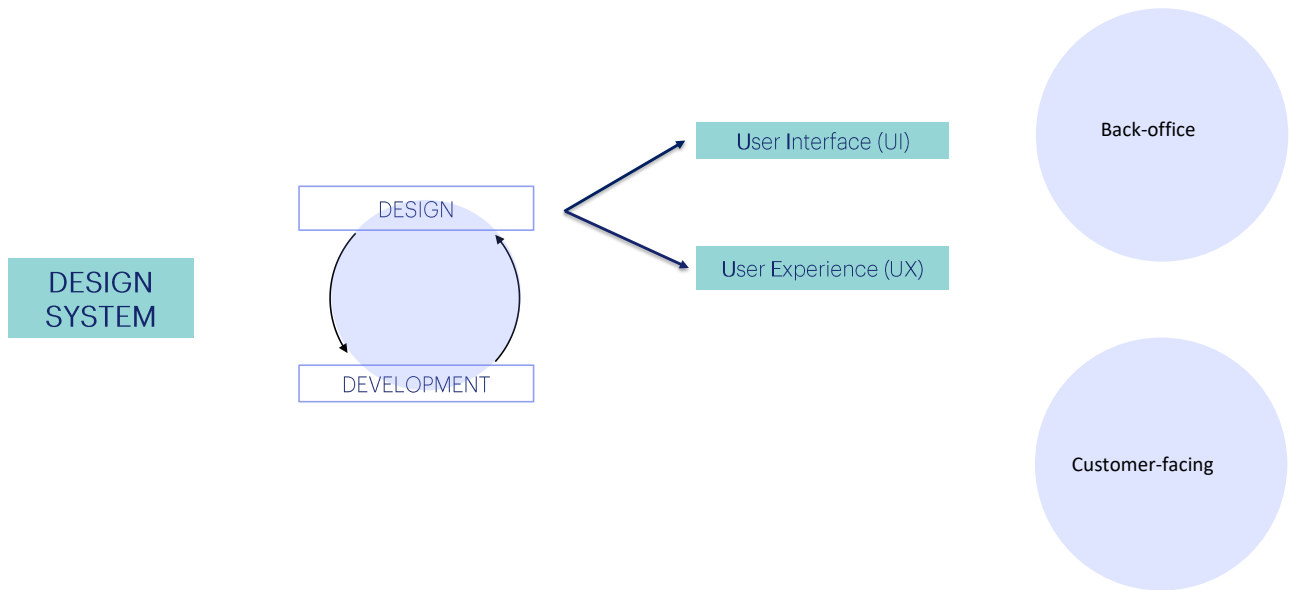


Design System in GeneXus

Customized

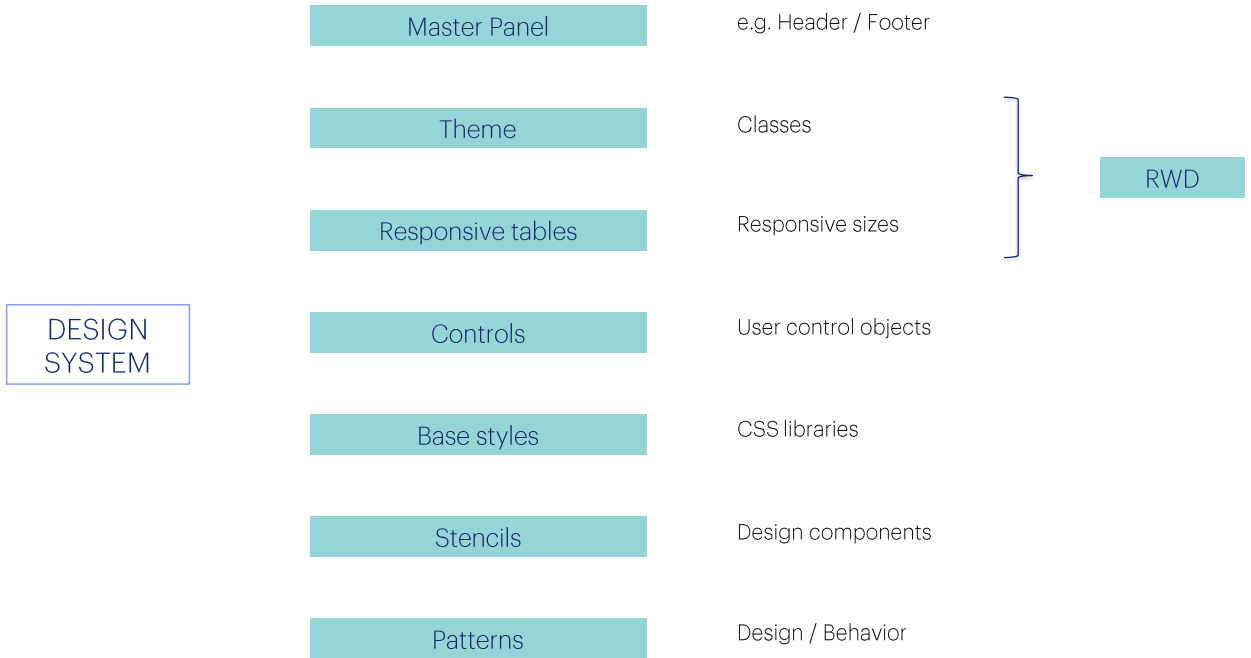
GeneXus™



In previous videos, we explained that both Back-office and Customer-facing applications have increasingly demanding requirements in terms of User Interface and User Experience, which led to the emergence of a new concept, the Design System.

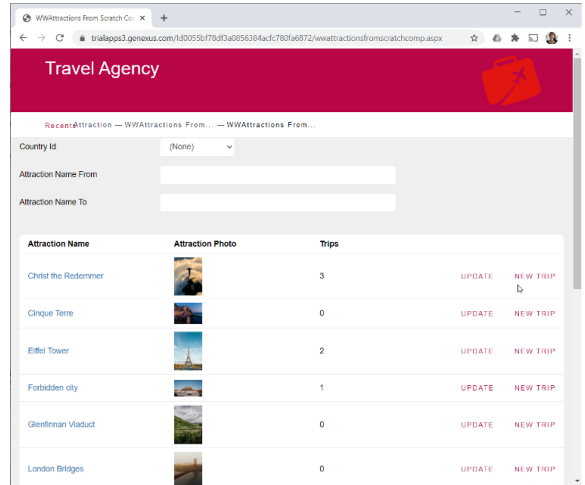
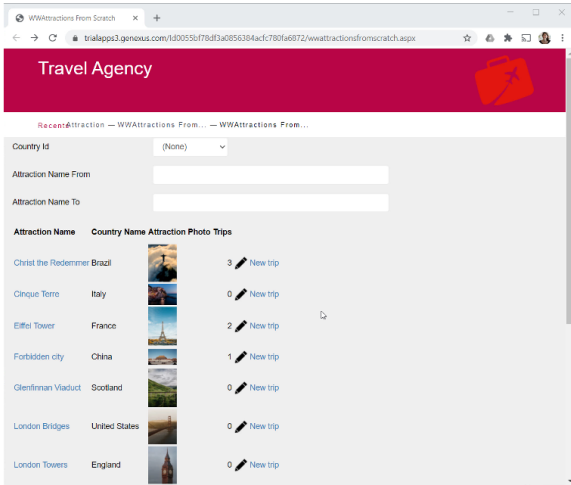
We saw that application development requires working together with graphic designers, who are the ones most involved in the definition of that design system and its use on our screens.

Then we focused on the Back-office application, to show how we already had a default Design System.



Later on, we saw the different players involved in its implementation, starting with the default master panel, followed by the default theme and everything that the Work With pattern already used from all that, through the classes it defined in that theme.

From there we saw how everything could start to be customized, creating new classes and applying them to the form controls, or changing the existing classes to keep consistency with the design of the pattern screens, ensure responsiveness, etc.



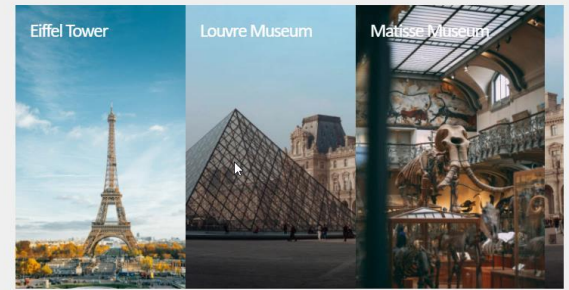
But when we had to design our own screens, those we created from scratch, we had nothing to fall back on but try to copy what the Work With element made. In real life, for these cases we will have graphic designers.



Country Name France

Attraction Name From

Attraction Name To



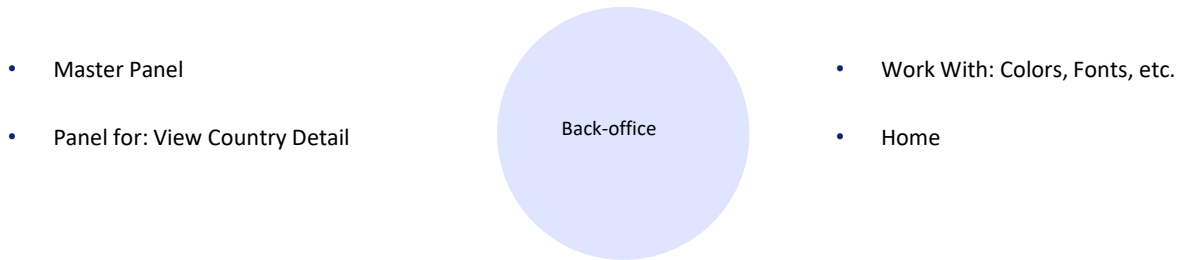
City Name

City Id City Name Attractions

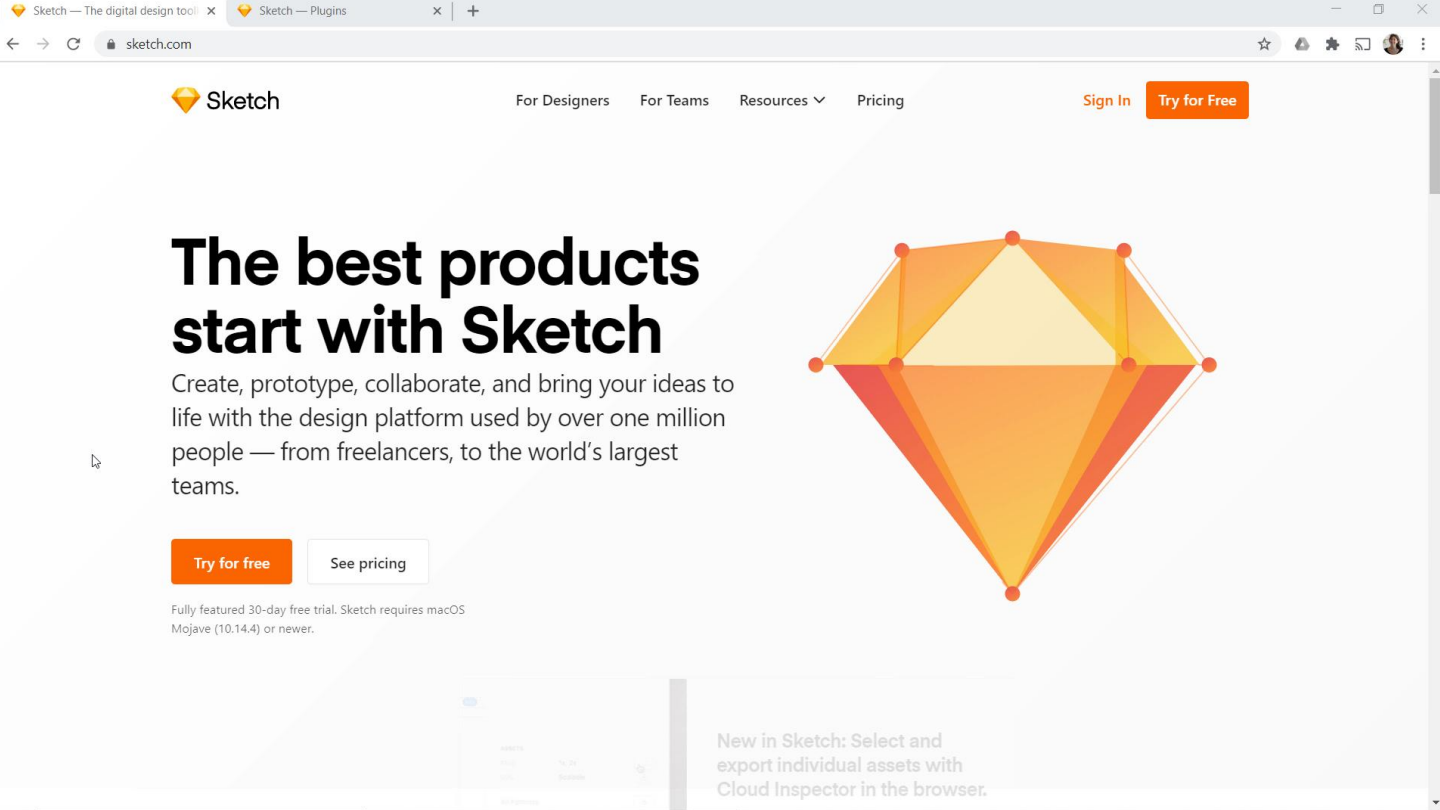
1	Paris	2
2	Nice	1
Total Attractions		3

Let's suppose that we have asked our graphic designer to design a screen for the Back-office to show the information of a country with its tourist attractions (like the one we implemented manually, but without the cities).

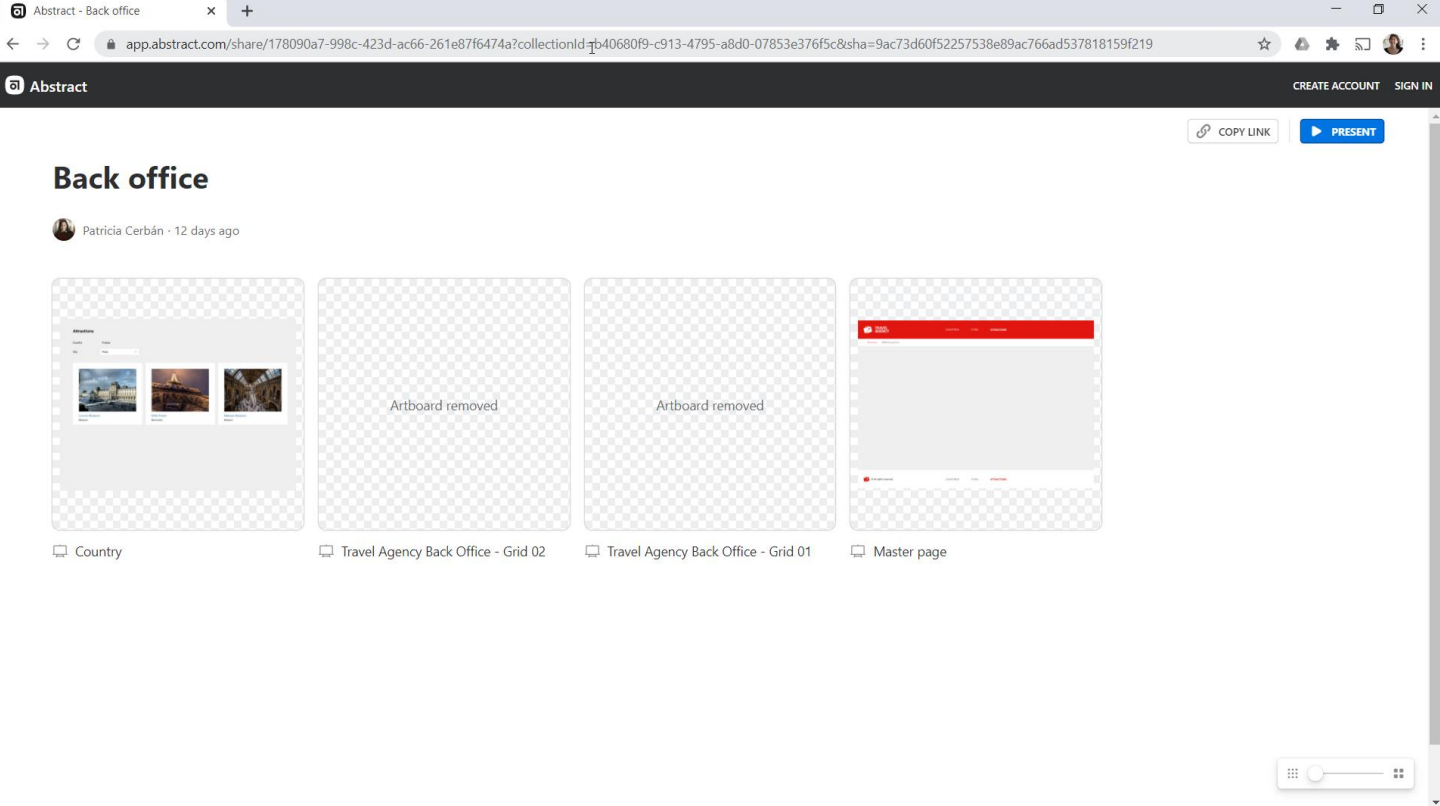
Actually, in real life we would never have implemented it from scratch if we had a graphic designer. Let's look at the steps we would have taken if we had known how.



To make it simple, we ask him or her to design initially only the master panel for the Back-office and that particular screen. But actually it will also adapt our Work With screens, possibly changing colors, fonts and so on. And, of course, he will design the Home page of the Back-office.

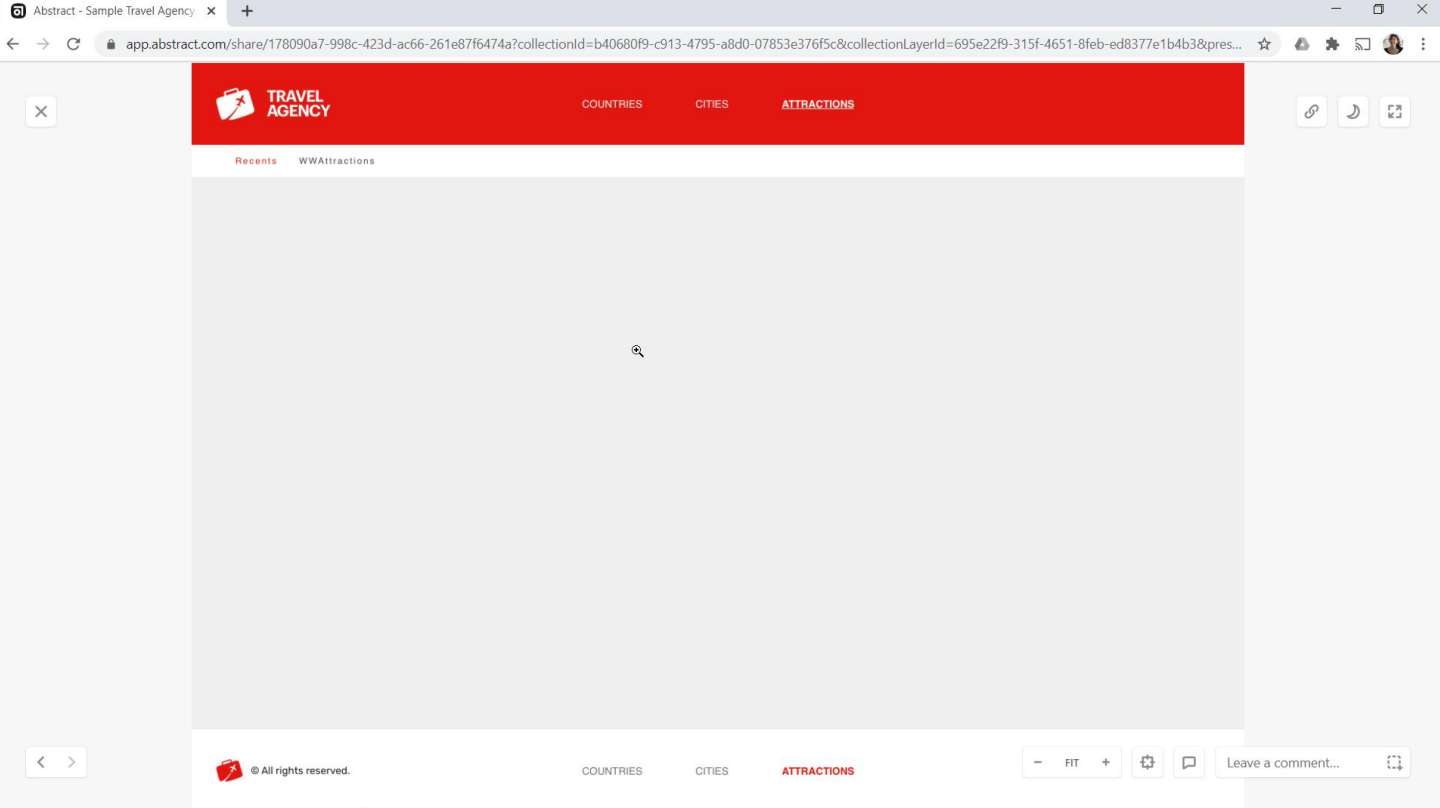


The designer uses the Sketch tool for his work, because it is a design tool to which you can add a GeneXus plugin that will do the amazing things we will see next, automatically integrating the design to our KB. It will even create objects.



So, the designer creates the master page and panel design that we asked for in Sketch. To do so, some guidelines must be followed so that the subsequent integration with GeneXus is simple and efficient.

Once the design is completed, it is shown to us for confirmation. In this case, we will be sent this link so that we can see the screens, as we don't have a Sketch user account.



This is what the master page will look like.


Abstrack - Sample Travel Agency x +

app.abstrack.com/share/178090a7-998c-423d-ac66-261e87f6474a?collectionId=b40680f9-c913-4795-a8d0-07853e376f5c&collectionLayerId=f444627f-6329-4c7d-940e-efc6d3c3839c&pres...

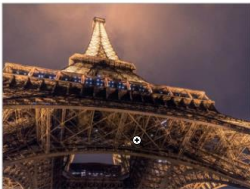
Attractions

Country: France


City: Paris



Louvre Museum
Museum



Eiffel Tower
Monument



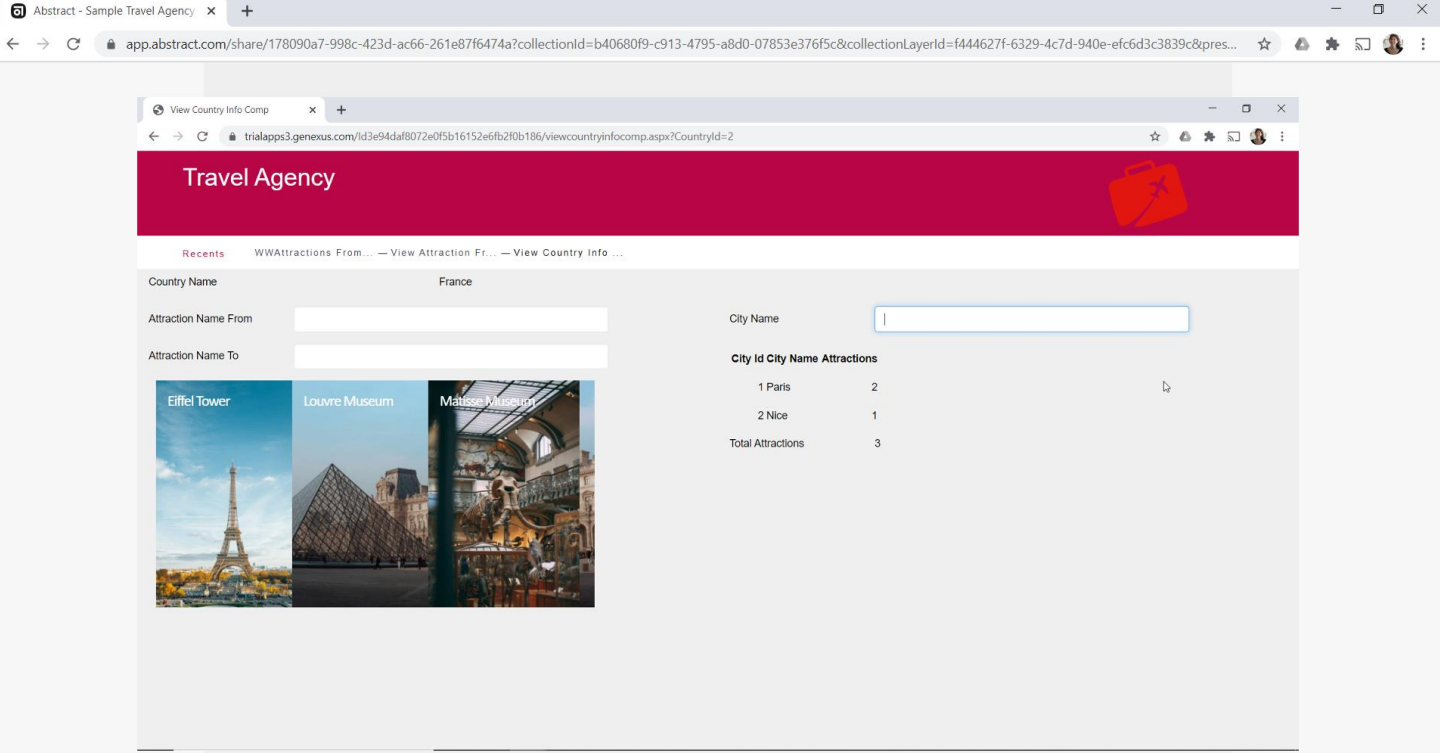
Matisse Museum
Museum

< >

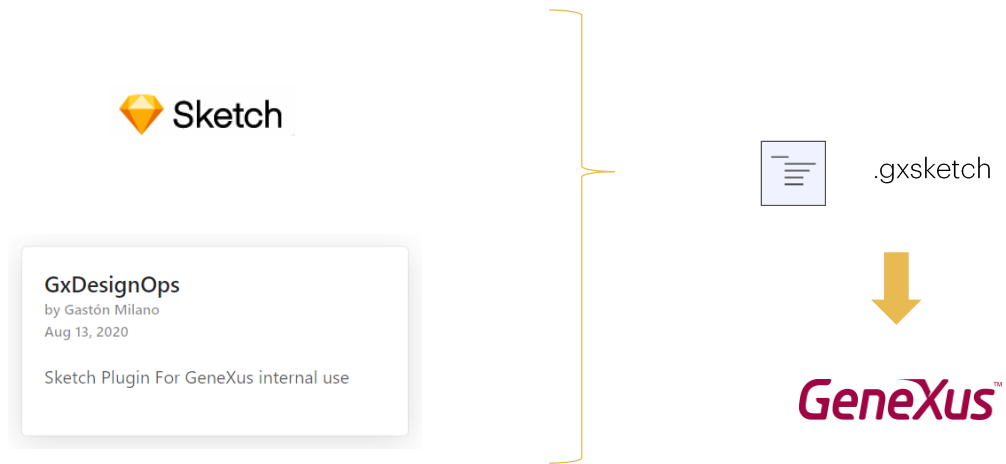
- FIT +

Leave a comment...

And this is the appearance of the country's attractions panel...

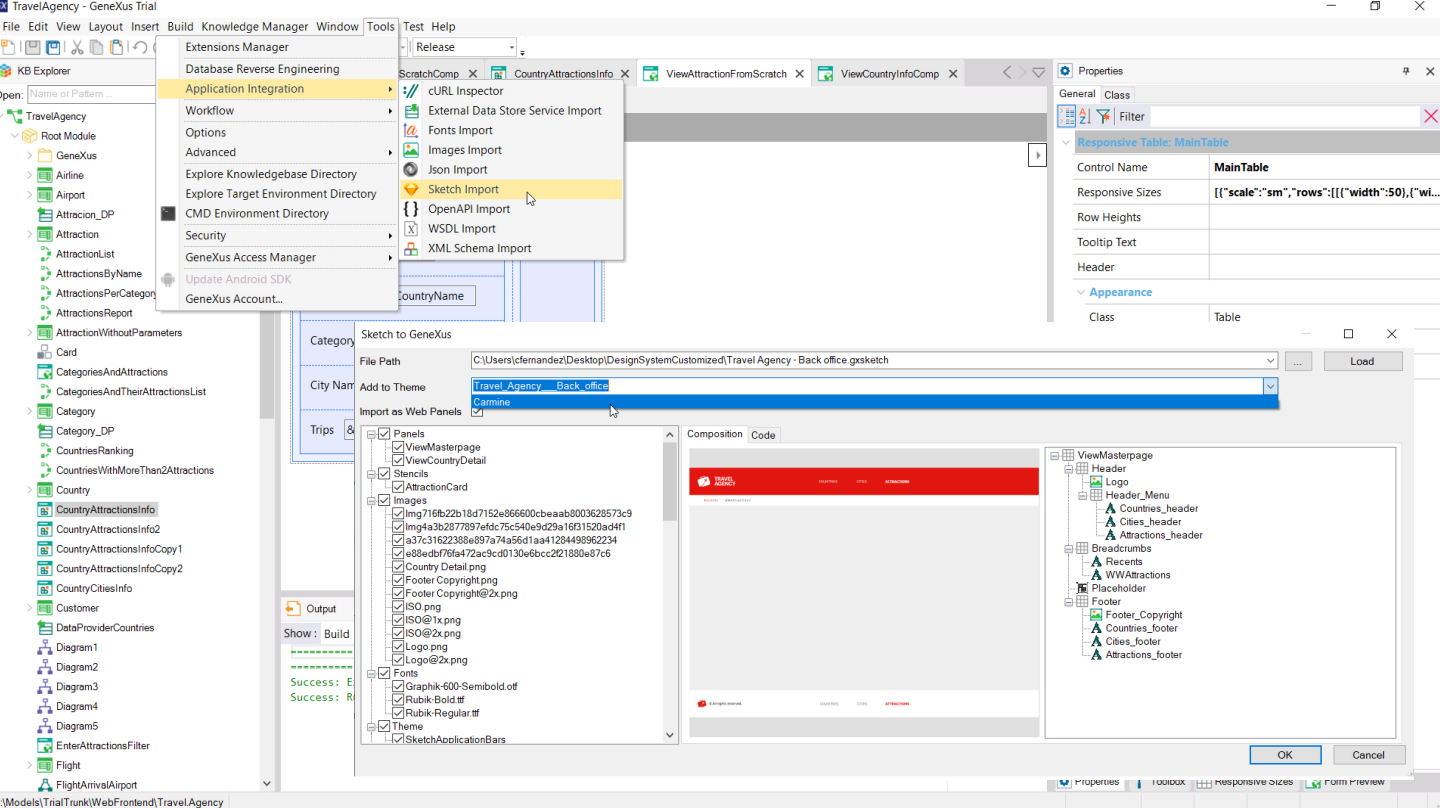


...that will replace this other one.



When we accept it, using that plugin we were talking about, it creates a gxsketch extension file, which is the one sent to us.

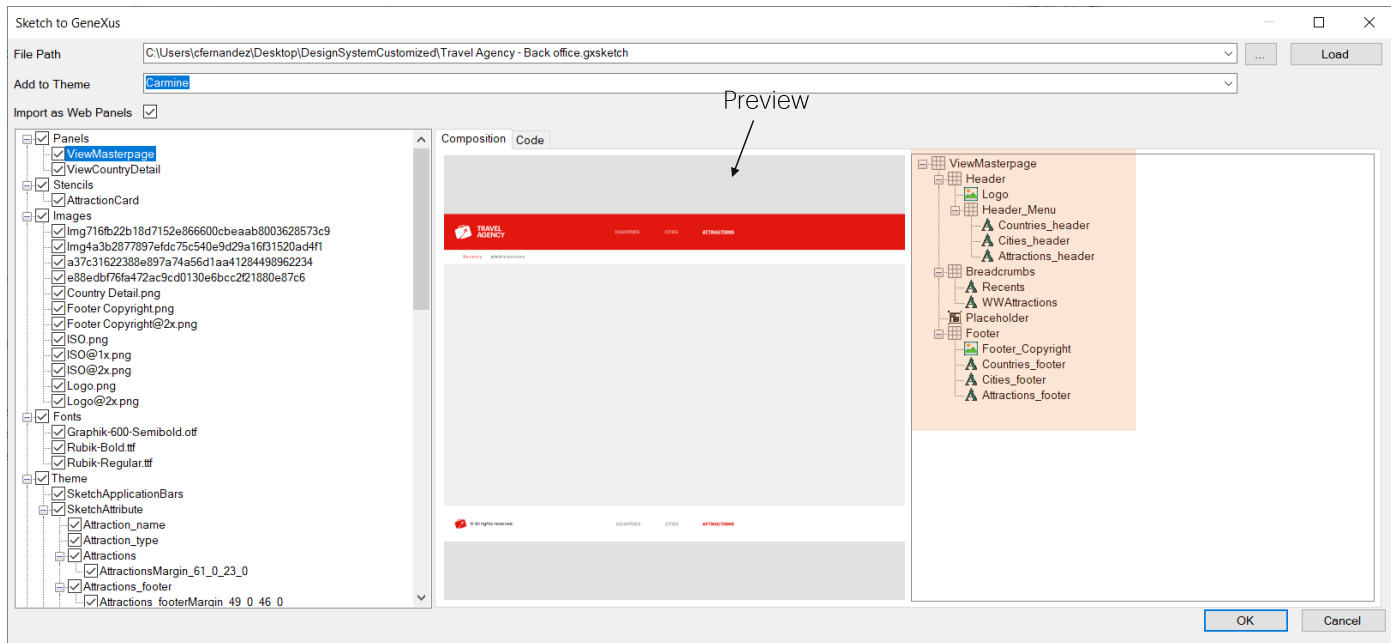
What we do next is import it into our KB. How?



By selecting **Sketch Import** in our KB.

There we must indicate the location of the gxsketch file sent to us by the designer, and check that we want to import Web Panels.

Also, we must indicate if all the classes that it will create to define the controls' design and behavior will be imported as a new theme object, or if we want them to be integrated into the Carmine Theme. We want to do the latter, so that everything is integrated into our back-office.



On the left it shows everything that will be imported. For example, we see two web panels.

The first one will correspond to the master panel. Here we can see the preview, and here it indicates the controls that it will contain.

It will have a Header table, which will contain an image control, and another table to specify the menu with the three options, presented as text blocks. Then another table for the recent links. One for the contentplaceholder. And a last one for the footer.

Sketch to GeneXus

File Path: C:\Users\cfermandez\Desktop\DesignSystem\Customized\Travel Agency - Back office.gxsketch

Add to Theme: Camrine

Import as Web Panels:

Composition Code

```

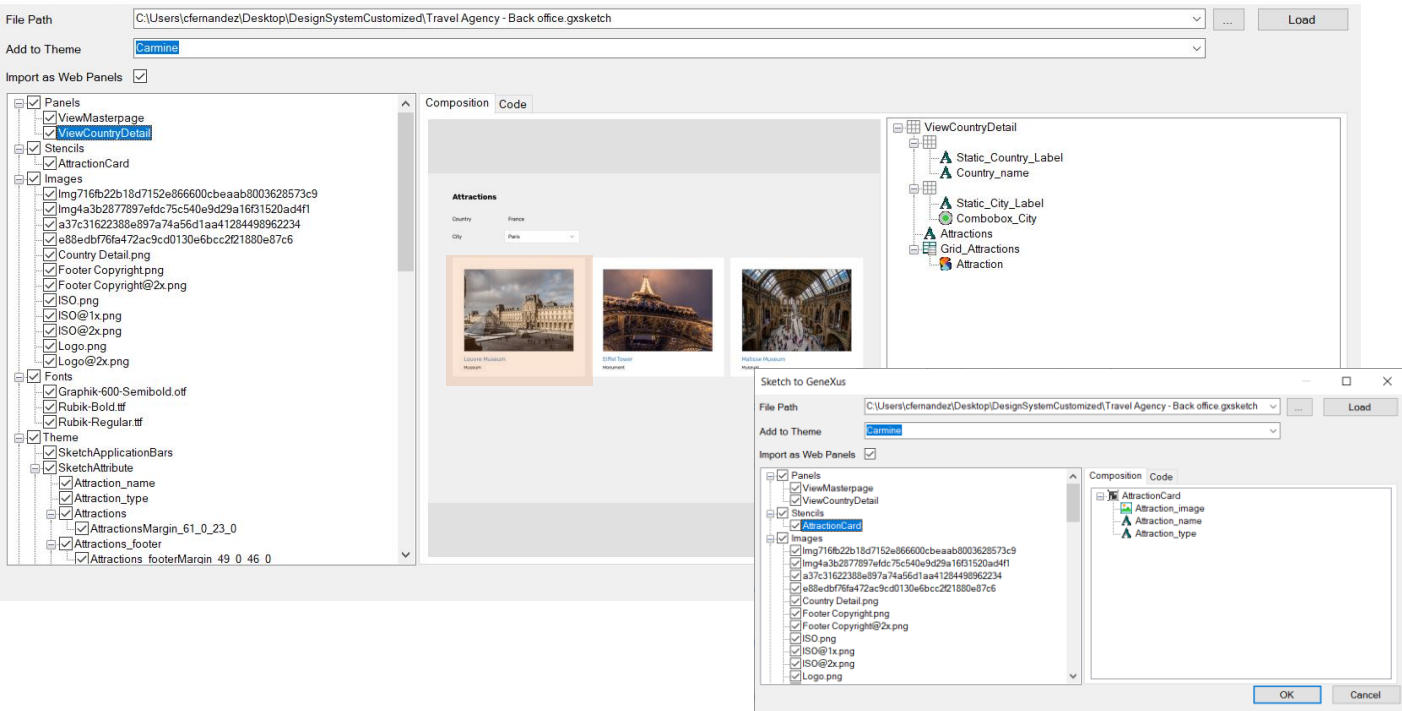
<Root Id="2ACD9572-97D0-45A6-89CD-3AF255EB757B" importFor="web">
  <Controls>
    <table tableType="SmartTable" controlName="ViewMasterpage" width="100%" height="100%"
      columnsStyle="100%" rowsStyle="112px;43px;100%;111px" class="Master_page" expandBounds="None"
      formClass="SketchForm">
      <row rowHeight="112px">
        <cell hAlign="Center" vAlign="Top">
          <table tableType="SmartTable" controlName="Header" width="100%" height="112px"
            columnsStyle="362px;1077px" rowsStyle="112px" class="Background_header" expandBounds="None"
            formClass="SketchForm">
            <row rowHeight="112px">
              <cell hAlign="Left" vAlign="Middle">
                <table tableType="SmartTable" controlName="WrapLogo" width="100%" height="100%"
                  columnsStyle="100%" rowsStyle="50%;44px;50%" class="SketchTable" expandBounds="None" formClass="SketchForm">
                  <row rowHeight="50%">
                    <cell />
                  </row>
                  <row rowHeight="44px">
                    <cell hAlign="Left" vAlign="Middle">
                      <data attribute="&Logo" class="LogoMargin_0_34_0_0" labelPosition="None"
                        readonly="True" type="BITMAP" controlNameForStencil="&Logo" initialValue="Image:Logo.Link()"
                        PATTERN_ELEMENT_CUSTOM_PROPERTIES="&Properties&Property&Name&AutoGrow&Name&";
                    </cell />
                  </row>
                </table>
              </cell />
            </table>
          </cell />
        </row>
      </table>
    </Controls>
  </Root>

```

OK Cancel

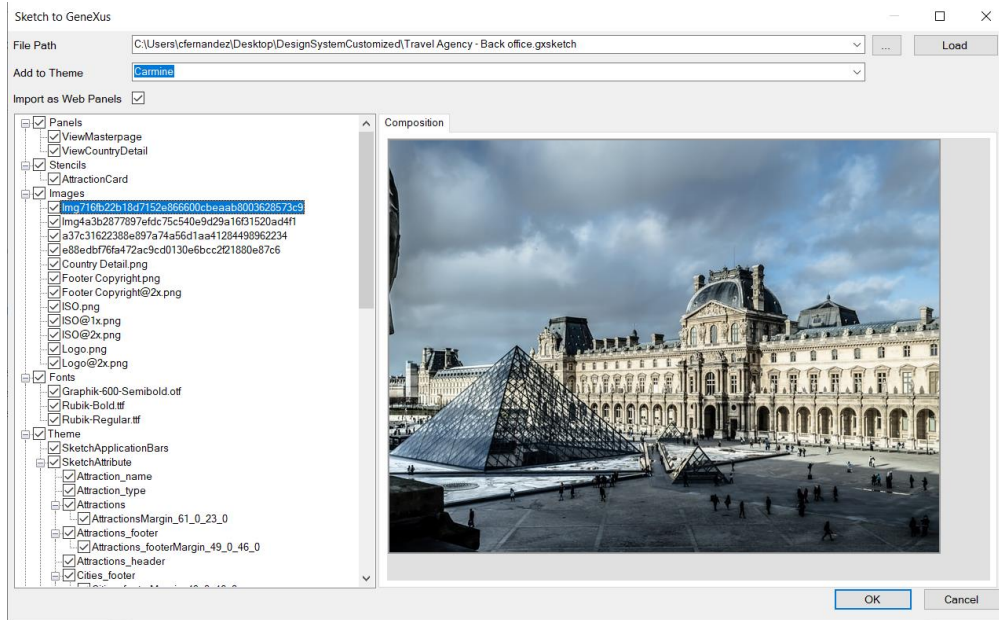
We can also see the corresponding HTML code.

Sketch to Genexus

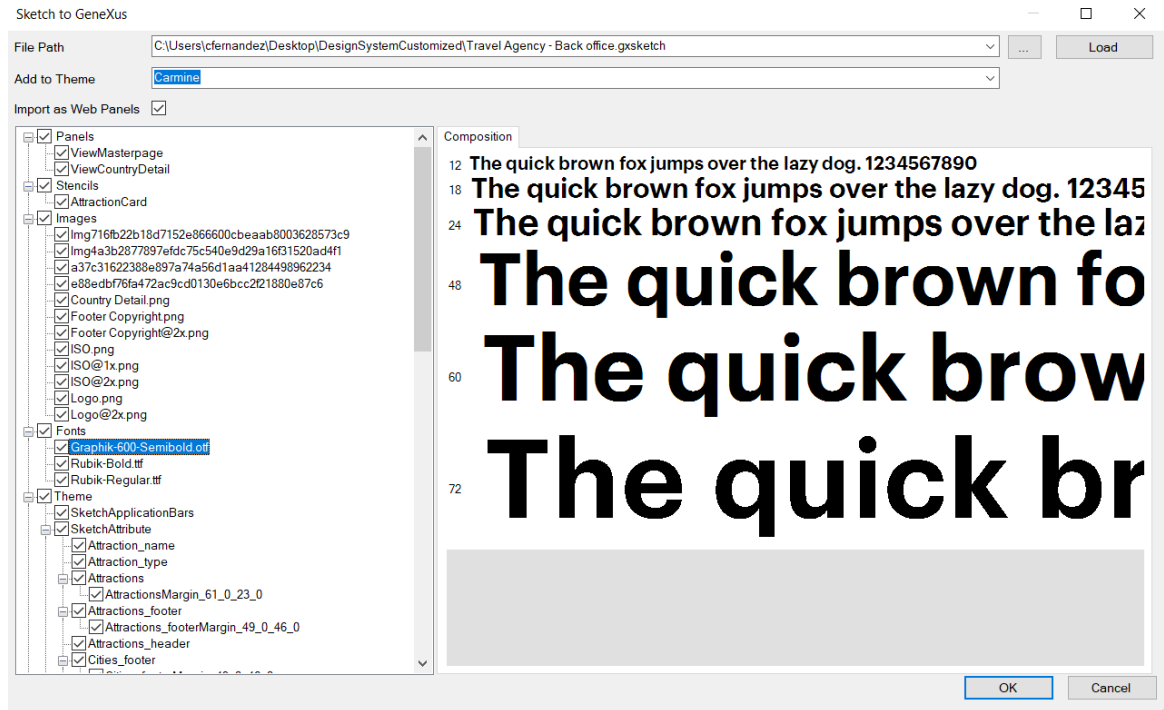


Then, the same thing in the next panel.

It will create a grid which, as we will see, will be horizontal, and a stencil control, precisely so as not to repeat this piece of design. We can see it here.



It will also import the necessary images. For example, these are to be able to run the panel with fixed data and show loaded attractions, which then the developers will have to replace with those in our application's database.

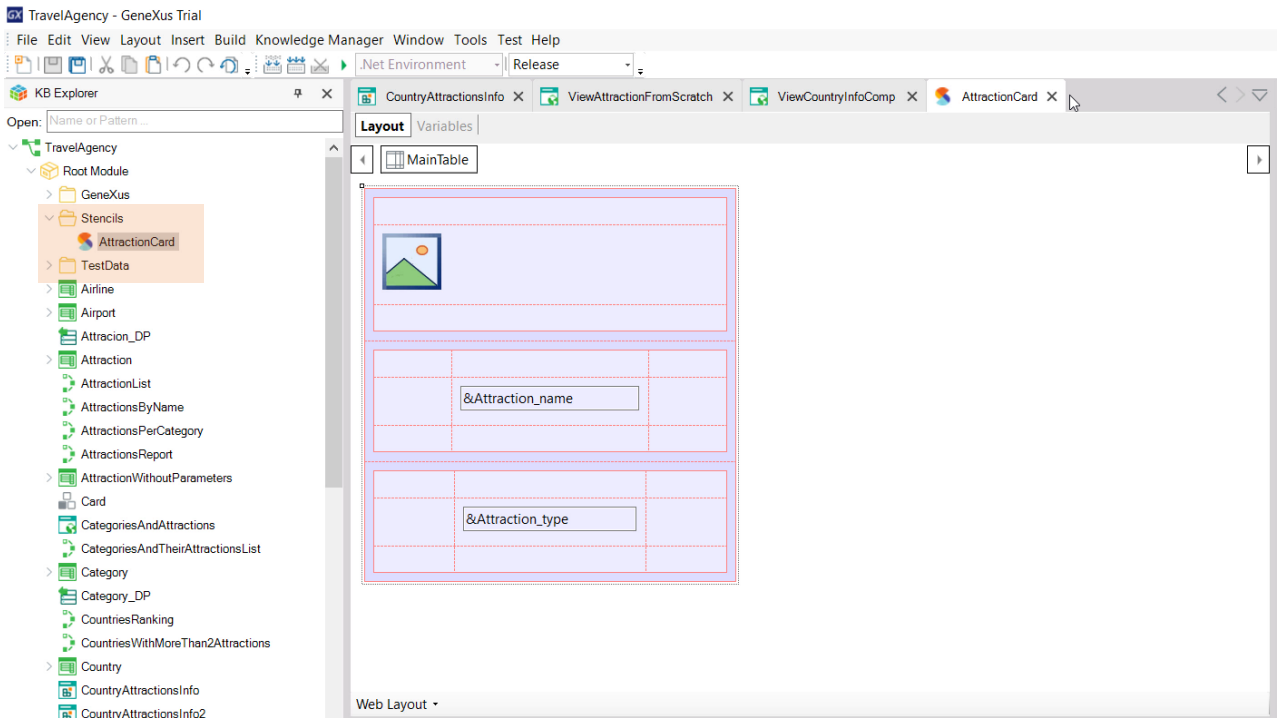


Here we see the fonts that will be used in this design.

And finally all the classes that will be added to our theme.

If new colors had been defined, they would be shown here too.

Since we want to import all this, we press OK.



Note that these two folders were created.

Here is the stencil we had identified for the card showing a photo, name, and category of the tourist attraction.

TravelAgency - GeneXus Trial

File Edit View Layout Insert Build Knowledge Manager Window Tools Test Help

.Net Environment Release

KB Explorer

Name or Pattern

TravelAgency

- Root Module
- GeneXus
- Stencils
- AttractionCard
- TestData
 - ViewCountryDetail_Grid_Attractions_DP
 - ViewCountryDetail_Grid_Attractions_SD
- Airline
- Airport
- Attraction_DP
- Attraction
- AttractionList
- AttractionsByName
- AttractionsPerCategory
- AttractionsReport
- AttractionWithoutParameters
- Card
- CategoriesAndAttractions
- CategoriesAndTheirAttractionsList
- Category
- Category_DP
- CountriesRanking
- CountriesWithMoreThan2Attractions
- Country

CountryAttractionsInfo ViewAttractionFromScratch ViewCountryInfoComp ViewCountryDetail_Grid_Attractions_DP

Source Rules Variables

```

1 ViewCountryDetail_Grid_Attractions_DP
2 {
3   ViewCountryDetail_Grid_Attractions_SDT
4   {
5     Layout = "Attraction"
6     Attraction_image = Image:Img716fb22b18d7152e866600cbeaab8003628573c9.Link()
7     Attraction_name = "Louvre Museum"
8     Attraction_type = "Museum"
9   }
10  ViewCountryDetail_Grid_Attractions_SDT
11  {
12    Layout = "Attraction"
13    Attraction_image = Image:Img4a3b2877897efdc75c540e9d29a16f31520ad4f1.Link()
14    Attraction_name = "Eiffel Tower"
15    Attraction_type = "Monument"
16  }
17  ViewCountryDetail_Grid_Attractions_SDT
18  {
19    Layout = "Attraction"
20    Attraction_image = Image:a37c31622388e897a74a56d1aa41284498962234.Link()
21    Attraction_name = "Matisse Museum"
22    Attraction_type = "Museum"
23  }
24 }
25

```

Properties

Filter

Data Provider: ViewCountryDetail_Grid_Attractions_DP

Name	ViewCountryDetail_Grid_Attractions_DP
Description	View Country Detail_Grid_Attractions_DP
Expose as Web Service	False
Main program	False
Call protocol	Internal
Module/Folder	TestData
Qualified Name	ViewCountryDetail_Grid_Attractions_DP
Object Visibility	Public
Output	
Infer Structure	No
Output	ViewCountryDetail_Grid_Attractions_SDT
Collection	True
Collection Name	ViewCountryDetail_Grid_Attractions_DP
Network	
Connectivity Support	Inherit
Miscellaneous	
Generate Object	True

And here is a Data Provider that will load with fixed data the three attractions that we saw in the preview.

The screenshot displays the GeneXus IDE interface for designing a web form. The main workspace shows a grid-based layout for a master page. The layout is organized into three horizontal sections: a header, a main content area, and a footer. The header section contains a logo on the left and three columns of content labeled '&Countries_header', '&Cities_header', and '&Attractions_header'. The main content area contains two columns of content labeled '&Recents' and '&WWAttractions'. The footer section contains a logo on the left and three columns of content labeled '&Countries_footer', '&Cities_footer', and '&Attractions_footer'. The Properties panel on the right shows the control name 'ViewMasterpage' and its appearance settings, including Columns Style (100%), Rows Style (112px:43px:100%:111px), Width (100%), Height (100%), Class (Master_page), and Form Class (SketchForm).

Control Name	ViewMasterpage
Tooltip Text	
Header	
Appearance	
Columns Style	100%
Rows Style	112px:43px:100%:111px
Width	100%
Height	100%
Class	Master_page
Columns Gap	0px
Rows Gap	0px
Form	
Form Class	SketchForm

And below is the web panel that implemented the master page. And here is the one that implemented the detail of the country.

If we open the first one, we see that all its visual content is already created.

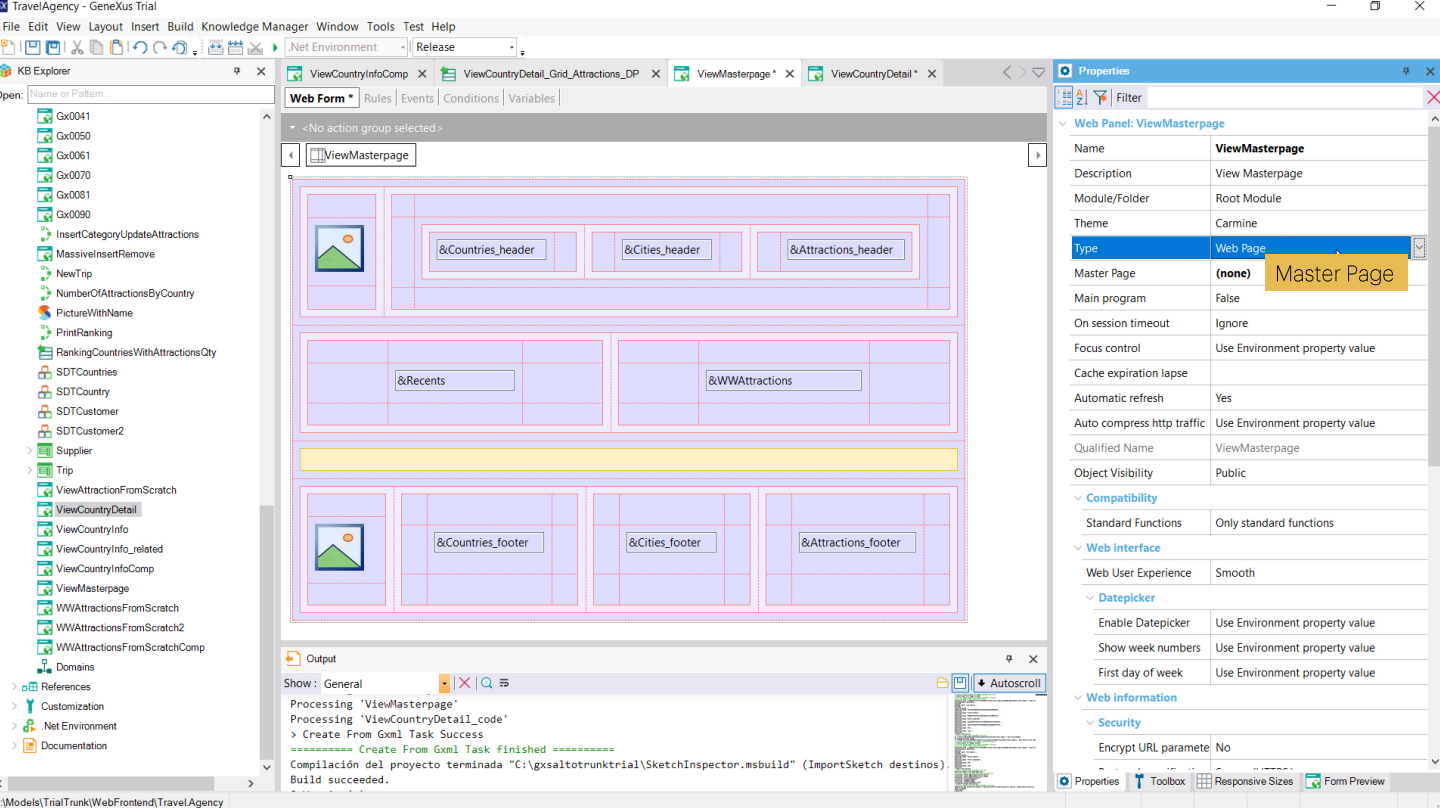
The screenshot displays the GeneXus IDE interface. On the left, a project tree shows various views, with 'ViewCountryDetail' selected. The main workspace shows a web form design for 'ViewCountryDetail'. The form is structured as follows:

- Top section: A text field labeled '&Attractions'.
- Middle section: A label 'Country' next to a text field labeled '&Country_name'.
- Bottom section: A label 'City' next to a dropdown menu labeled '&Combobox_City'.
- Bottom-most section: A 'GRID' section, partially visible.

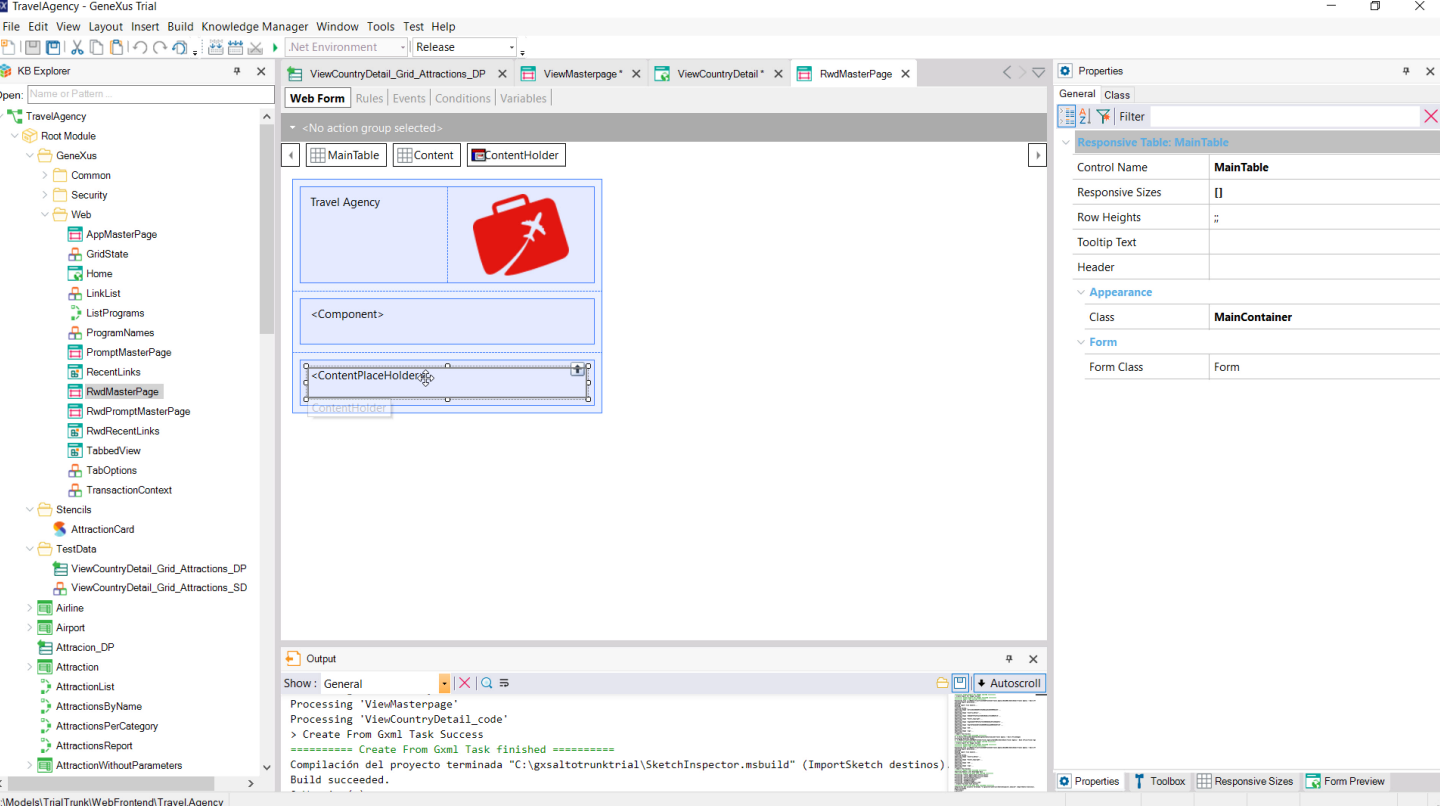
The Properties window on the right shows the following settings for the 'ViewCountryDetail' control:

Control Name	ViewCountryDetail
Tooltip Text	
Header	
Appearance	
Columns Style	100%
Rows Style	112px;5%;8%;86%
Width	100%
Height	100%
Class	SketchTable
Columns Gap	0px
Rows Gap	0px
Form	
Form Class	SketchForm

And the same if we open the second one.

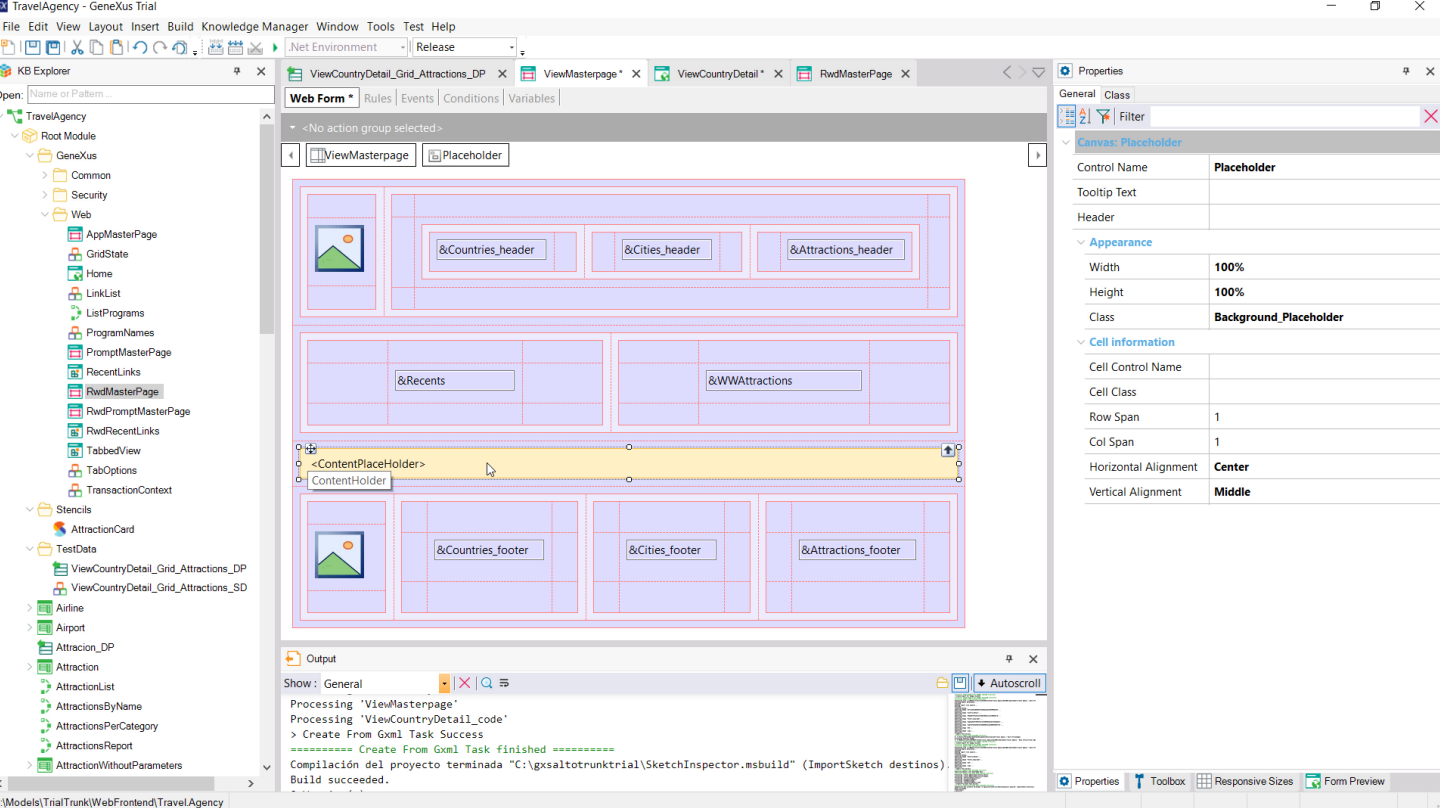


However, what the Sketch import program doesn't do yet (because it is still under development) but will do in the future, is to import this panel as a master page. Note that it imported it as a common web page. Then we will have to change its type.

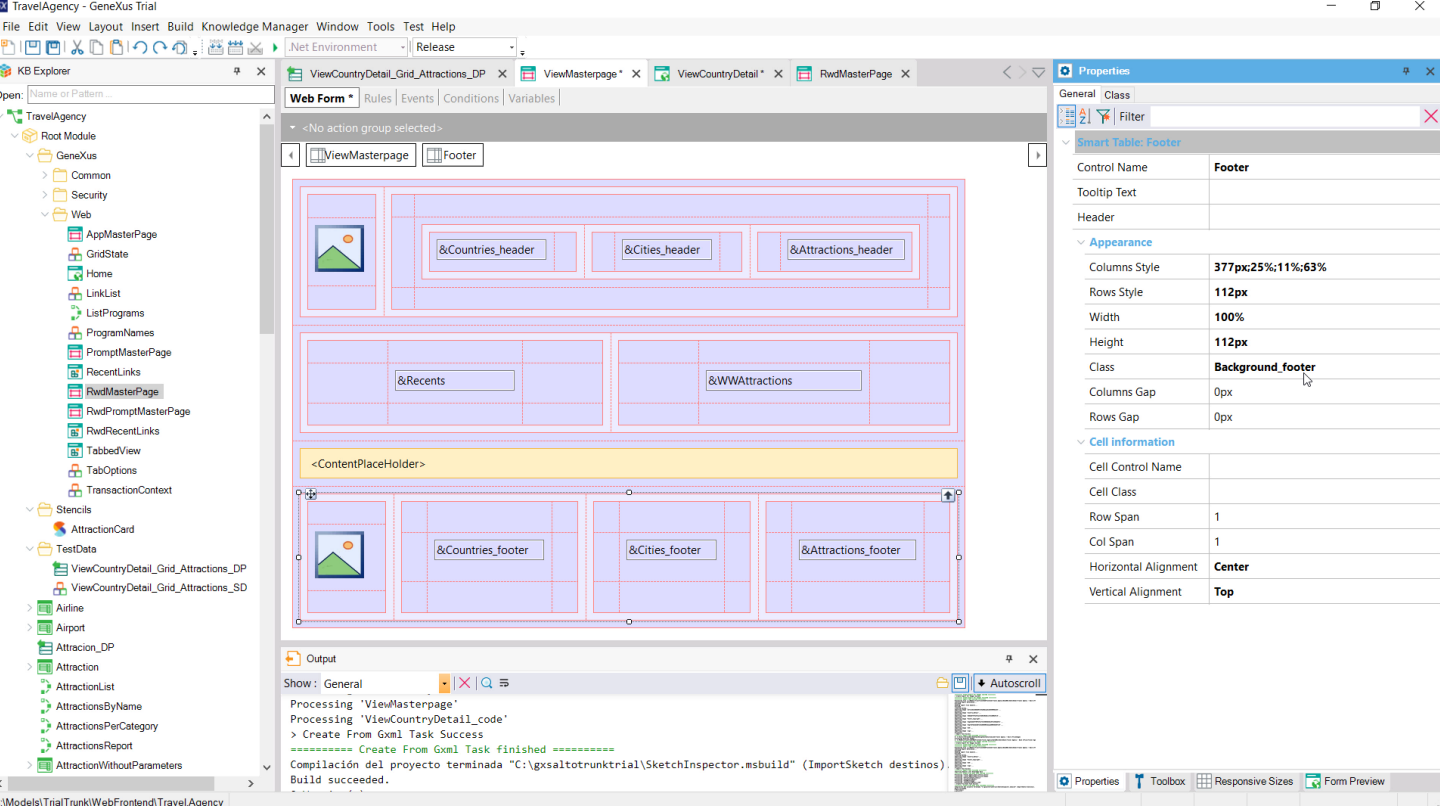


Remember that in our KB the web objects were using the default master page, which we had customized a bit.

Let's take advantage of the fact that we have it open to copy the contentplaceholder for the new one, which we have just set as master page, so it will require to have this control in its form (otherwise, we will not be allowed to save the object).

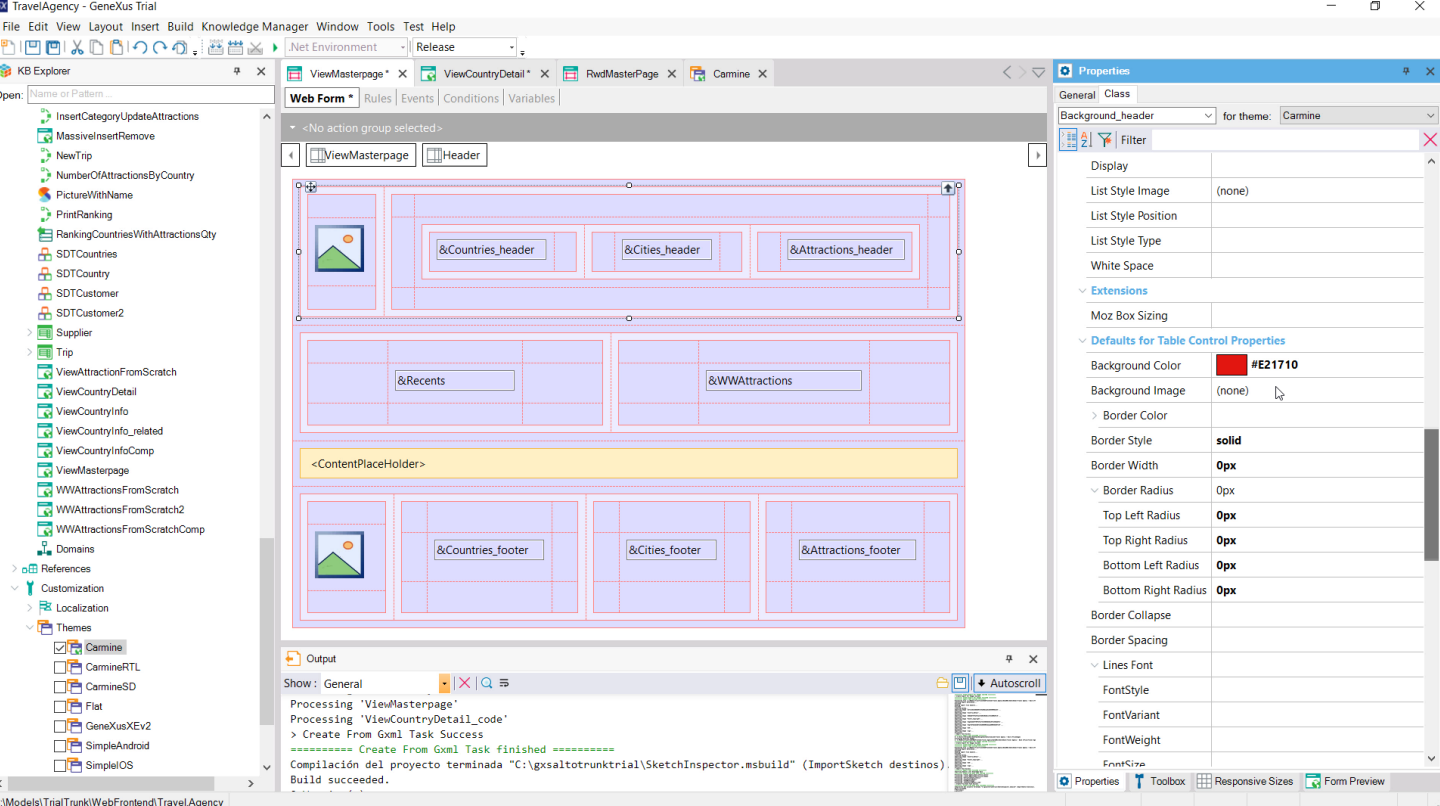


We copy it in the place left by the designer for this purpose.

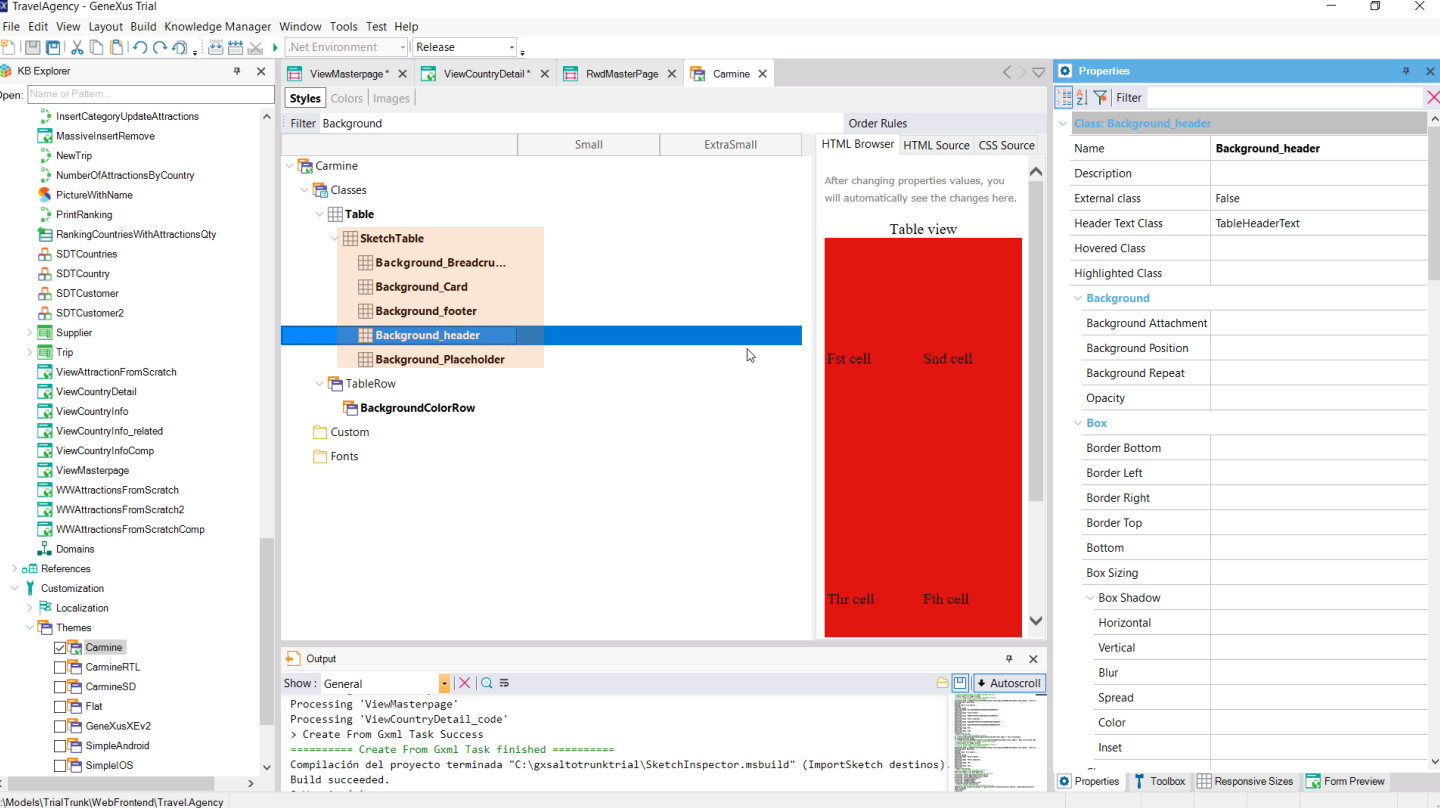


Here we see the Header table, which we had seen before importing. Here is the recent links one.

And here that of the footer. We see that it was assigned this class, which was imported into our Carmine theme....

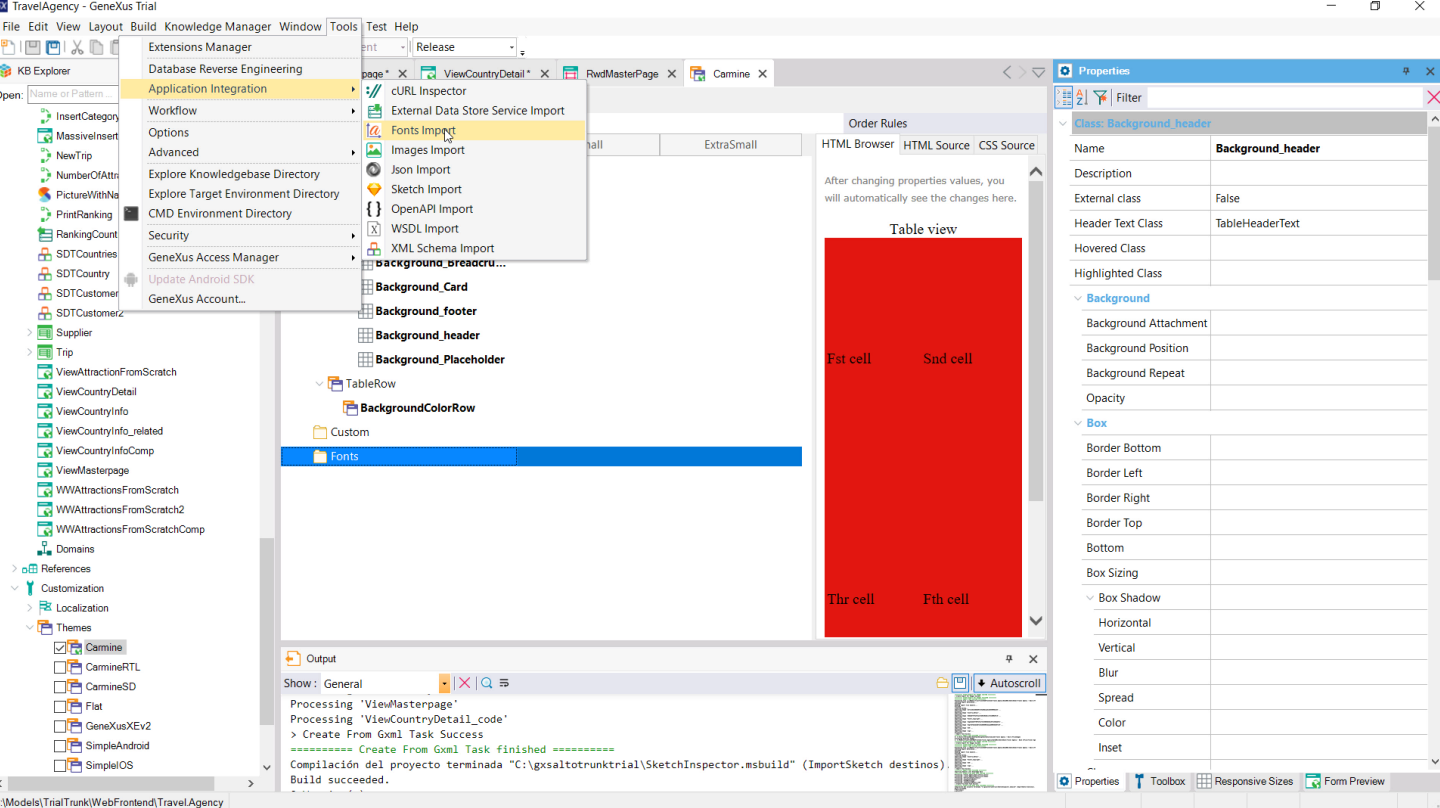


If we open it... and look for the class for the Header table that has this background color specified...

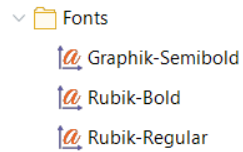
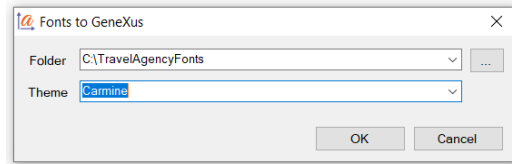
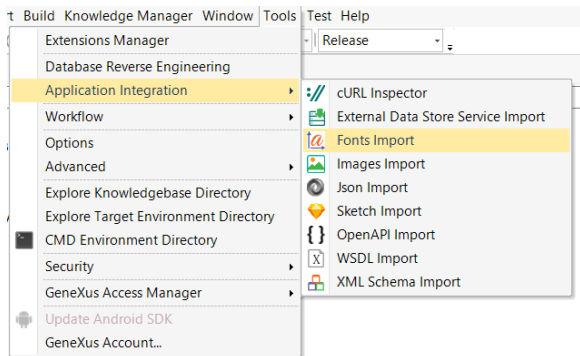


...We can see it here.

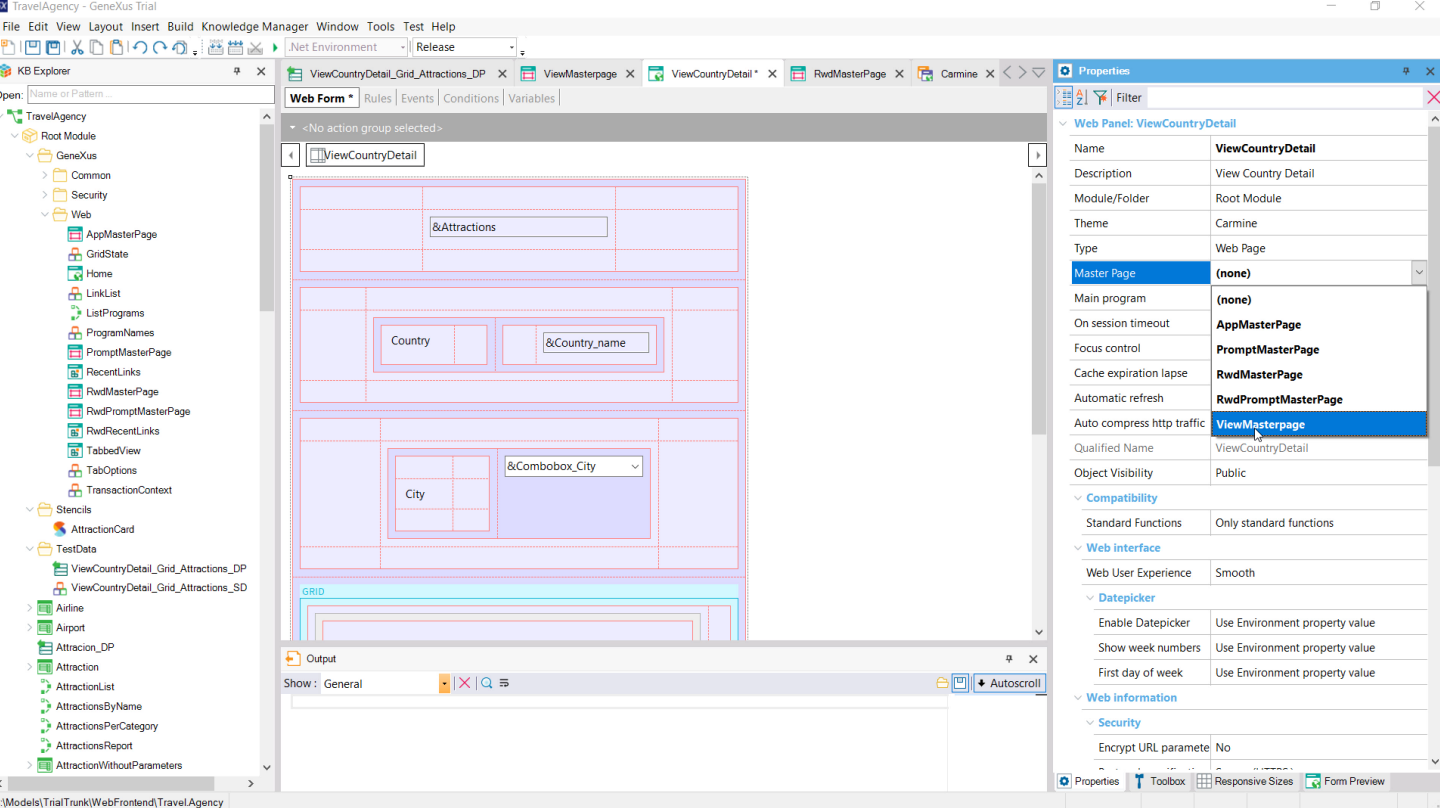
With the import all these classes of tables were created that were not there before.



The fonts specified in Sketch should also have been created below here. This will be done in future versions. For now, we have to import them explicitly.

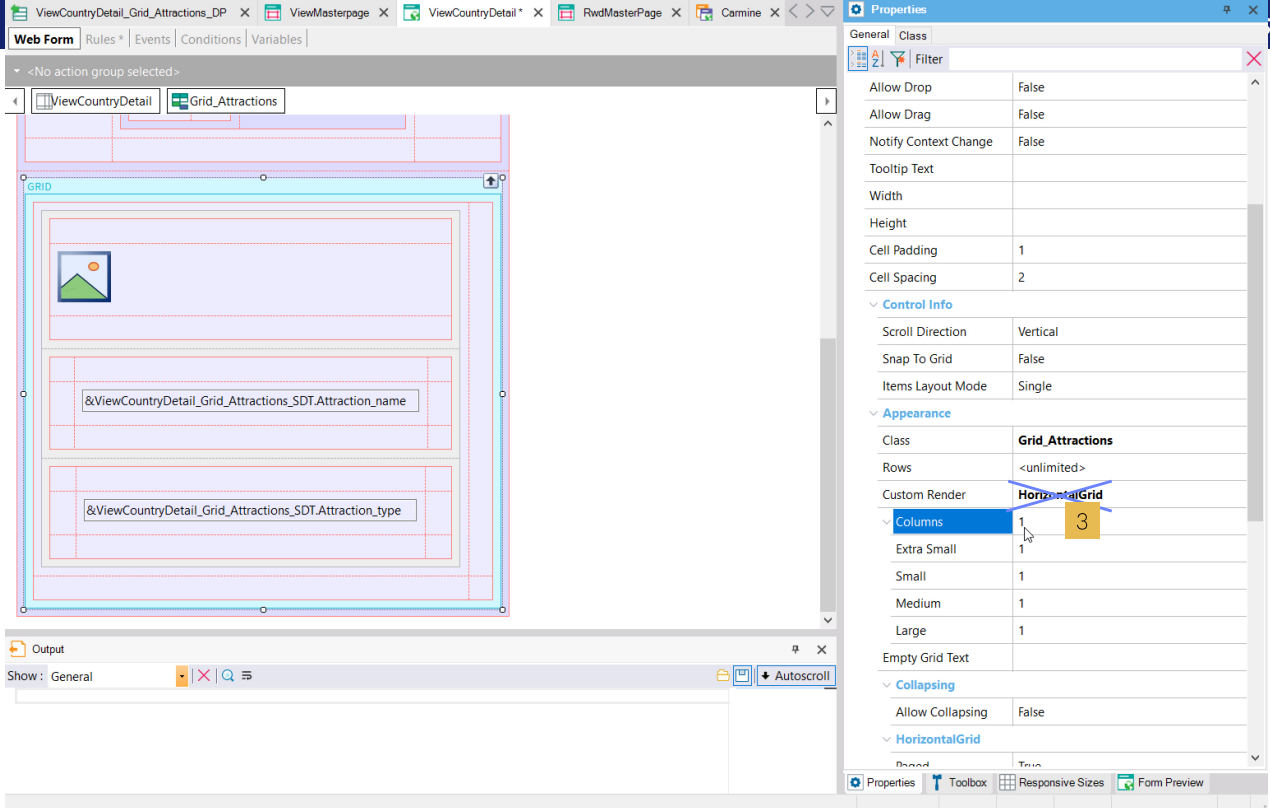


We indicate which folder they are in... and that we want them to be incorporated into the Carmine theme. There they are.



We save the new master panel.

And now we must have a country detail panel loaded into it.



Before running it, note that a horizontal grid has been inserted, even though the 3 columns per horizontal page have not been specified correctly (yet). This is because the import function from Sketch for web panels is still not fully implemented.

So we change it.

We could also customize it to first test how it would look if the grid was not horizontal but standard, although with 3 columns.

Let's try it.

Attractions

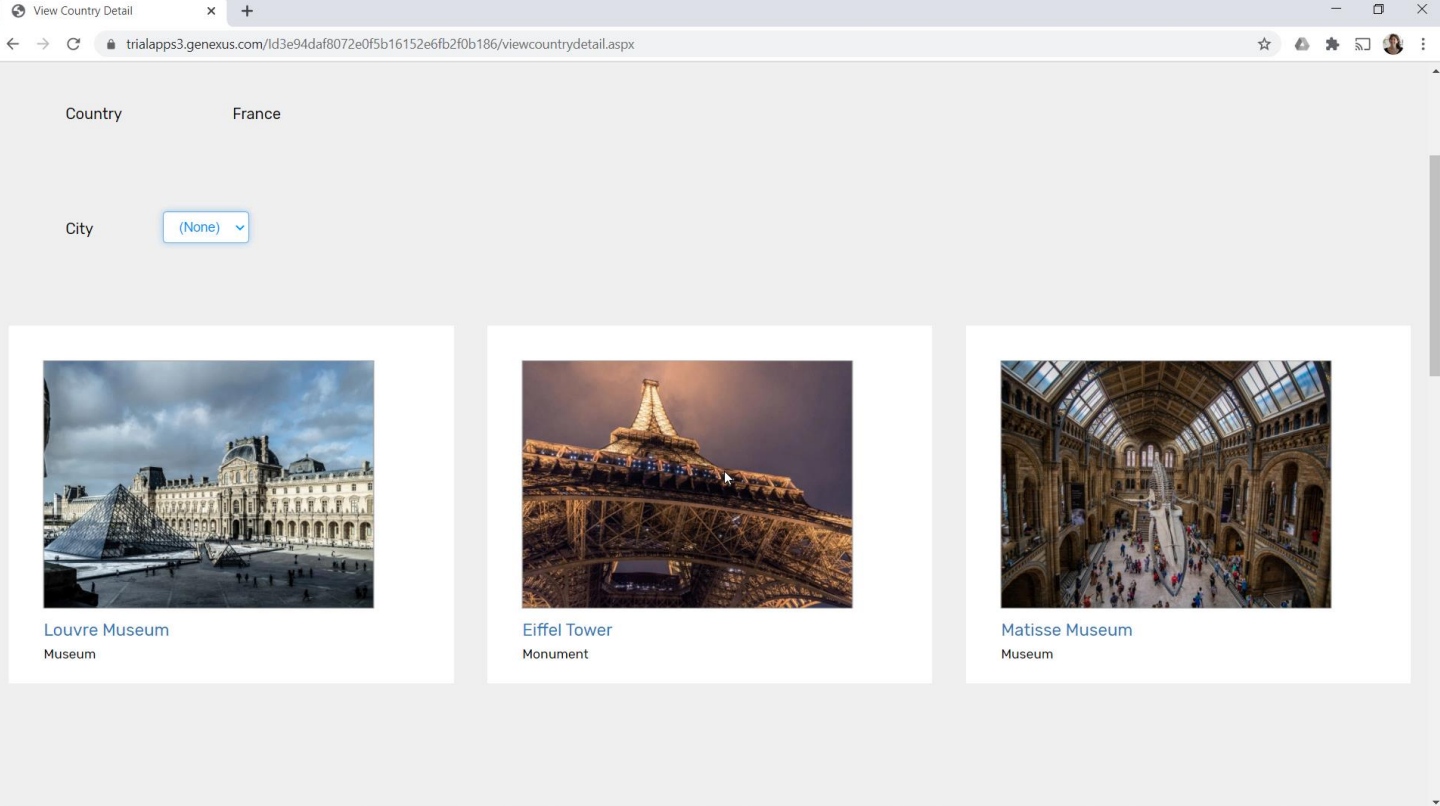
Country France

City

- (None)
- (None)
- Paris

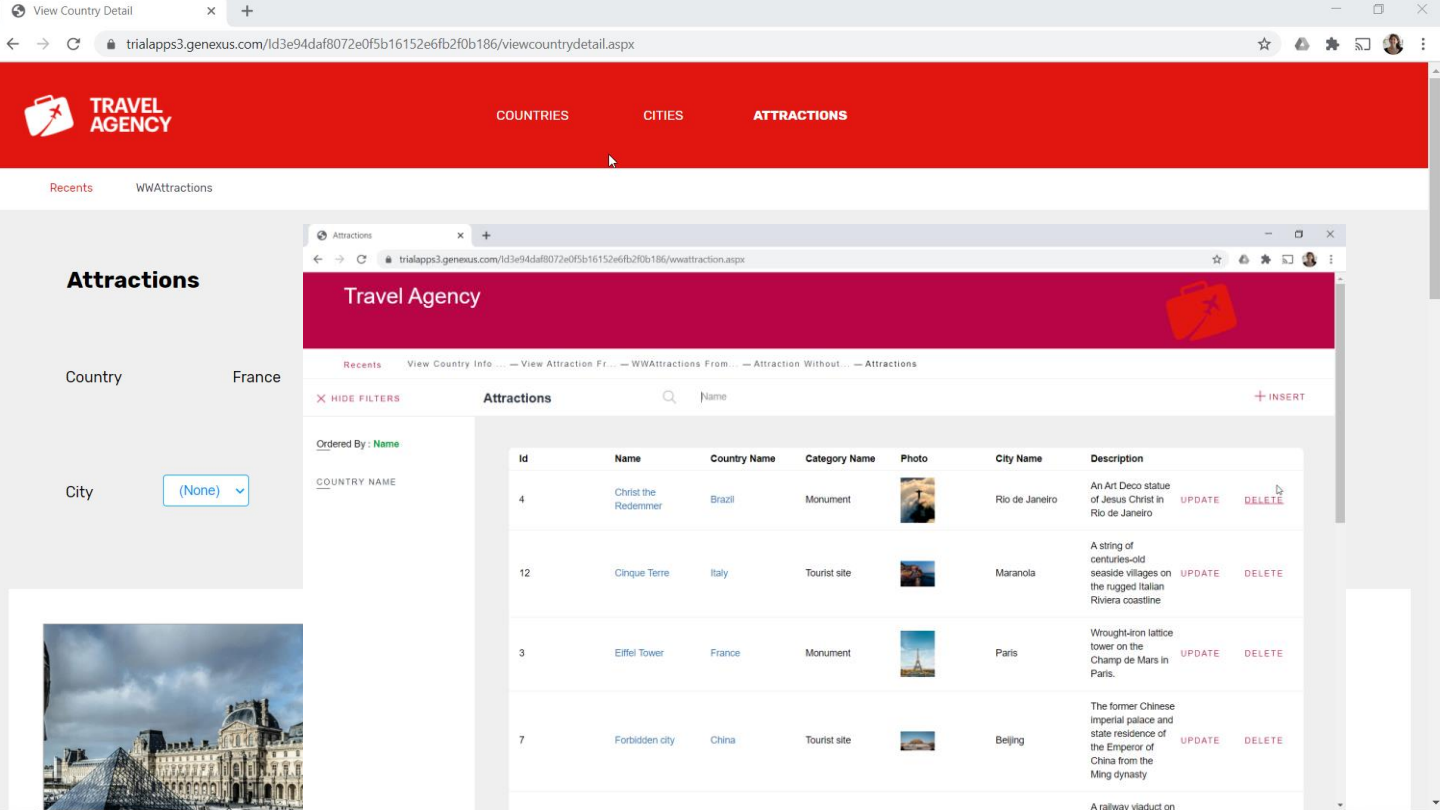


We see that this panel is loading with fixed data, which are the ones imported with Sketch, in order to immediately see how it would look. That's why we see the country name France and a combo that allows choosing between None and Paris.



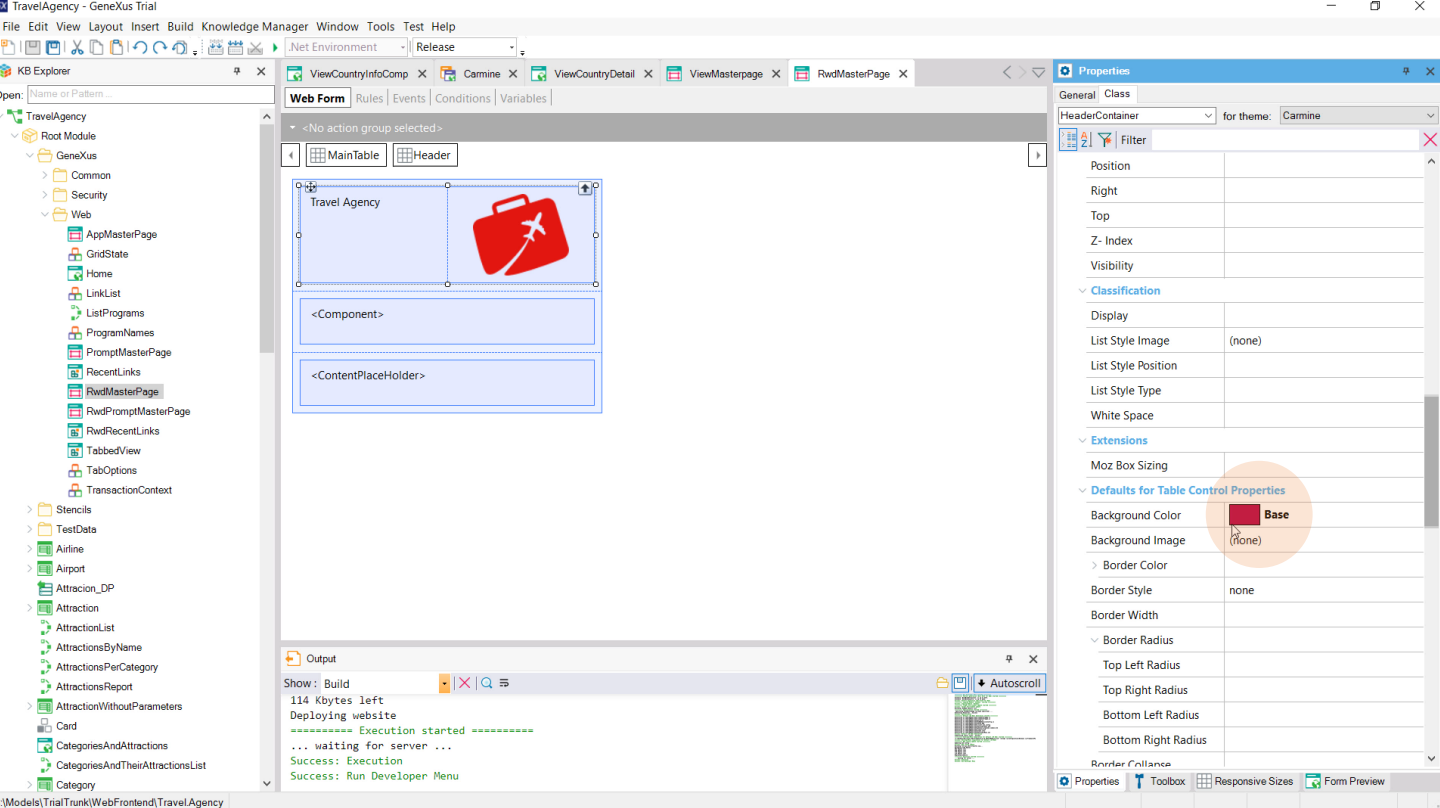
Also, we see three tourist attractions that are hardcoded; that is, specified manually. Of course, we will have to modify all this to take the data from the database.

And maybe change some things like the spacing between controls, since this import is not pixel-perfect, as it will be soon when it matures. But, for starters, directly importing the design has saved us a lot of work. Let's think about the difference between this and implementing everything from scratch.

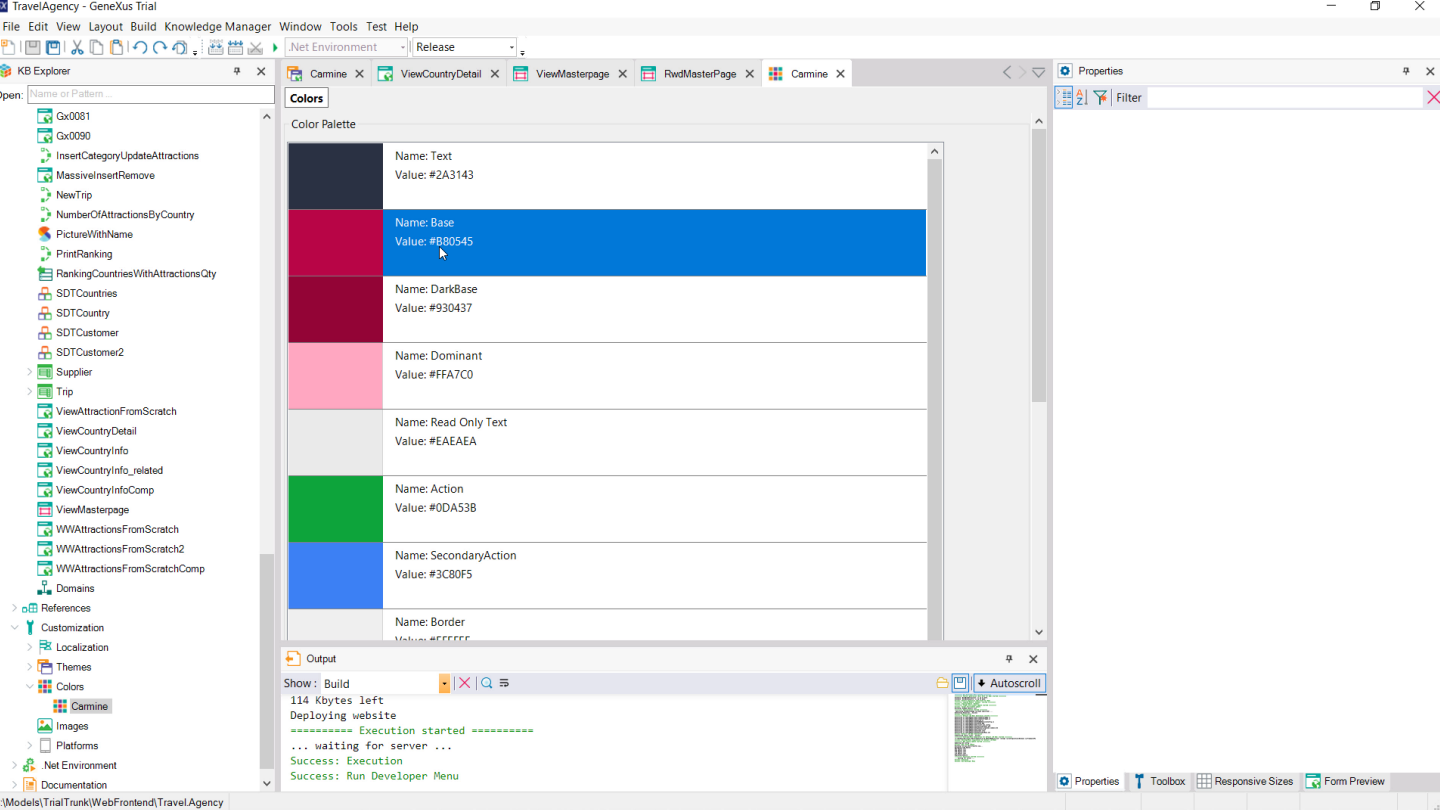


Obviously, if our intention is to turn the master page of this panel into the master page of the entire back-office, the base color of the rest of the application should be this one and not the previous one.

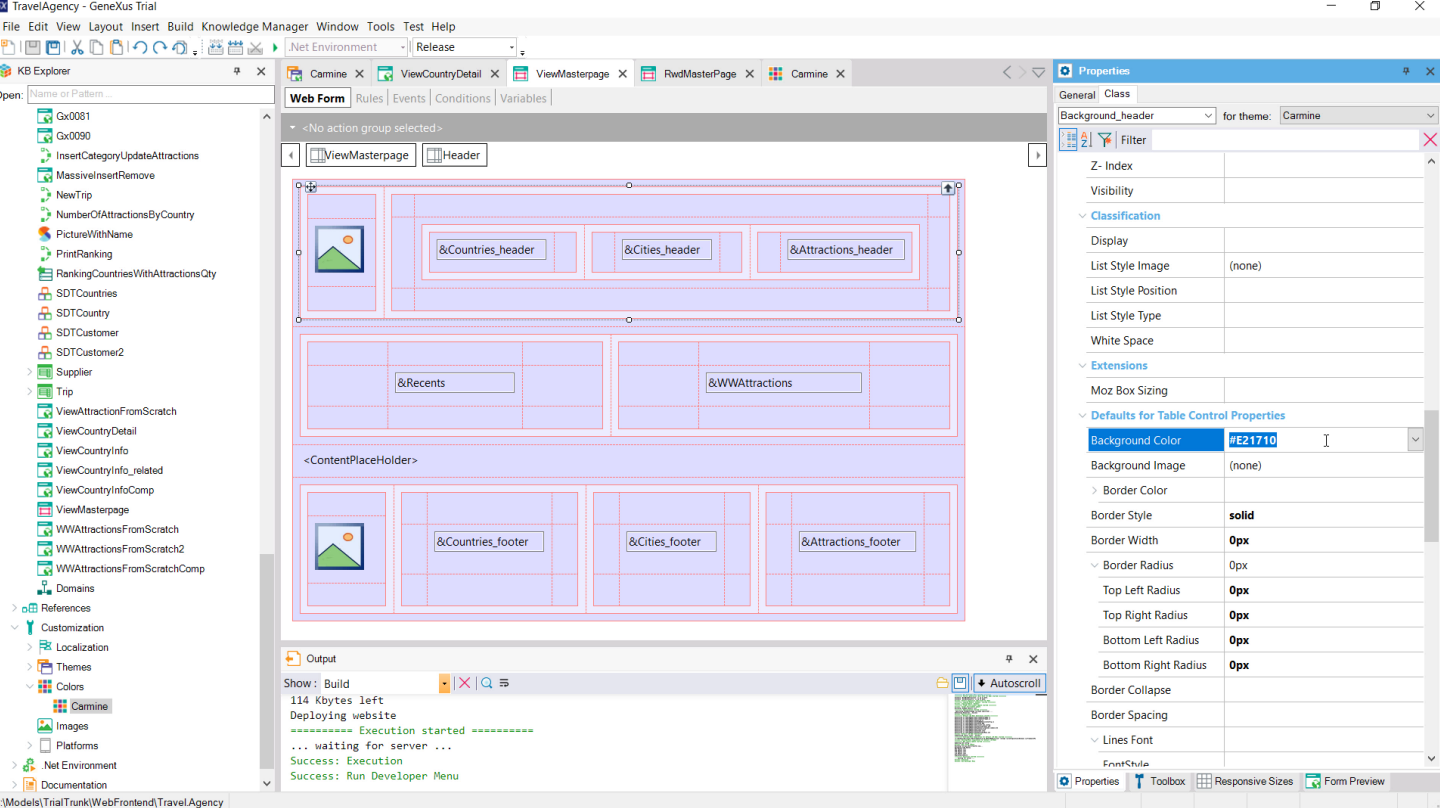
Not only the application bar, but all actions and buttons should take this other color.



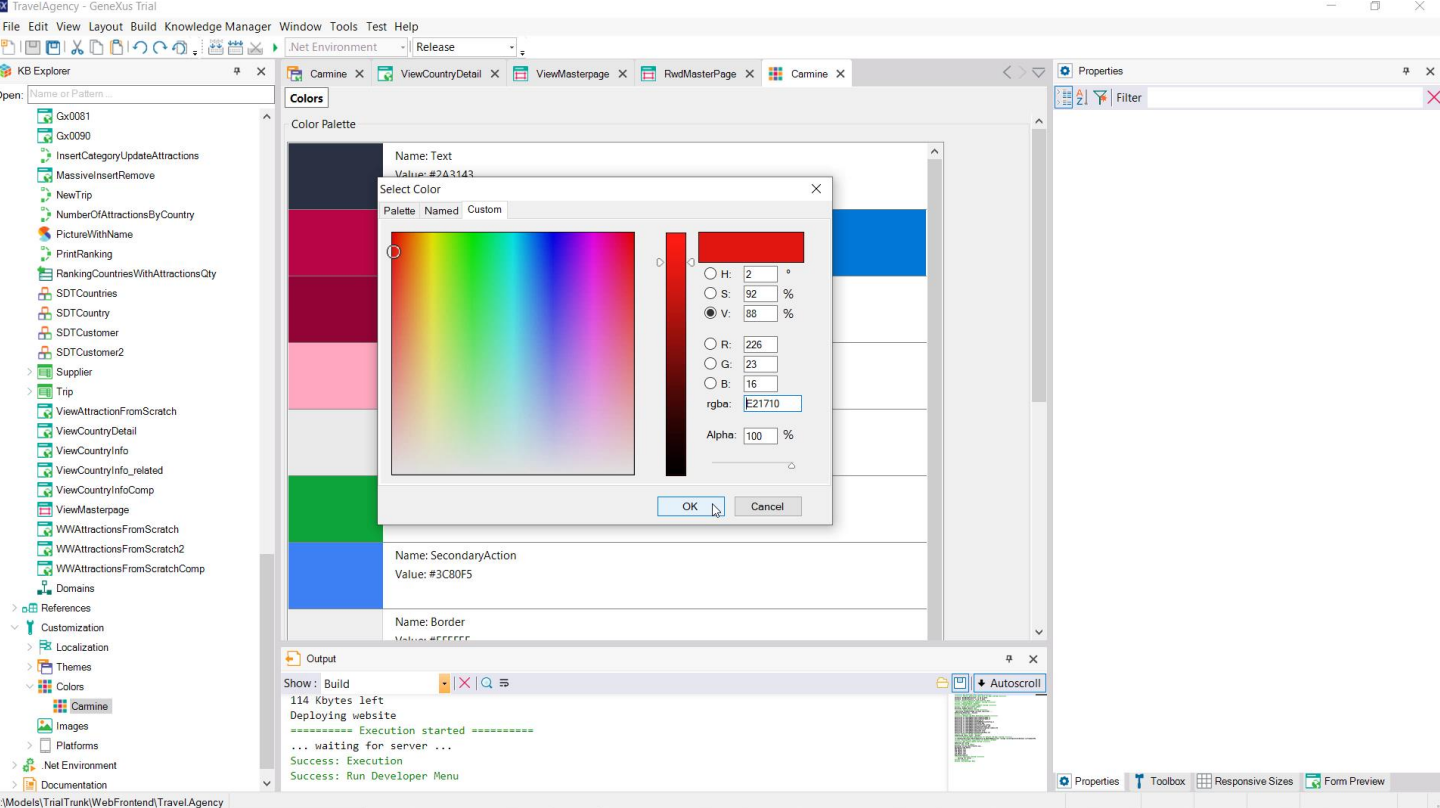
If we look at the class of the table that implements the application bar in the default master page, we see that the background color was this: Base. We need to change it. Where is it configured?



In the color palette associated with the Carmine theme, with the same name. So, there we change its color to the one the designers used in the application bar of the new master page.



We copy the code...



...Double-click here, and paste it in this other place.

That's it. We save and run.





Travel Agency

Recents WWAttractions From... — Attractions

X HIDE FILTERS Attractions + INSERT

Ordered By : Name

COUNTRY NAME

Id	Name	Country Name	Category Name	Photo	City Name	Description		
4	Christ the Redemmer	Brazil	Monument		Rio de Janeiro	An Art Deco statue of Jesus Christ in Rio de Janeiro	UPDATE	DELETE
12	Cinque Terre	Italy	Tourist site		Maranola	A string of centuries-old seaside villages on the rugged Italian Riviera coastline	UPDATE	DELETE
3	Eiffel Tower	France	Monument		Paris	Wrought-iron lattice tower on the Champ de Mars in Paris.	UPDATE	DELETE
7	Forbidden city	China	Tourist site		Beijing	The former Chinese imperial palace and state residence of the Emperor of China from the Ming dynasty	UPDATE	DELETE
						A railway viaduct on		

Travel Agency

Recents Attraction — Attractions — WWAttractions From... — View Attraction Fr... — View Country Info ...

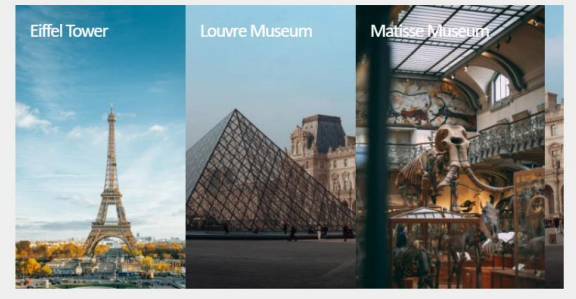
Country Name

Attraction Name From

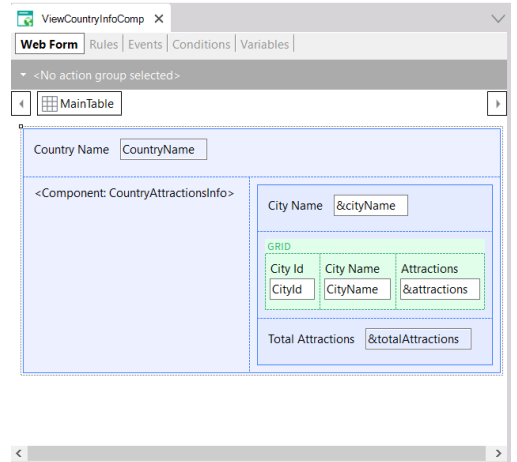
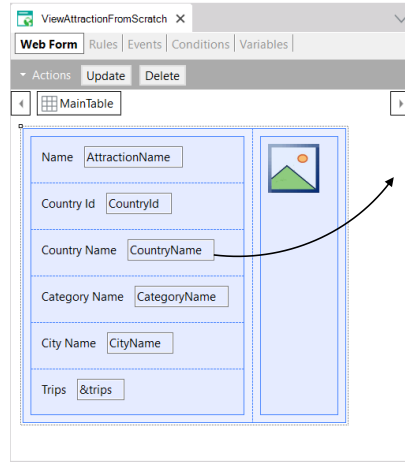
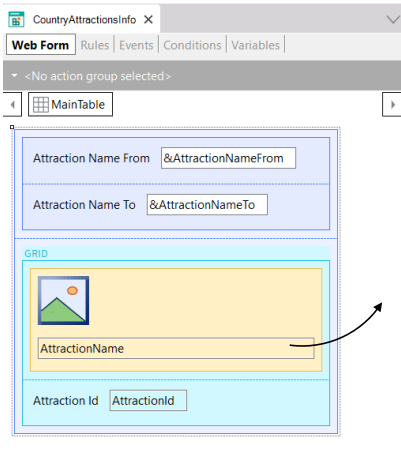
Attraction Name To

City Name

City Id	City Name	Attractions
1	Paris	2
2	Nice	1
Total Attractions		3



What if now we want to change the master page of these web panels?



Master Page: ViewMasterpage

Master Page: ViewMasterpage

We go to GeneXus...

This is the component that shows the attractions in a horizontal grid, and when we click on the name of one of them, it opens this other one, which shows the information of the attraction. If we click on its country name, it opens the last one, which is the one we will replace later with the new one.

So, let's change the master page of this one, and of the previous one.

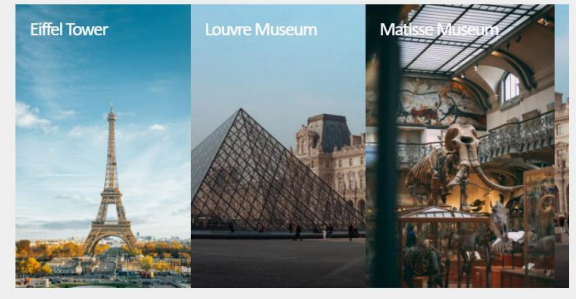
Country Name France

Attraction Name From

City Name

Attraction Name To

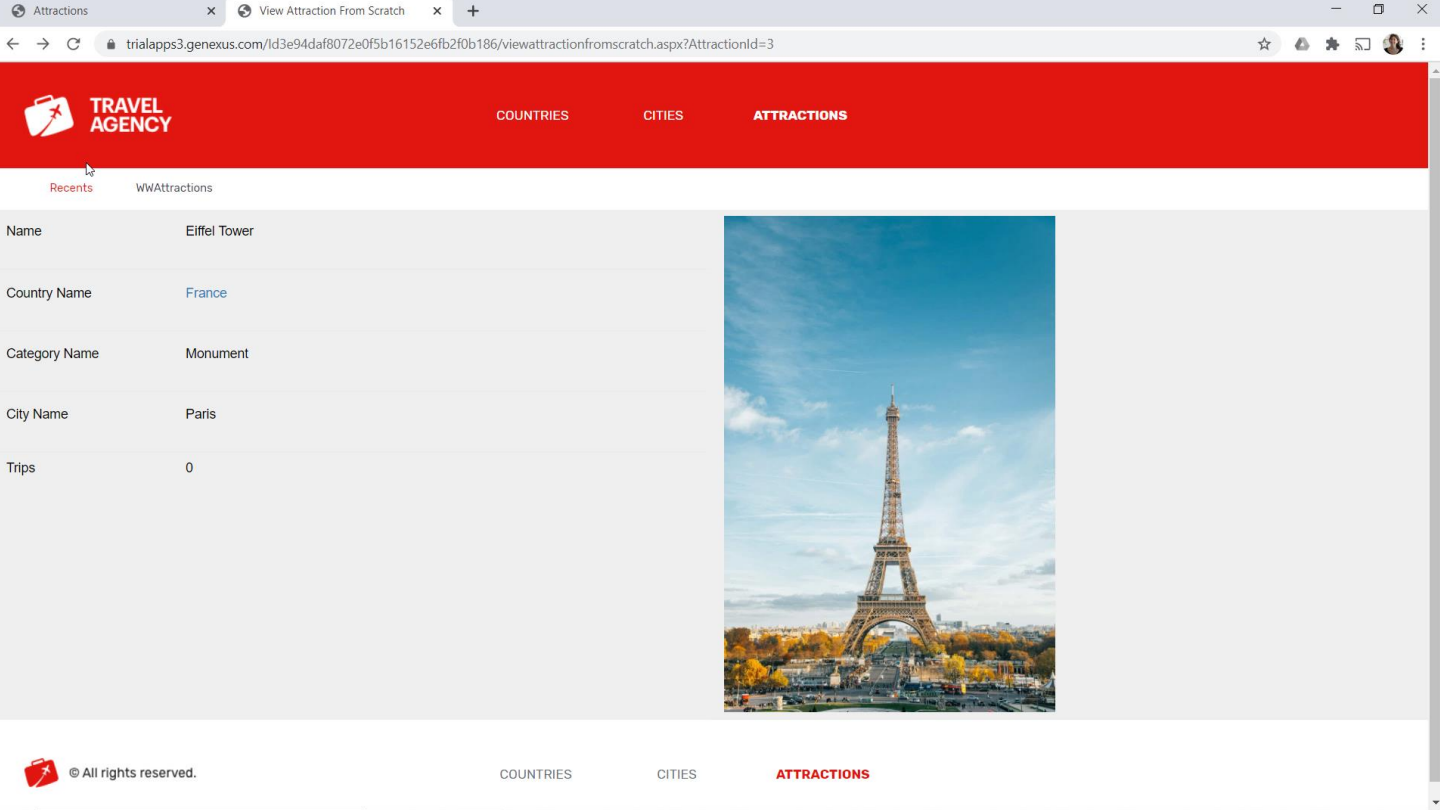
City Id City Name Attractions



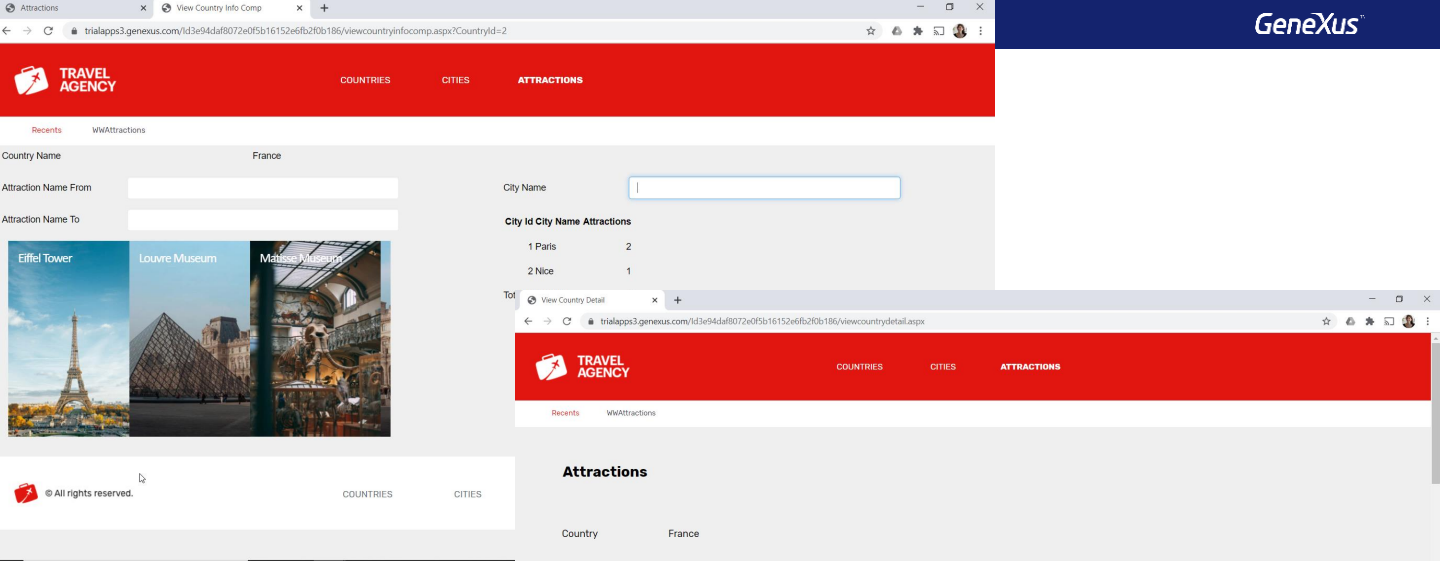
1 Paris	2
2 Nice	1
Total Attractions	3

Let's try it.

We refresh, and here we see the master page perfectly applied.



And also for the previous panel: the one that called this last one. Not so for the first one, because we didn't change it.



Now we will want to replace this panel with the new one, but loading all its data from the database.

Web Form Rules Events Conditions Variables

```
1 parm(in: &CountryId);
2
```

The screenshot displays the GeneXus IDE interface. At the top, there are tabs for 'CountryAttractionsInfo', 'ViewAttractionFromScratch', 'ViewCountryInfoComp', 'Camrine', and 'ViewCountryDetail *'. Below the tabs, a 'Web Form' is being designed. The design area shows a grid with several fields: '&Attractions', 'Country', '&Country_name', and '&Combobox_City'. A 'GRID' label is visible at the bottom left of the design area.

On the right side, the 'Properties' panel is open, showing the properties for the selected control '&Country_name'. The properties are organized into sections: 'General', 'Appearance', and 'Control Info'. The 'General' section includes 'Control Name' (&Country_name), 'Attribute' (&Country_name), 'ReadOnly' (True), 'Return On Click' (False), and 'On Click Event'. The 'Appearance' section includes 'Label Position' (None), 'Class' (Country_name), 'Invite Message', 'Auto Resize' (True), 'Width' (40chr), 'Height' (1row), 'Maximum Number of Lines' (0), 'Format' (Text), and 'Tooltip Text'. The 'Control Info' section includes 'Control Type' (Edit), 'Input Type' (Values), 'Suggest' (No), and 'Auto completion' (True).

In the center, a 'Variables' panel is open, showing a list of variables. The 'Standard Variables' section is expanded, showing the following variables and their types:

Name	Type
& Variables	
Standard Variables	
Attractions	VarChar(40)
Combobox_City	Name
Country_name	Name
CountryId	Attribute:CountryId
ViewCountryDetail_Grid_At...	ViewCountryDetail_Grid_Attractions_SDT

The first thing we must do is receive the country identifier as a parameter. We add the variable to the object, which will be based on the attribute with the same name.

Then, we see that the variable Country_name should have the same data type as that of the CountryName attribute, which is Name, so we change it.

The screenshot shows the GeneXus IDE interface. The main window displays a code editor with the following code:

```

1 /* Generated by GeneXus Sketch Import [Start] */
2
3 Event Load
4   For &ViewCountryDetail_Grid_Attractions_SDT in ViewCountryDetail_Grid_Attractions_DP()
5     Load
6     EndFor
7   EndEvent
8
9 /* Generated by GeneXus Sketch Import [End] */
10
11
12 Event Start
13   For each Country
14     where CountryId = &CountryId
15     &Country_name = CountryName
16   endfor
17 Endevent

```

A blue-bordered box highlights the following line of code:

```
&Country_name = find(CountryName, CountryId = &CountryId, "")
```

On the right side, the 'Properties' panel is open, showing the 'Variable: &Country_name' definition:

Name	Country_name
Description	Country_name
Column title	Name
Class	Attribute

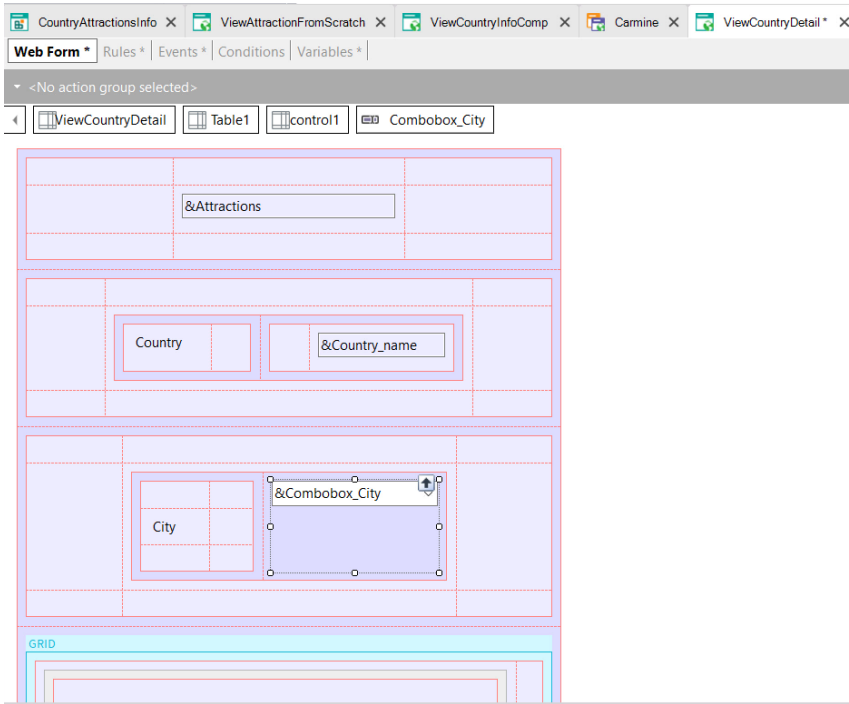
Below this, the 'Type Definition' section is also visible:

Based on	Name
Data Type	Character
Length	50
Collection	False
Dimensions	Scalar
Initial value	"France"

This web panel doesn't have a base table, as we will show. So, we have to find in the database the name of the country received in a parameter. For this, we can program a For Each command in the Start event. In the Country table we will look for the record whose CountryId is equal to the parameter variable, and for that record we assign the value of the CountryName attribute to the variable of the &Country_name form.

Another way to achieve the same would have been to use the inline Find formula, instead of the For Each command.

Now let's remove that initial value it was using, because we don't need it anymore.



Properties

General Class

Filter

Attribute/Variable: **&Combobox_City**

Control Name	&Combobox_City
Attribute	&Combobox_City
ReadOnly	False
Return On Click	False

Appearance

Label Position	None
Class	SketchAttributeWithType
Invite Message	
Auto Resize	True
Width	40chr
Tooltip Text	

Control Info

Control Type	Dynamic Combo Box
Data Source From	Attributes
Item Values	CityName
Item Descriptions	CityName
Sort Descriptions	True
Conditions	CountryId=&CountryId; ...
Instantiated Attributes	
Empty Item	True
Empty Item Text	GX_EmptyItemText
Notify Context Change	False

We must load this combo box variable with the cities of that country. Therefore, we must change the type of control so that it is a combo whose values are loaded dynamically from the database, and not statically as before. The values taken by the combo will be those of the CityName attribute and the descriptions will be the same, CityName (it will save internally and display the same). But also we don't want to show all the cities, but only those of the country received in a parameter. We leave the property set to true to add the empty item (that None value we saw at runtime).

The screenshot shows the GeneXus IDE interface. The main window displays a web form with a grid. The Properties window is open, showing the configuration for the grid's attribute, `&ViewCountryDetail_Grid_Attractions_SDT`. The configuration includes:

- Attribute: `&ViewCountryDetail_Grid_Attractions_SDT`
- Field Specifier: `Attraction_name`
- Readonly: `True`
- Return On Click: `False`
- On Click Event: (empty)
- Appearance:
 - Class: `Attraction_name`
- Behavior:
 - Input History: `True`
 - Is Password: `False`
 - Nulls in Forms: `Empty as Null`

The Structure window shows the SDT's layout and data types:

Name	Type	Description
ViewCountryDetail_Grid_Attractions_SDT	VarChar(40)	View Country Detail_Grid_At
Layout	Image	Attraction_image
Attraction_image	Image	Attraction_image
Attraction_name	VarChar(40)	Attraction_name
Attraction_type	VarChar(40)	Attraction_type

A separate window shows the design of the `AttractionCard` stencil, which includes a placeholder for an image, a text field for `&Attraction_name`, and another text field for `&Attraction_type`.

Well, now we need to change what is shown on the grid, and also filter it.

As we see, the grid has no specified Base Transaction, no Order, and no Conditions. It doesn't even have attributes, but elements of an SDT that was created by importing from Sketch.

So, this grid will load a collection of that SDT. For each item, its elements will be shown according to the stencil design we saw.

Here the SDT is defined, with one element that will store an image of the attraction; another will store the name, and another one the category. These elements will replace these generic variables in the grid stencil.

```

ViewCountryDetail_Grid_Attractions_DP X
Source Rules Variables
1 ViewCountryDetail_Grid_Attractions_DP from Attraction
2 where CountryId = &CountryId when not &CountryId.IsEmpty()
3 where CityName = &CityName when not &CityName.IsEmpty()
4
5 {
6   ViewCountryDetail_Grid_Attractions_SDT
7   {
8     Layout = "Attraction"
9     Attraction_image = AttractionPhoto
10    Attraction_name = AttractionName
11    Attraction_type = CategoryName
12  }
13
14 // ViewCountryDetail_Grid_Attractions_SDT
15 // {
16 //   Layout = "Attraction"
17 //   Attraction_image = Image:Img716fb22b18d7152e866600cbeaab8003628573c9.Link()
18 //   Attraction_name = "Louvre Museum"
19 //   Attraction_type = "Museum"
20 // }
21 // ViewCountryDetail_Grid_Attractions_SDT
22 // {
23 //   Layout = "Attraction"
24 //   Attraction_image = Image:Img4a3b2877897efdc75c540e9d29a16f31520ad4f1.Link()
25 //   Attraction_name = "Eiffel Tower"
26 //   Attraction_type = "Monument"
27 // }
28 // ViewCountryDetail_Grid_Attractions_SDT
29 // {
30 //   Layout = "Attraction"
31 //   Attraction_image = Image:a37c31622388e897a74a56d1aa41284498962234.Link()
32 //   Attraction_name = "Matisse Museum"
33 //   Attraction_type = "Museum"
34 // }
35 }

```

```

ViewCountryDetail_Grid_Attractions_DP X
Source Rules Variables
1 param(in: &CountryId, in: &CityName);
2

```

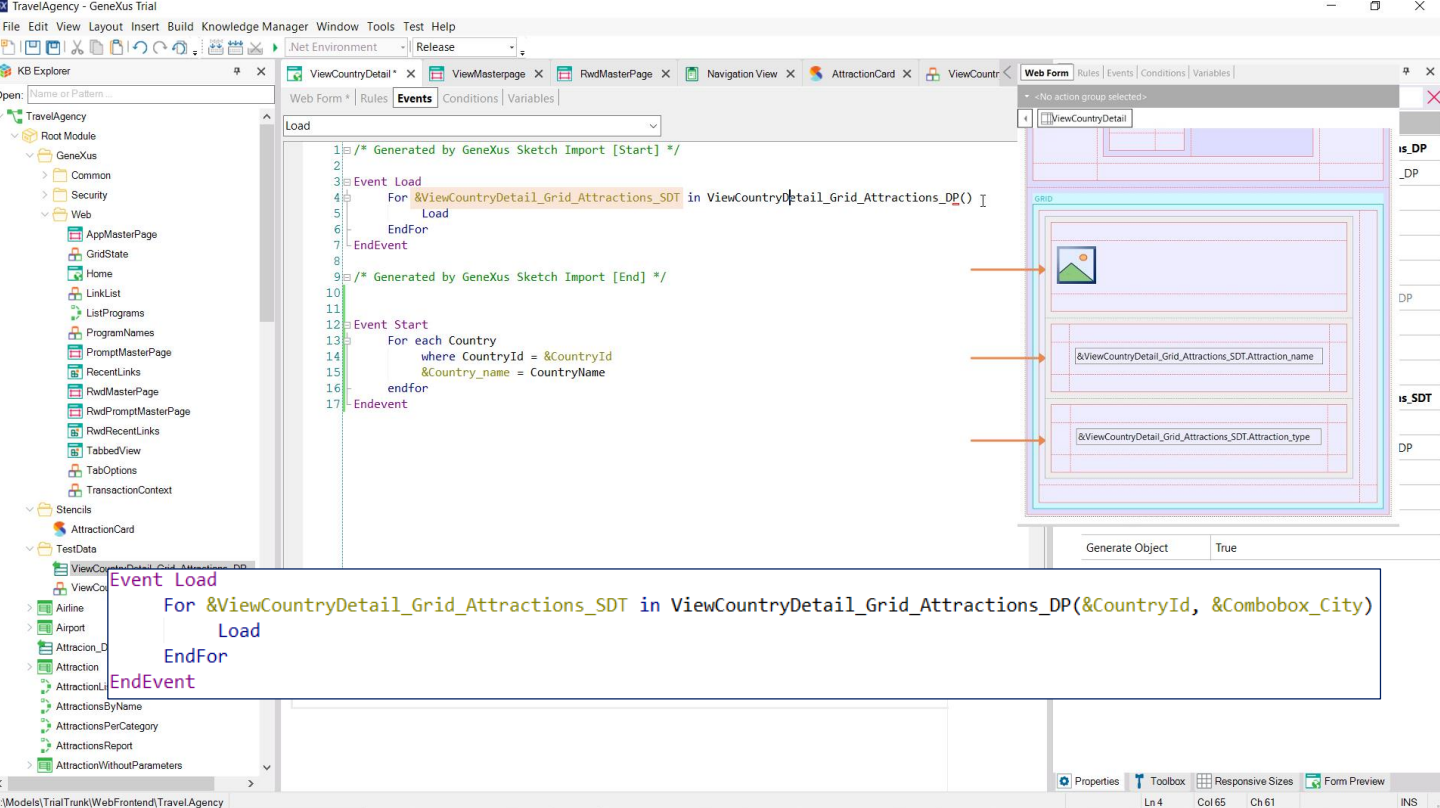
But the one that will actually return a data collection from this SDT with the attractions loaded will be this DataProvider imported with Sketch. The only thing we have to do is modify it so that the items are loaded from the attractions in the database, and not from these fixed data.

So we leave these three groups with comments and add one that will be variable, specifying that we will take the data from the Attraction base transaction, filtering by country identifier when it is not empty. And by city name, when it is not empty.

We must define both variables in the parm rule.

But first, for each record found, we want an item to be added to the resulting collection, with the value of the attraction photo, its name, and the name of its category.

Now, let's define the two parameters that this data provider must receive when it is invoked.



And now let's look at its invocation.

It is done in the Load event, of course!

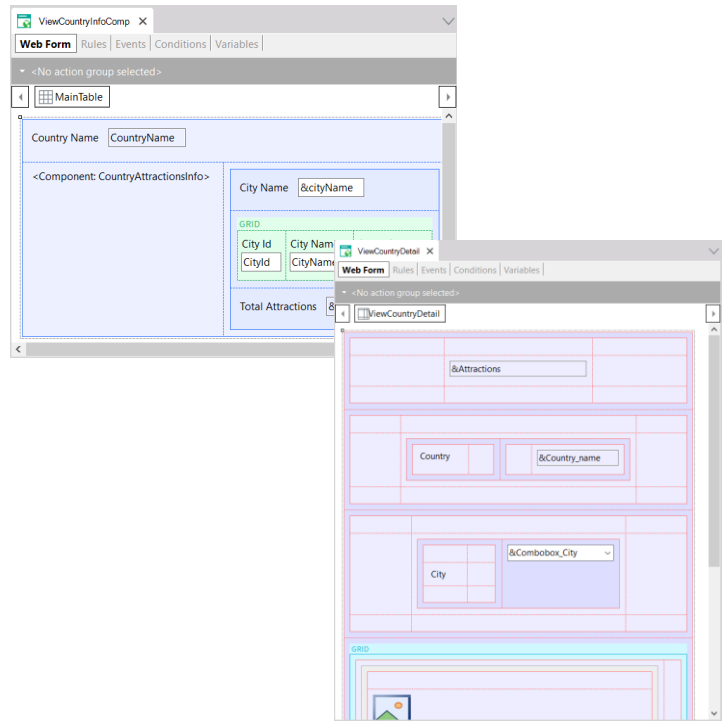
We see that we have a For in command that runs through each item in the collection returned by the Data Provider, and loads it into the grid.

This is where we will have to send it the parameters, so that it filters by country and city.

```

ViewAttractionFromScratch * X
Web Form | Rules | Events * | Conditions | Variables |
CountryName.Click
1 Event Load
2   &trips = Count(TripDate)
3 -Endevent
4
5
6 Event CountryName.Click
7   //ViewCountryInfo(CountryId)
8   //ViewCountryInfo_related(CountryId)
9   //ViewCountryInfoComp(CountryId)
10  ViewCountryDetail(CountryId)
11 -Endevent
12
13
14 Event Start
15   CountryId.Visible = False
16 -Endevent
17

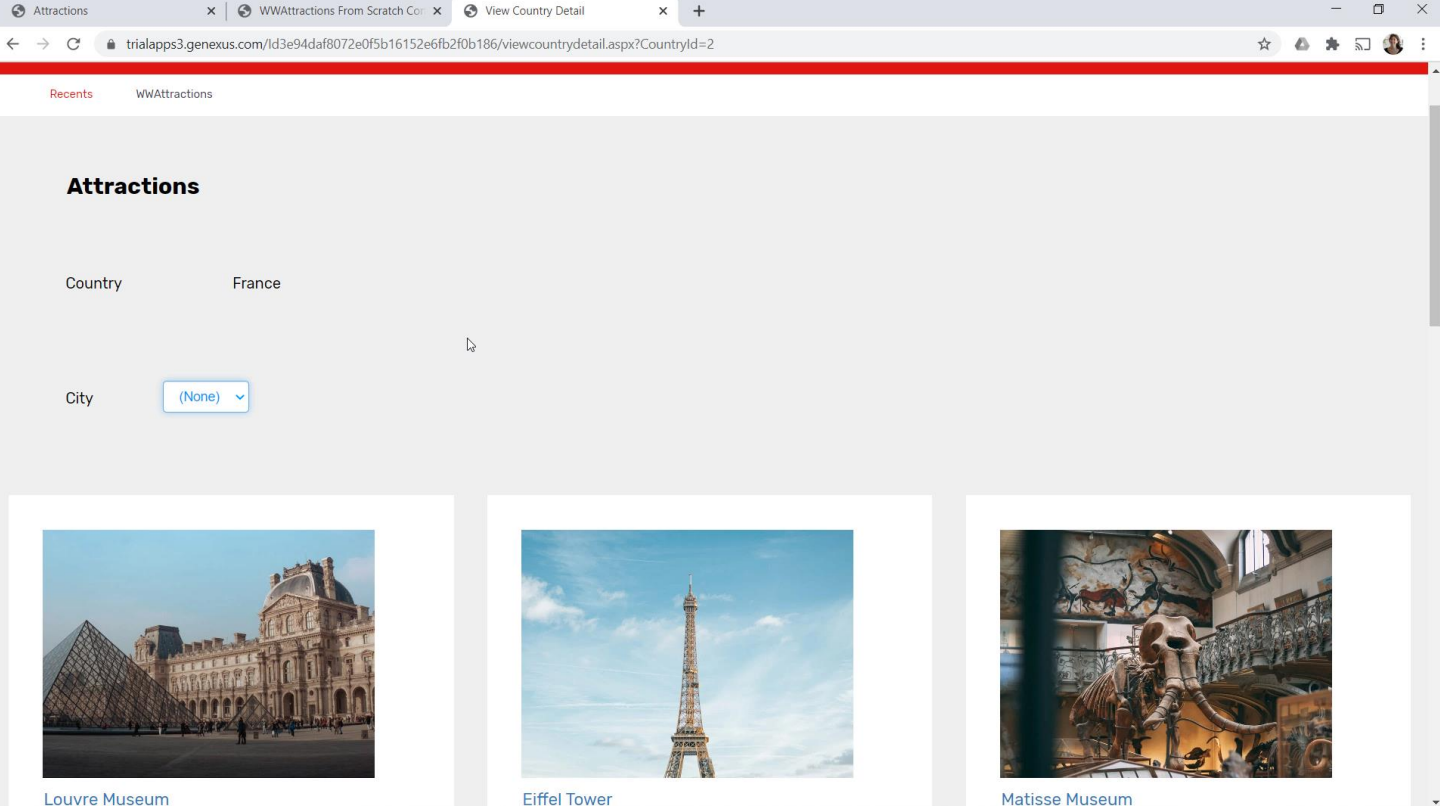
```



The only thing left to do is to invoke this web panel instead of the one we had.

That is, instead of this one, this other one.

We save and run.



Let's select this attraction. And see the information about its country, France.

There we see deployed the three attractions that we have from France in the database.

Attractions

WWAttractions From Scratch Co x View Country Detail

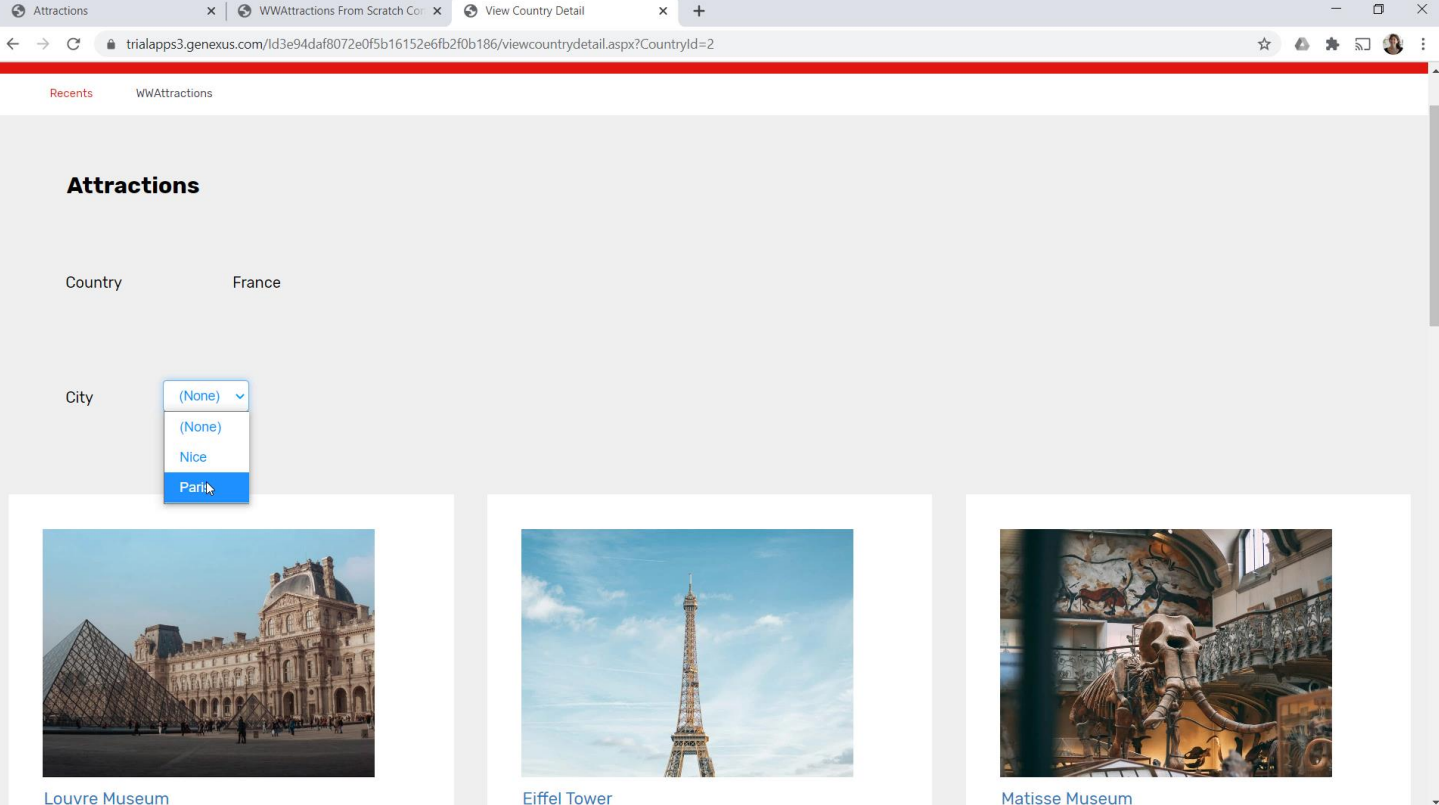
trialapps3.genexus.com/ld3e94daf8072e0f5b16152e6fb2f0b186/viewcountrydetail.aspx?CountryId=2

Recents WWAttractions

Attractions

Country France

City (None) (None) Nice Paris

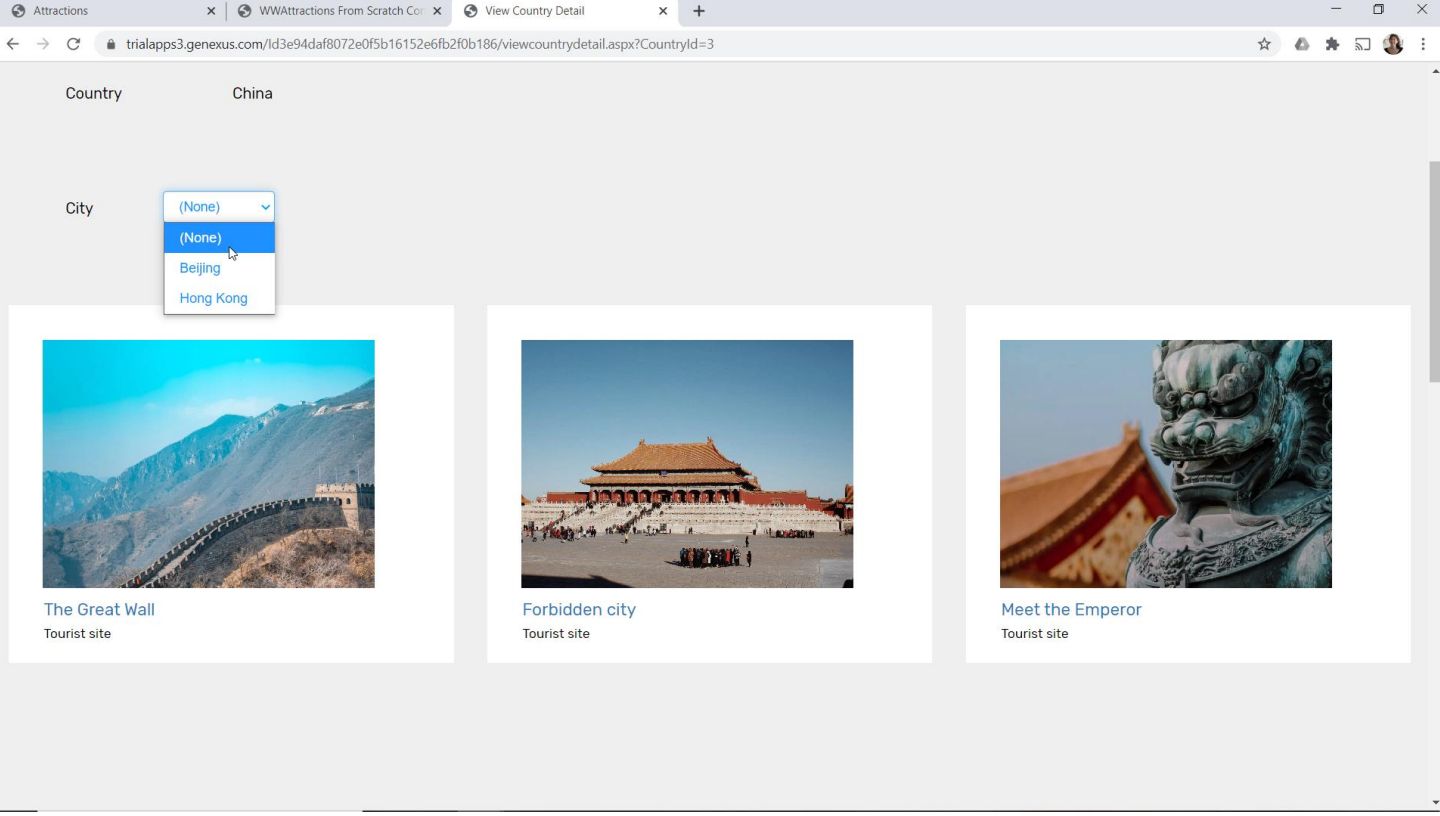


Louvre Museum

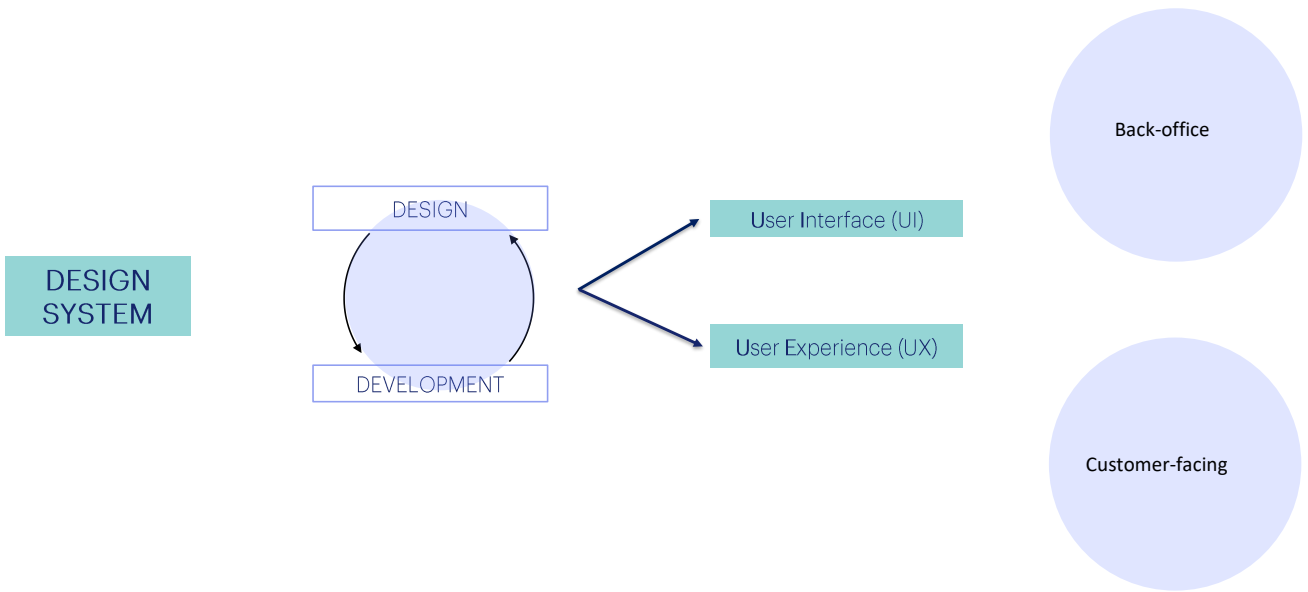
Eiffel Tower

Matisse Museum

And the two cities, and filtering by Paris we see two attractions and by Nice, one attraction.



If we choose another attraction, from China this time, we see the three attractions we have in China.



With this small example we can have an idea of how the incremental development process of an application will be, both in terms of logic and design, and to what extent we should work as a team with people with different skills.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications