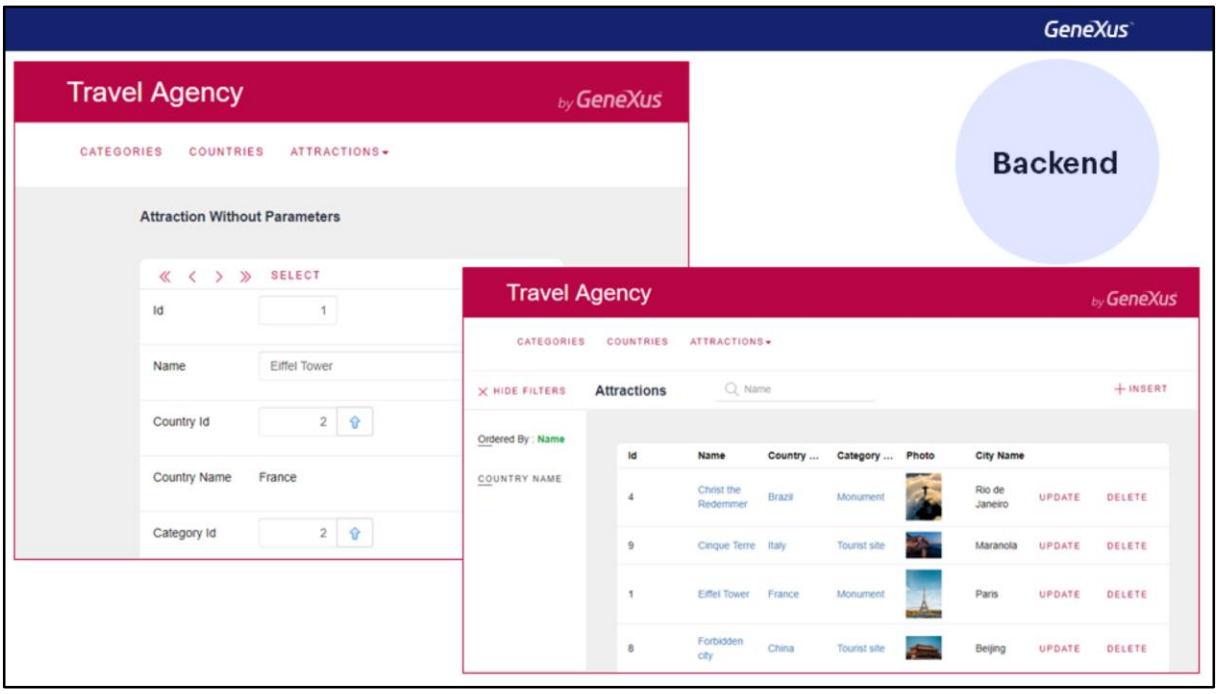
GeneXus 16



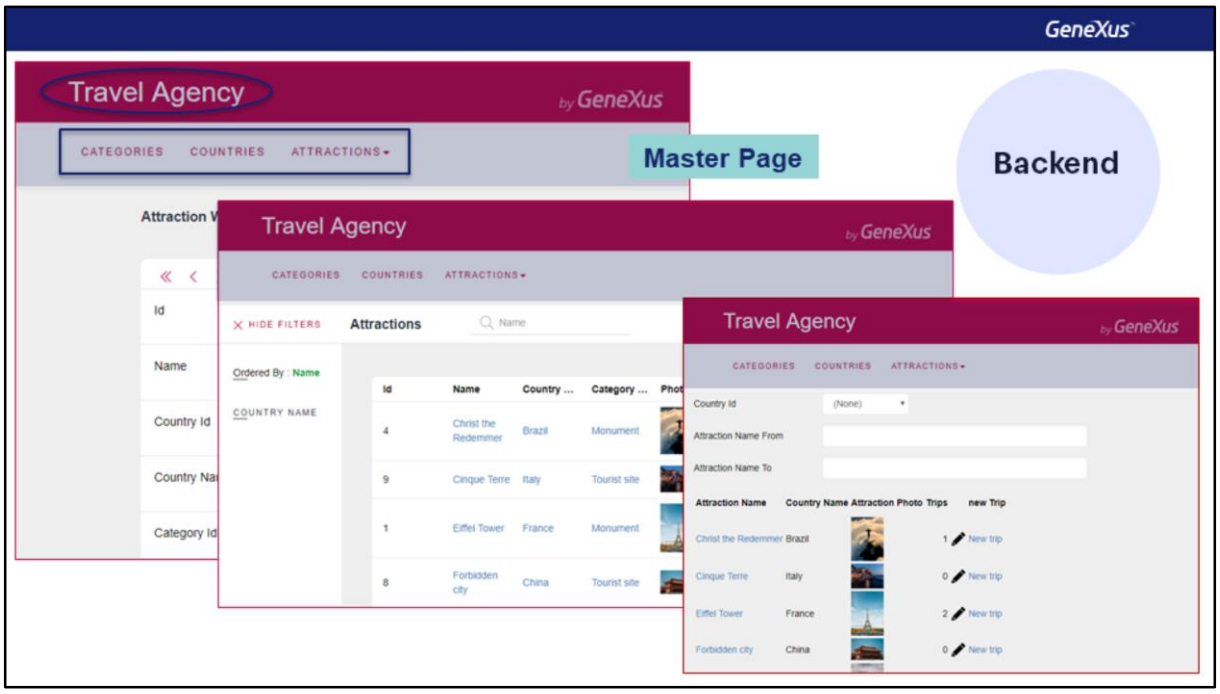
[DEMO: https://youtu.be/l_VA02JfGYQ]

As we've said before, until now we had been working above all on **backend** screens. However, as we will see at the end, the **frontend** is completely analogous. The biggest difference is the greatest effort we will put into its design.

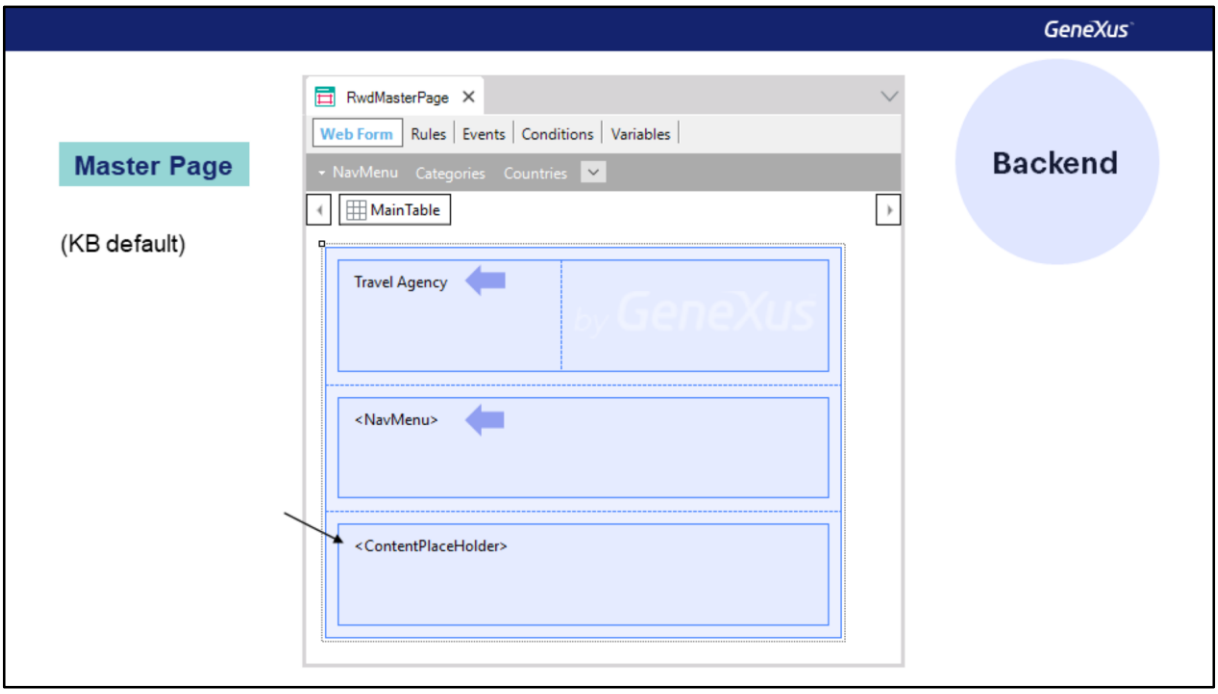
Let's start by the **backend**...



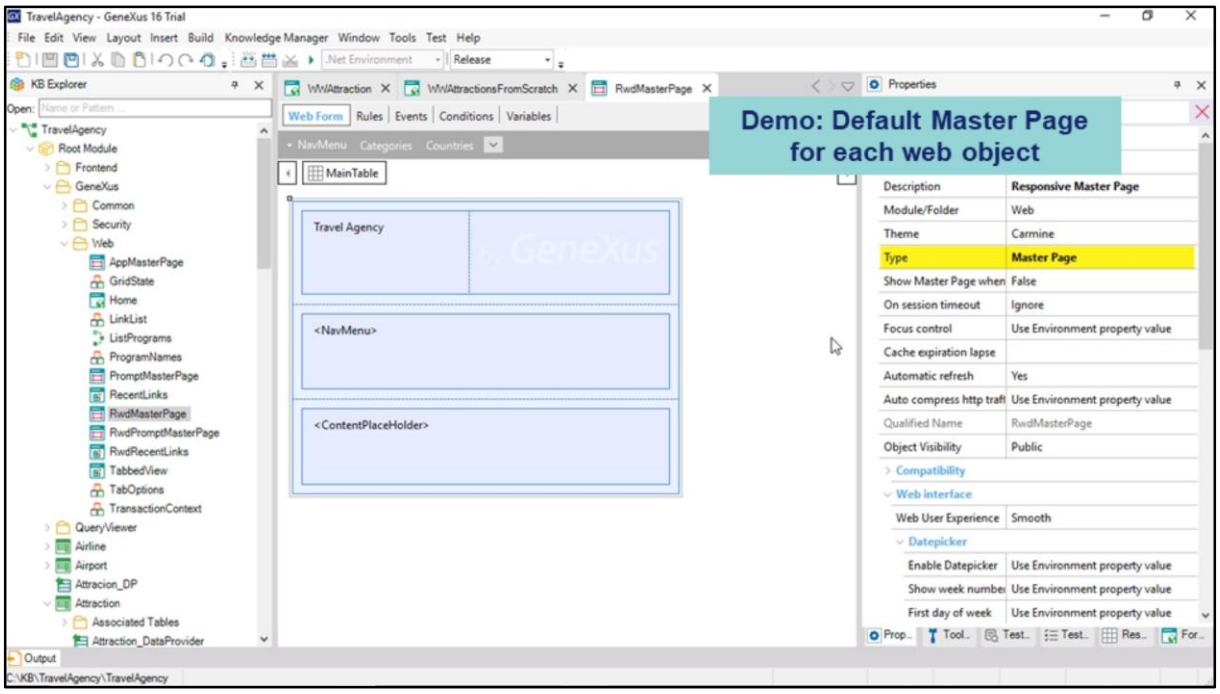
If we pay attention to the transactions we have been using, and to the panels created by the Work With pattern, we see that there are elements that give consistency and coherence, without us having worried about achieving this. This means that GeneXus is already doing something for us in terms of a predetermined Design System, even if it is elementary.



For example, we see that all screens (even the ones made from scratch, like this web panel) have a common header. In this case we have customized it, changing the default text to "Travel Agency" and removing the "Recents" section that was predefined (let's not forget that we've added a menu bar!). The way to achieve it is with the web Master Page object (Here we see the modified text block and the added menu)



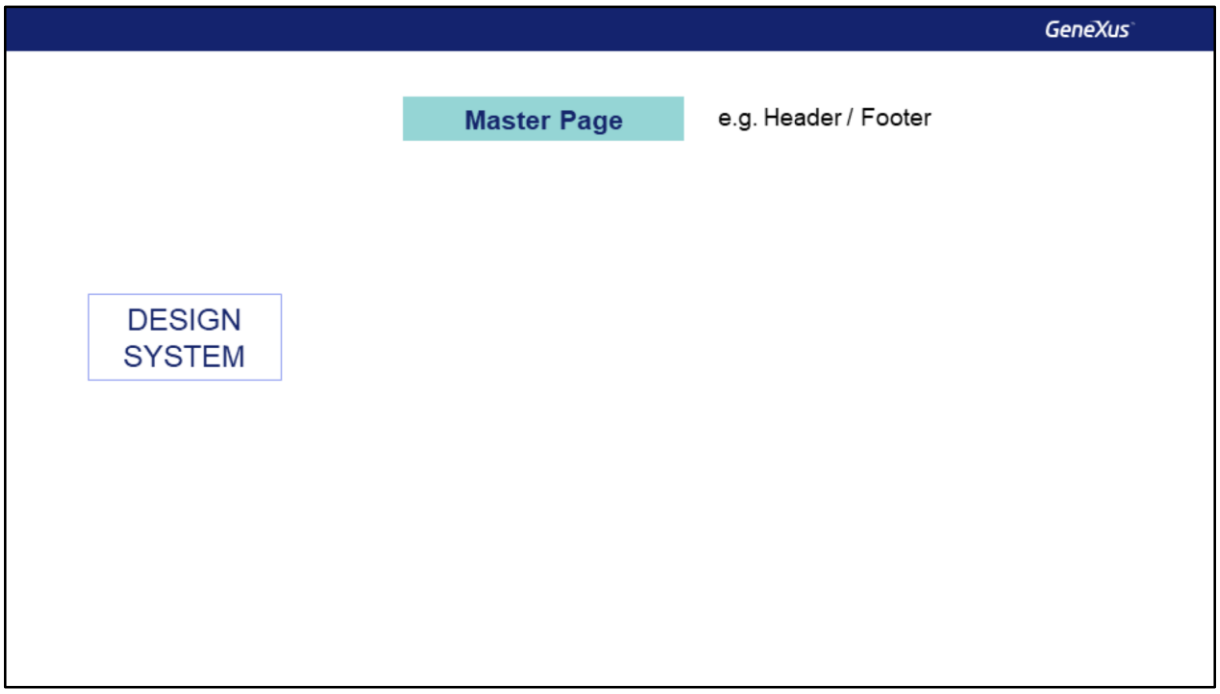
By default, the KB version comes with this predefined Master Page, which means that it will be applied to all web objects that are created.



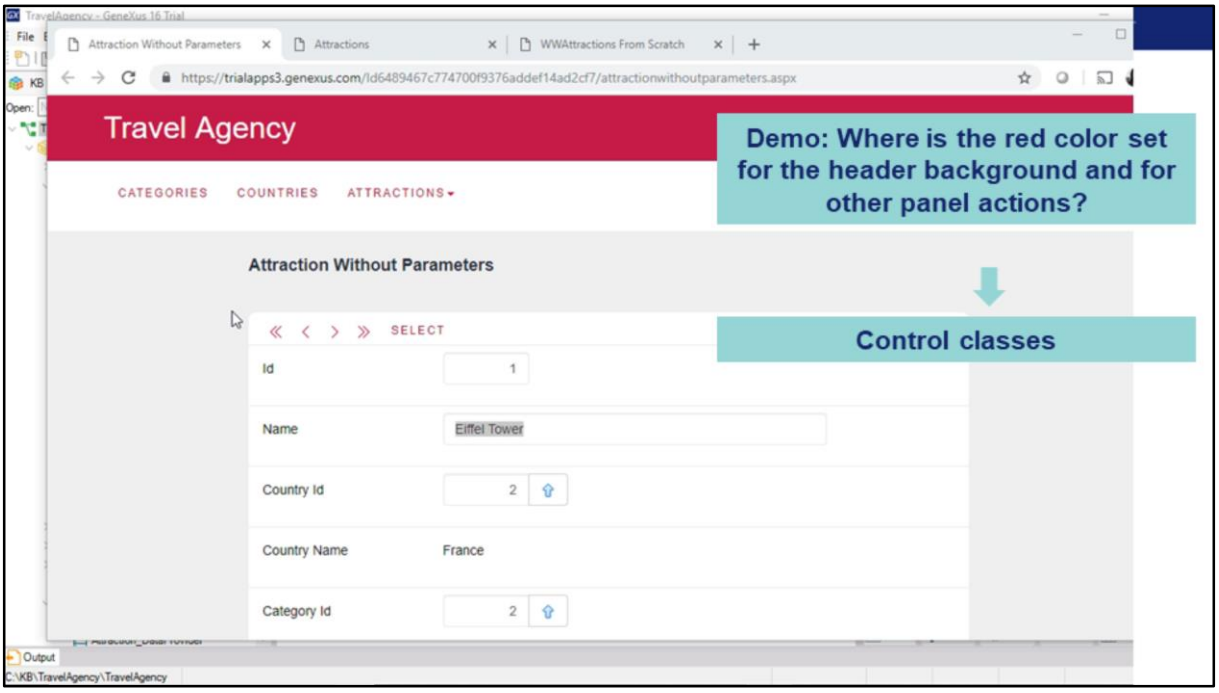
[Demo: https://youtu.be/PYBRdASQBvs]

So, in the properties of the transaction we saw at runtime, we see that it has defined as its Master Page this one, the default one. The same applies to the Work With element and the panel.

Note that the Master Page contains this control which indicates that this is where the content of each web page will be loaded.



So, GeneXus already offers a Master Page object to centralize what will be the common information that we want all pages to share.

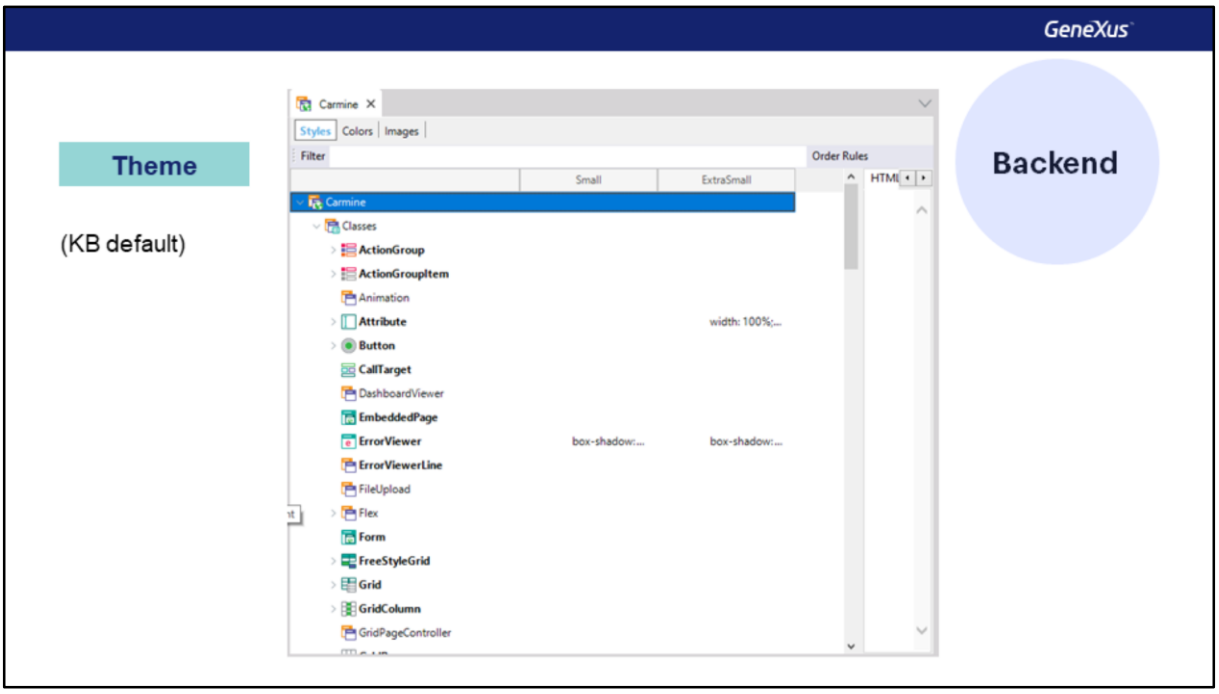


[DEMO: https://youtu.be/vkybJzp3uE0]

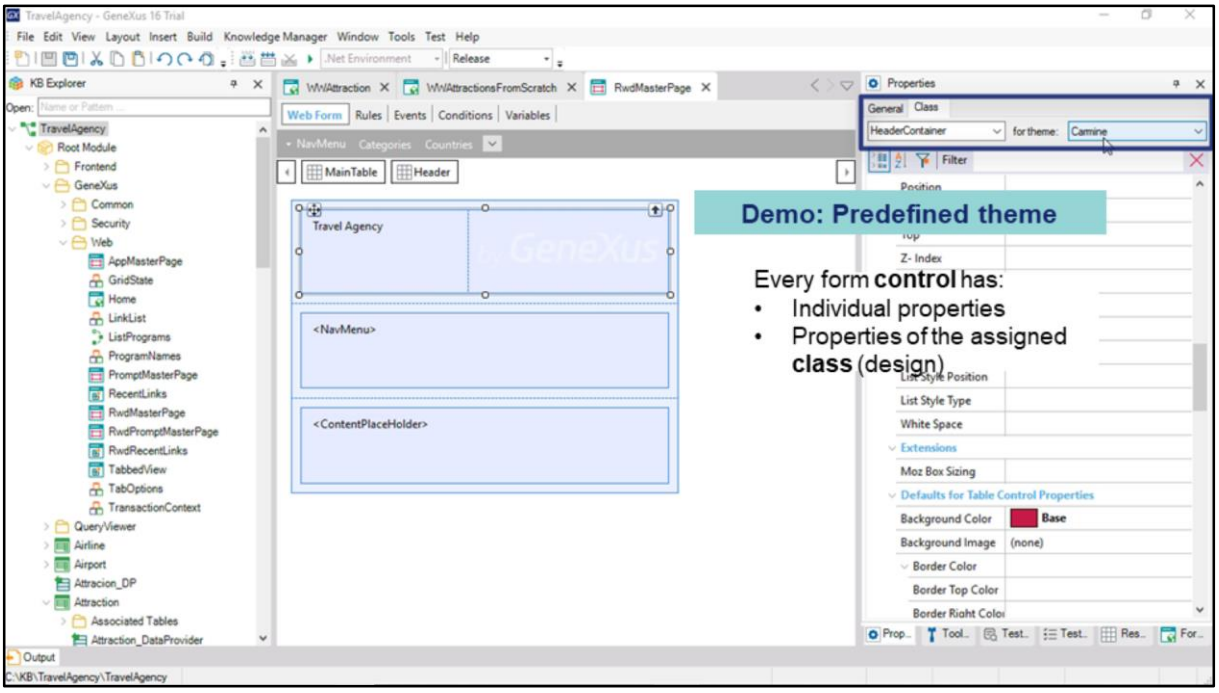
In addition, note that the base color, the one that stands out, is a type of red, which is used not only for the header, but for the buttons and other actions that are given to the user. Where is this configured?

We know this table has a red background at runtime, but we're not seeing it! Note that the control (in this case of the **responsive table** type) has a pair of defined **classes**, not coincidentally under the group "Appearance." If you select the **Class tab**, you'll see that the properties of the second one of the classes associated with that table are being edited. Also, that the things such as margins, paddings, and the red background color that we saw earlier are configured there!

Classes are useful because they allow you to centralize the design of that type of control (in this case, table). Then, in this case, other tables located in the same or in other forms can share the same definition, so if you want to change, for example, the background color from red to blue for those controls, it is not necessary to do it one by one in each control -that is to say, in each table- but it is done directly in the class and that already affects all the controls that have it associated.



Where are all classes configured? In the Theme object! Note that just as there was a predefined Master Page, there is also a predefined theme called Carmine.



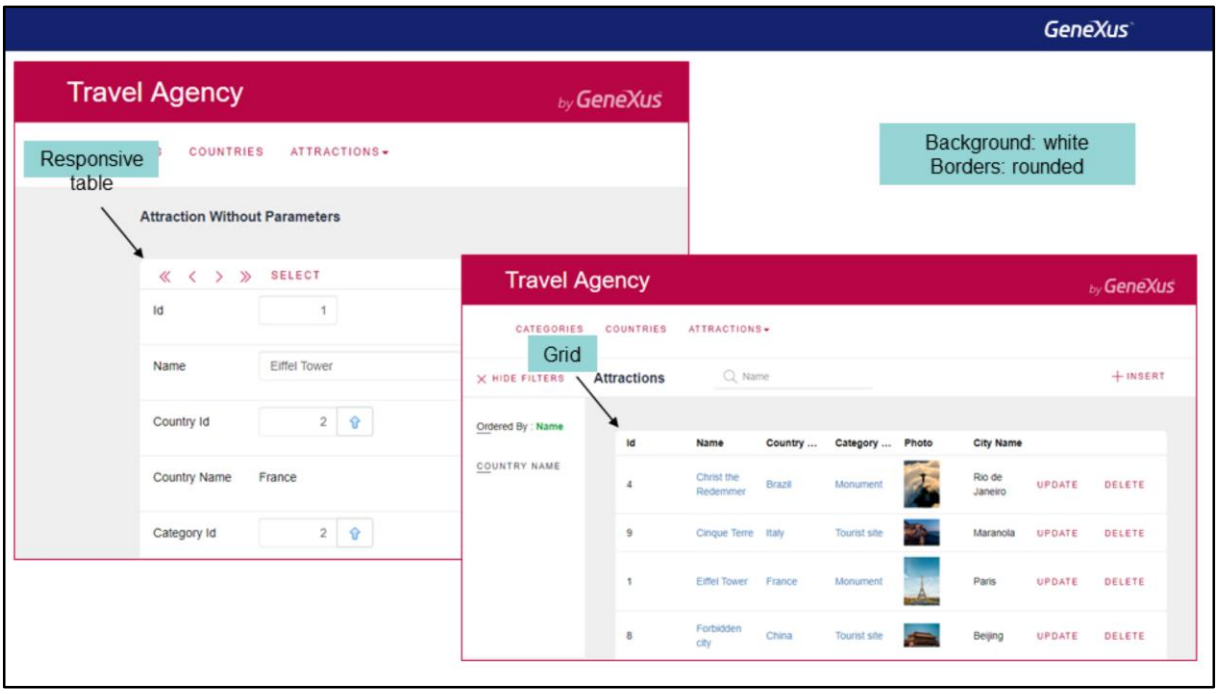
[DEMO: https://youtu.be/tkdOB-JPSgg]

As we will see later, we have created one of our own that will be used for the frontend. In the backend we use the default one. Note that when opening it there are classes grouped by type of control. This is where everything is centralized. Also, note that the Colors tab shows the color palette of the same name... This is where the Base color is configured.

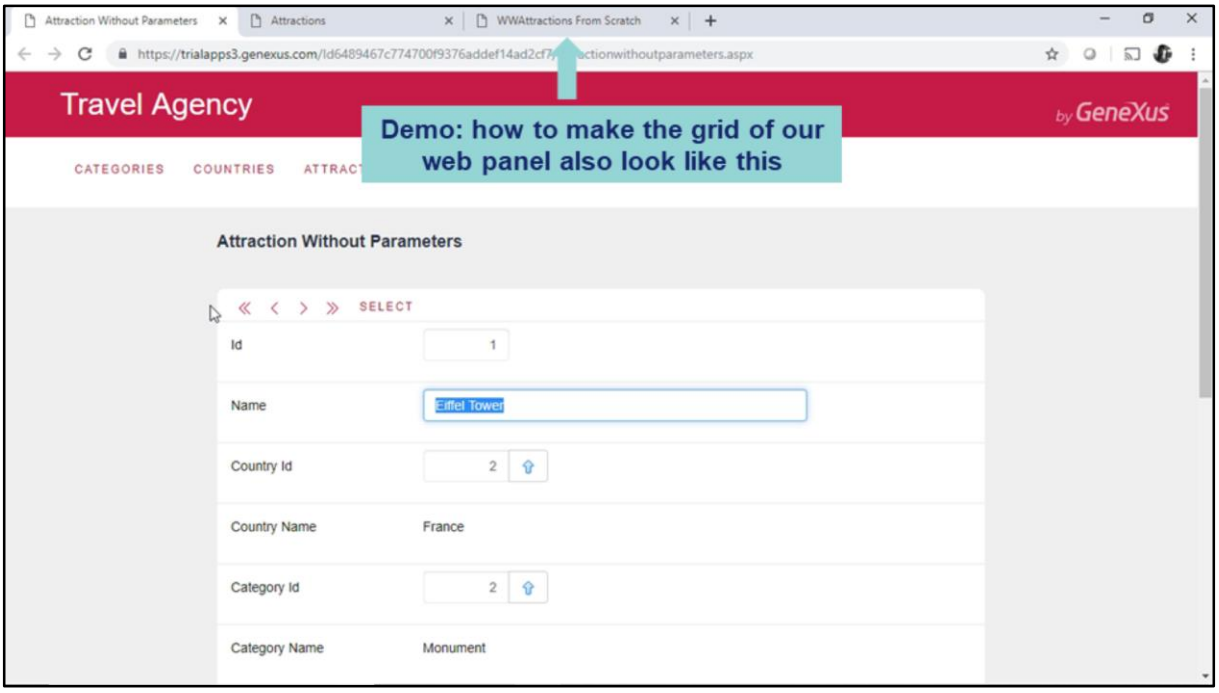
If you look at the Update or Delete options of the Work with pattern which, as we know, have the base color at runtime, you'll see that the type of control is Attribute/Variable. Also, that its assigned class is this predefined one. We can edit its properties from here knowing that what we modify here will be modified at central level, in the Theme object, so that any other attribute/variable control that has this class will also be modified.

In short, every control in a form has individual properties, which can be configured for it alone... And properties that will come from its assigned class, which are those that involve design aspects.

We can already note that in this particular case in which the control is, in addition, a grid column, another class is displayed to be associated with it, which is the class that rules the design and behavior of the column.



And if we look again at the predefined design of the transaction and the Work With element, we see that the most relevant data is shown in white on grey, with rounded edges.



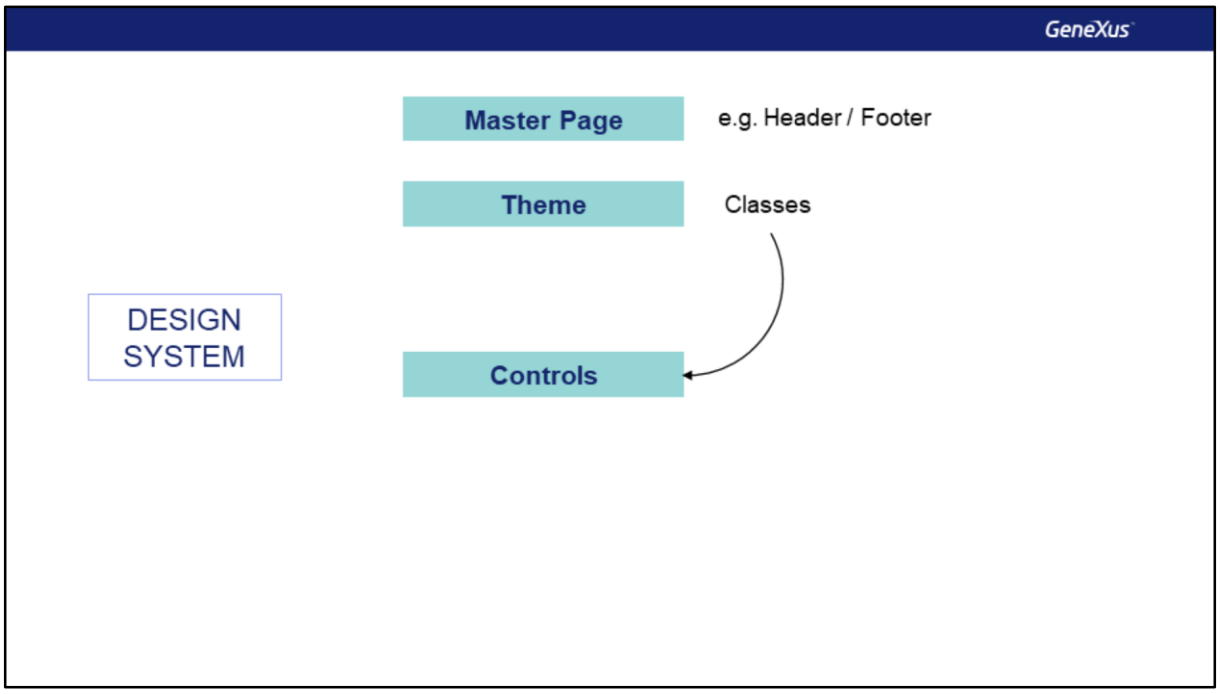
[DEMO: https://youtu.be/XhDqvve5E74]

This is because the table where this data is found in the transaction was assigned this class... which specifies white as the background color and a radius for the four edges...
...the Work With grid was assigned this other class, which also specifies white as the background color, and the same values as the other for the radius.

On the other hand, if we look at the grid of the web panel we created, we can see that its class is called Grid, which doesn't have any of this configured... And this is how it looks at runtime, with all the columns next to each other, with the default gray background. Note what happens if we change its class for the same one of the WorkWith grid...

And to maintain the consistency of the Design System, this grid action should look like these others...

Also, the column title shouldn't be here...



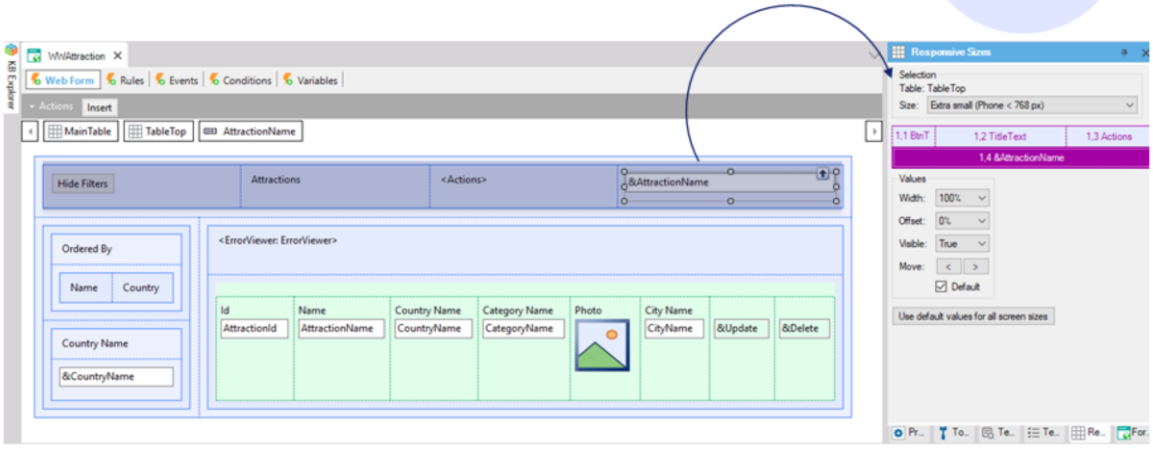
From what we've seen, we can infer that a very high percentage of the Design System is specified in the Theme object, through its classes (the predefined ones and the ones we add). These classes are the ones assigned to the controls of the form, thus making it possible to abstract the design and establish a certain logic, so as to unify it according to the function played by each control.

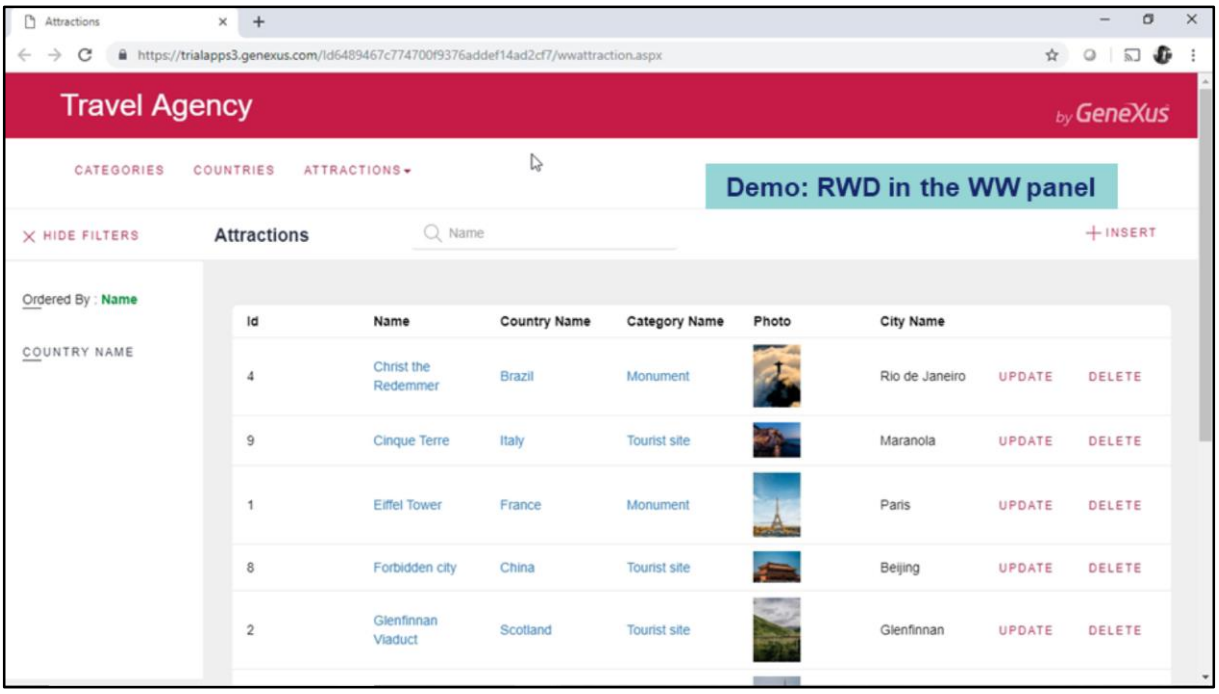
Therefore, a crucial aspect of design will be to identify the classes that we'll need for the controls, and apply them.

Responsive Web Design

Backend

- Responsive Sizes of Responsive Tables





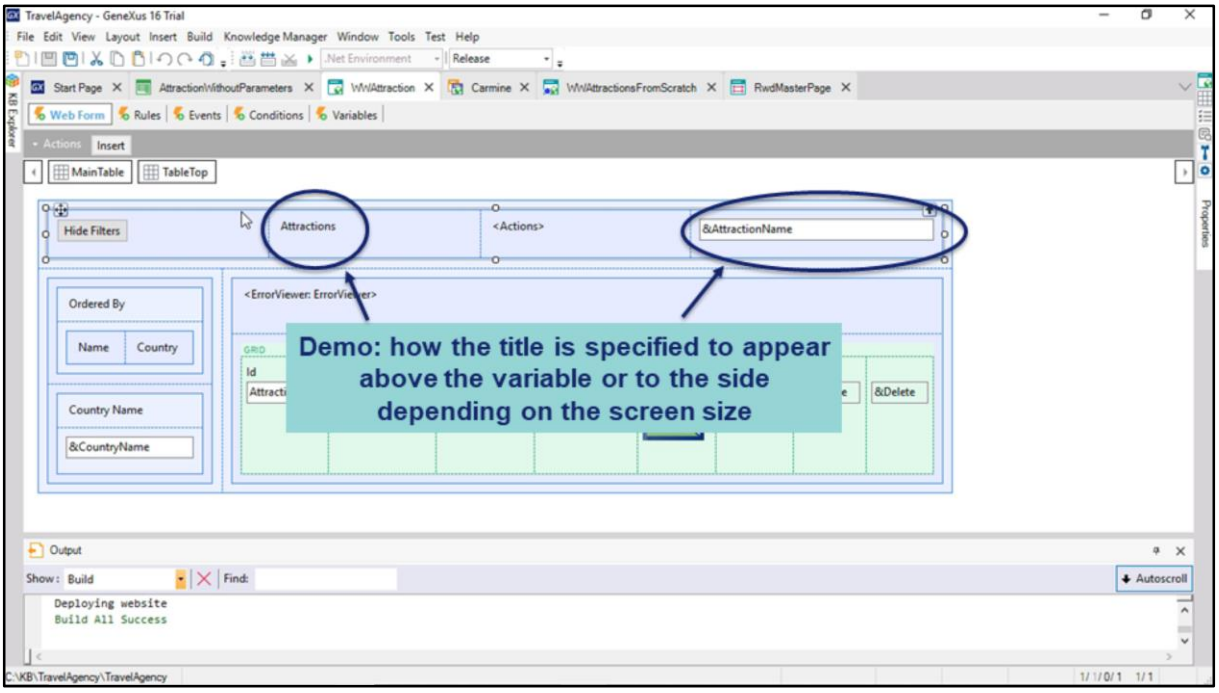
[DEMO: https://youtu.be/aRtwIstT7BU]

Returning to the predefined settings (remember that some browsers offer a display mode –in this case by pressing F12– that allows you to vary the size of the screen to see how it would look)...

We can see that if we reduce the screen of the Attractions Work With enough to match a phone screen, the filters appear differently, and the menu we had added appears as the typical hamburger menu. And, for example, the field for the user to enter a filter by attraction name appears below and not to the side.
And with the screen width reduction, almost all columns also disappear except for the one showing the name and Update and Delete actions.

We had named this behavior as "responsive," and the web design that considers it as Responsive Web Design. A design that is not responsive is not acceptable today.

Part of responsiveness is established through responsive tables...



[DEMO: https://youtu.be/qsbLGSHDzHM]

For example, this table, which we see is responsive, contains these four controls. The second one corresponds to this textblock, and the last one corresponds to the filter variable that we saw. If we edit the table properties, we find this one labeled “Responsive Sizes.” Here you establish how you want to display the table contents according to four screen sizes: **Extra small**, which corresponds to phones, usually with a screen width smaller than 768 pixels; **Small**, corresponding to tablets of a width greater than or equal to 768 pixels; **Medium** that corresponds to notebook or PC screens larger than 992 pixels, and smaller than 1200 pixels; for screens larger than or equal to that size, the size **Large** is used.

Here we can see that if the size is **Extra small**, the variable will appear below the title; and if the size is **Small**, the variable will appear to the right.

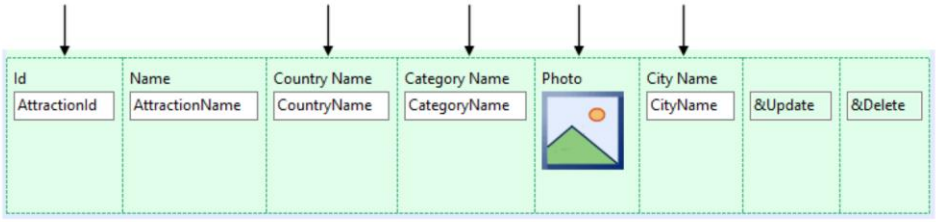
We’ve said that an important part of responsiveness, the most general one, is achieved by making use of these tables and properties.
But another part, a bit more specific, is achieved through the **classes**.

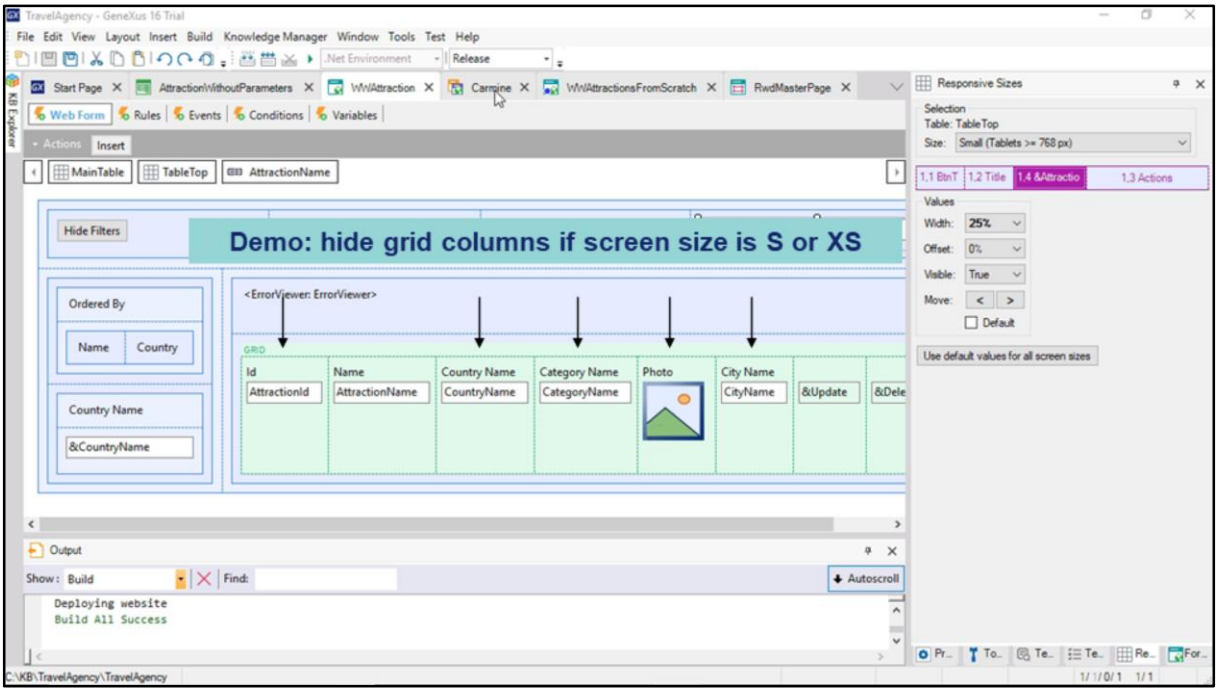
Responsive Web Design

Backend

- Responsive Sizes of Responsive Tables
- Theme Rules: Small, Extra-small and Default

For example: hide grid columns if Extra small or Small size. Not for larger sizes.





[DEMO: https://youtu.be/WXz42v6mpNI]

For example, hiding grid columns depending on screen size.

Note that the class of the attraction name column is WWColumn, but that to the other columns the class WWOptionalColumn is added.

If you select the theme, you'll see these two columns that allow you to vary the values of the class properties, according to the size of the screen.

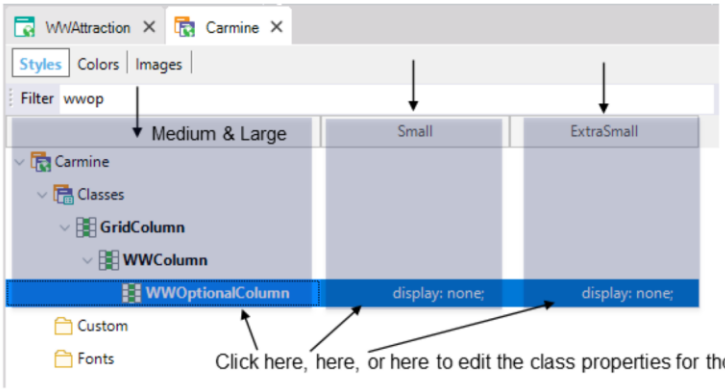
By default, only two are set: Small and ExtraSmall. But, actually, there are three, because the values in the default column correspond to Medium and Large.

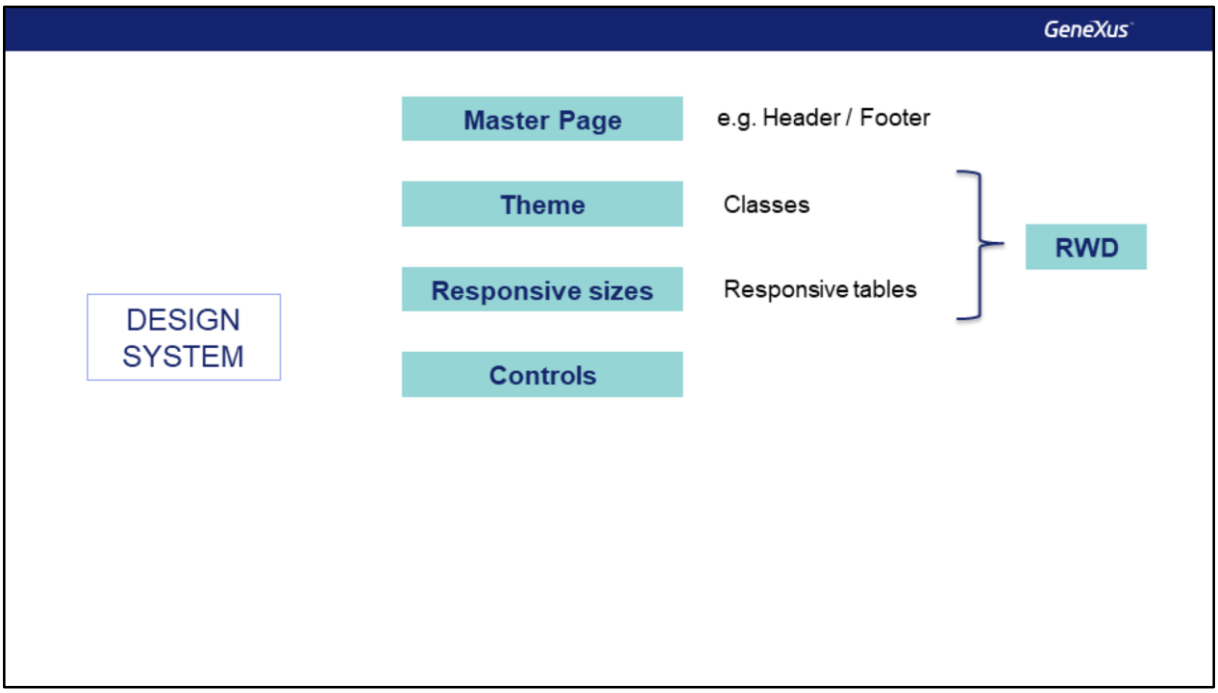
So, even if the Display property of this class doesn't say anything, it means that it will be displayed at runtime for Medium and Large screen sizes. For Small and ExtraSmall it has the "none" value. That's why they are not displayed.

Responsive Web Design

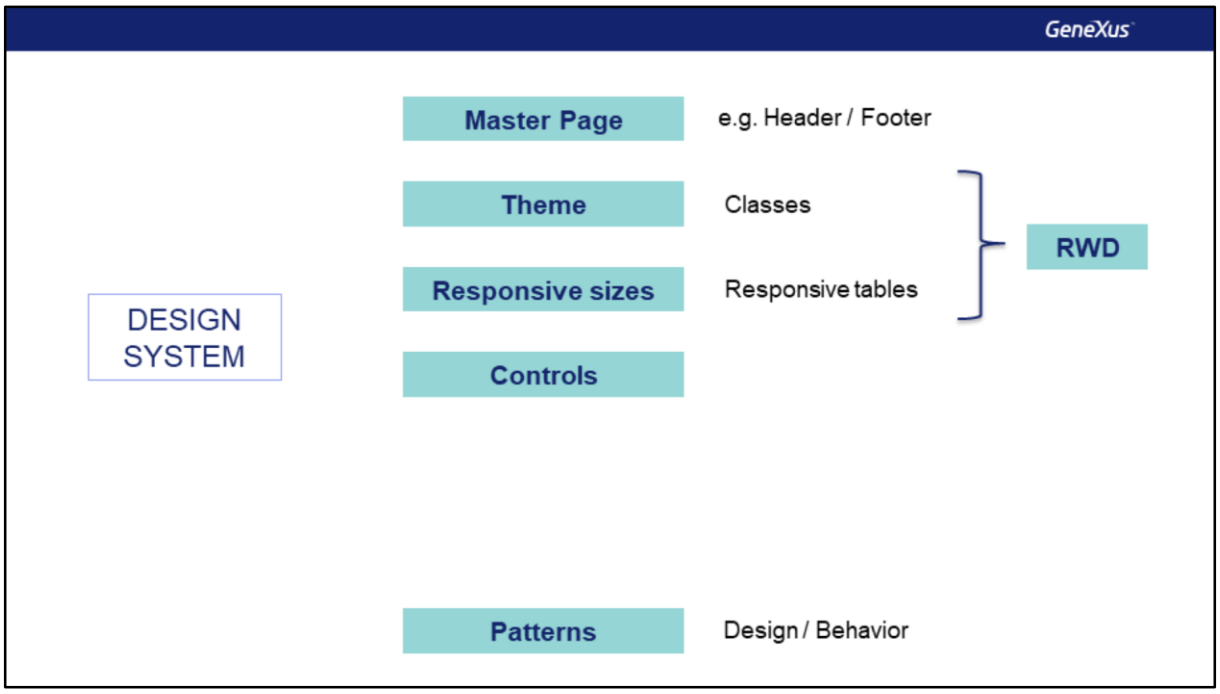
Backend

- Theme Rules: Small, Extra-small and Default





In summary: another important part of the Design System has to do with responsiveness, which is achieved both by varying the positions and visibility of the responsive table controls, as well as by varying the class properties according to size as well (through these columns of the Theme we saw).

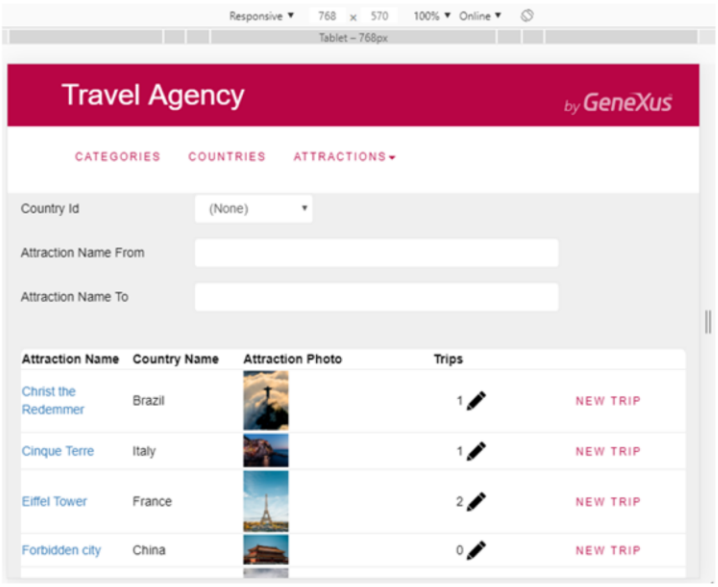


To achieve all this, we didn't have to do anything. The Work With pattern did it automatically, together with the theme. For this reason, we say that GeneXus provides a basic, predefined Design System that we can use in our panels.

GeneXus

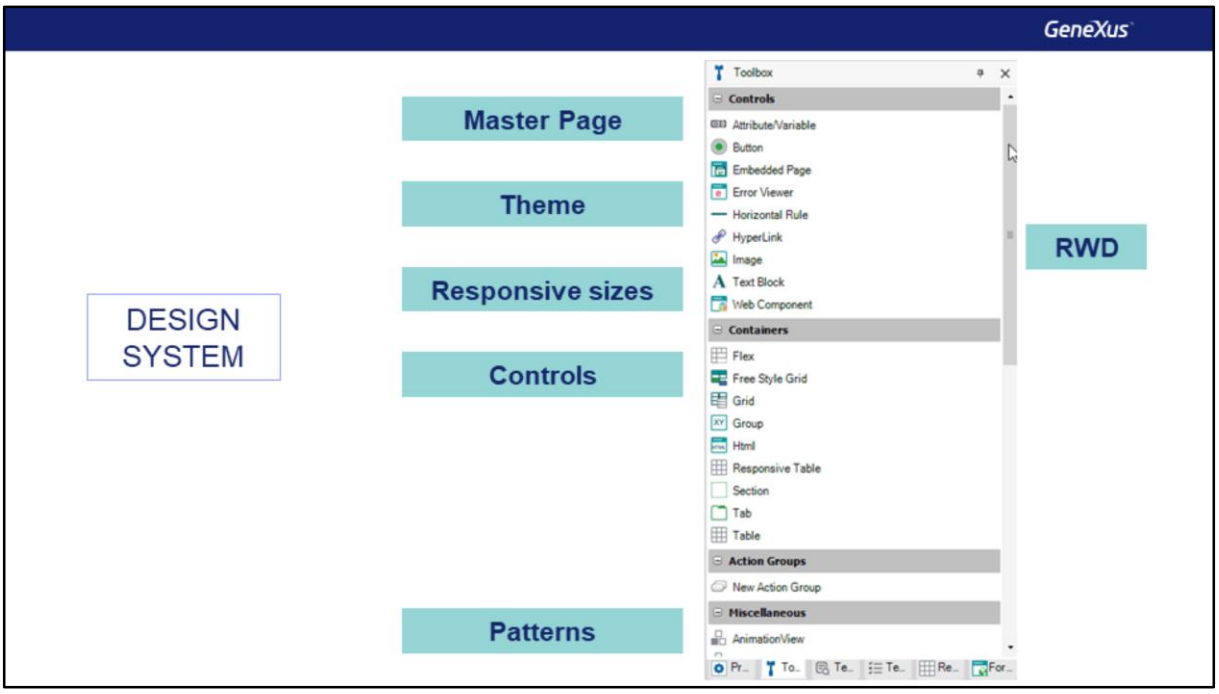
Task

- Try to hide columns from the grid of the web panel created from scratch



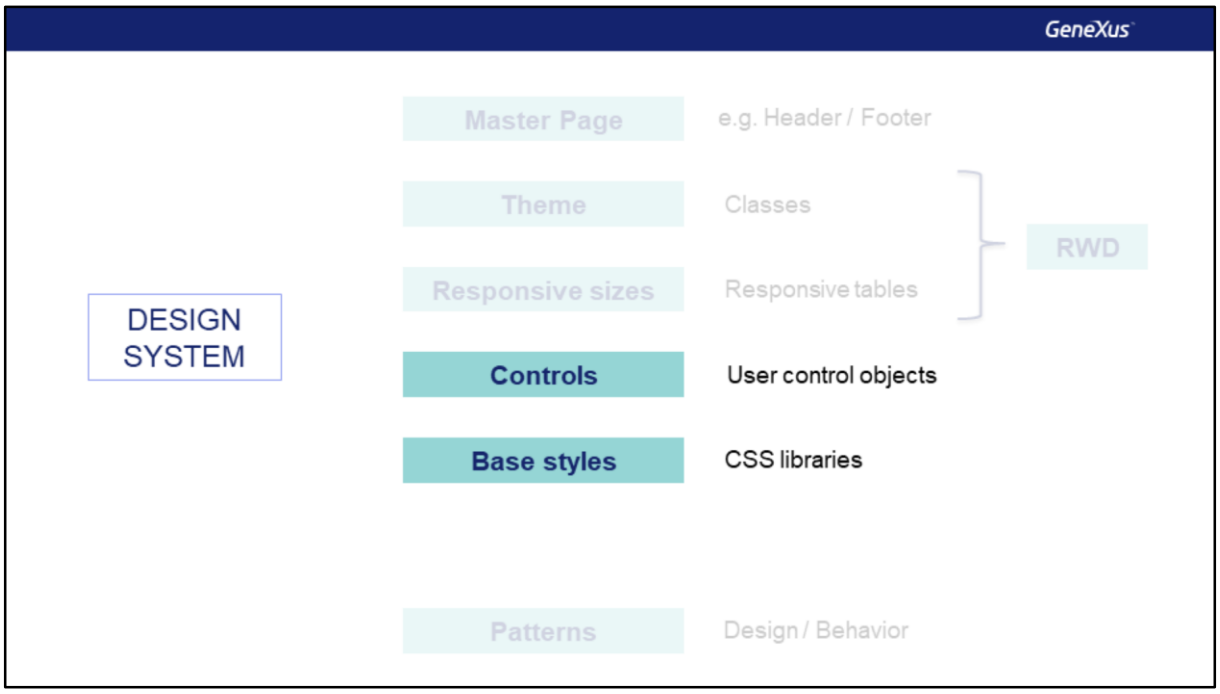
For example, how did we manage to make the panel we had implemented from scratch have a responsive behavior similar to that of the pattern? We invite you to do it.

There is plenty of additional information about this topic. Here, this introduction is enough.

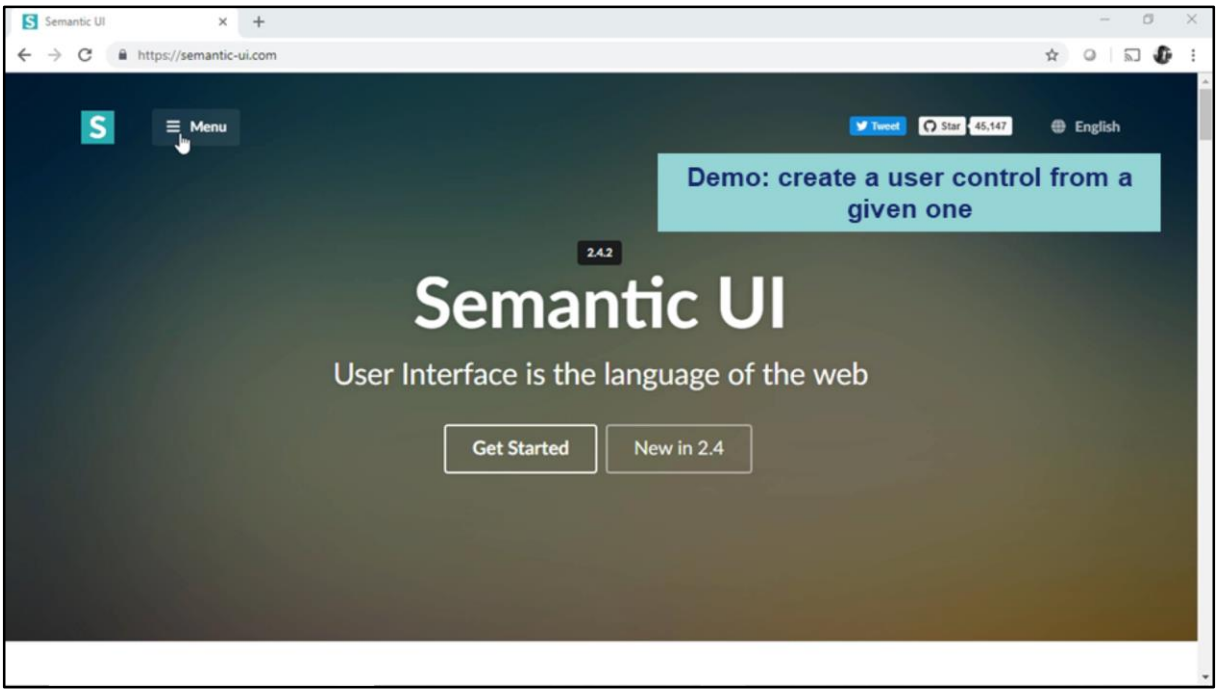


There are other players, together with those we have seen, that allow using a powerful Design System in GeneXus. We will not study them at this level of the course; only an introduction will be provided.

We can not only use in our forms the controls included in the GeneXus toolbox in the predetermined way...



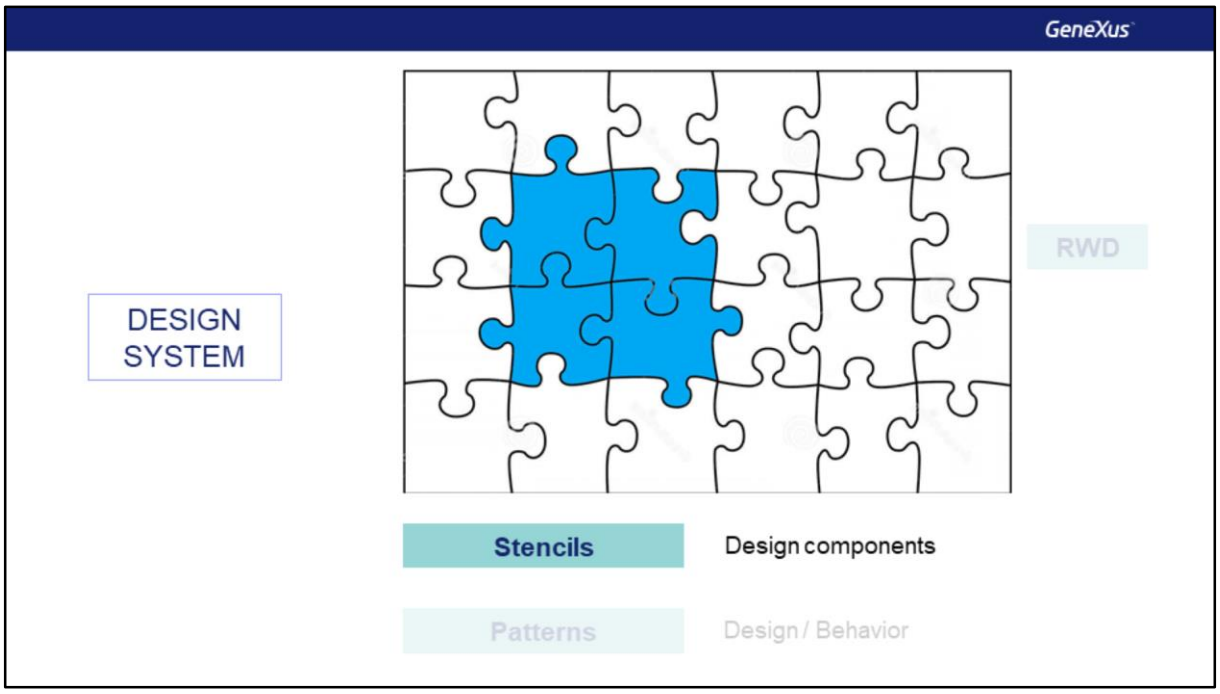
...but we can also define user controls that can be copied from platforms that offer those controls along with CSS libraries (which, like themes in this case, specify their style –the style of those controls).



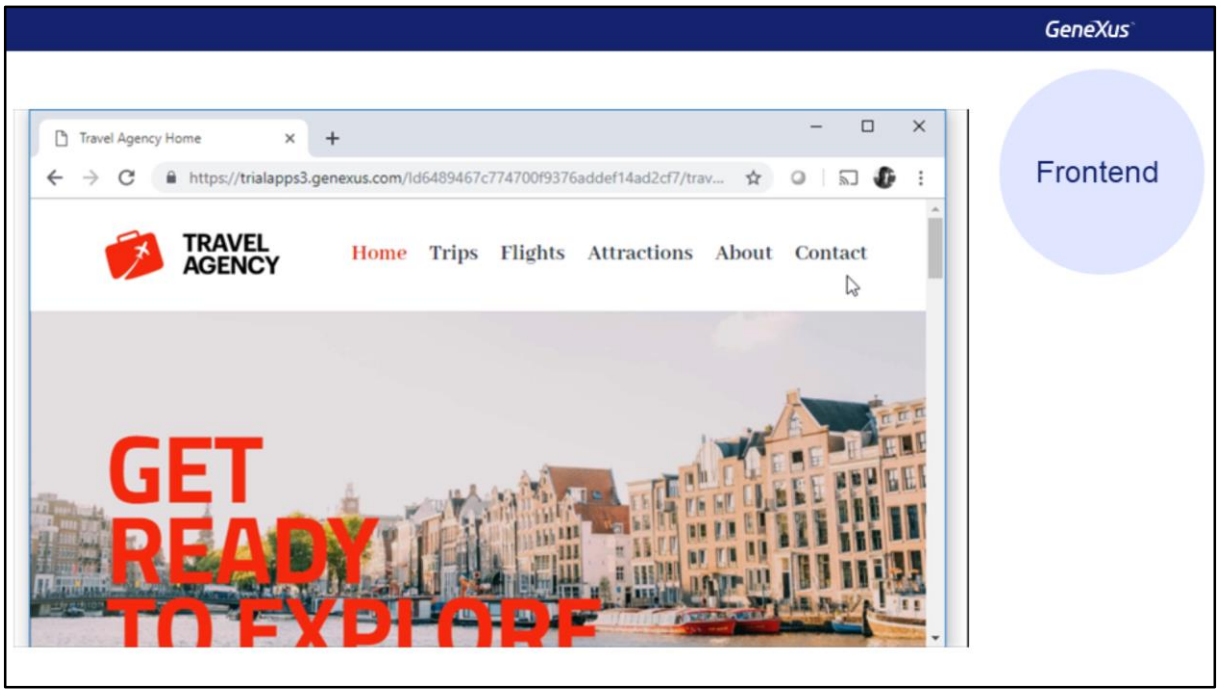
[DEMO: https://youtu.be/lKGkv7VM_ms]

For example, this one, the Semantic UI. We may want to use a Card control such as this one.

To do this, you just need to create a User control object (with the same name), copy and paste its HTML code, and change the fixed data to something like names of SDT elements (to be able to make it dynamic; that is to say, to dynamically load that control, with data that you are going to be able to specify, for example, of the database).
Also, indicate from where it will take the CSS –the class design–, and do something else. In this way, you'll have it available in the Toolbox to use it.



In addition, GeneXus offers Stencils, which allow repeating the design of the same portion of the screen (a set of controls), in many screens. It's a way to abstract design at a higher level. If you think of the screens of a web application like puzzles made up of atomic pieces (the controls), stencils are a way of grouping together a set of controls whose design will be repeated on many screens. It would be a part made of other smaller parts.



[DEMO: https://youtu.be/X4fb58cxmrA]

In the next video we will locate these concepts in the Travel Agency frontend.

GeneXus™

The power of doing.

More videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications