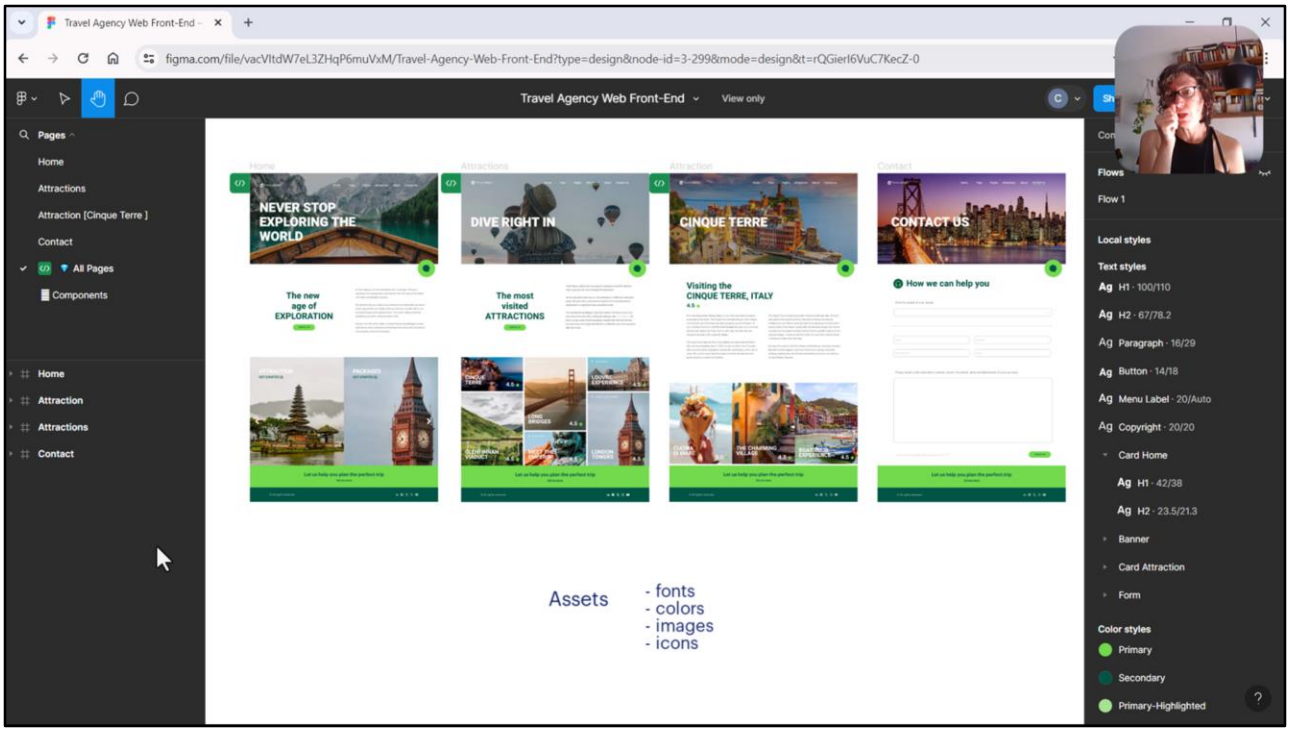
GeneXus by Globant

Assets

Fonts, Colors, Images

Cecilia Fernández

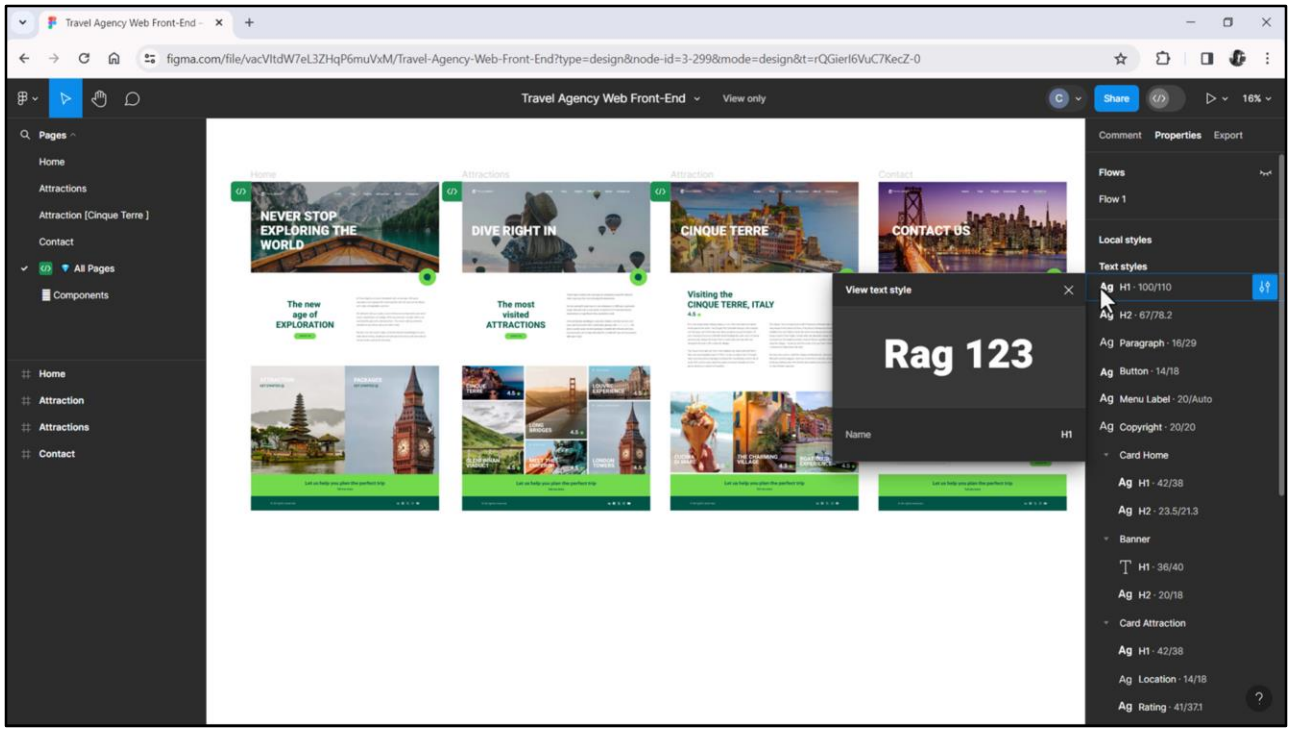


In the previous module, we saw how to implement these two parts of the layout, this one and this one, so that they looked just like the design. If we use the metaphor of the forest, we were analyzing two leaves of two trees, but we hardly looked up to see the forest as a whole.

Now that we know the **basic elements** involved in the development of a Frontend – and that's why we have paid attention to these two leaves – we can think about how we should start the initial stages of the development of a Frontend project.

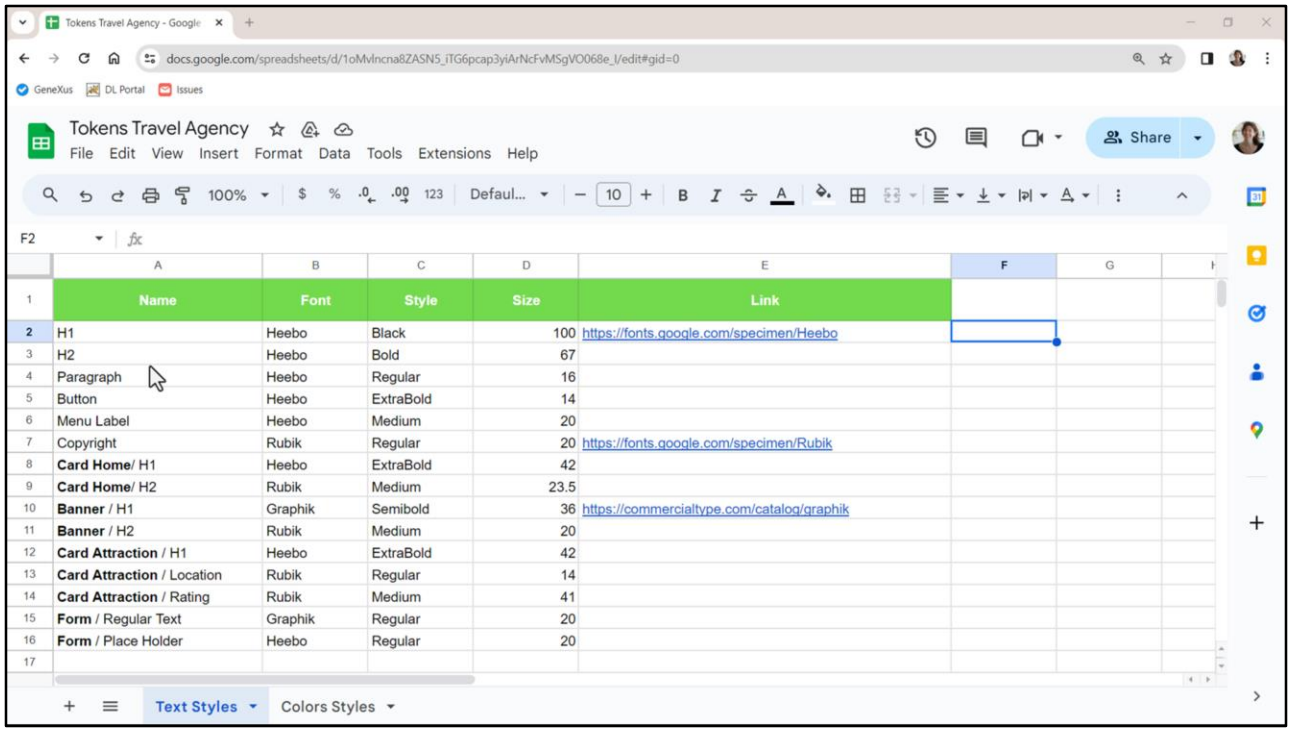
To streamline the development stage itself, in general it is convenient to integrate the assets from the beginning. This includes all the resources that we will have to use in the project, such as fonts, colors, images, and icons, so as not to stop the development every time we need them and go looking for them to incorporate them in the KB. Also, because doing everything from the beginning, with an overall view, can help to be more systematic, that is to say, more consistent.

So if I were just starting out with the project, first of all I would get the necessary fonts and integrate them into the KB.



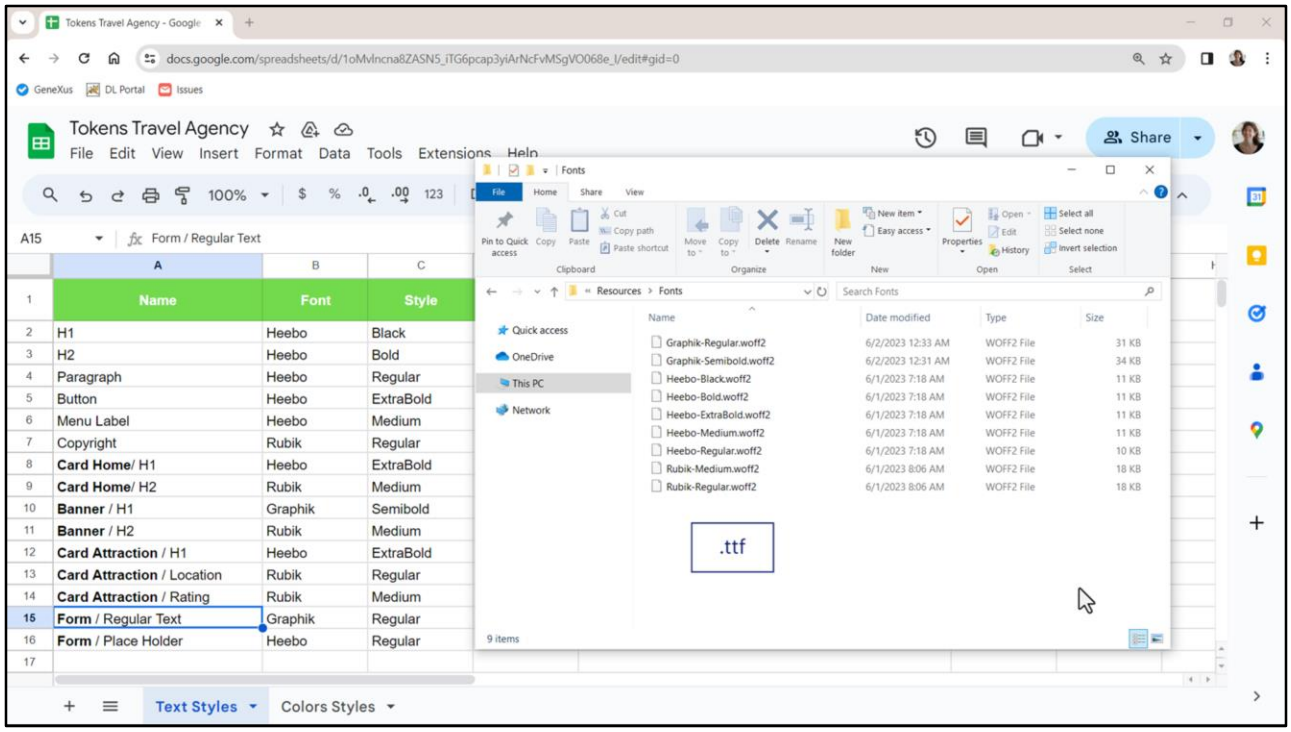
Here, on the page that contains all the screens, I can see the list of the typographic styles used in them... But in view mode (which is the free one) we can't see the fonts that each one uses. We only see their names, size and line height.

If we are going to continue in view mode, we should ask our designer to share with us the fonts that each of these styles will use. That way we don't have to inspect the elements of the frames looking for where each typographic style is used.



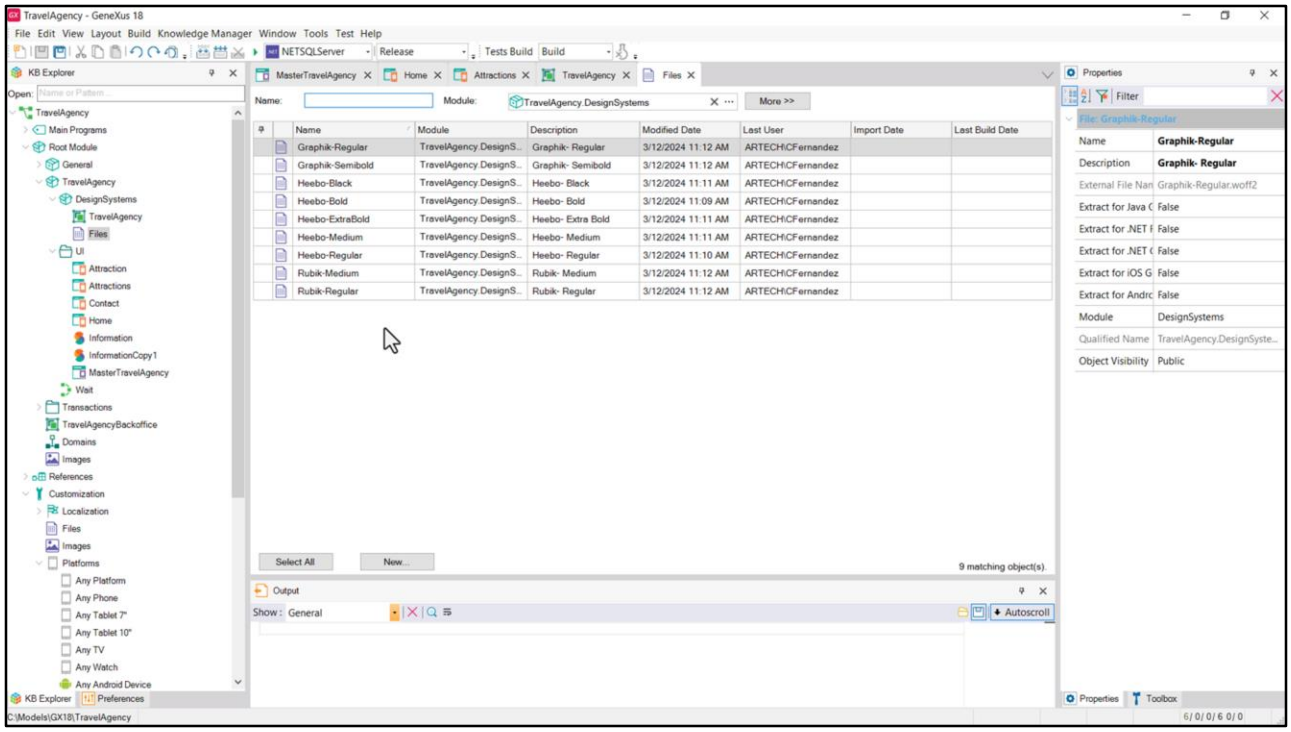
	Name	Font	Style	Size	Link
2	H1	Heebo	Black	100	https://fonts.google.com/specimen/Heebo
3	H2	Heebo	Bold	67	
4	Paragraph	Heebo	Regular	16	
5	Button	Heebo	ExtraBold	14	
6	Menu Label	Heebo	Medium	20	
7	Copyright	Rubik	Regular	20	https://fonts.google.com/specimen/Rubik
8	**Card Home**/ H1	Heebo	ExtraBold	42	
9	**Card Home**/ H2	Rubik	Medium	23.5	
10	**Banner** / H1	Graphik	Semibold	36	https://commercialtype.com/catalog/graphik
11	**Banner** / H2	Rubik	Medium	20	
12	**Card Attraction** / H1	Heebo	ExtraBold	42	
13	**Card Attraction** / Location	Rubik	Regular	14	
14	**Card Attraction** / Rating	Rubik	Medium	41	
15	**Form** / Regular Text	Graphik	Regular	20	
16	**Form** / Place Holder	Heebo	Regular	20	
17					

So, Chechu sent me this spreadsheet with all the typographic styles and in this column is the font family. She even shared with me links so I can download those families. Graphik is not free, but I already had it. If you don't have that family, then use Rubik or Heebo in these two cases.

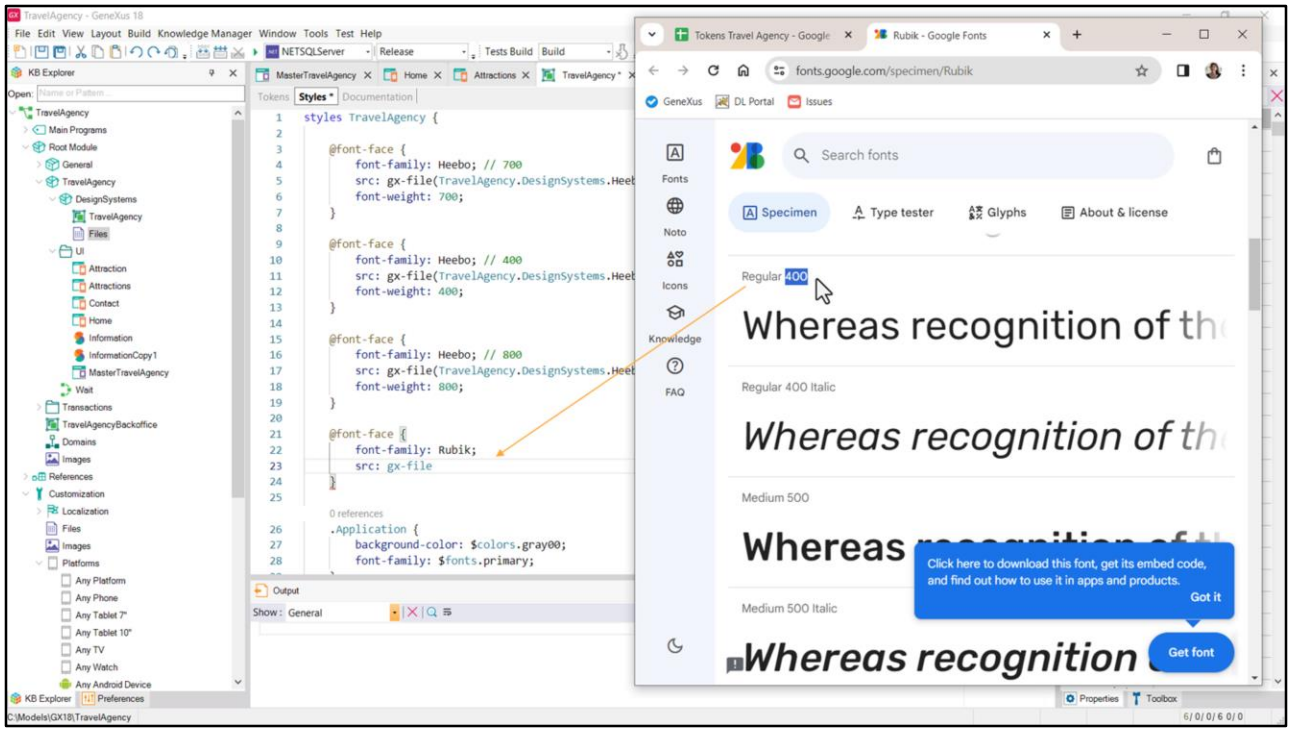


Here I had already downloaded the files for the fonts before. Do you remember many videos ago, I told you that I had downloaded them in woff2 format because it is very light and useful for Angular? The disadvantage is that it can't be used for native applications. Instead, we could have downloaded the fonts in TrueType format (ttf) and then they would be exactly the same for the native application.

We have to weigh pros and cons to decide. For now, I will leave those that are best for Angular, and when we move on to consider the native application, if we get there, then we will see how to do it: if we will integrate two versions of the same font (a woff2 and a ttf), or we will modify the Angular application to use the same version as the native one, that is, ttf.

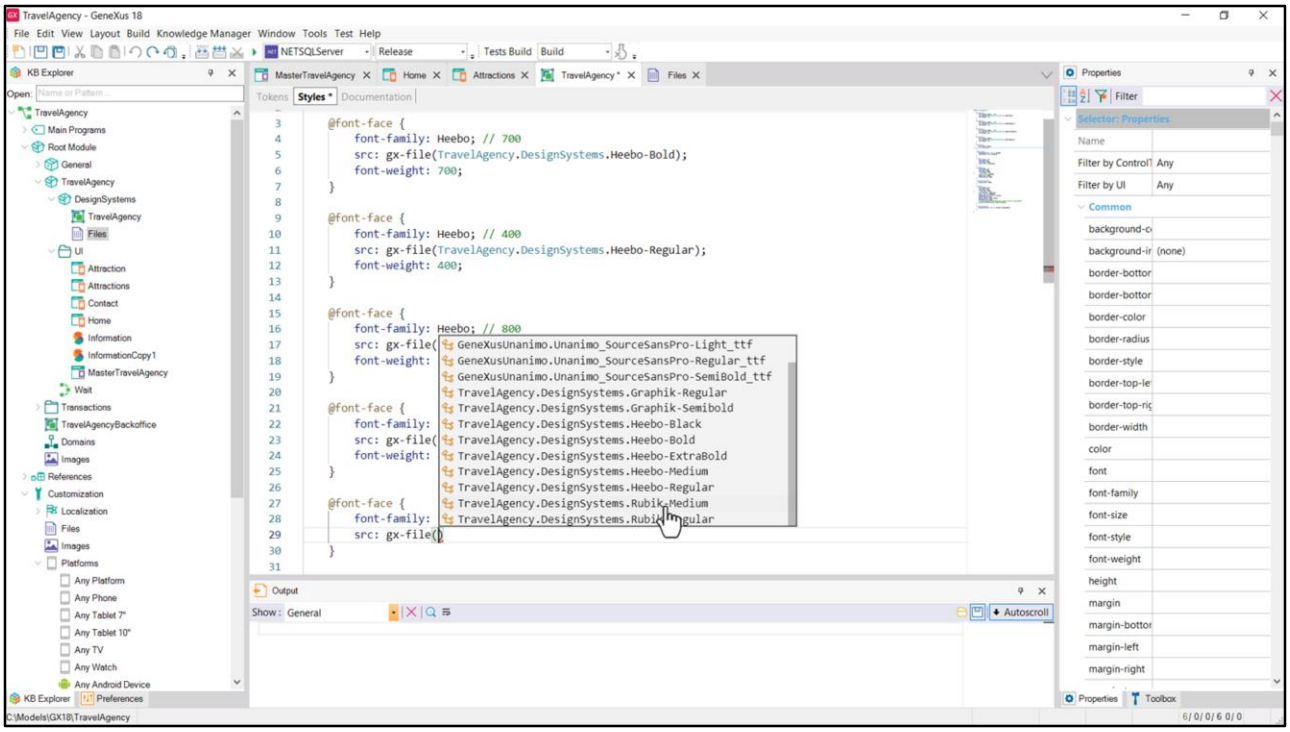


The next step will be to import the font files into GeneXus. I already have done it (we did it in one of the first videos, remember?).

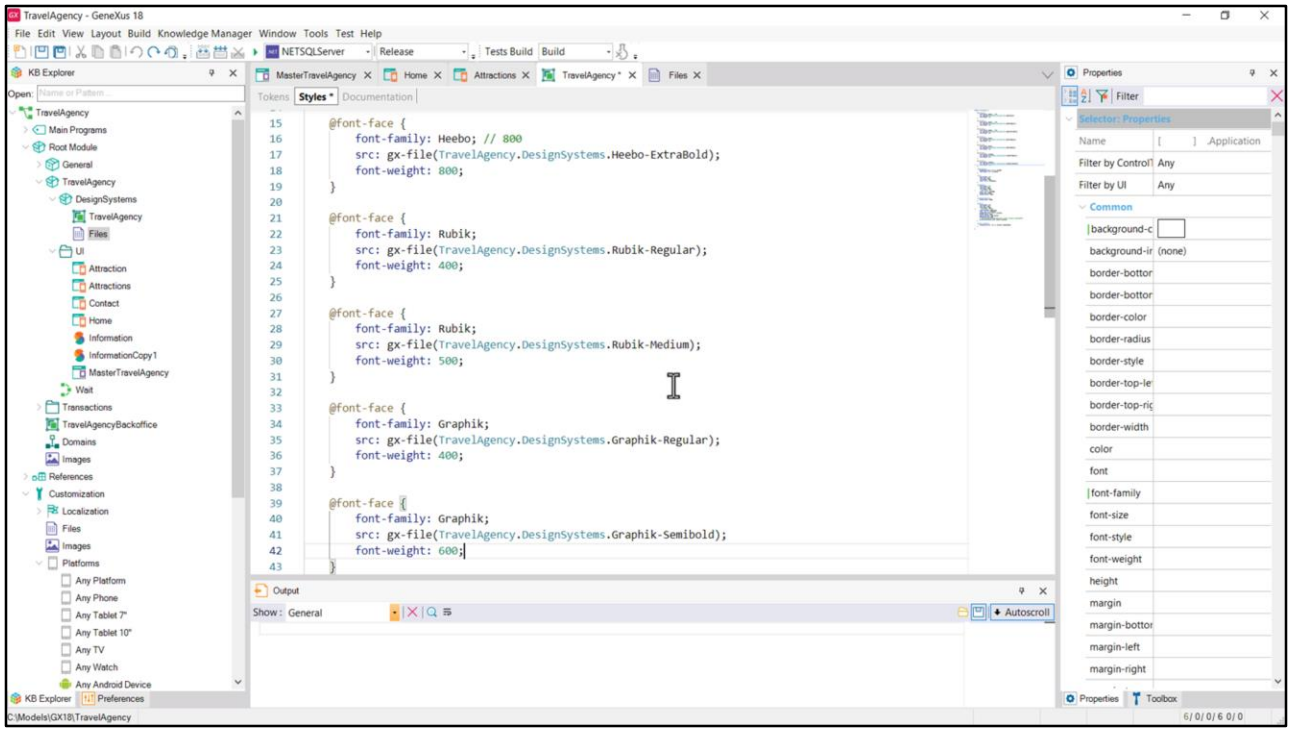


Next, we would declare font-face rules to incorporate all these fonts in our DSO. Here we had these 3 declarations, let's complete with all the rest.

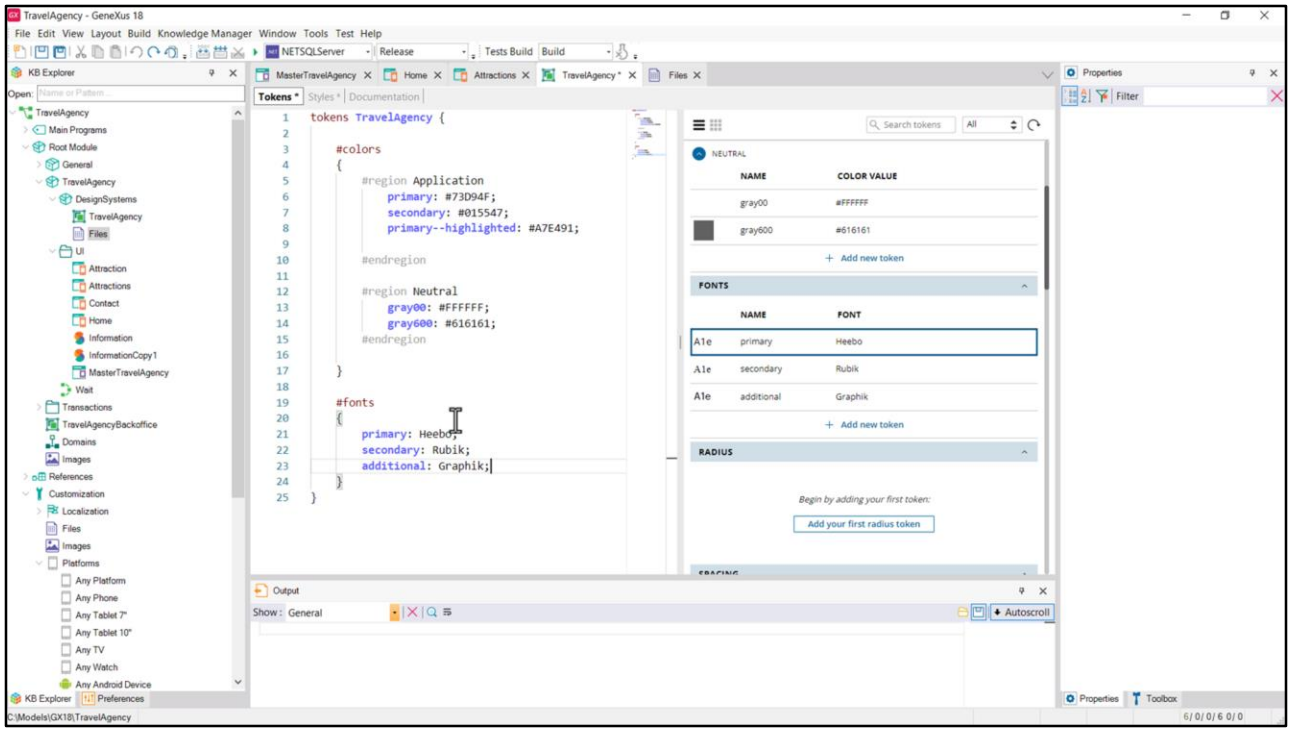
Rubik-Regular, which has 400 weight, and the file is this one here.



Rubik... Medium, which has 500 weight.

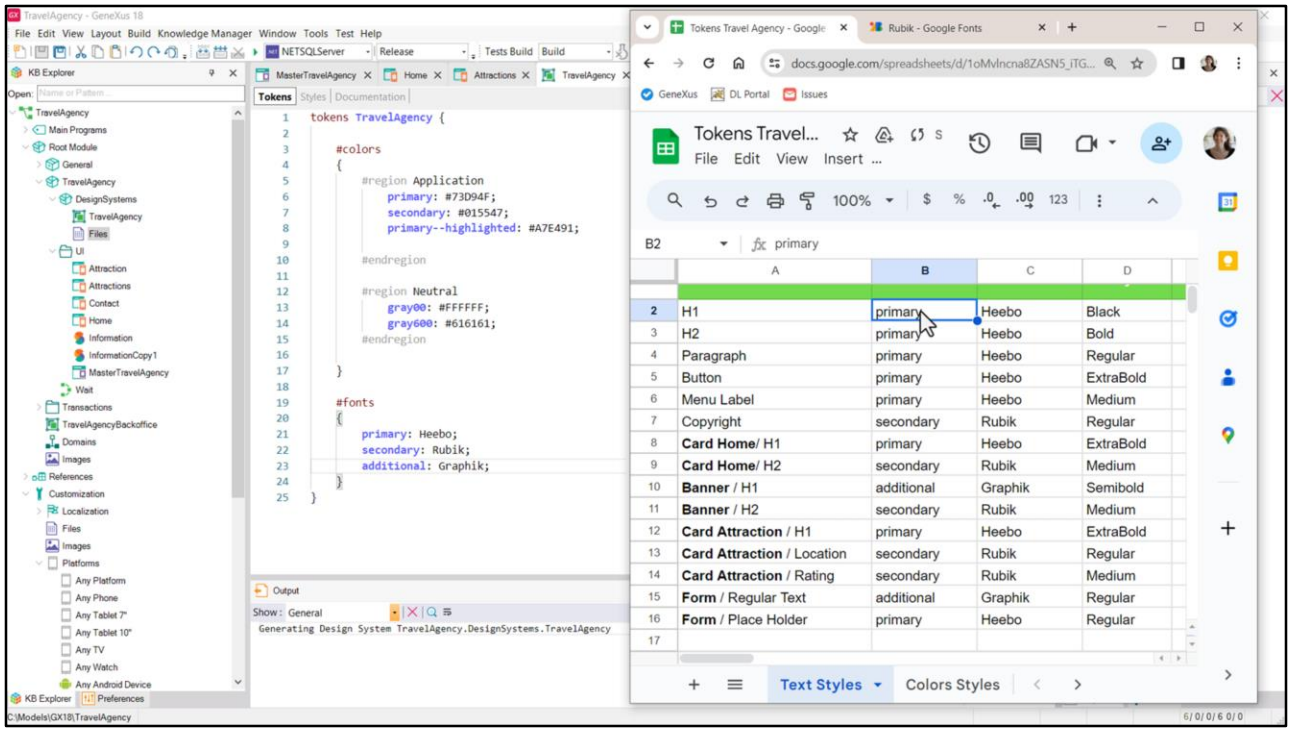


Graphik... we see that we have two, semi-bold and regular. The weight of the regular one is 400, and that of the semi-bold is 600.



We have 3 font families, and Heebo is clearly the main one. This other one, Rubik, is used in several cases, so we could consider it as a secondary font, and this one, Graphik, only in two cases.

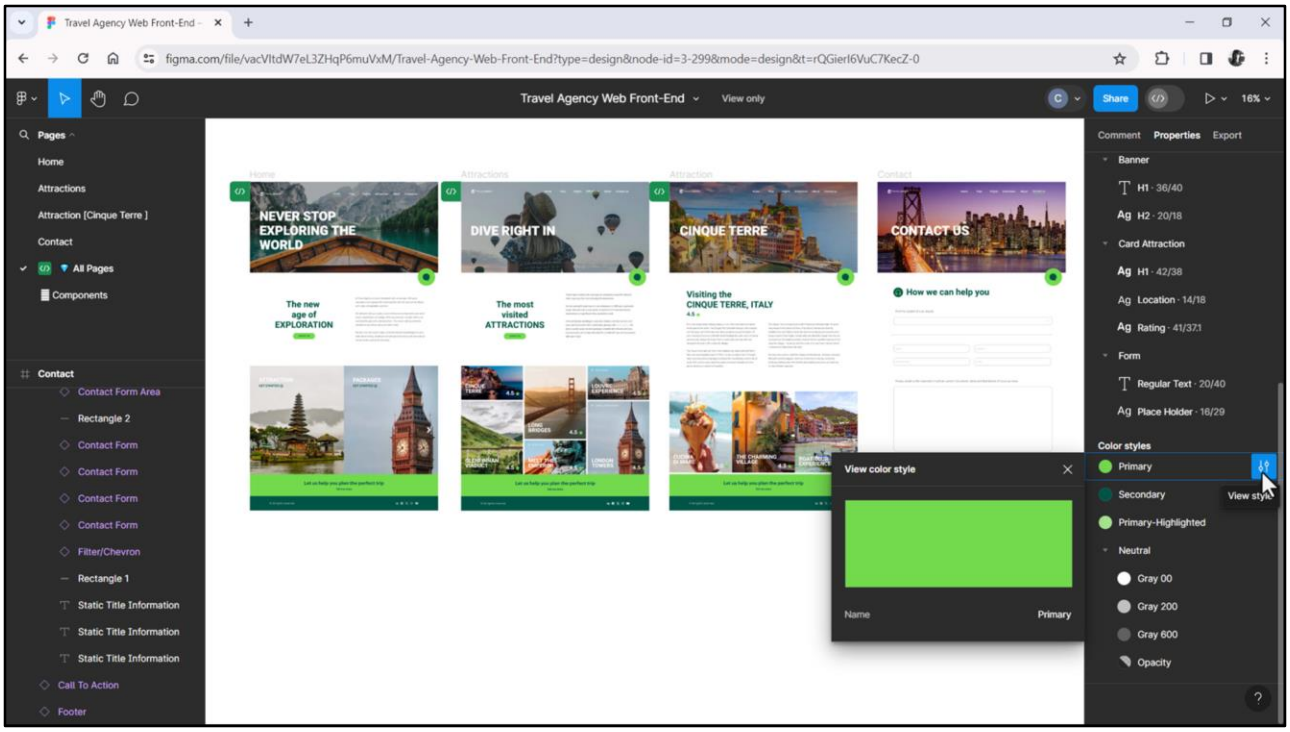
We had already created the primary token for the primary font, and now I'm going to add two more tokens for the other two.



We could come to Chechu's spreadsheet and instead of the specific font, we could place the token, for when we create the classes. And with this, we already have everything ready and initialized, so that it is easier to incorporate the classes later.

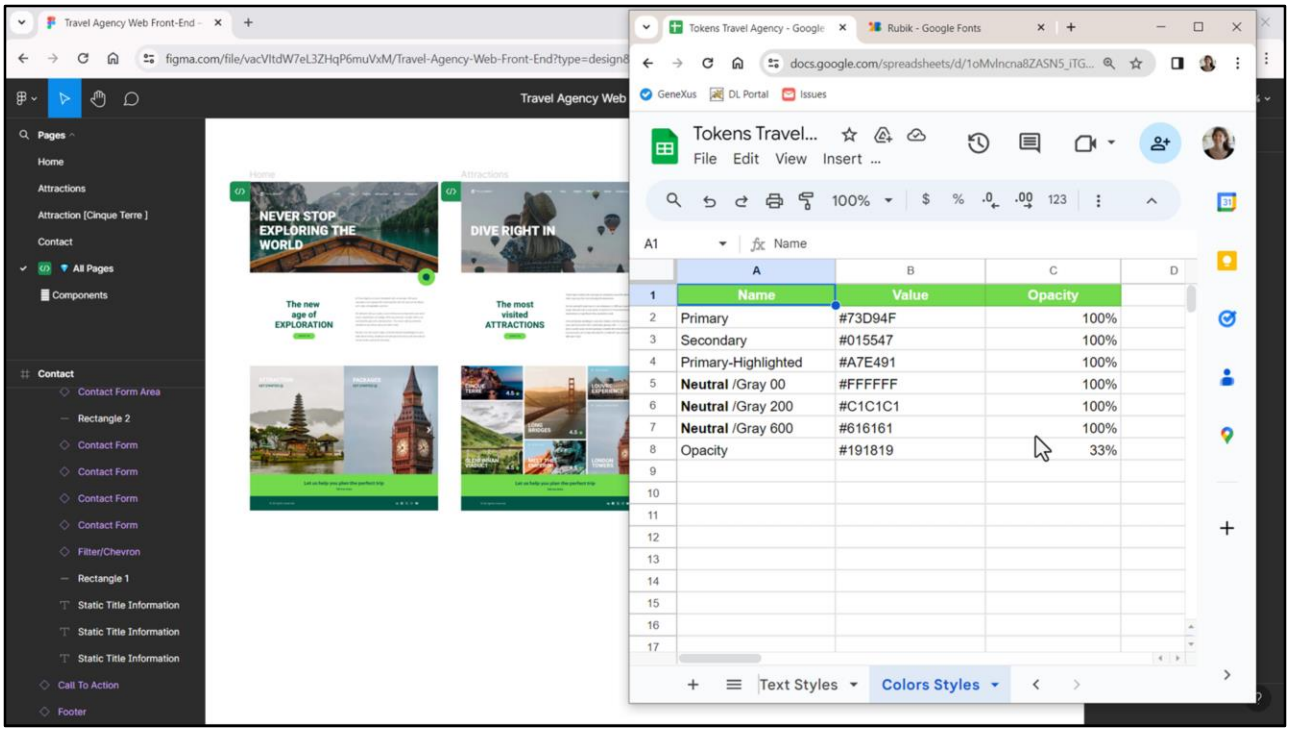
GeneXus by Globant

Colors



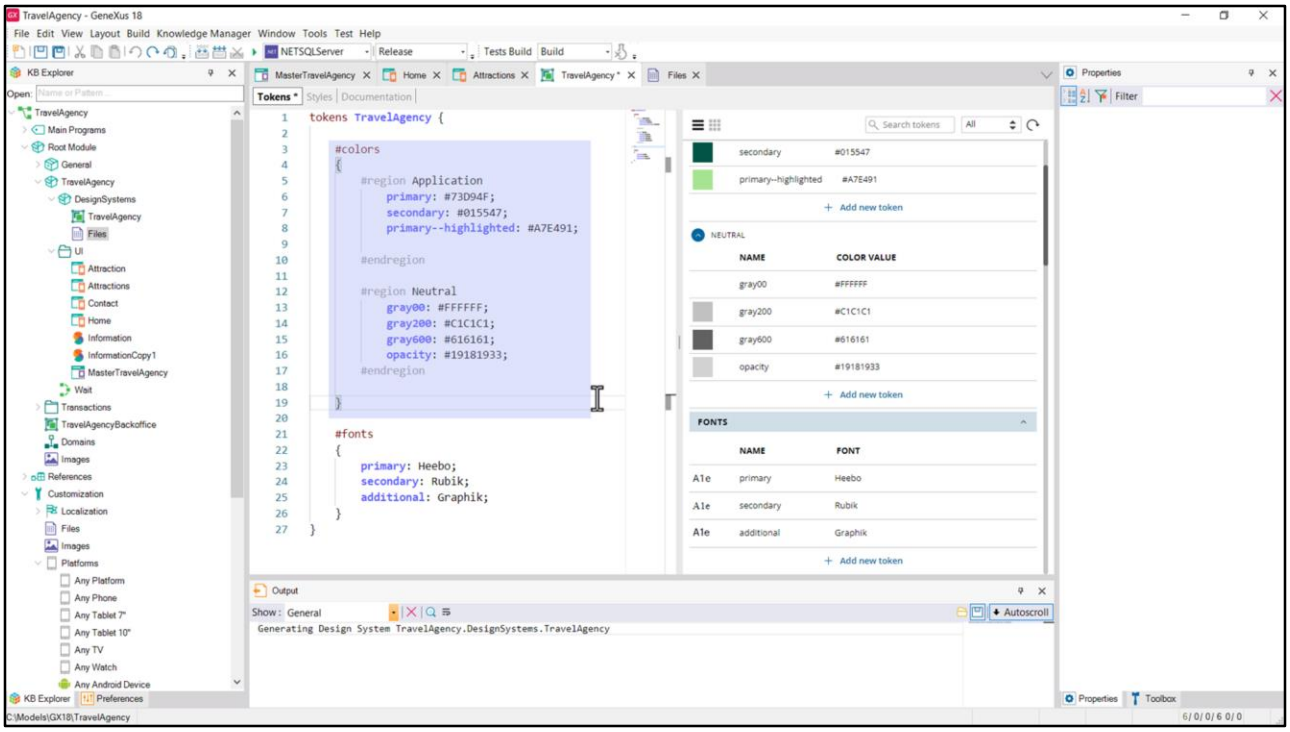
Then we would come to Figma to locate the color styles, so we can define them as tokens in our DSO.

Again, here we only see the names but not the value.



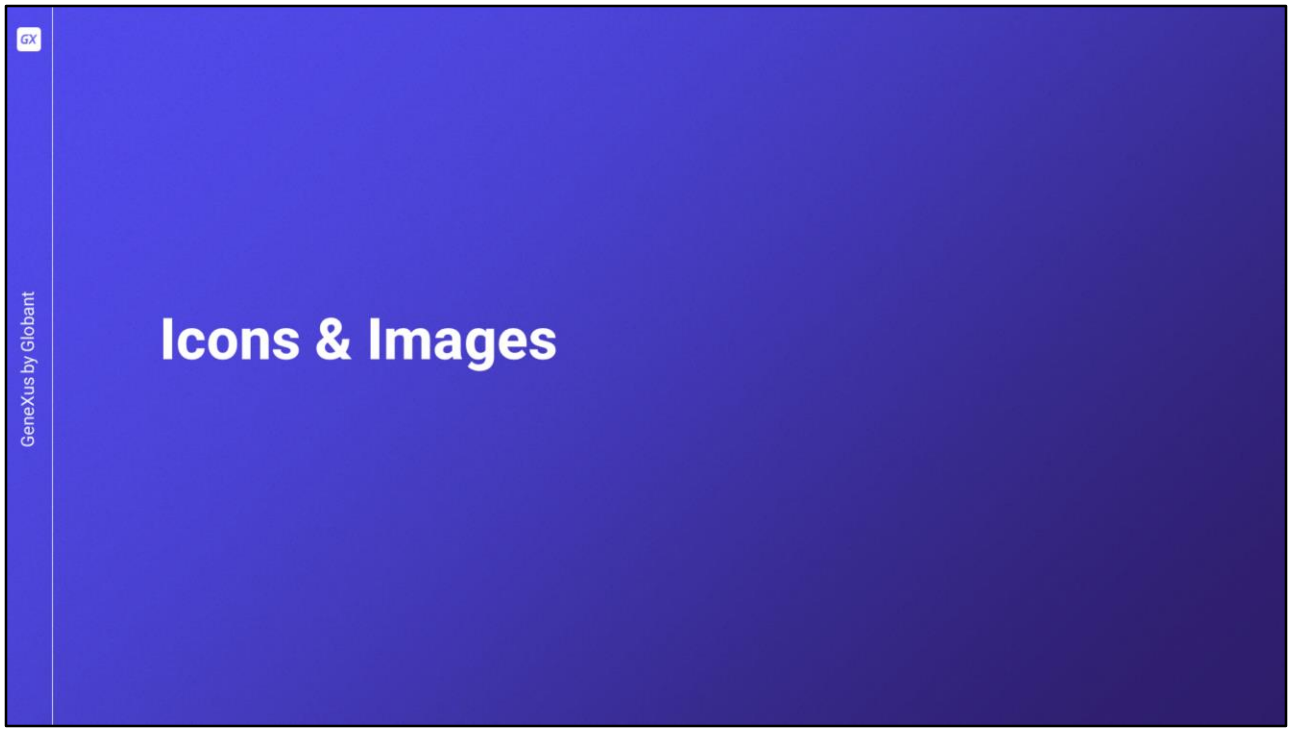
	A	B	C	D
1	Name	Value	Opacity	
2	Primary	#73D94F	100%	
3	Secondary	#015547	100%	
4	Primary-Highlighted	#A7E491	100%	
5	**Neutral** /Gray 00	#FFFFFF	100%	
6	**Neutral** /Gray 200	#C1C1C1	100%	
7	**Neutral** /Gray 600	#616161	100%	
8	Opacity	#191819	33%	
9				
10				
11				
12				
13				
14				
15				
16				
17				

So Chechu also listed them for me in the spreadsheet, in this other tab, so that I don't have to go looking, inspecting each item on the pages until I find them.

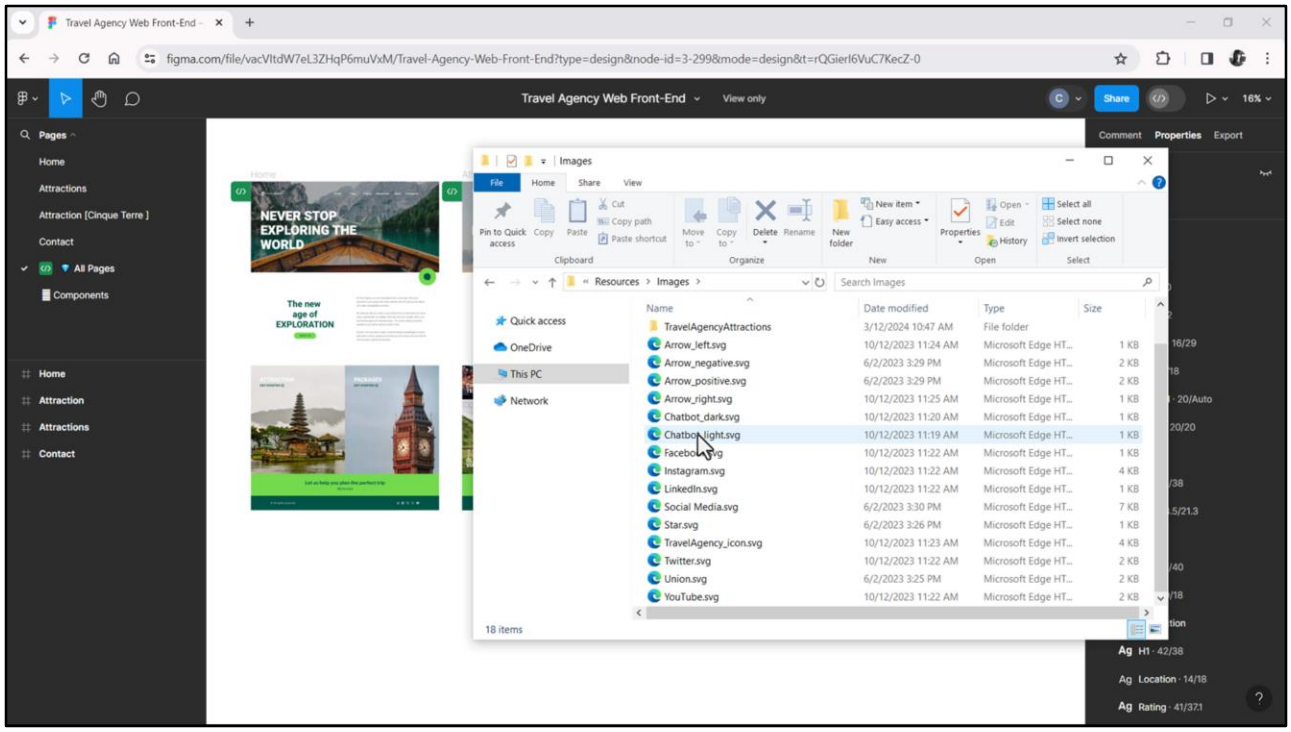


In the previous videos, we had defined the ones we needed and separated them into two regions. Now we will define the missing ones, which are two: gray200... and opacity... which has 33 percent opacity, so we add the number 33.

Well, we can ask ourselves right now if this basic level of abstraction is enough, that is, of these global tokens where it only defines the color palette that the application will use, but doesn't show the colors organized in relation to the function that they will have in the system. In the next video, we will address that topic in particular, which is more related to the concept of the design **system**. For now, we introduced the color palette.



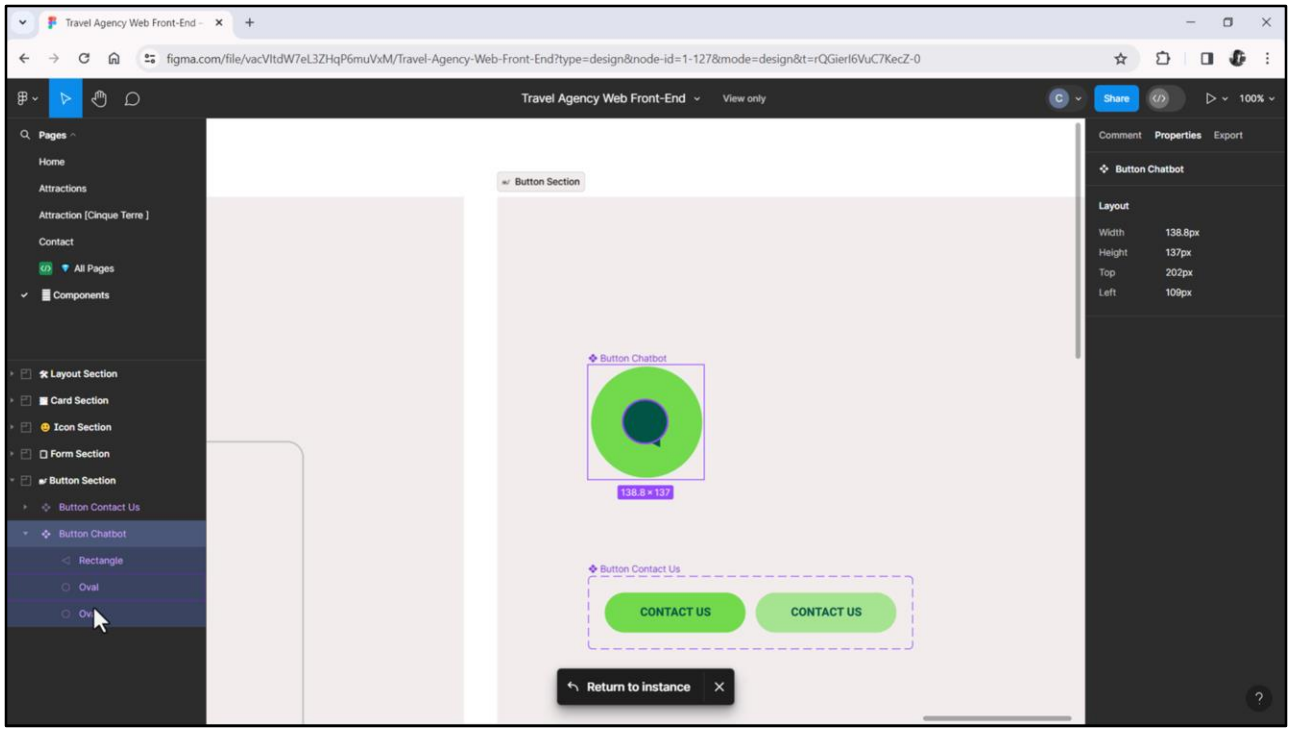
It would also be good to introduce in the KB the icons and images that will be needed.



I have already downloaded all of them. I'm going to show them first in the folder, and then I'll show you how I got each thing from Figma.

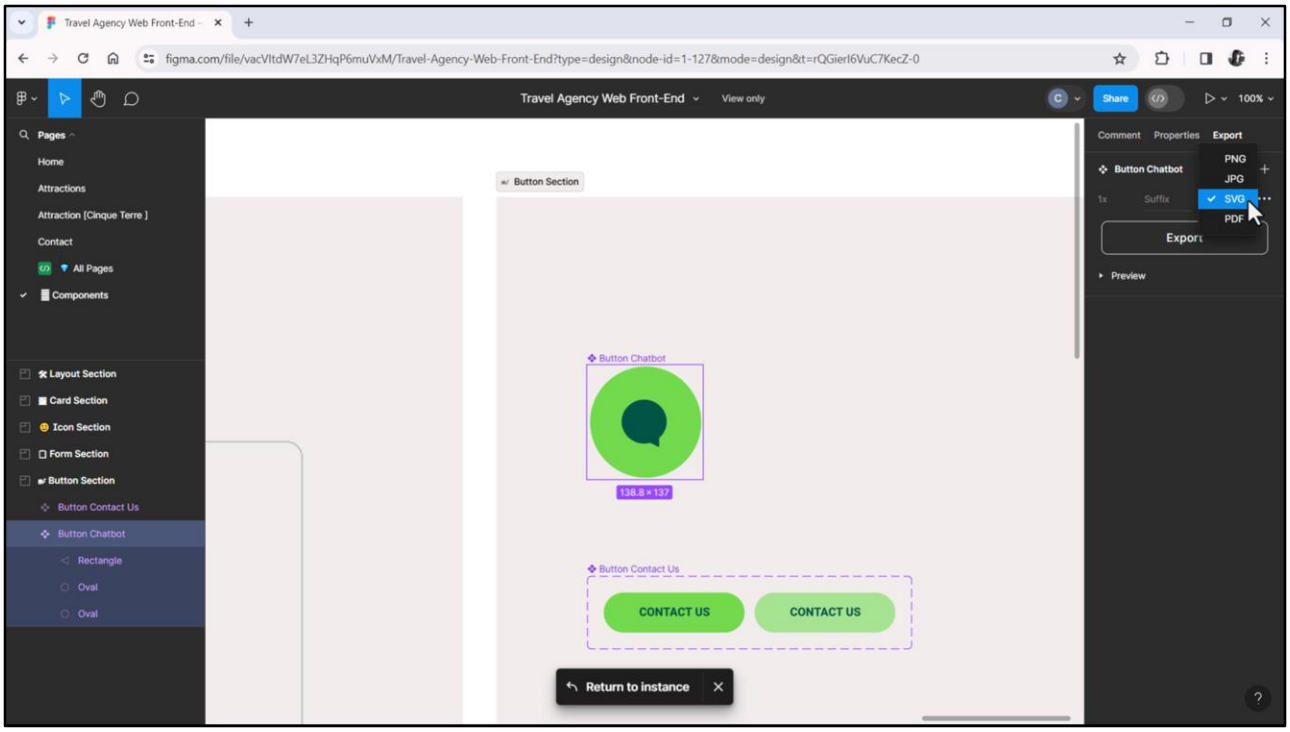
Here we see the icons... For example, that of the chatbot, in light mode, because later on we will see the dark one, or for example, the star... Note that these are SVG files.

One option is to ask our designer to send us all these resources so that we don't have to do any of the things we are going to do now. But the other option is that we download it ourselves from the file in Figma.



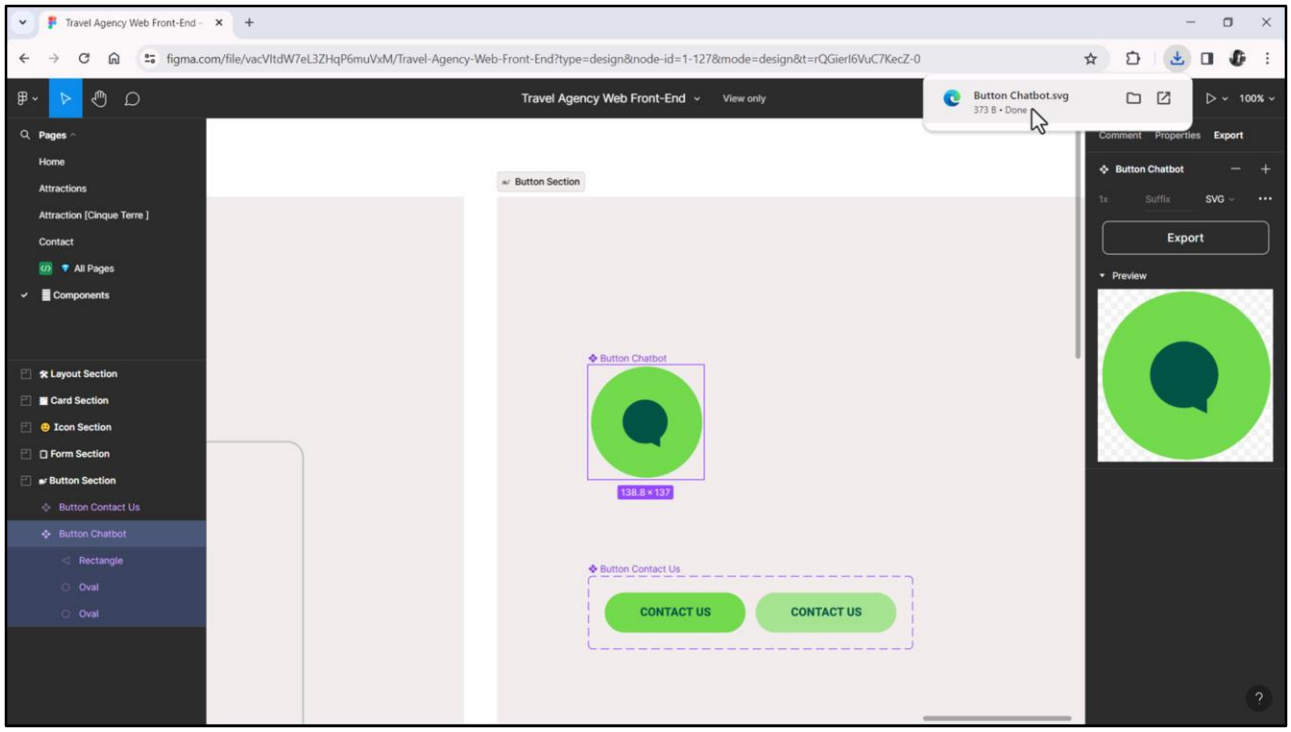
For example, let's see it with the chatbot image. Note that the element is modeled as a component, and when we inspect it, we see it's not an image, but that Chechu put it together with these simple elements.

We will need it as an image because we will implement it in GeneXus as a button with that image.

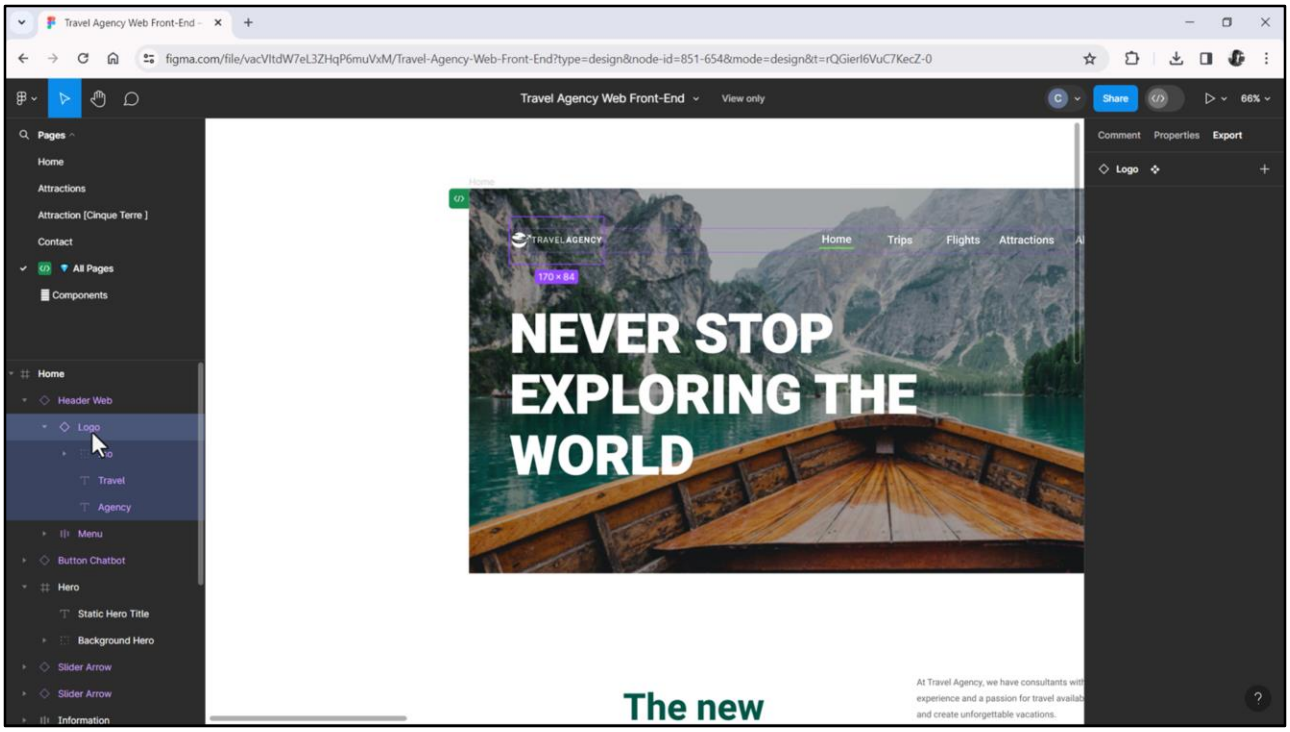


Since it is a simple image, not a complex one, we opted for the SVG format, which has several advantages over formats such as PNG or JPEG, including that SVG files scale without losing quality, since they are vector graphics, and have a smaller size (which helps reduce page loading time). These are two very important advantages.

So, icons, which are simple images, and all the simple images we find in the design should all be SVG files.

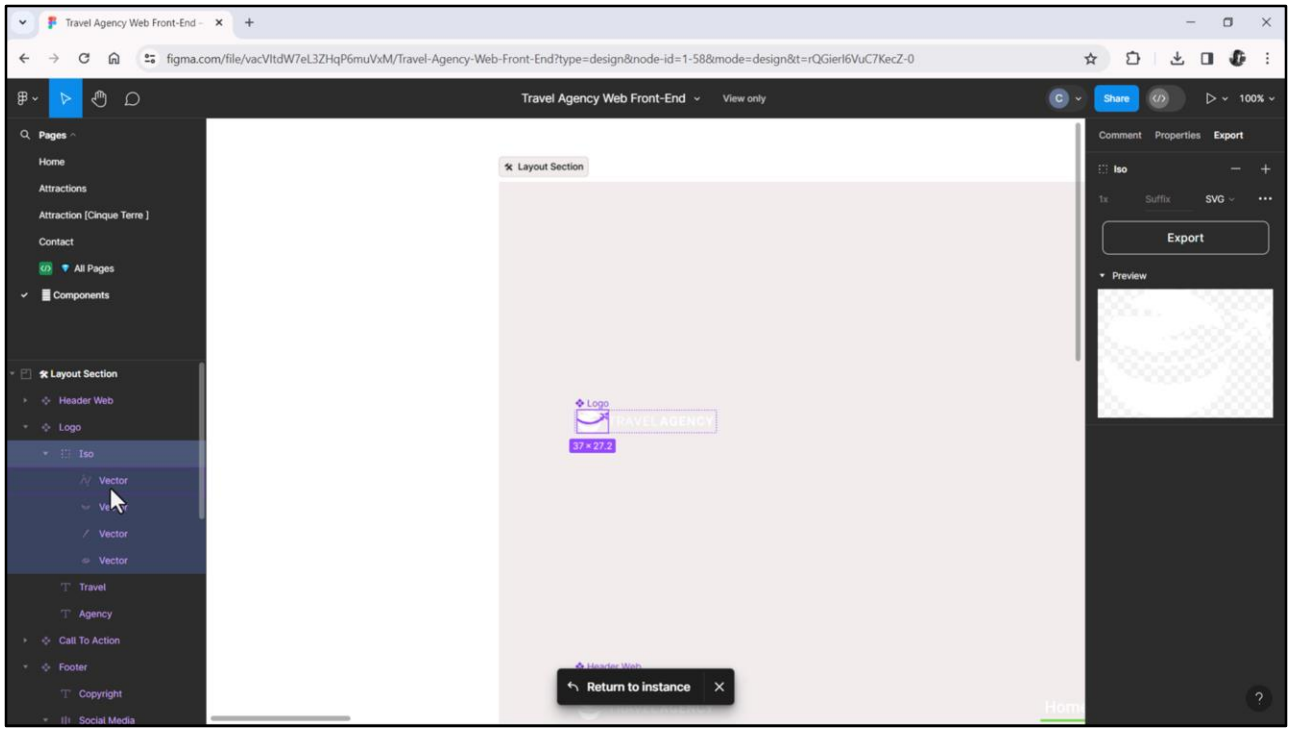


What we do then is click here to download it. And there we see it.

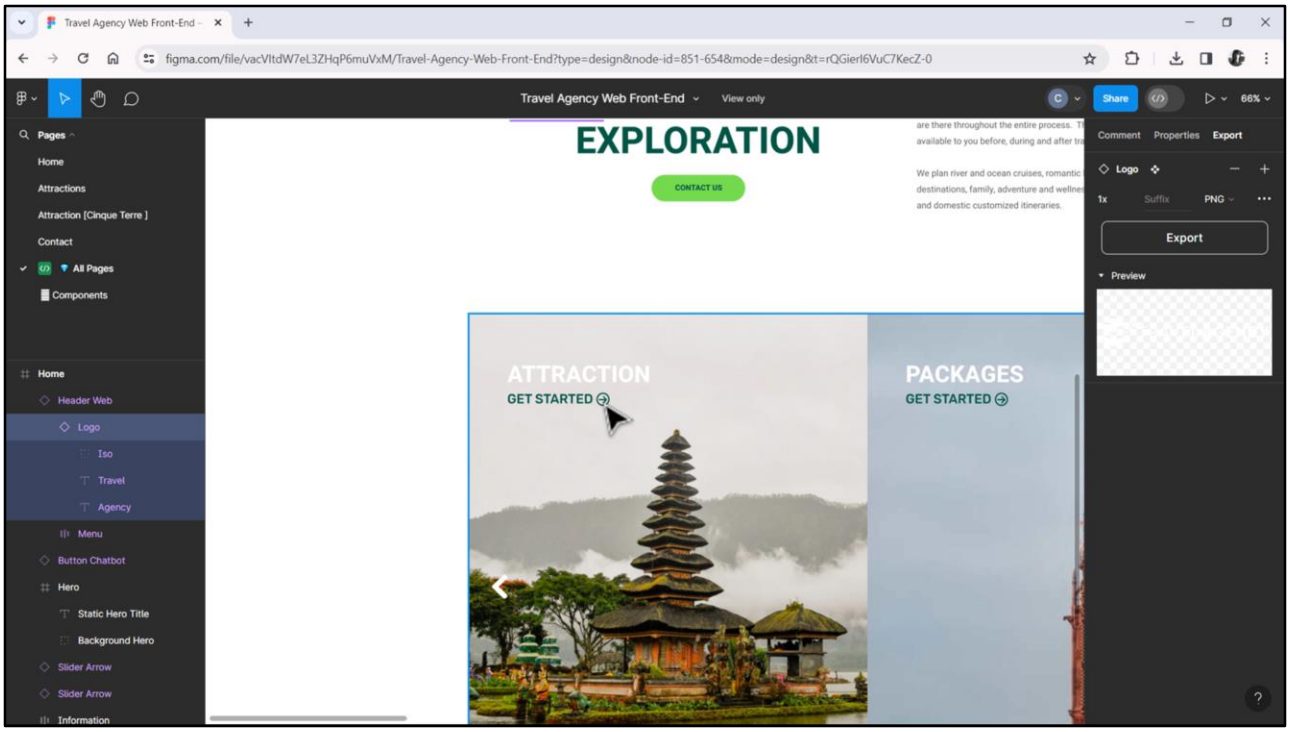


Then we would have to go through all the screens looking for the icons to download them, as we did with that of the chatbot. For example, we have here, in the Header, the logo. The logo is composed of three elements: one is the logo icon, the text “Travel” and the text “Agency” (I'm going to enlarge it to make it clearer). See?

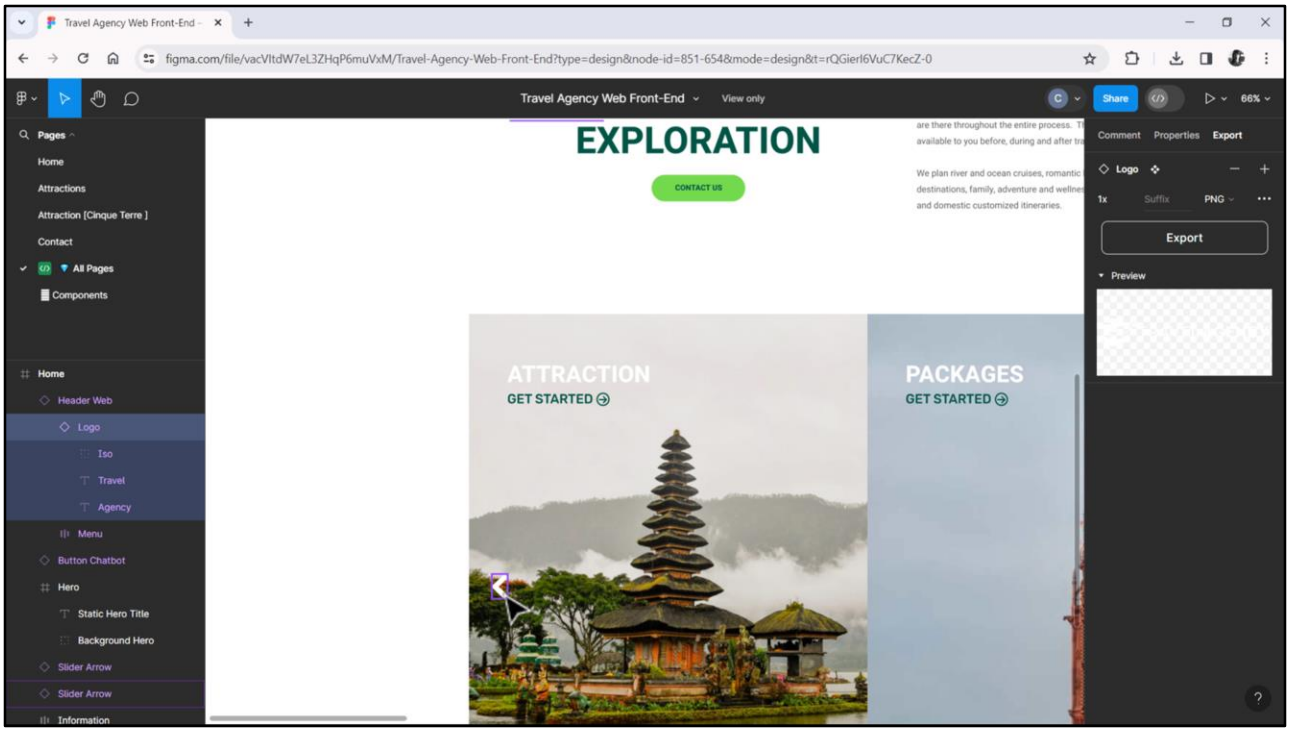
I could download the whole thing: the icon with the two words, exporting it from here (we see that it is a component).



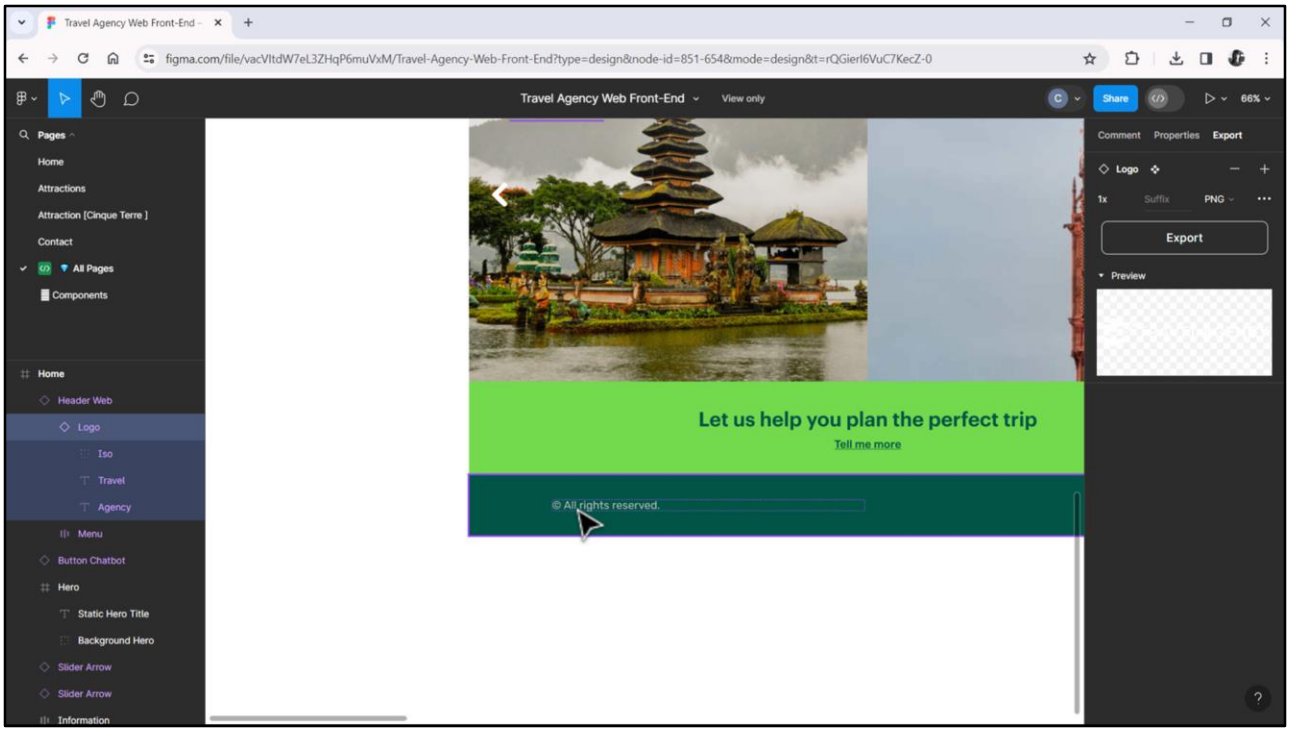
Or I could just export the icon itself, which you can see that Chechu also built with these elements here. It is a similar idea as for the chatbot, and then I can download only that, which is what I did, and as in the case of the chatbot, I click here and it will download that icon as SVG.



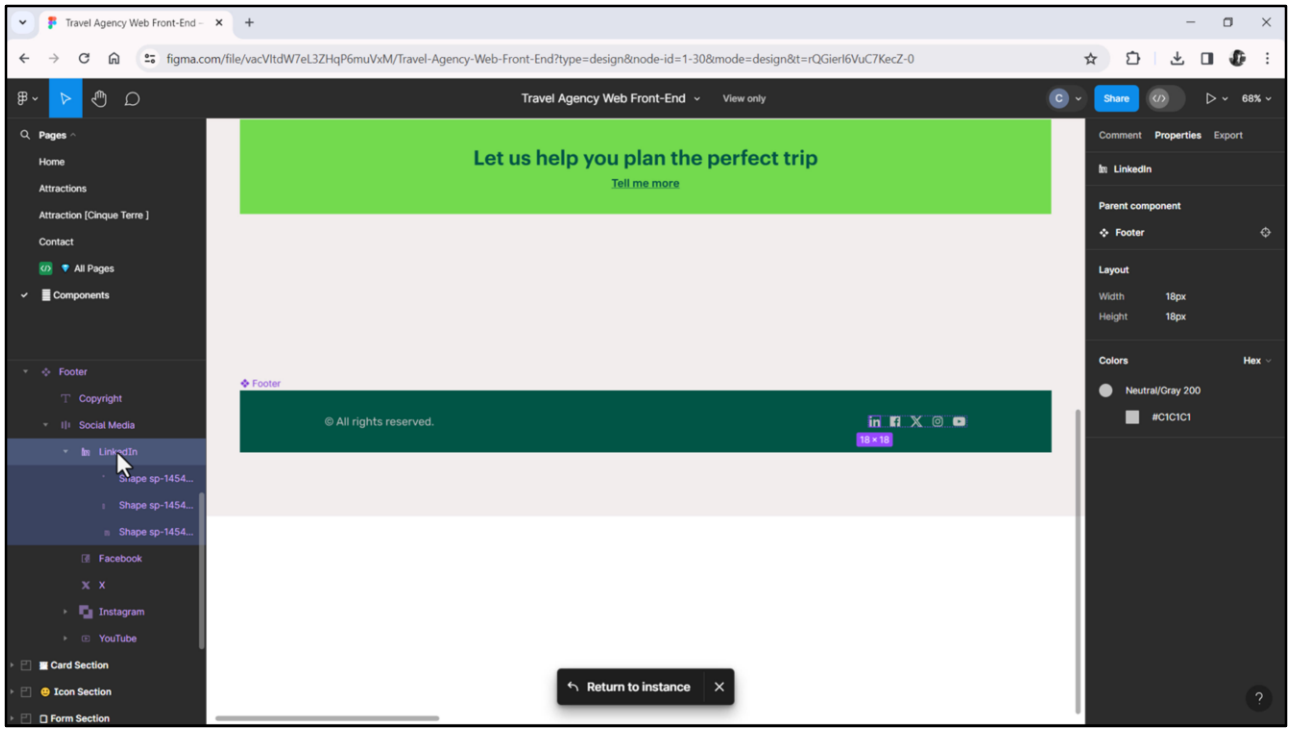
And so on... I'm going to show you a few others; some that are a bit different. Note that we have this little arrow here, for example...



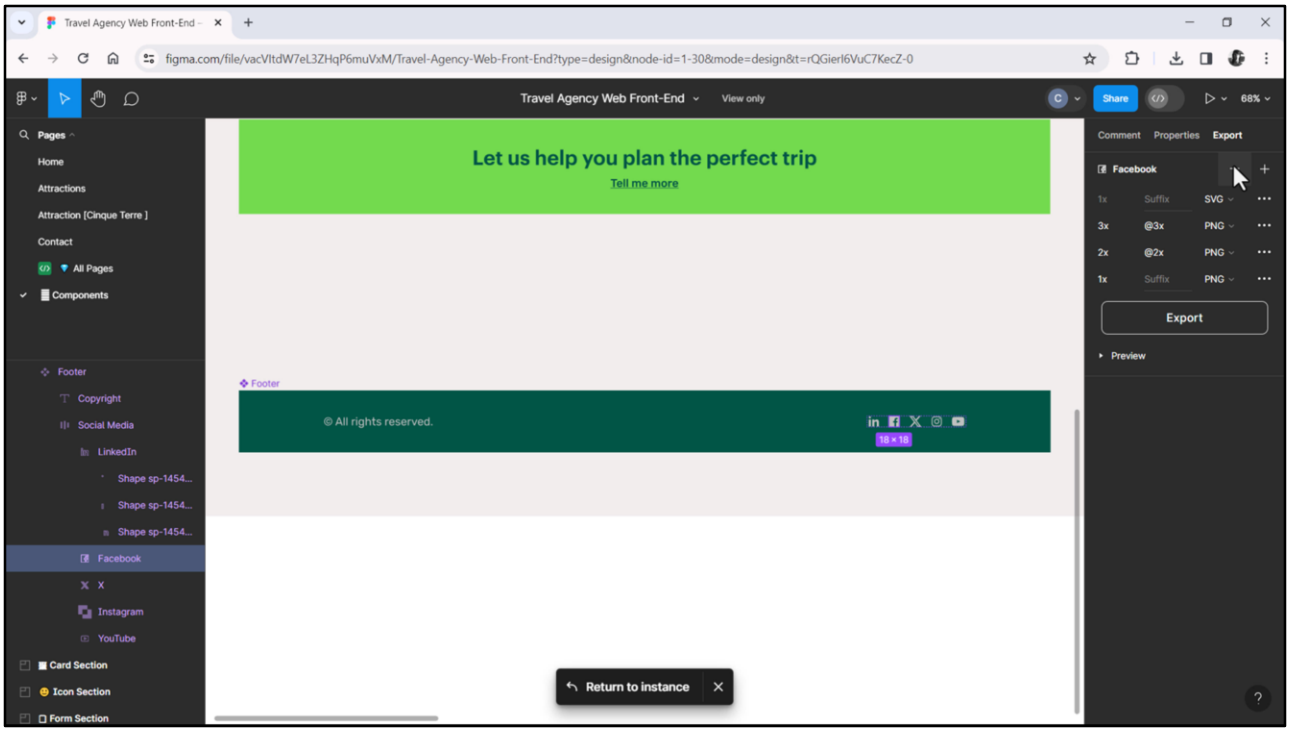
…we have these arrows.


Travel Agency Web Front-End
figma.com/file/vacVItdW7eL3ZHqP6muVxM/Travel-Agency-Web-Front-End?type=design&node-id=851-654&mode=design&t=rQGierl6VuC7KecZ-0
Travel Agency Web Front-End
View only
Share
66%
Pages
Home
Attractions
Attraction [Cinque Terre]
Contact
All Pages
Components
Home
Header Web
Logo
Iso
Travel
Agency
Menu
Button Chatbot
Hero
Static Hero Title
Background Hero
Slider Arrow
Slider Arrow
Information
Comment
Properties
Export
Logo
1x
Suffix
PNG
Export
Preview
Let us help you plan the perfect trip
Tell me more
© All rights reserved.

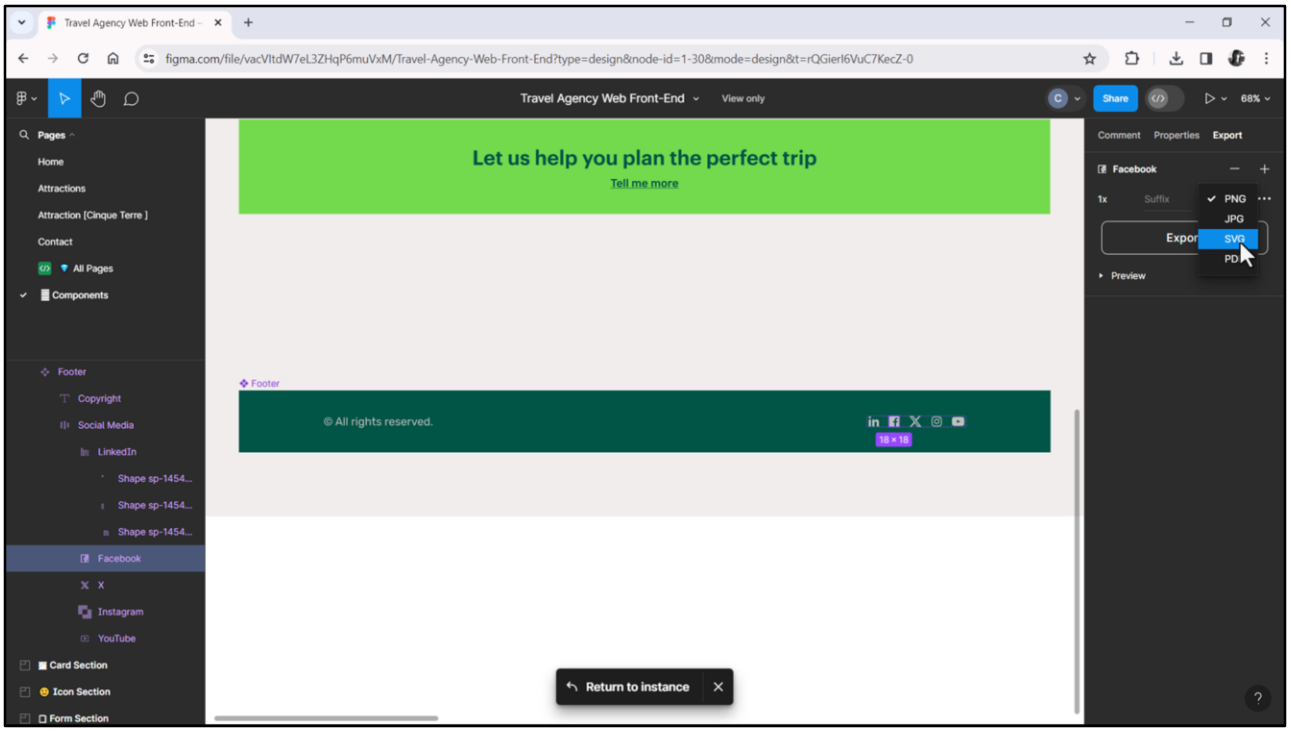
This is text...



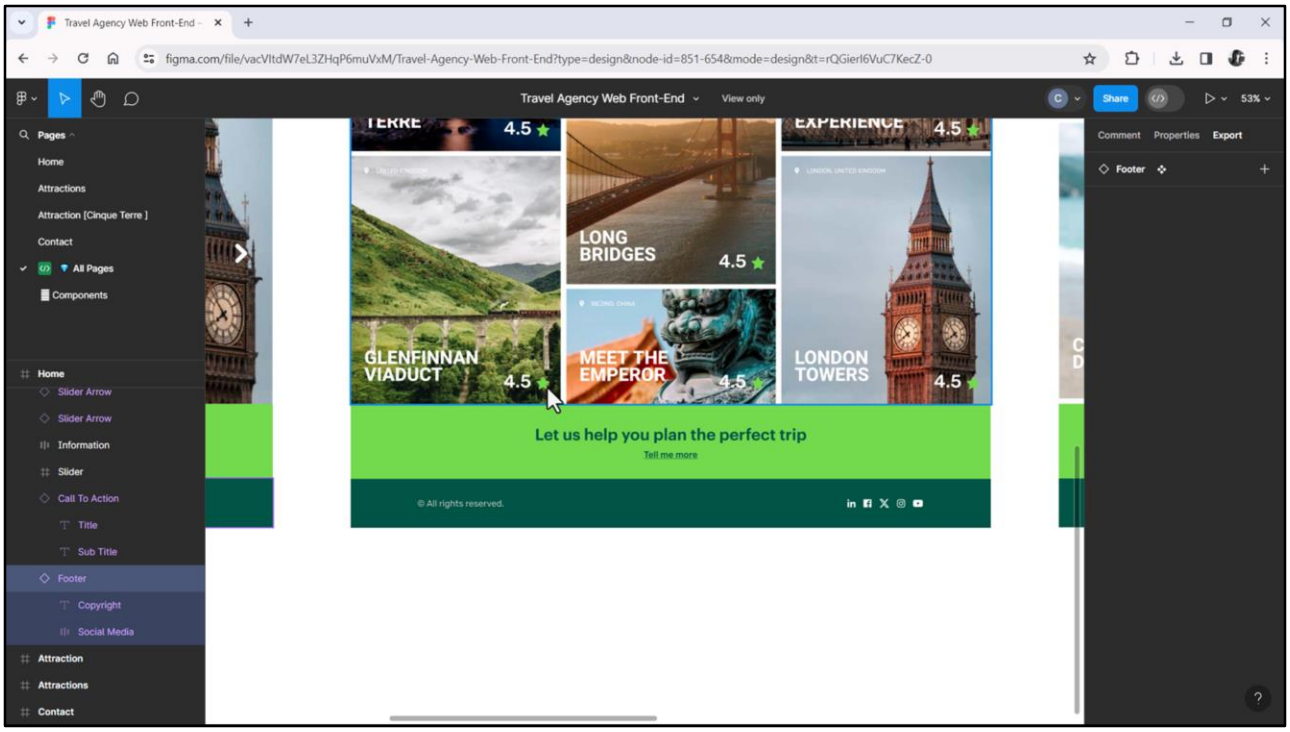
...but here we have, for example, the social media icons. It is a component and it is in the footer. We go over here, and we see it here, right?
Then we have... this is the one for LinkedIn...



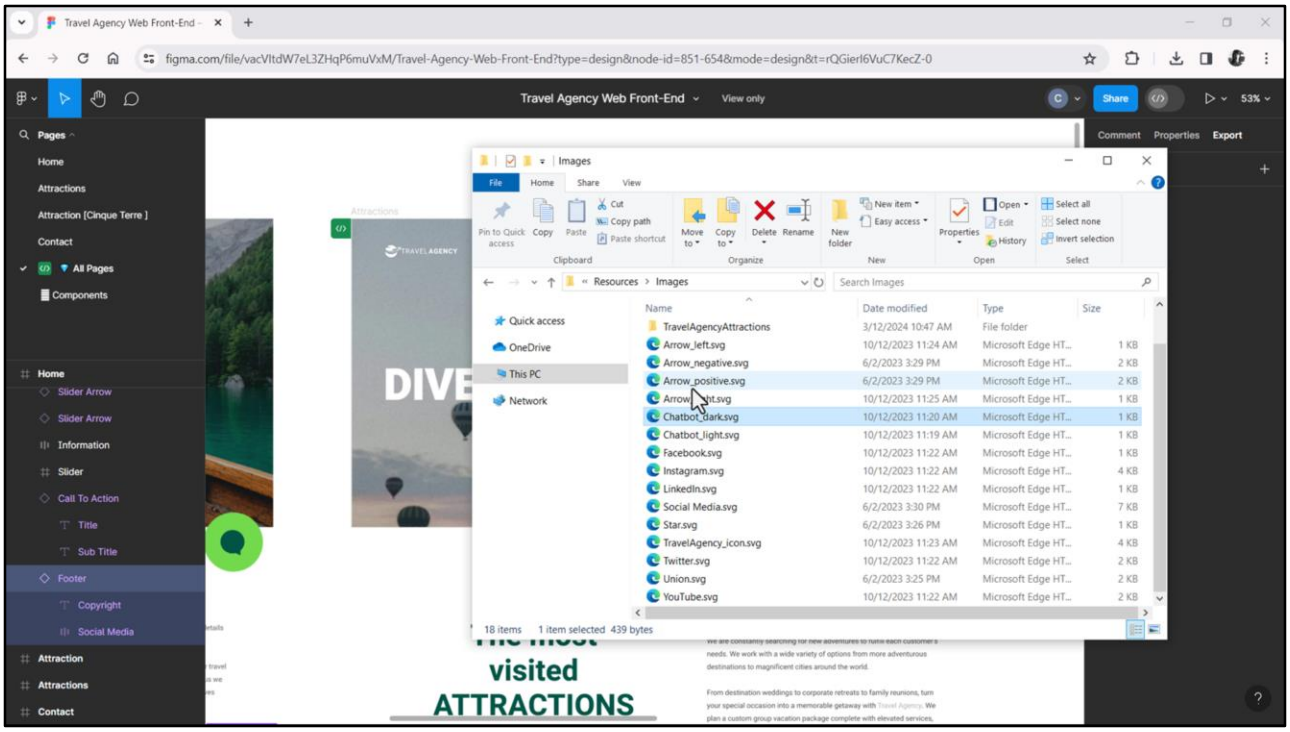
...this is Facebook's... and well, the same thing. Here you can see that it has entries for the different densities.



I will remove them to have only one left, which will be the SVG format, and then I will export the icon in that format.

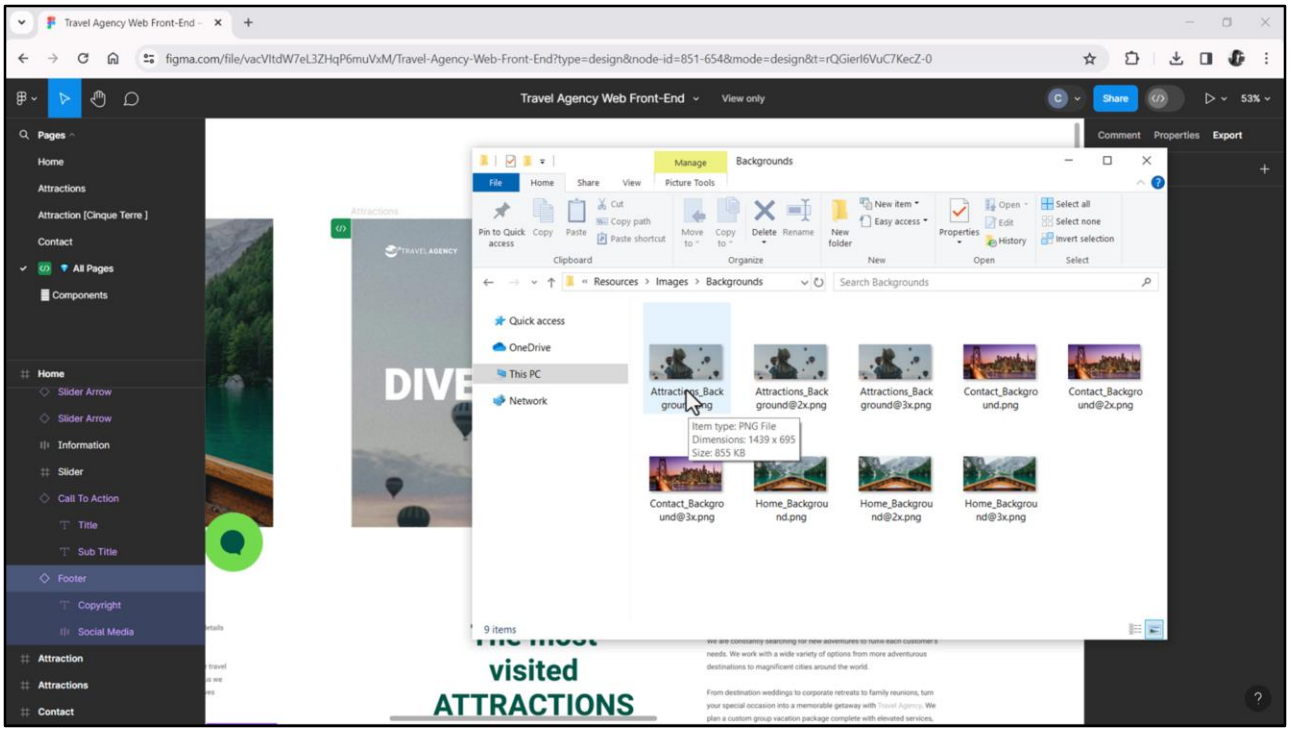


And so on... Let's see if there is anything else worth mentioning... we have already seen the little star... well, but that would be it, wouldn't it? We would have to carefully inspect the pages to download all those icons to have them already available.



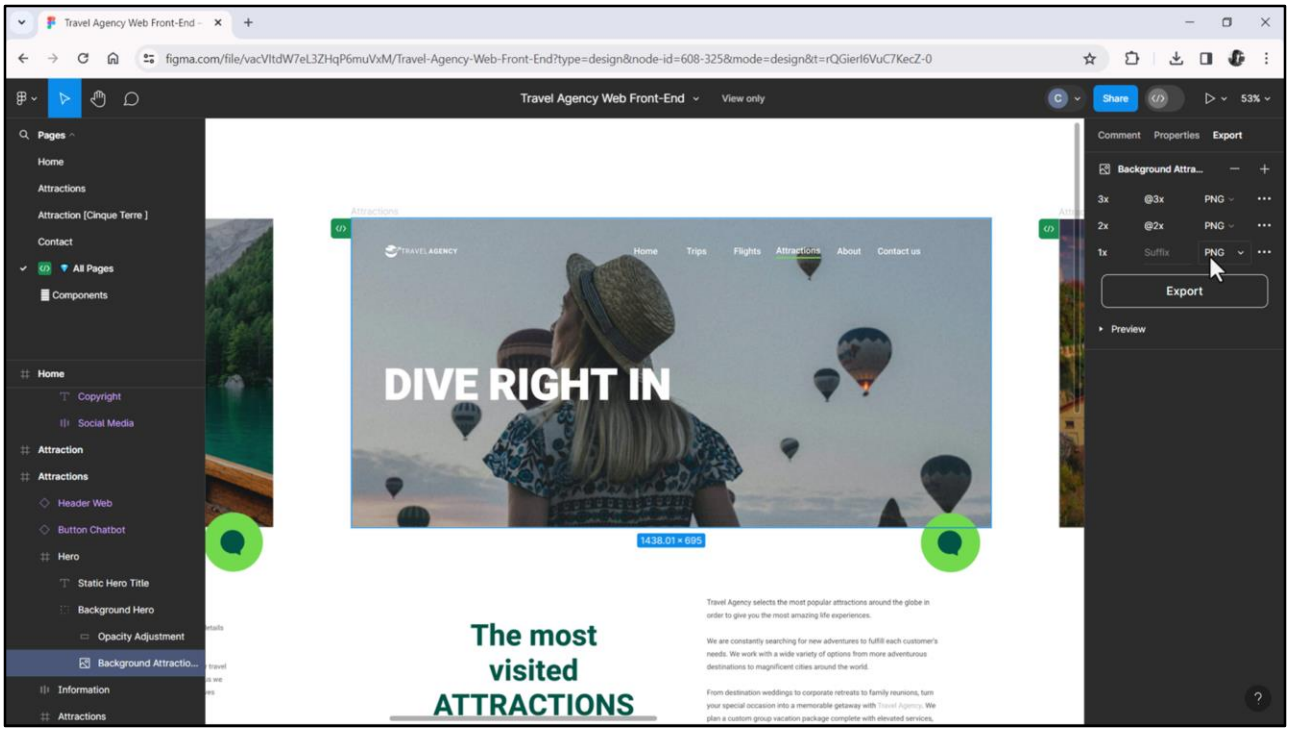
And that's what I have here.

Okay, but on the other hand, I was saying, some images will not be SVG because they are not simple images but complex images.



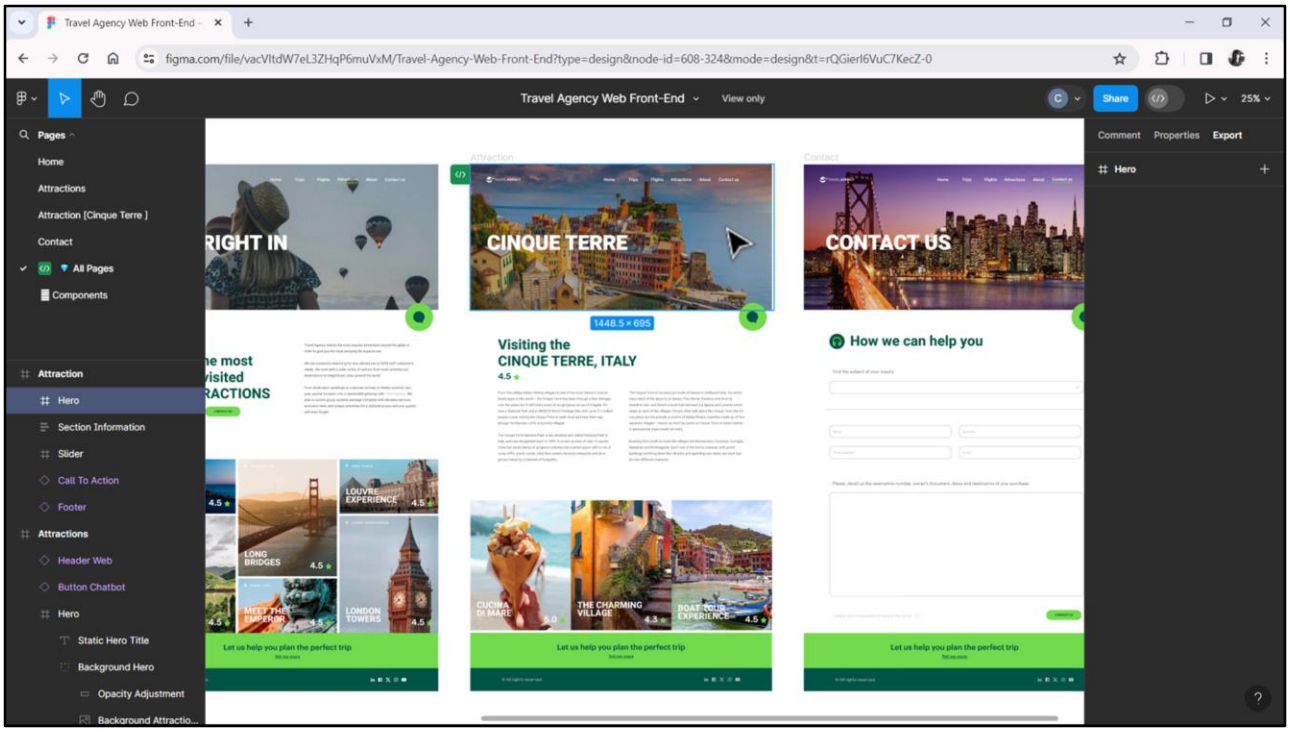
For example, here we see these images, which are Hero images; that is, the ones that will be part of the Header of each page.

As you can see, I downloaded several versions of each image. Why? Because it is convenient to have versions by density so that, if the device has a low resolution, I don't waste time downloading heavier images. And if it is high resolution, it will not look pixelated. Then depending on the resolution of the device I can choose which of the images to use according to its density. Therefore, in this case we are not going to download the images in SVG format, but in some other format.

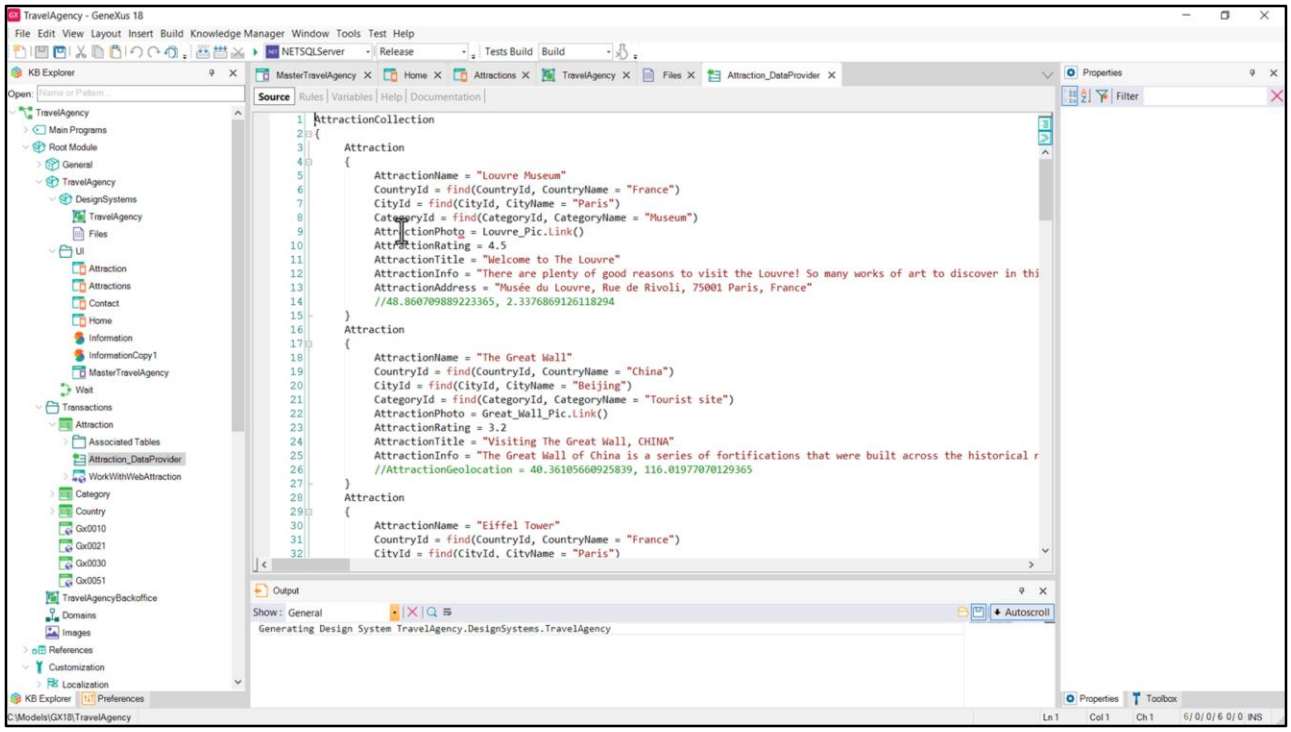


For example, let's see here that we have the Hero element, of which we want only the image. This one here. And here we can download it with density 1x, 2x, 3x, in PNG format. Then, if we select Preview we make sure that it is the image we want to download. We can remove and add options to the image. For example, here we have another one. I'll remove it because we don't need it. All right, we have the three options and what we do is export; note that it exports the 3 images in a zip file, with the 3 densities.

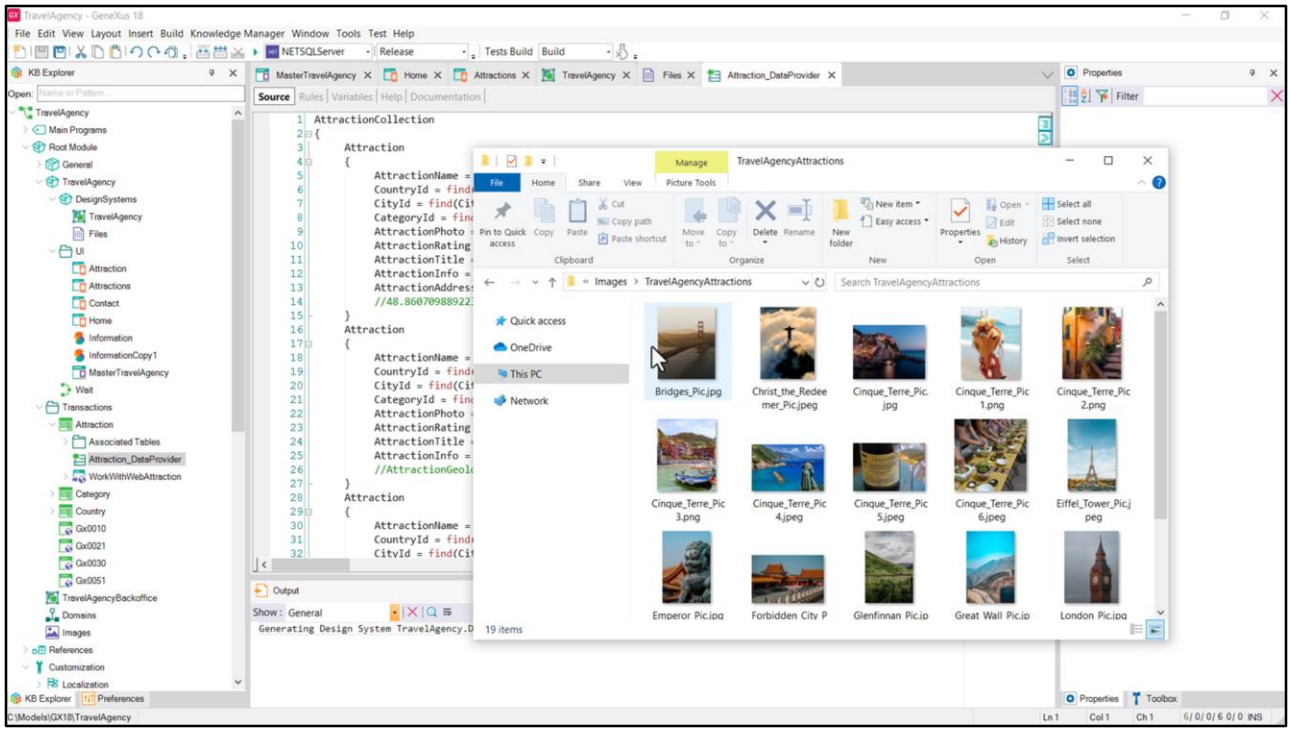
We would do the same to download this image. And also this other image.



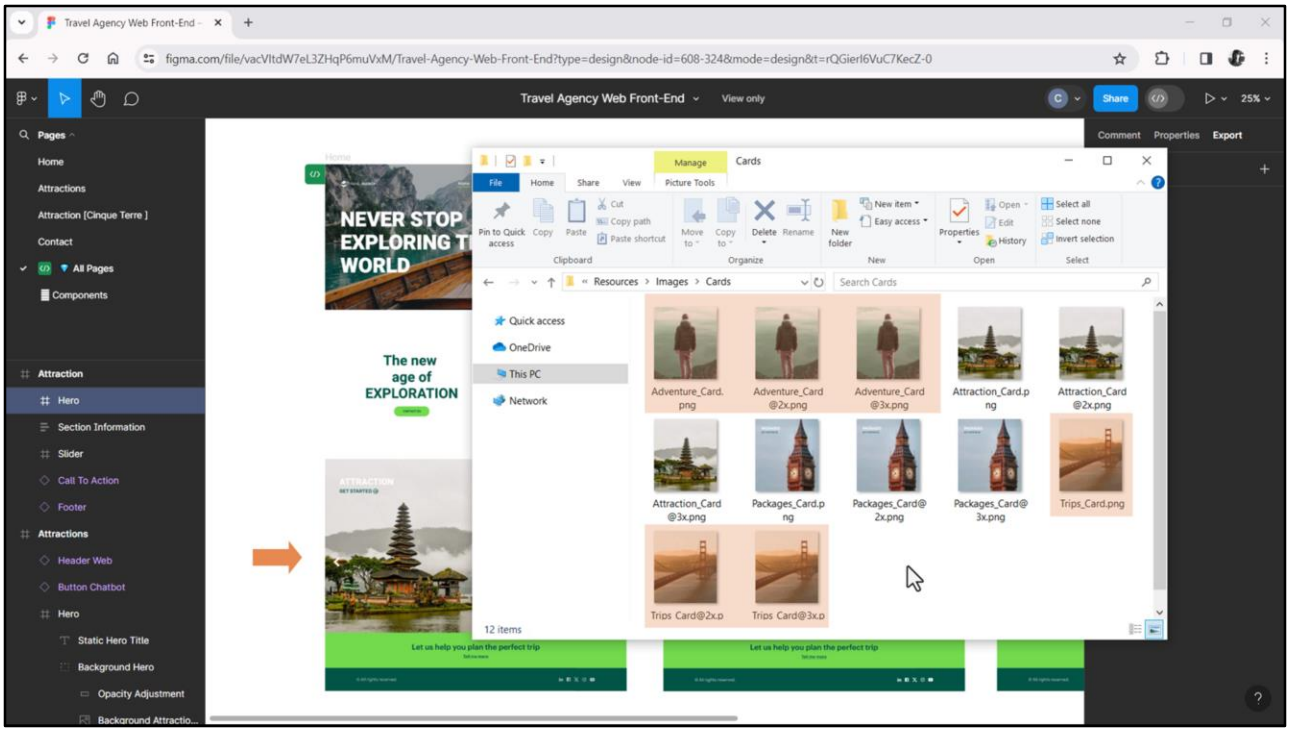
Why not the Attraction one? Because this image here, Hero, is going to be taken from the database. It's going to be the image, the picture, of the tourist attraction selected from this other panel.



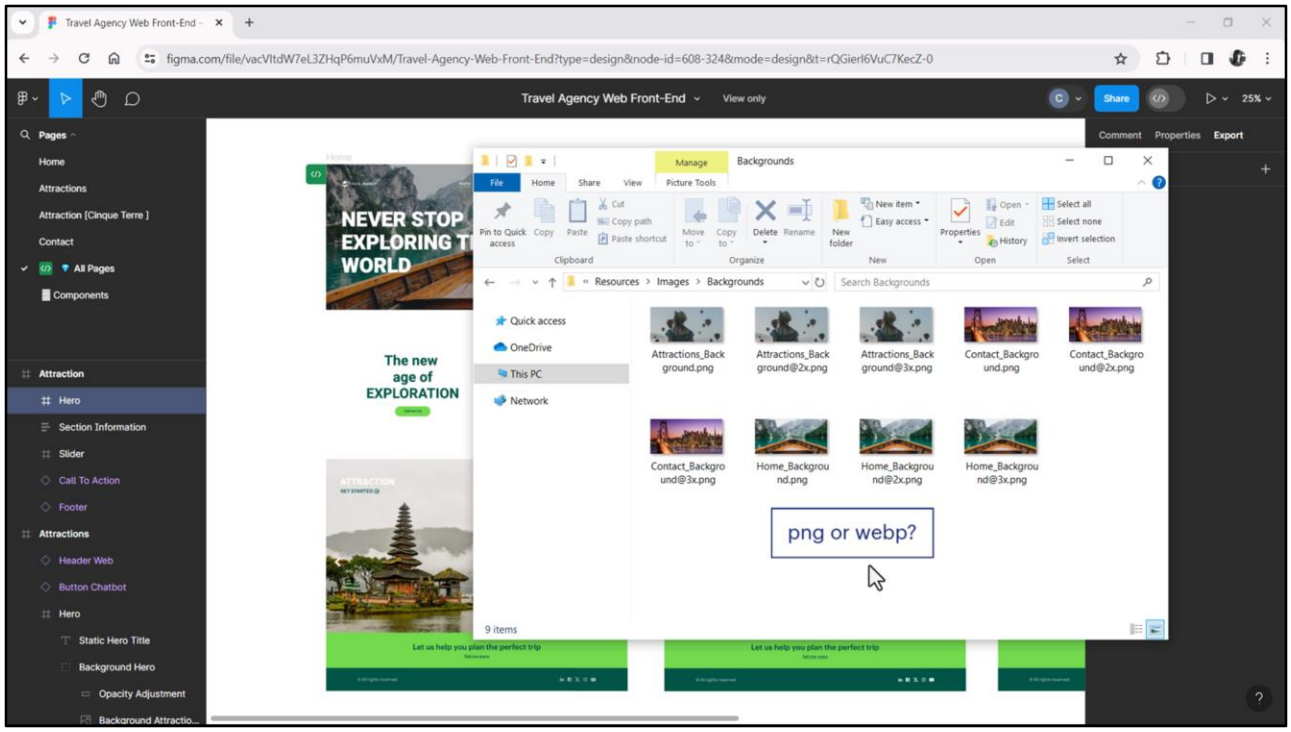
Remember that these images are loaded to the database from the Data Provider that we have in the Backoffice, precisely to load the data of the Attraction transaction.
You can see that the picture is among the data.



Anyway I have those images here too, the ones uploaded by this Data Provider.



In addition, I have these images, Cards, which correspond to which images? Those that are not taken from the database. These are from the database… the ones here, which are 4, because we have these two, and note that this is a carousel, there are 2 more. But I downloaded them all following the same criteria, right? I have 3 versions for each one, for each one of the resolutions. Of course, here we would have to talk to Chechu because I downloaded the images that are not being displayed now from a previous file that Chechu had sent me, in which those other 2 images were displayed, but we are not seeing them here. So, either we ask the designer what is happening with those images so that she can send them to us if they are not modeled in the file, or we ask her to make those images visible to us.

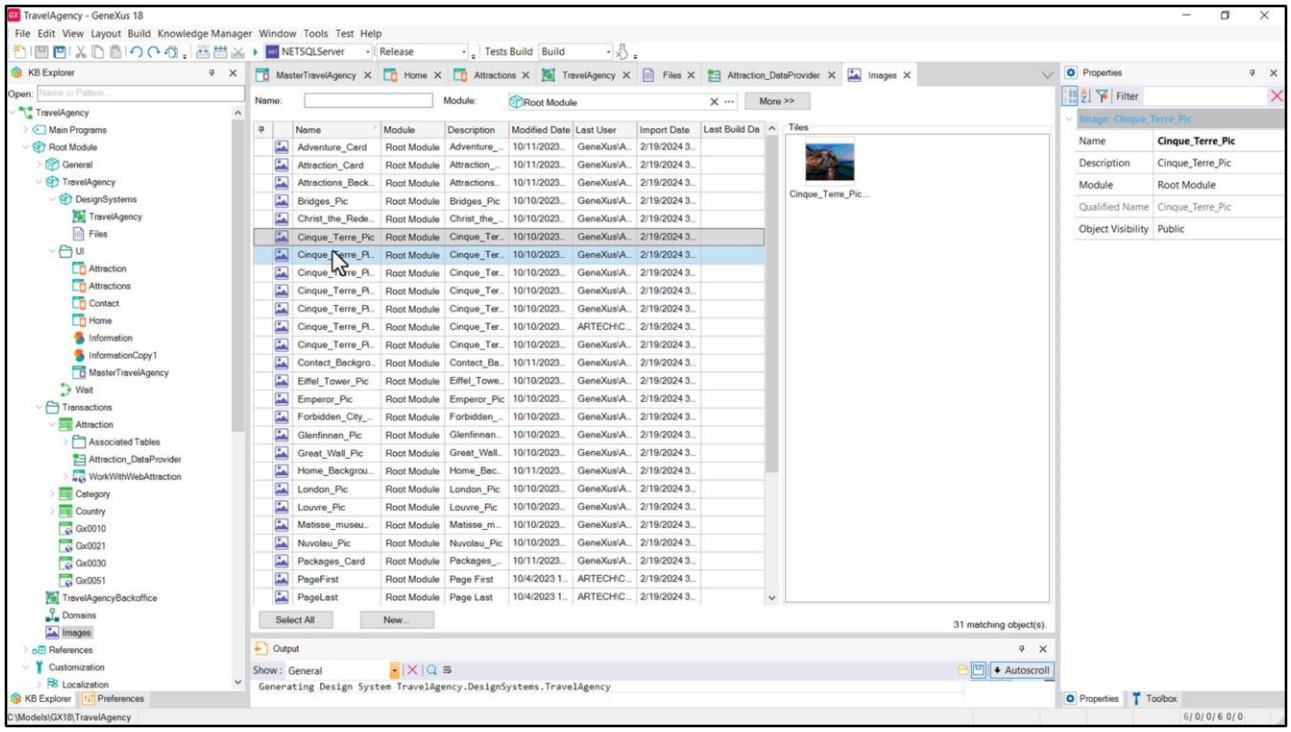


OK, in the Images folder I already have images here and here, and here are the icons of the application, that is to say, I already have all the assets that I will need inside the KB. The icons as SVG images and the more complex images as PNG.

For now as PNG. Why?

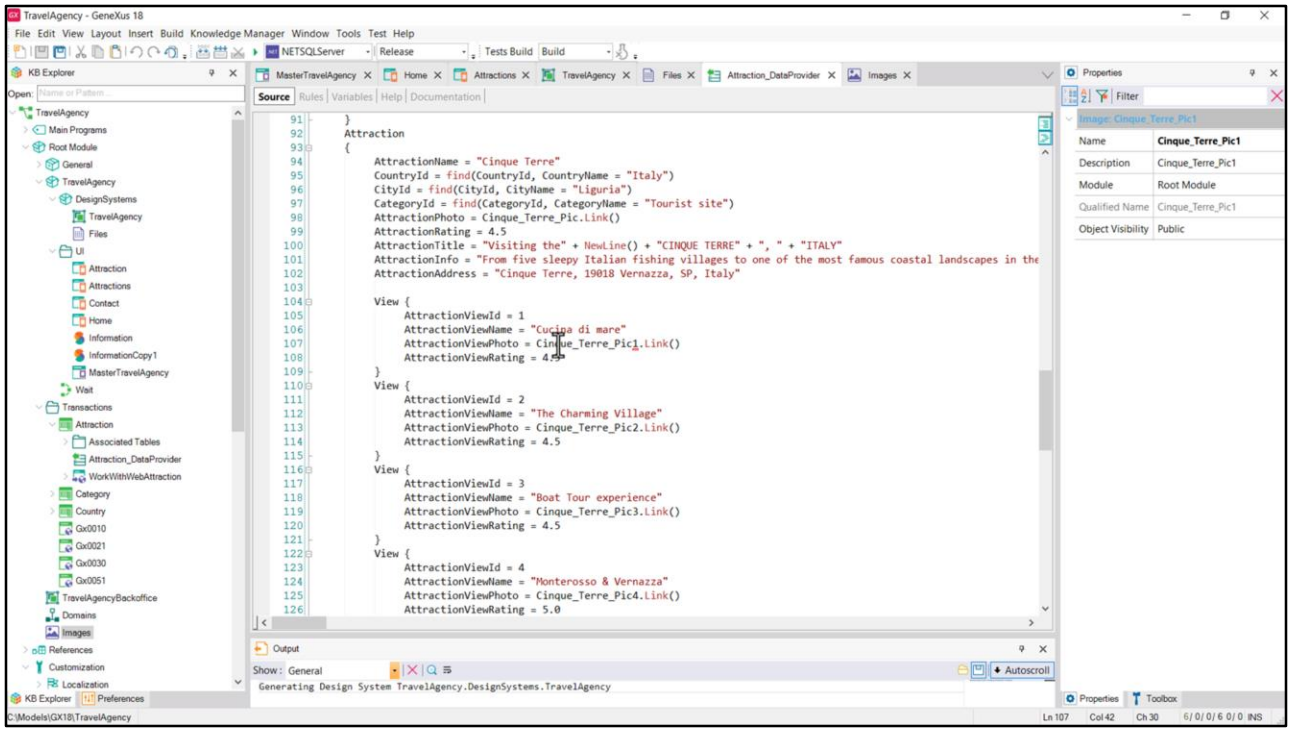
Actually, WebP is much more convenient than the PNG format, since the same quality is achieved as with PNG but it is a more compressed format, so one could think that it's a huge advantage. Except that GeneXus doesn't support uploading images in this format to the KB at the moment. The only way to use them is if they are in the database or are consumed from services.
On the other hand, the Web version of GeneXus will support WebP, but the Win version, which is the one we are using, does not. If we were using the Web version of GeneXus, we would use some online converter from PNG to WebP format, because we saw that Figma didn't offer that format for downloading. Something I didn't say and it is important: it is supported by native applications.

Anyway, let's stay for now in the scenario we are in. We have no choice but to use the PNG format.
So what's the next step? To insert all these images and icons that we gathered from the design as assets in our KB.

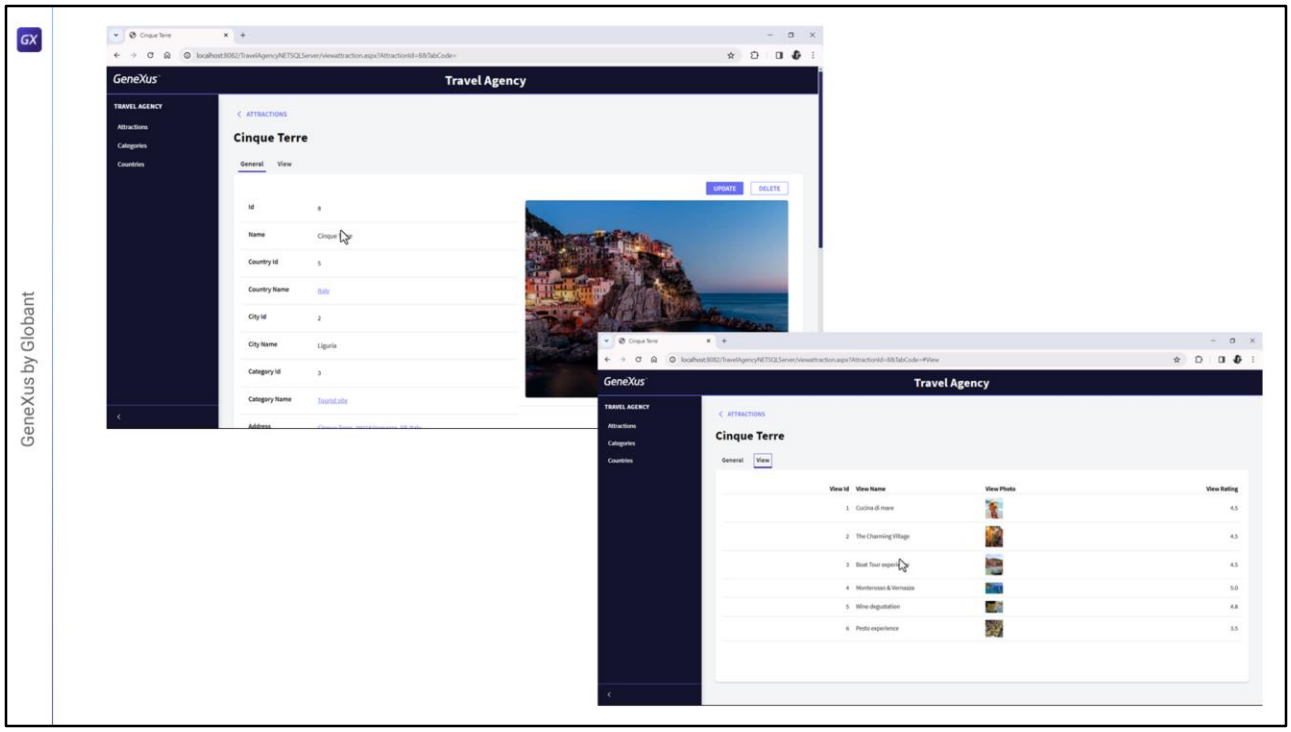


I have them inserted from the very beginning, even before I created the TravelAgency module, so they are associated with the Root Module.

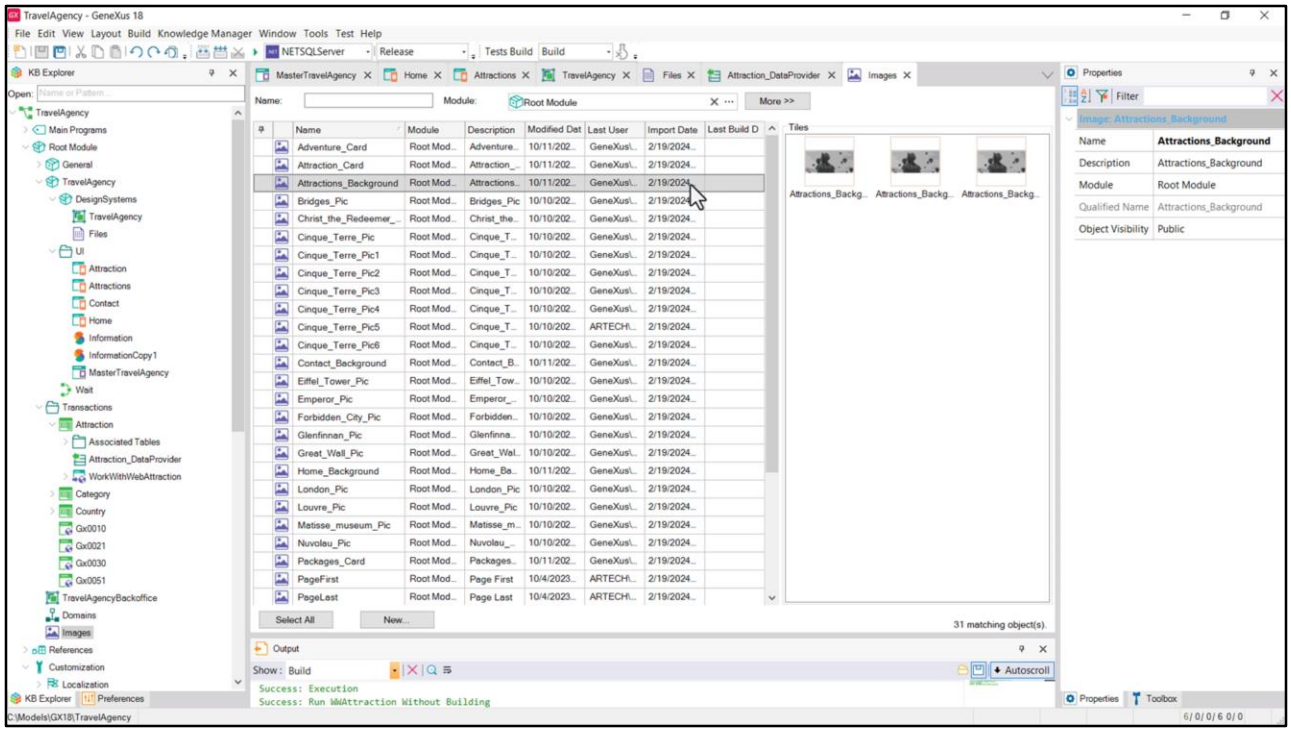
They will all correspond to GeneXus objects of Image type. So if I click here, look at everything that appears. In fact, these are the images I inserted in the KB when I implemented the Backoffice... you didn't see it because it was already done. Let's see this one, for example... and these other ones, which are the options of that tourist attraction...



I used them in the Data Provider to populate the Attraction transaction with data initially.

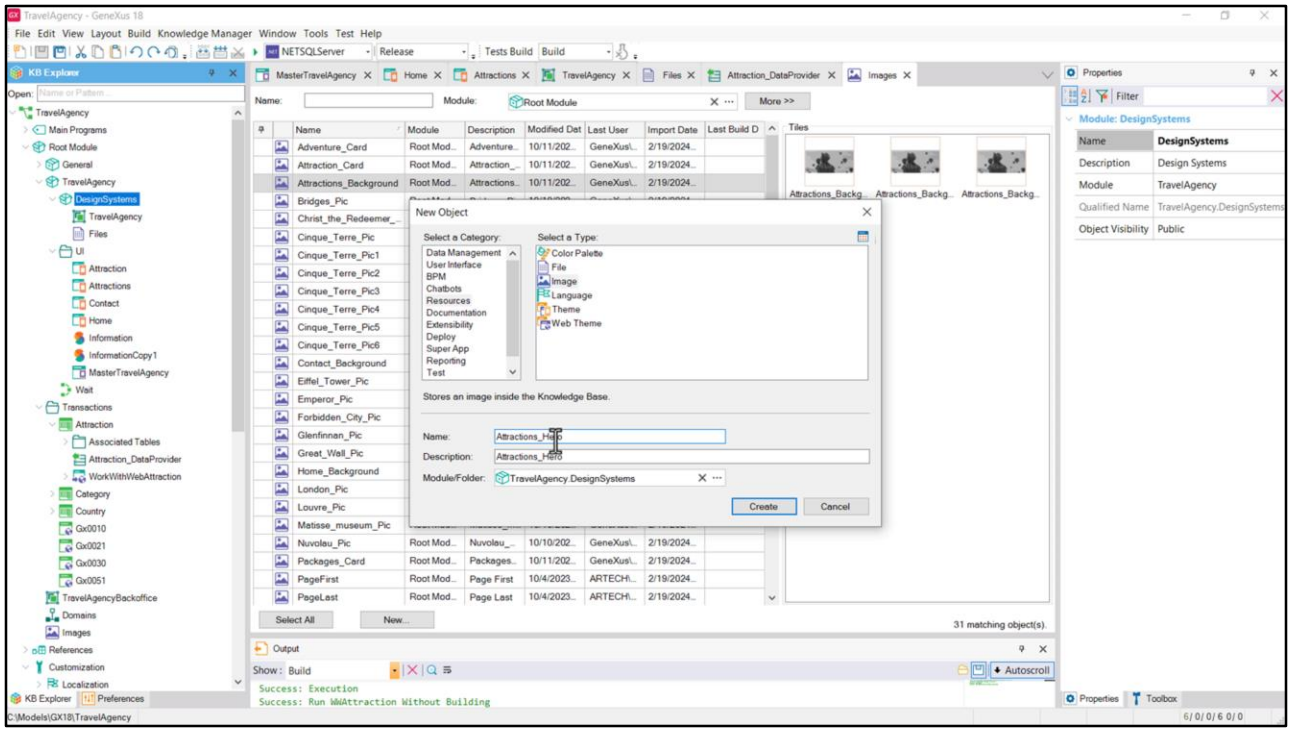


Here we see them already loaded in the Backoffice.

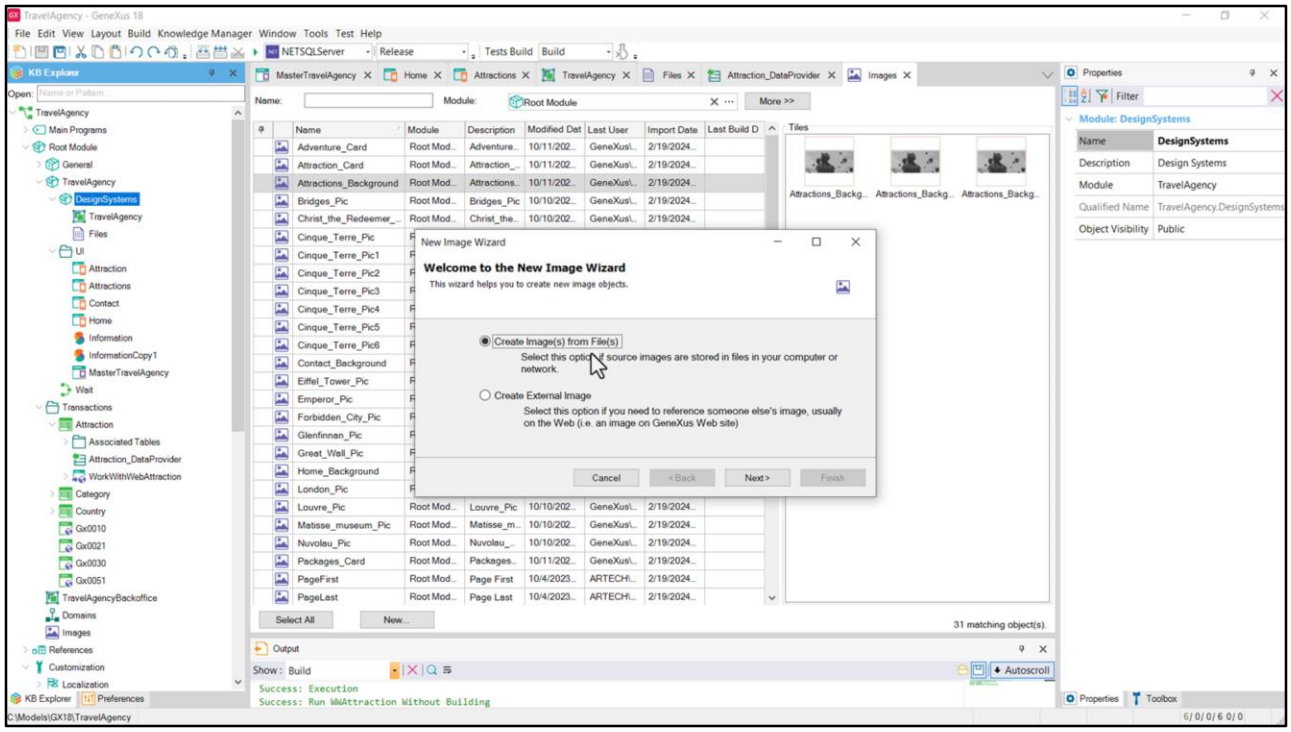


But in that stage I not only loaded the ones that I would use in the database, like these that end in Pic, but I also loaded several of the ones that we were seeing in our design.

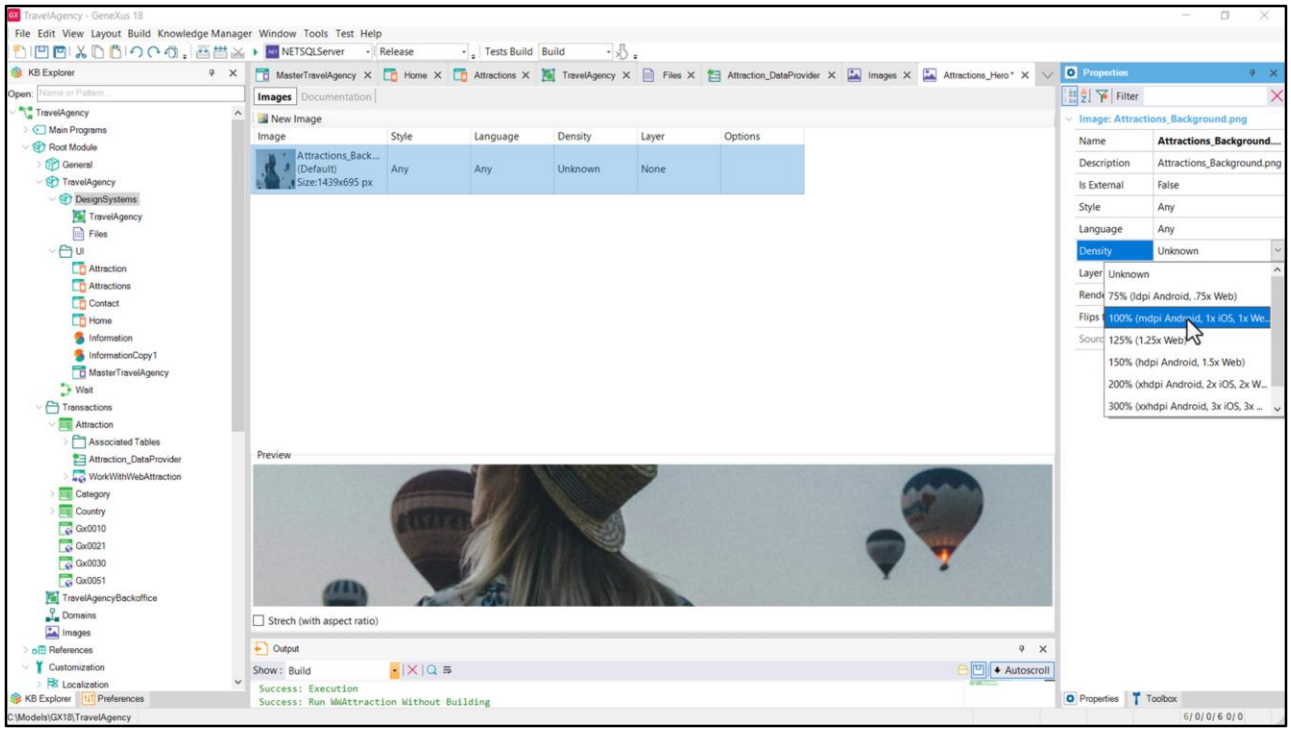
I'll choose one, for example, the Attractions background, this one, to show you how those three files would be loaded into the same Image object. Most of you already know this but I'm going to explain it for those who do not.



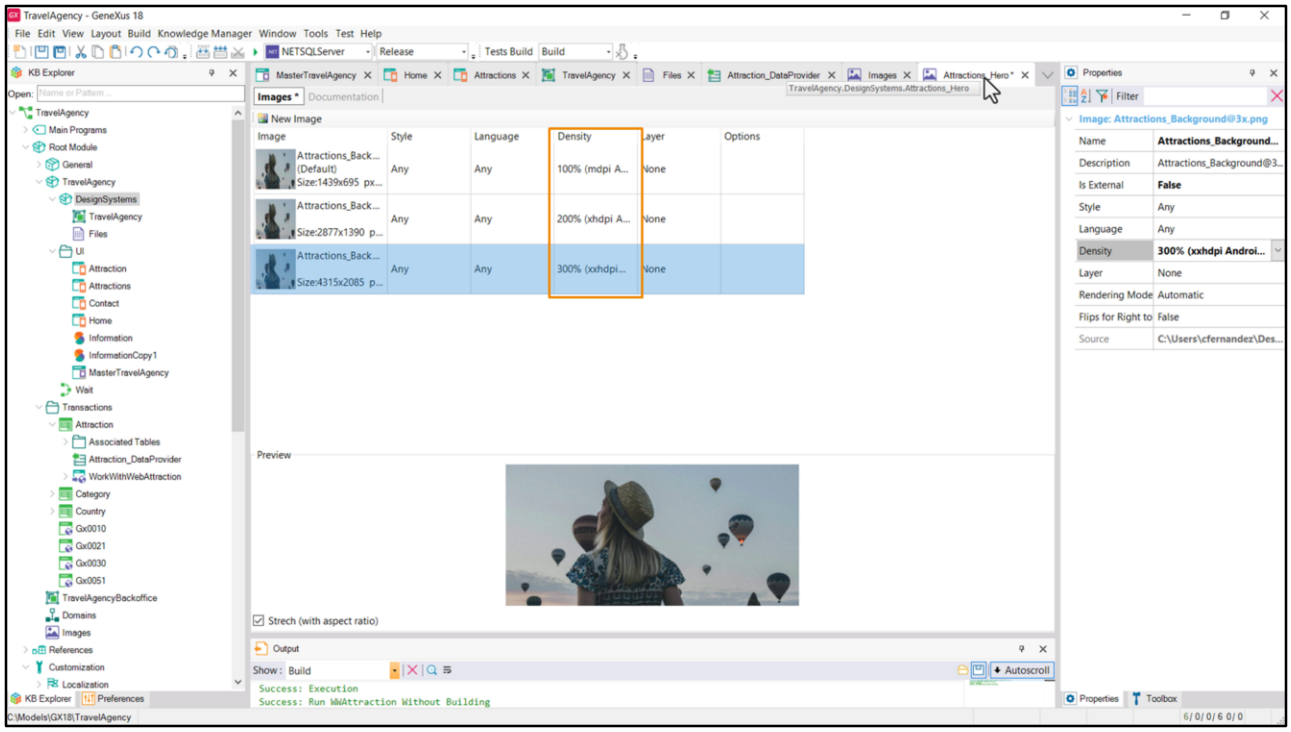
Since I'm already at the module level, I'm going to insert the Attractions Background image, or Hero, or whatever I want to call it, creating a GeneXus object of Image type in the DesignSystems module. And I will name it Attractions_Hero, for example, instead of background.



And here we are asked if we want to create the image from files or from an external source such as a website.



And I come here and enter, for example, the file corresponding to density 1x. So what I do from there... we see that these properties appear, and I enter the 1x density.

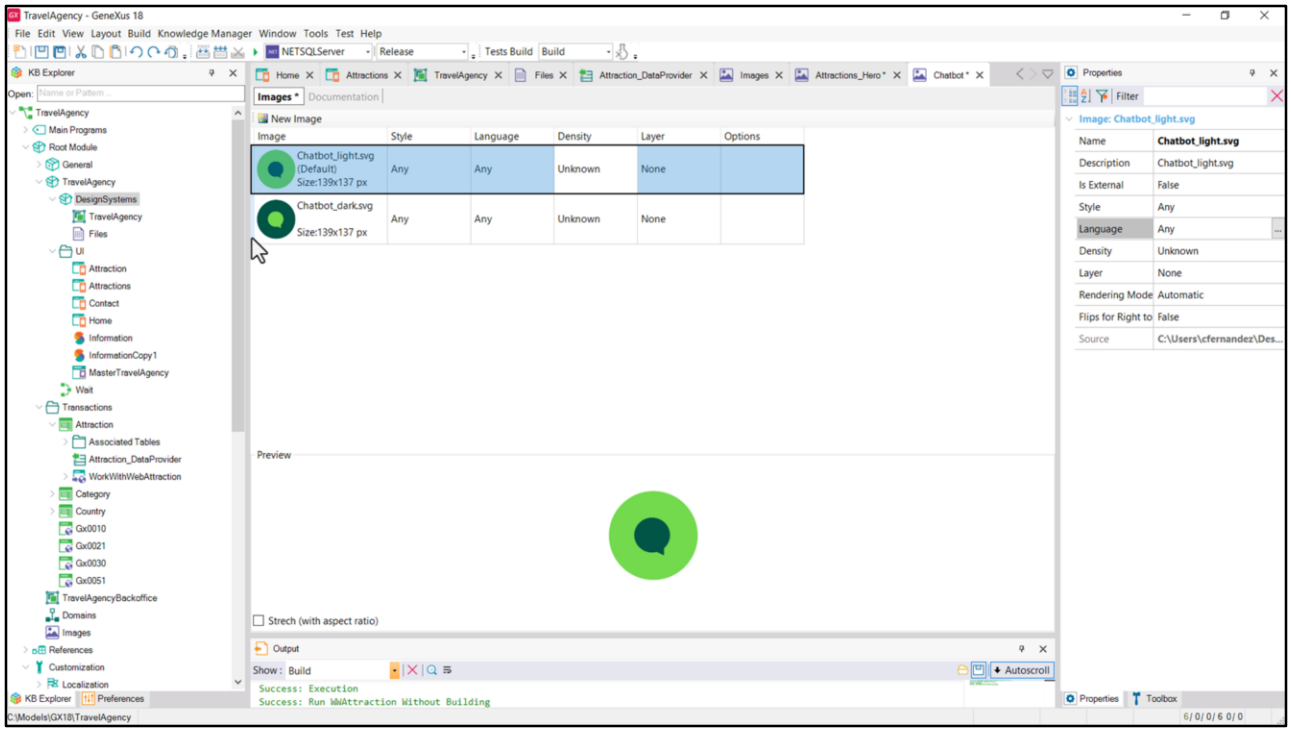


Next, I add the other 2 images... the 2x image, I set the 2x density, and also the 3x one, right?

Note that a series of properties are displayed, including Density, but there are others.

We must take into account that the Image object is only one, this one, named Attractions_Hero, but when we want to use it somewhere in the KB, the specific images that will be used will depend on these dimensions. In this case, for example, the only difference will be about density, because the three images apply for any style, that is to say, for any DSO of the KB, for any language, and without layer specification. That is to say, one or another will be used according to the resolution of the device that is running the application.

Now, I could also vary the image by DSO, or by language if I have texts at the image level, for example.

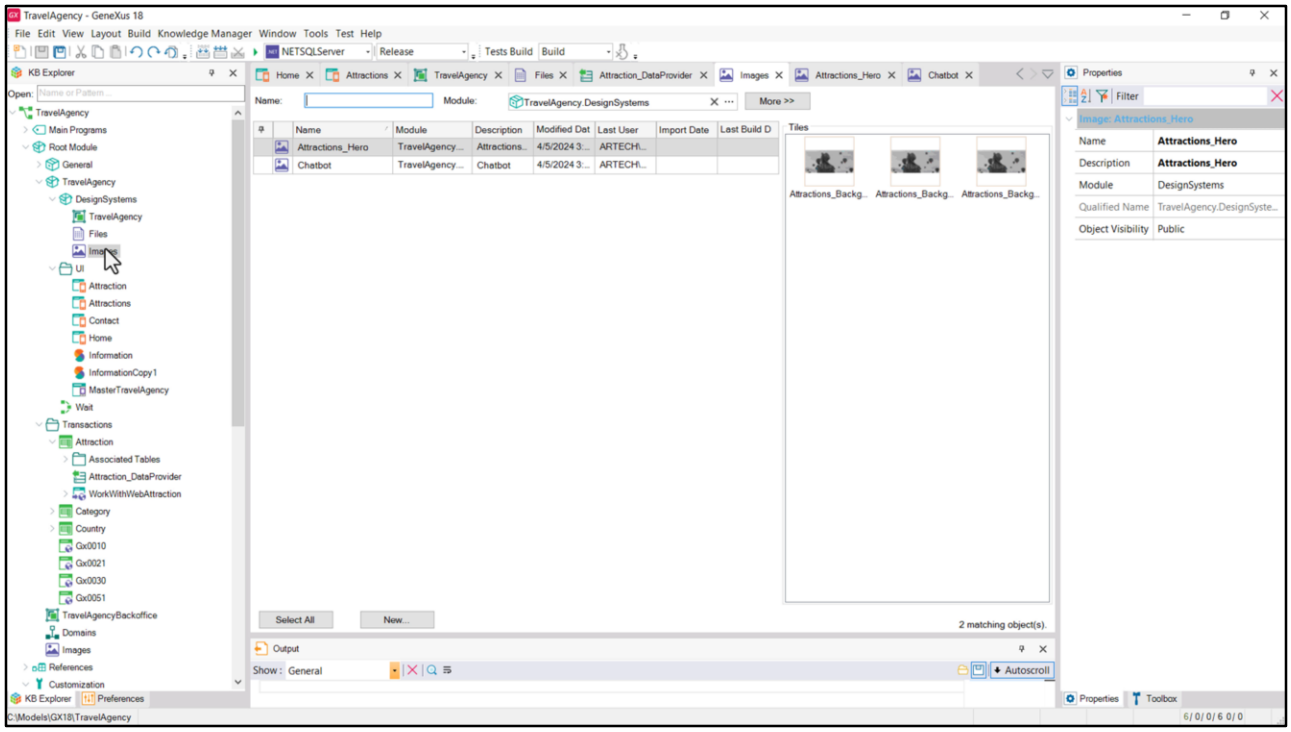


Now I will enter the chatbot image. This image, precisely because it is SVG, does not require variation by density. So it will stay like this, with a single file valid for any circumstance. Even though I'm going to leave it entered, it's not going to be fixed. This chatbot image will vary according to the mode: whether it is light or dark.

We haven't seen yet what the application design should look like with the dark mode. Chechu already did it but in another file.

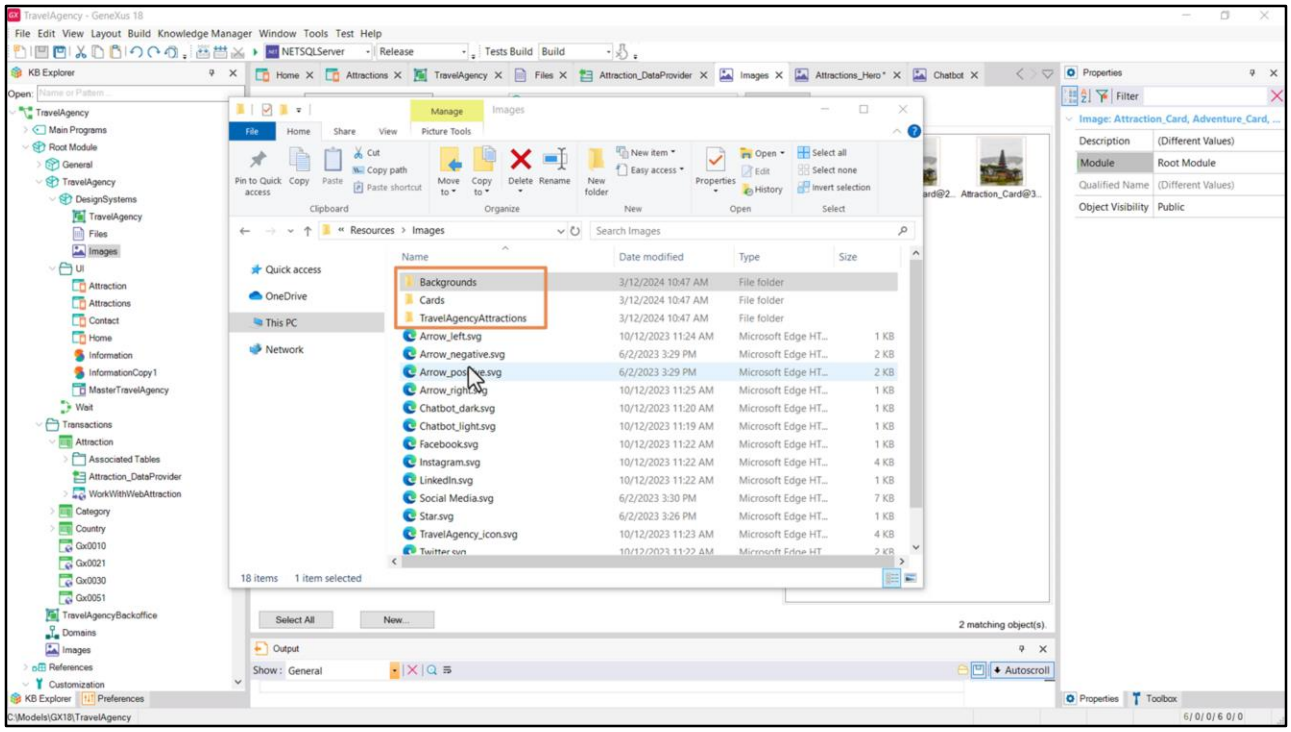
So, what I'll do is only so that it is already expressed here, without worrying now; later on we will add another version of the image, which will be for the dark mode. How will I be able to specify that one is for the light mode and the other for the dark mode? That will be given by the options, which are not enabled yet because to enable them we will have to specify that we are going to use those two modes. At what level? At the DSO level.

Now I'll leave the file with these two options and save. And how will it know, when I insert this image in one of our panels – in fact, it will be in the Master Panel – how will it know which of these two to use? Well, since the dimensions are the same, it will use the first one it finds. I will record this other one.

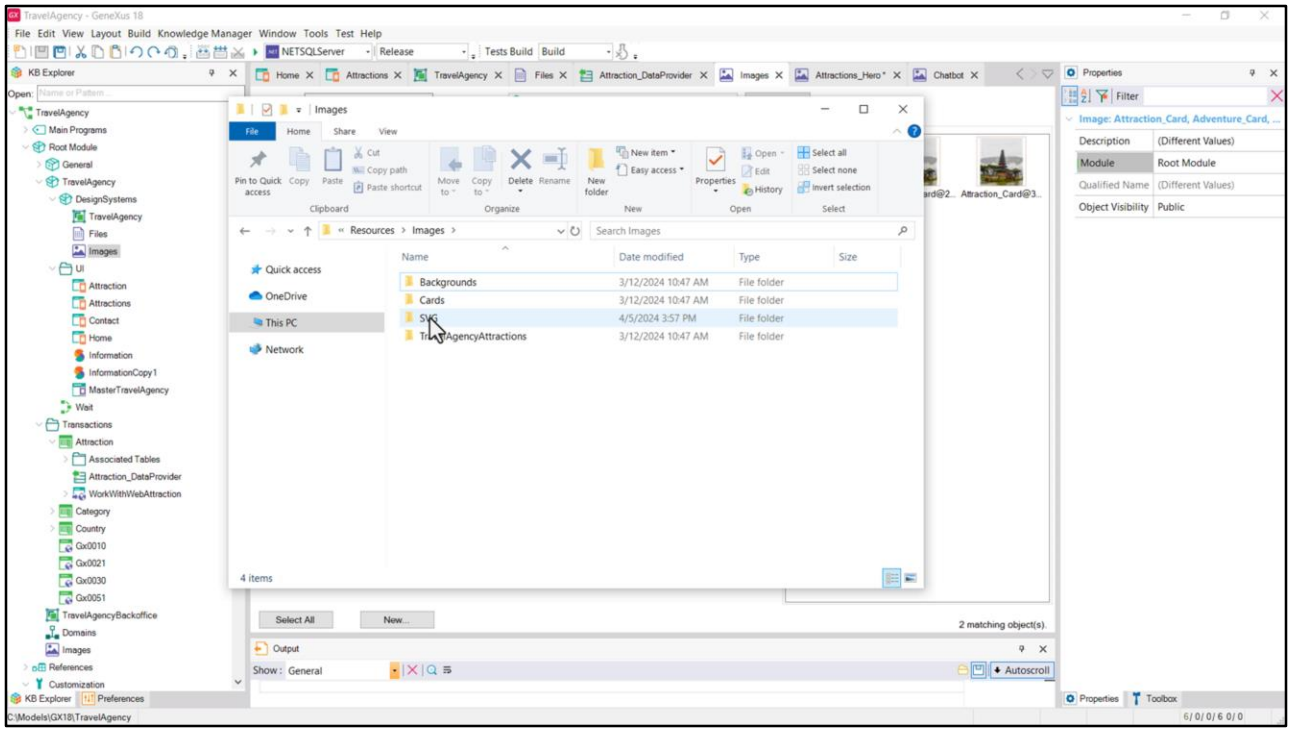


There we see that by placing them inside the Design Systems module, this node appears here, separated from this one here that was the Root. And the two images that we created.

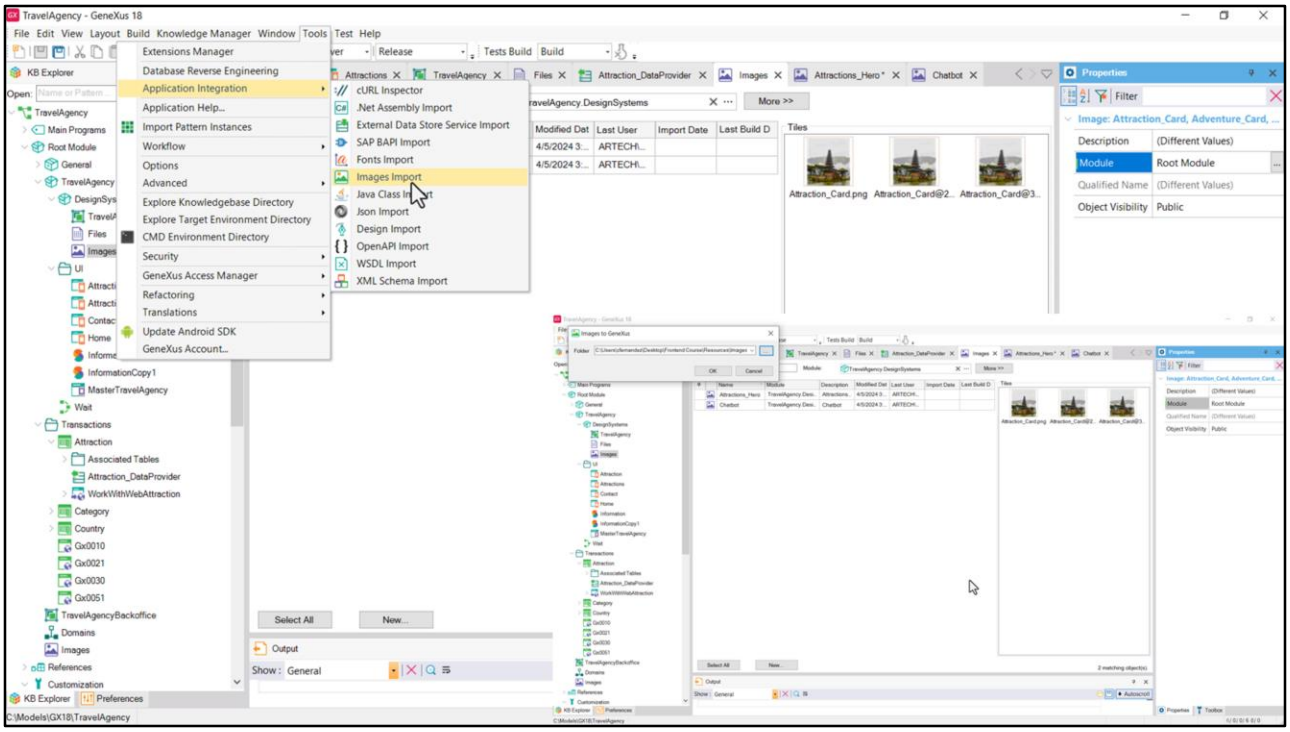
Well, working this way we would introduce all the images. We can also take into account that we have a tool for it, which is this one here. It allows us to import the images from a folder.



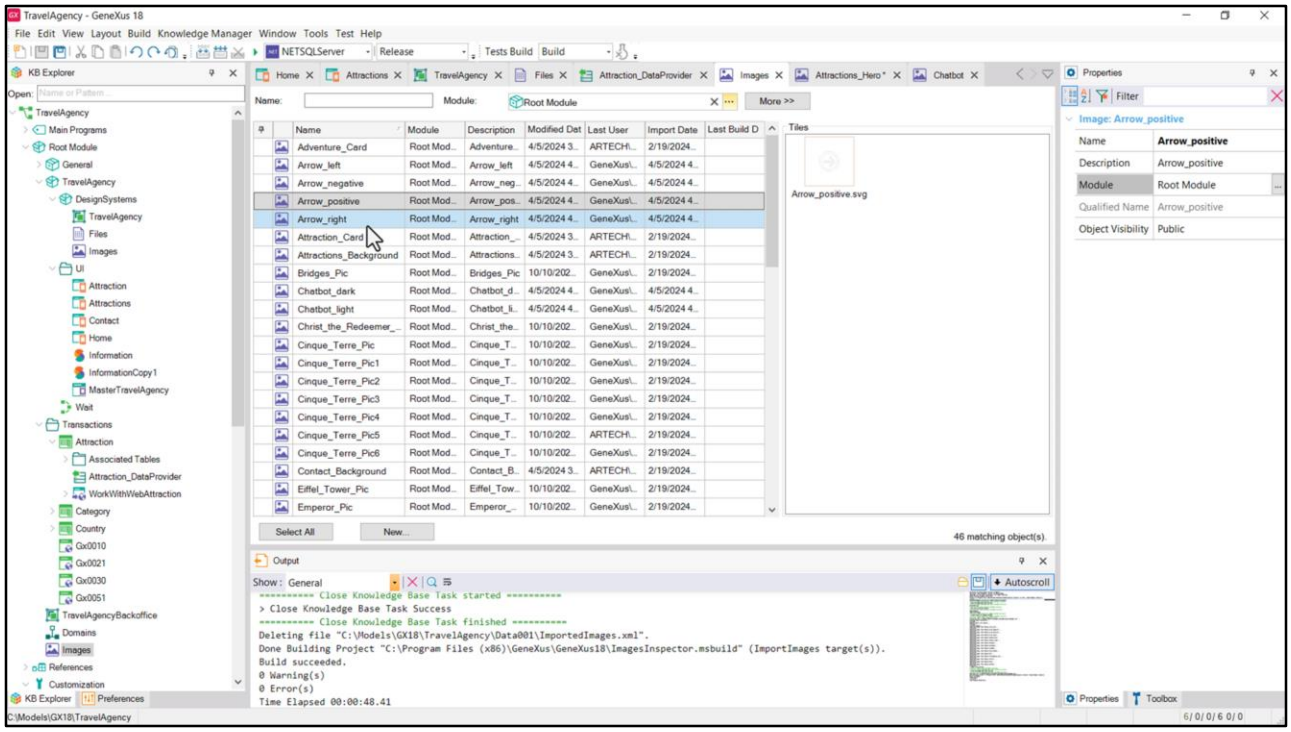
So, for example, to insert all these SVGs, taking into account that I hadn't previously inserted them in the KB (I had inserted the images, the more complex ones – the PNGs – but not the SVGs).



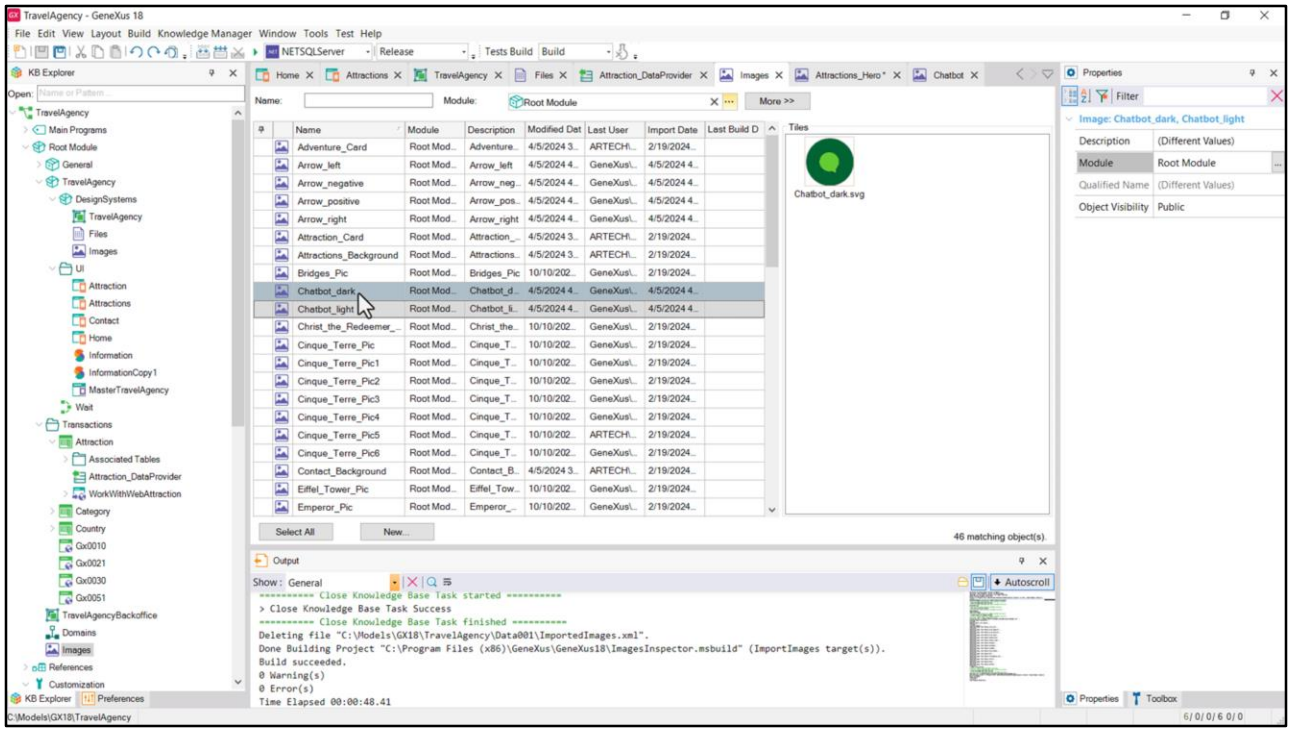
With that in mind, I'm going to create an SVGs folder to make it more organized, and I'll move all these elements there.



And I'm going to come back here and import everything from that folder. It imported them, but where?

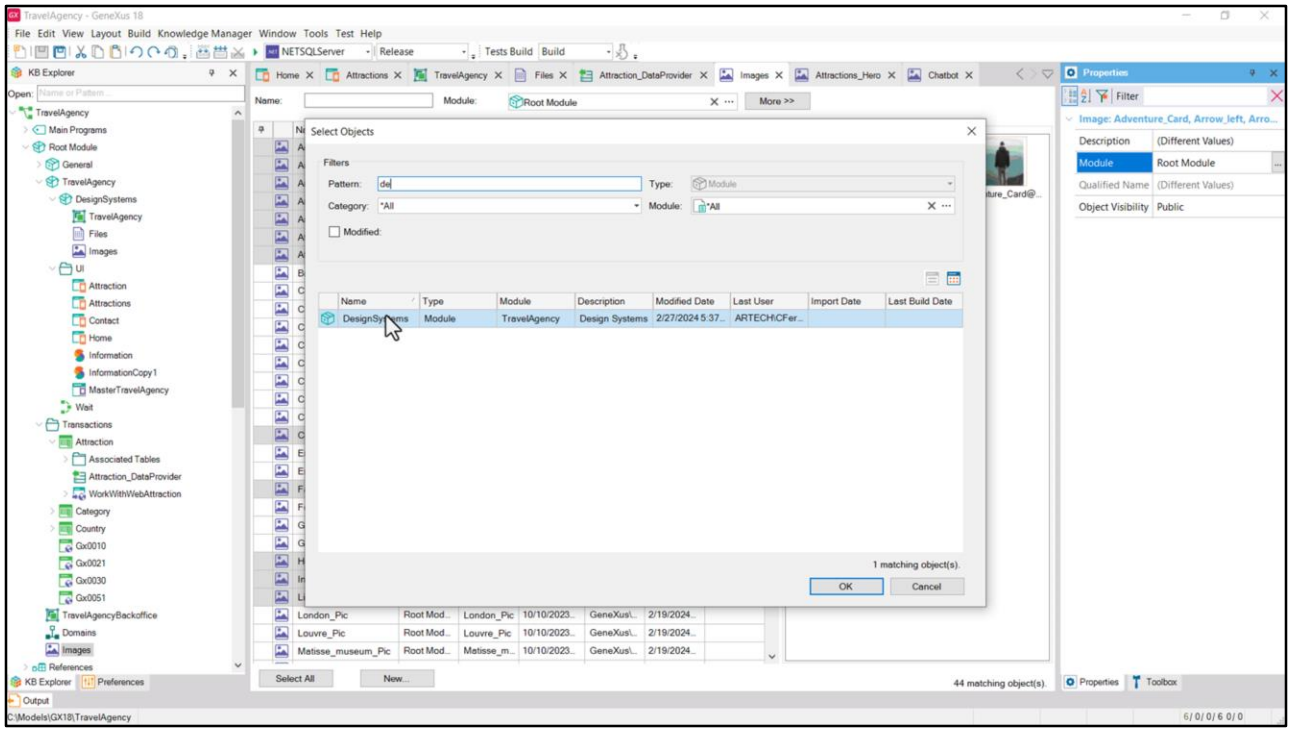


It didn't import them into our module, clearly, but instead it imported them into the Root module. Here we see them.

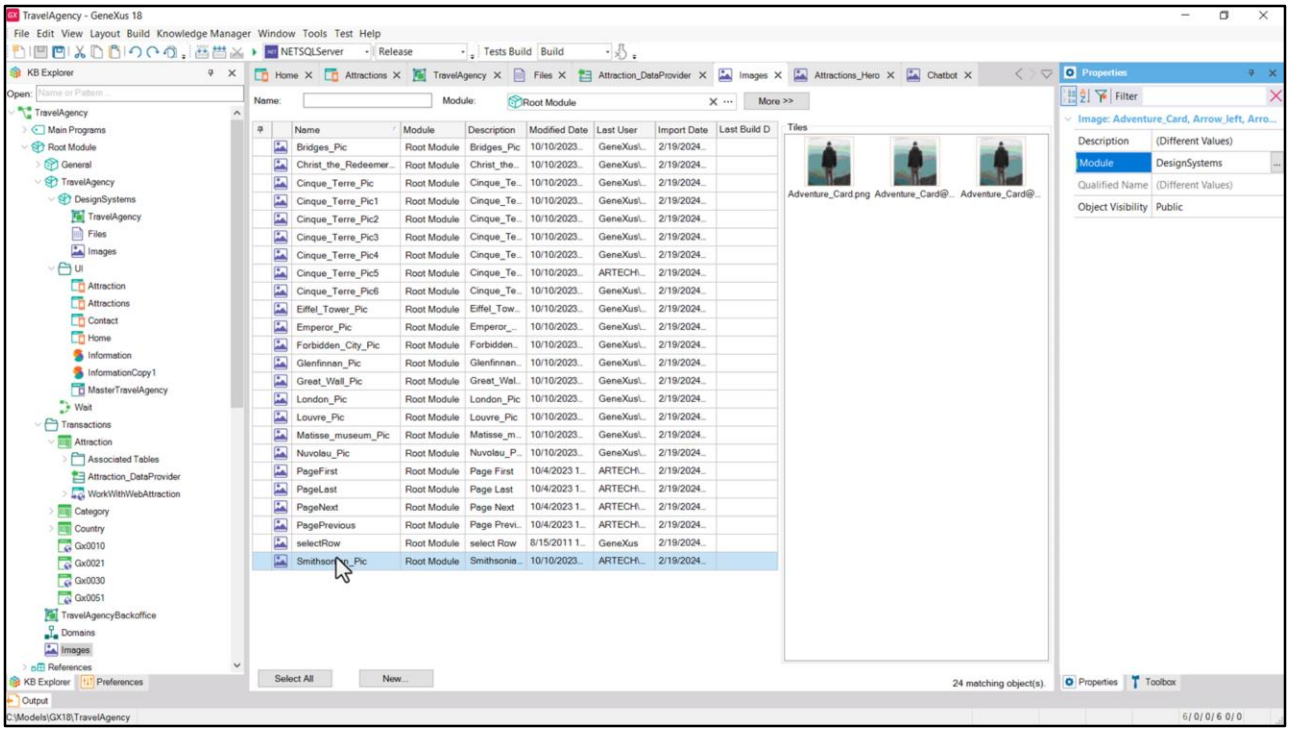


The disadvantage of this mass import is, for example, that the two files that should have been imported as a single image are imported as two separate image objects.

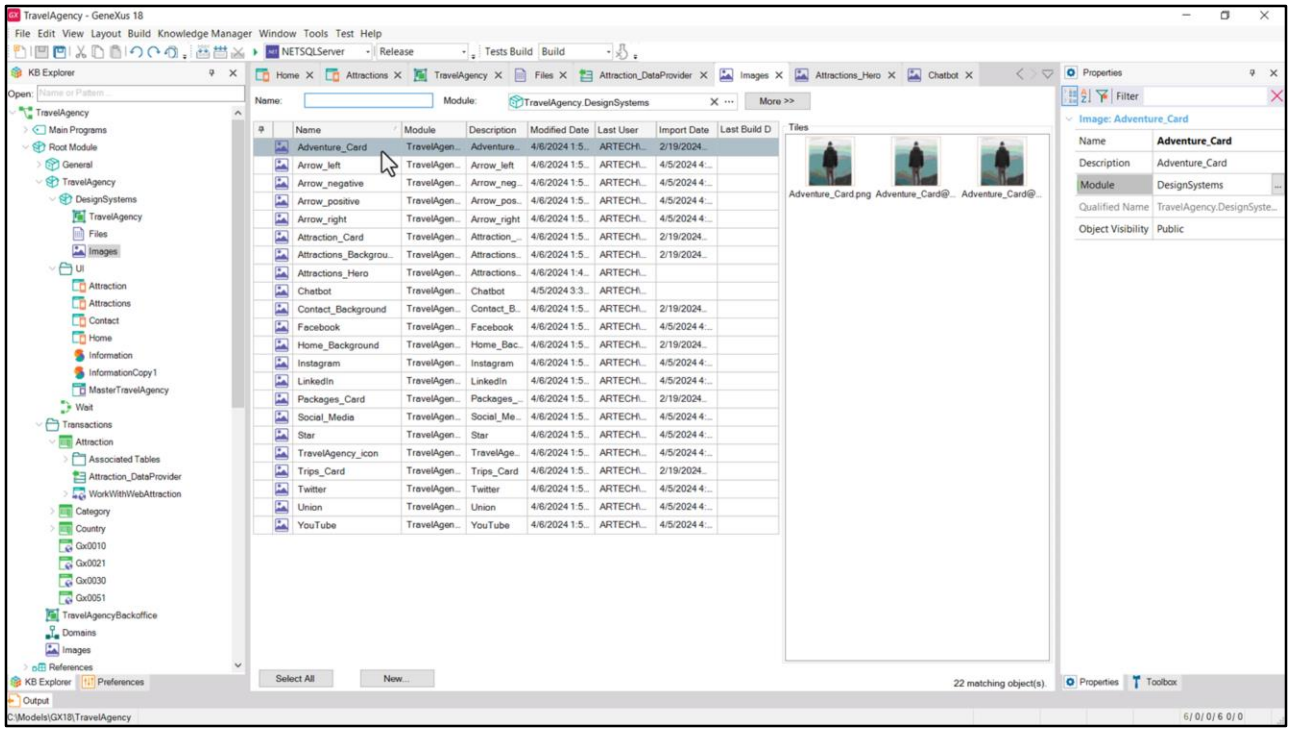
Since we already created this one in our Design Systems module we can remove it from here.



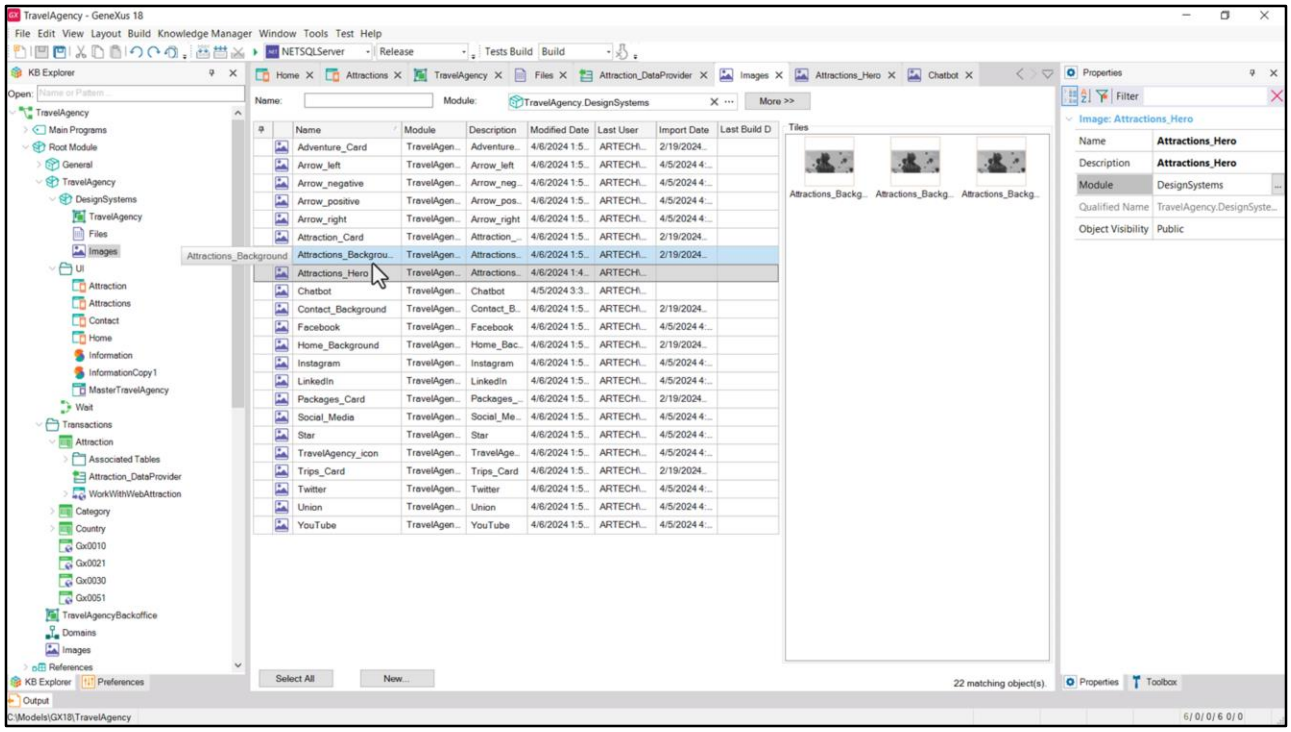
Now, to leave all the assets in order, I will move all the images that will be linked to our Design System to the DesignSystems module.



Only the BackOffice ones are left here.

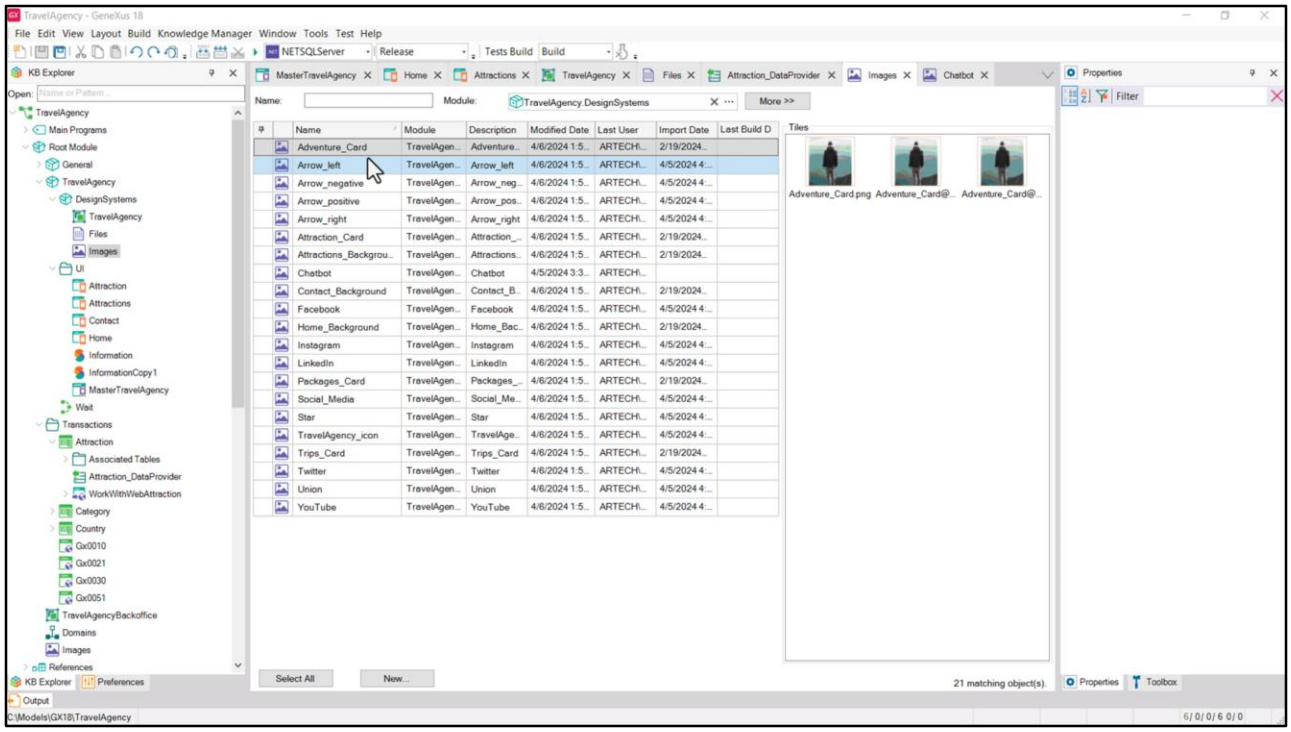


All the others are in the Design Systems module. For example, one, two, three, four cards, which will be from the Home page.



And here, for example, we have the one we created and this one that was already there, which is the same, so we delete the one we created, which was just to show you.

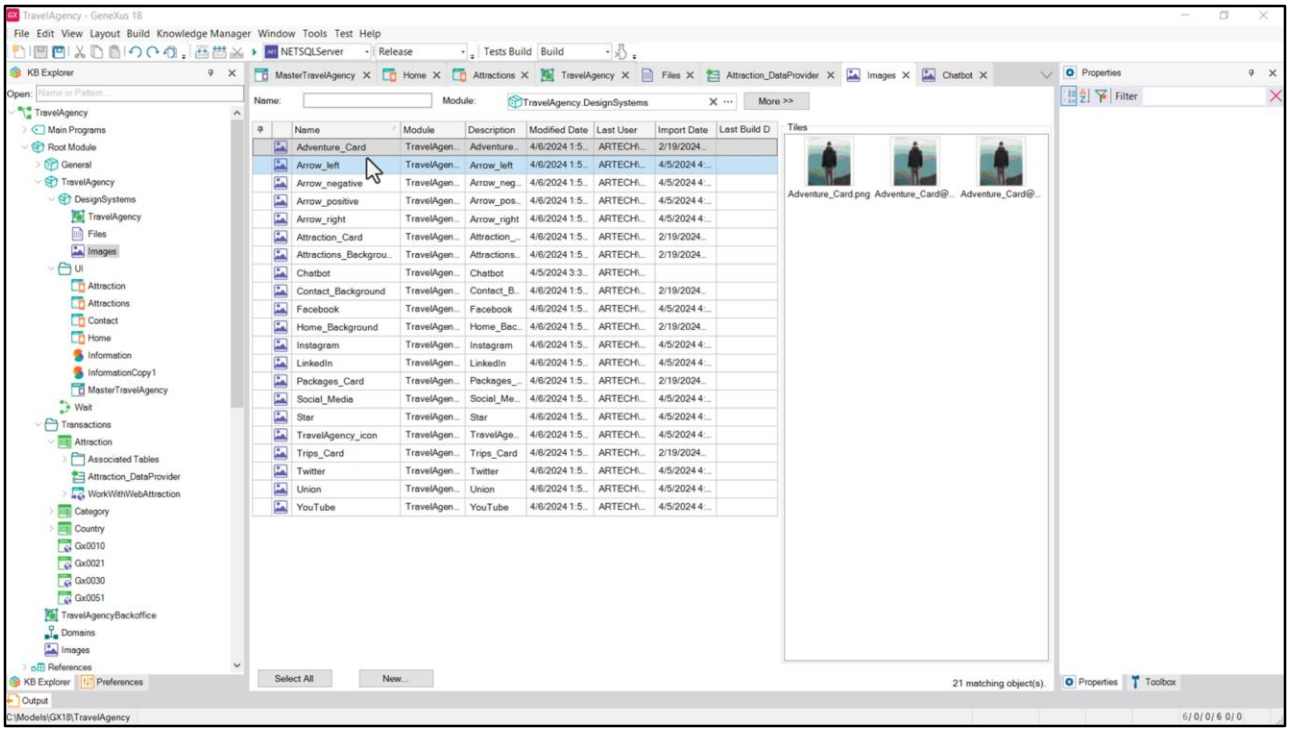
OK.



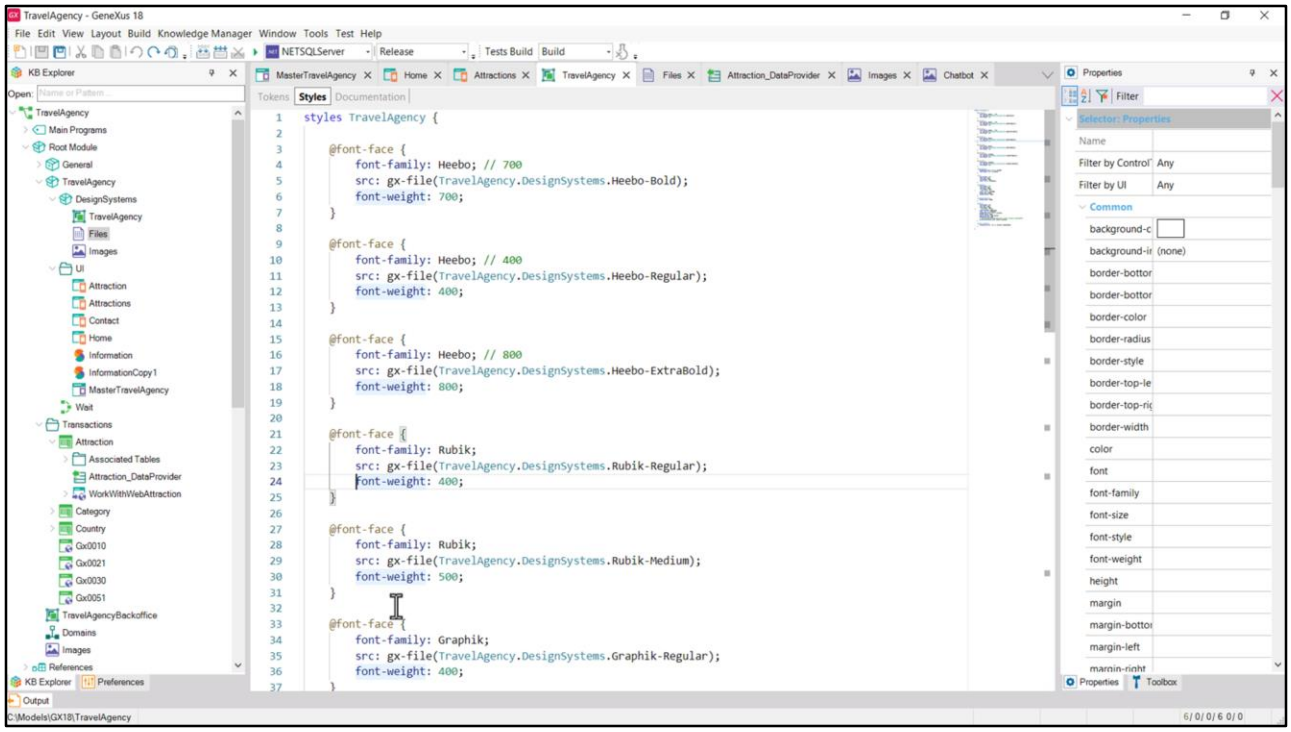
To summarize:

In the initial development stage, it is convenient to incorporate in GeneXus all the assets, that is to say, all the resources that we will have to use later in our screens and DSOs.

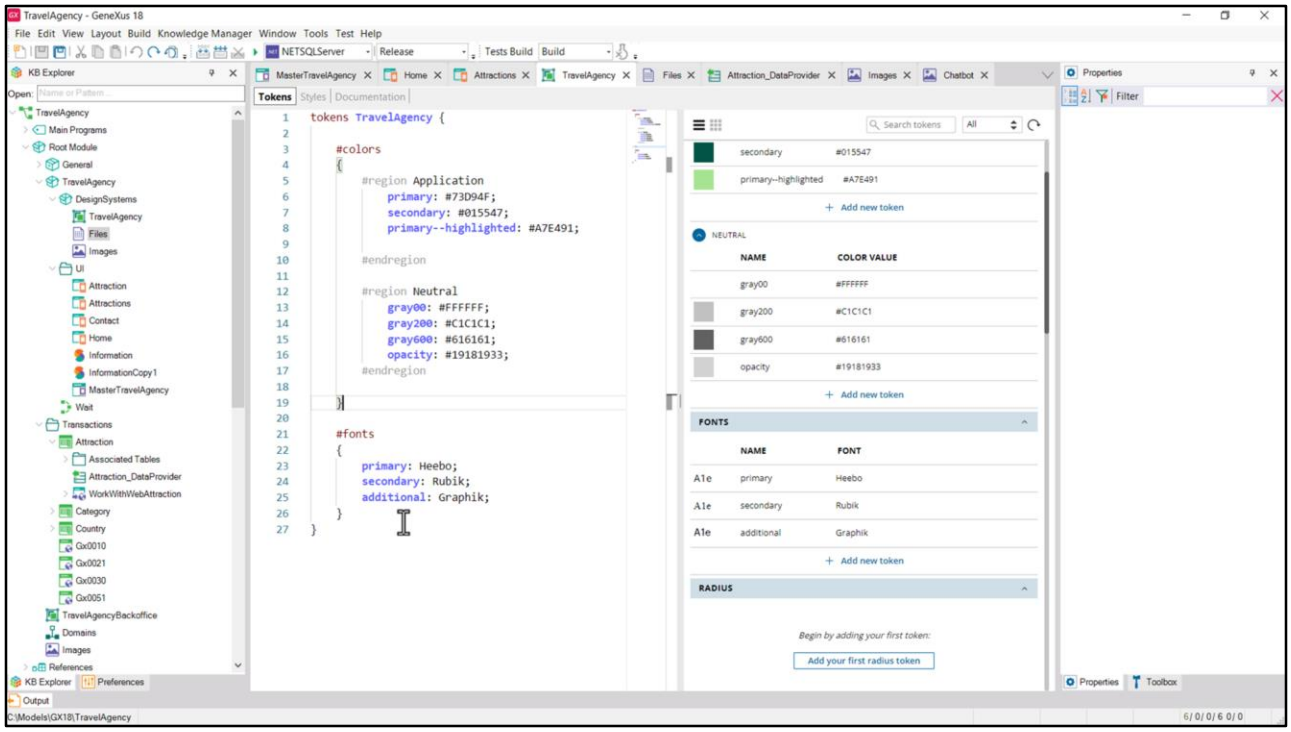
Then already having the images...



... with non-default font files...



...integrated into the DSO through the font-face rules... seems to be a good idea.



We could even decide to have these font families tokenized...

And of course, include the color palette of our application as tokens.

In the next video, we will expand on this and consider a second stage of color tokenization to allow for a better modeling of the system, in a more semantic way. See you there!


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