

First Layout in GeneXus

Some additional considerations



Cecilia Fernández

Revisiting our implementation

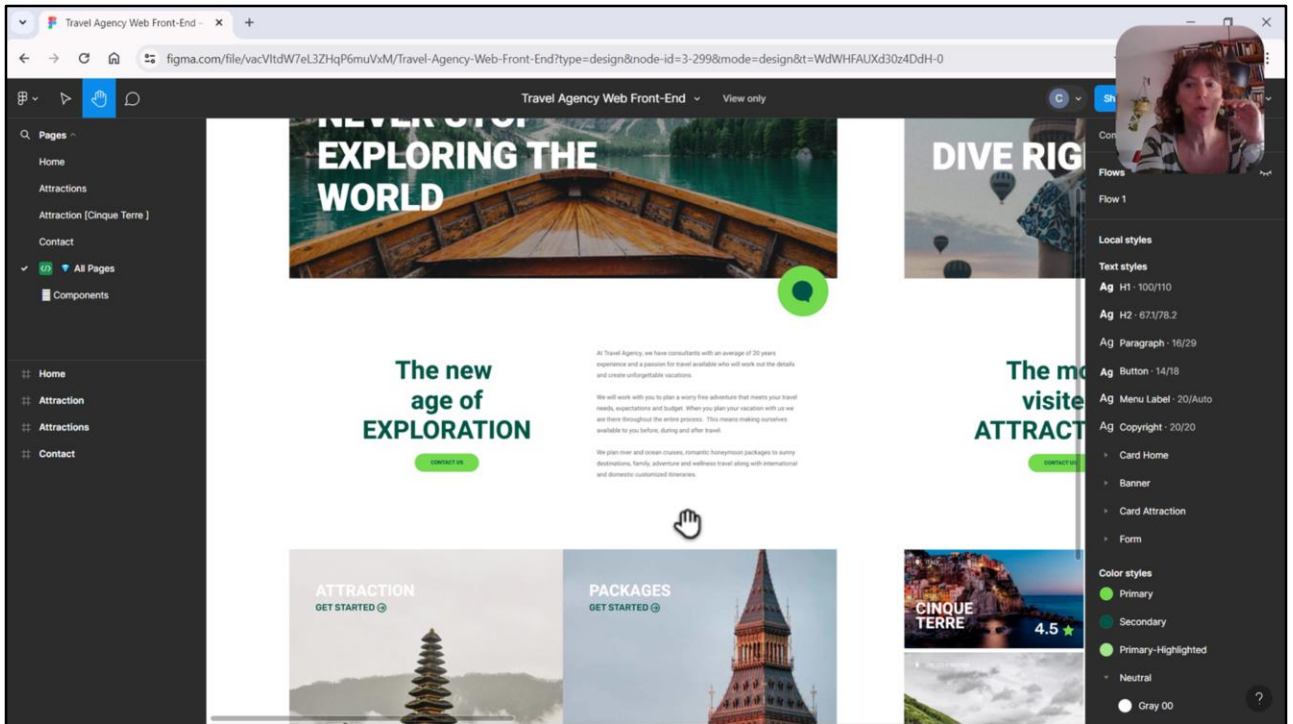
Exploring alternatives

Uncovering unnoticed aspects

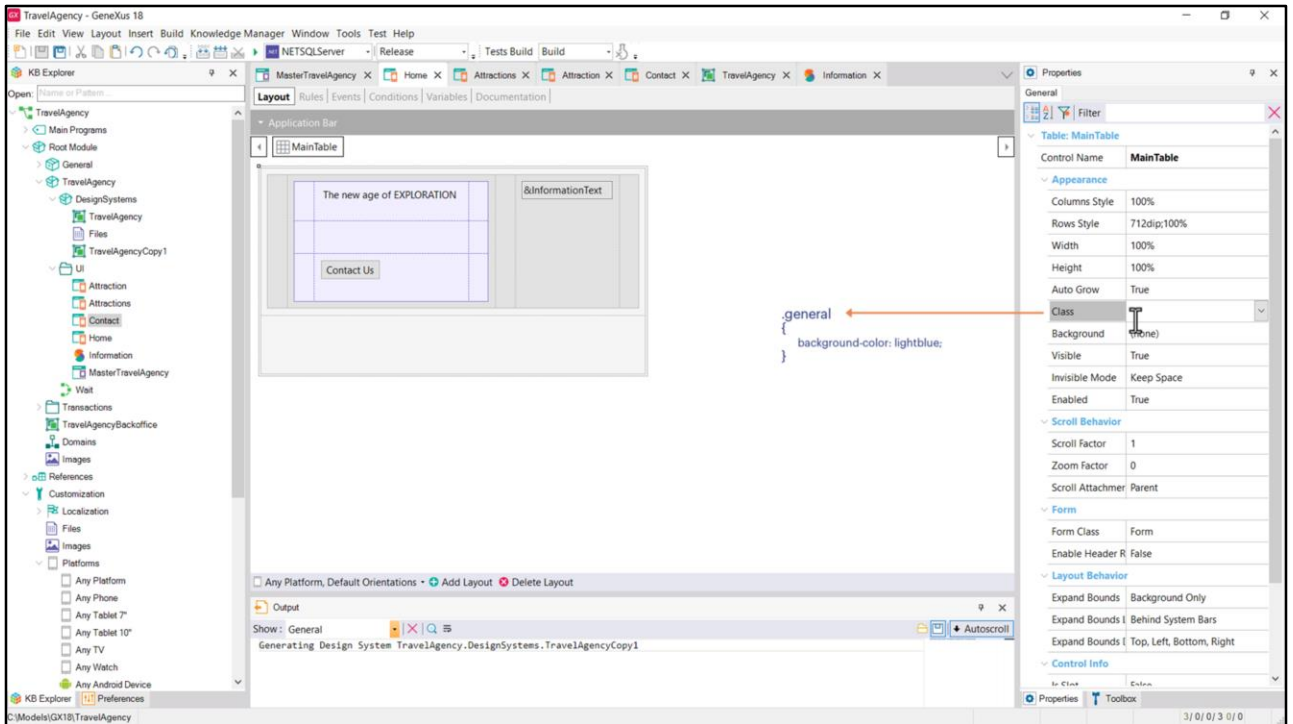
Enhancing Prototyping

Before moving on with the development of the application, I'd like to take a few minutes, in this video and the next one – that is, in the videos of this module – to rethink some of the things that we have implemented, to consider other alternatives, and even to consider aspects that we didn't see at the time, that we didn't notice before.

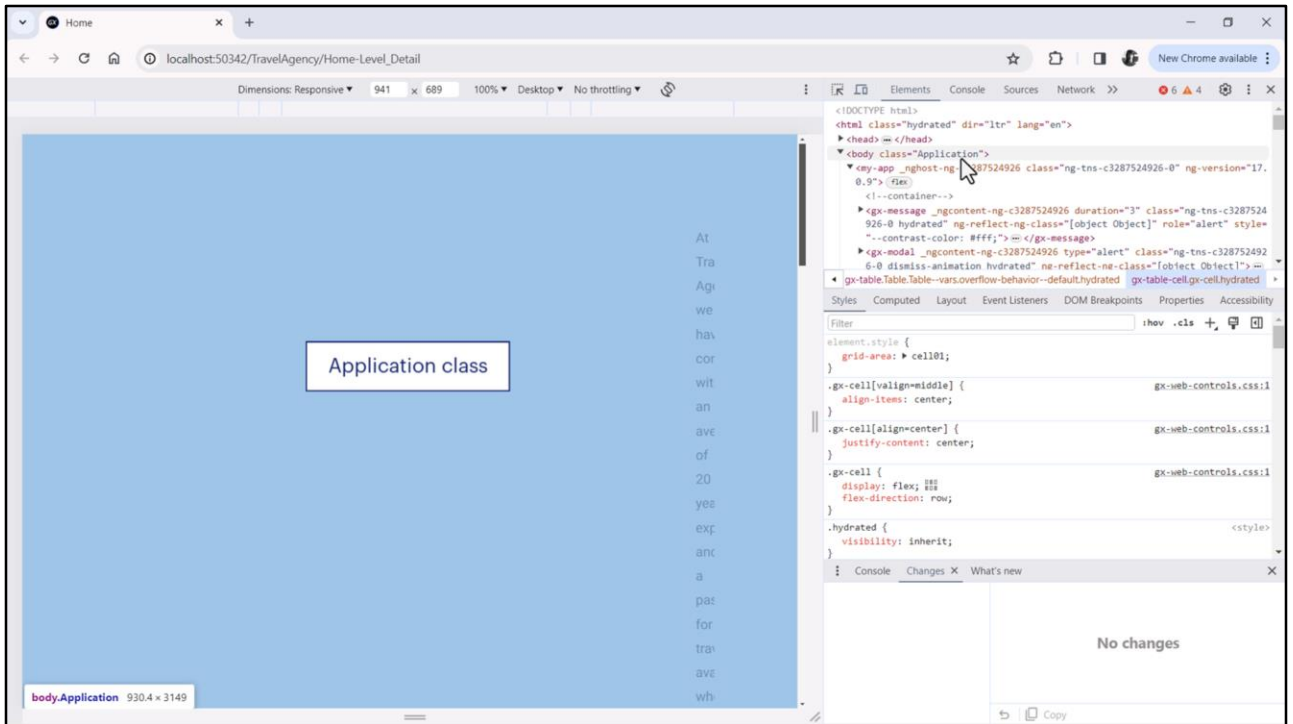
Finally, at the end of the video, we are going to look at matters related to efficiency in the development stage, in prototyping. How to reduce our work times.



So let's start with something that we overlooked and that is, for example, if we wanted the background color of our screens not to be white, but for example "lightblue", how would we go about it?



We could create a class with the property `background-color lightblue` and associate it with the main table of each panel. But every time we will have to remember to make this association, which is not so convenient.



If we inspect the HTML of our Home page, the body tag has the Application class associated with it. This happens in general with frameworks such as Angular or React. Every HTML body tag will have this class associated with it, and therefore, it will apply to everything that is placed inside the layout.

What's interesting is that this class will **also** apply to native application layouts: Android, iOS. It doesn't matter that there is no HTML there and the implementation is completely different. The Application class will have the same meaning.

Therefore, for DSOs that will apply to the Panels world, using that class to configure what we want to be valid for all screens is a good idea, and a cost-effective one.

The screenshot shows a web browser window displaying a 'Travel Agency' application. The application has a dark blue sidebar with 'TRAVEL AGENCY' and navigation links for 'Attractions', 'Categories', and 'Countries'. The main content area is titled 'Attractions' and features a search bar and an 'INSERT' button. Below the search bar is a table of attractions with columns for 'id', 'Name', 'Country', 'City', 'Category', 'Photo', and 'Rating'. The table contains 11 rows of data. A developer console is open on the right side of the browser, showing the HTML structure of the page. The console highlights the 'body' tag with the class 'form-horizontal Form form-horizontal-fx'. The console also shows the 'element.style' object and the 'Layer' object for the selected element. The console output includes the following HTML structure:

```

<!DOCTYPE html>
<html lang="en" class="history flexbox csstransitions hydrated">
  <head> =</head>
  <body class="form-horizontal Form form-horizontal-fx" style="background-color: #f9f9f9; color: #212121; font-family: sans-serif; font-size: 14px; margin: 0; padding: 0 0 0 0;" data-hasenter="false" data-skipenter="false">
    <:before>
      <form id="MAINFORM" autocomplete="off" name="MAINFORM" method="post" tabindex="-1" class="form-horizontal Form" data-gx-class="form-horizontal Form" novalidate action="wattraction.aspx">
        <div style="height:0;overflow:hidden"> =</div>
        <div class="ex-call-target ton off emotv"> =</div>
        <div class="row" data-cs="2" data-kind="parent" data-bbox="225 225 255 255">
          <div class="table-top" data-bbox="225 225 255 255">
            <table border="1">
              <thead>
                <tr>
                  <th>id</th>
                  <th>Name</th>
                  <th>Country</th>
                  <th>City</th>
                  <th>Category</th>
                  <th>Photo</th>
                  <th>Rating</th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td>4</td>
                  <td>Christ the Redeemer</td>
                  <td>Brazil</td>
                  <td>Rio de Janeiro</td>
                  <td>Monument</td>
                  <td><img alt="Christ the Redeemer" data-bbox="418 255 448 285"/></td>
                  <td>4.0</td>
                </tr>
                <tr>
                  <td>8</td>
                  <td>Cinque Terre</td>
                  <td>Italy</td>
                  <td>Liguria</td>
                  <td>Tourist site</td>
                  <td><img alt="Cinque Terre" data-bbox="418 285 448 315"/></td>
                  <td>4.5</td>
                </tr>
                <tr>
                  <td>3</td>
                  <td>Eiffel Tower</td>
                  <td>France</td>
                  <td>Paris</td>
                  <td>Monument</td>
                  <td><img alt="Eiffel Tower" data-bbox="418 315 448 345"/></td>
                  <td>4.0</td>
                </tr>
                <tr>
                  <td>7</td>
                  <td>Forbidden city</td>
                  <td>China</td>
                  <td>Beijing</td>
                  <td>Tourist site</td>
                  <td><img alt="Forbidden city" data-bbox="418 345 448 375"/></td>
                  <td>3.9</td>
                </tr>
                <tr>
                  <td>9</td>
                  <td>Glenfinnan Viaduct</td>
                  <td>Scotland</td>
                  <td>Glenfinnan</td>
                  <td>Tourist site</td>
                  <td><img alt="Glenfinnan Viaduct" data-bbox="418 375 448 405"/></td>
                  <td>4.5</td>
                </tr>
                <tr>
                  <td>10</td>
                  <td>London Towers</td>
                  <td>England</td>
                  <td>London</td>
                  <td>Monument</td>
                  <td><img alt="London Towers" data-bbox="418 405 448 435"/></td>
                  <td>4.5</td>
                </tr>
                <tr>
                  <td>11</td>
                  <td>Lome</td>
                  <td>United States</td>
                  <td>San Francisco</td>
                  <td>Tourist site</td>
                  <td><img alt="Lome" data-bbox="418 435 448 465"/></td>
                  <td>4.5</td>
                </tr>
              </tbody>
            </table>
          </div>
          <div class="table-actions" data-bbox="495 225 525 255">
            <span>UPDATE</span> <span>DELETE</span>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>

```

The console also shows the 'element.style' object and the 'Layer' object for the selected element. The console output includes the following CSS rules:

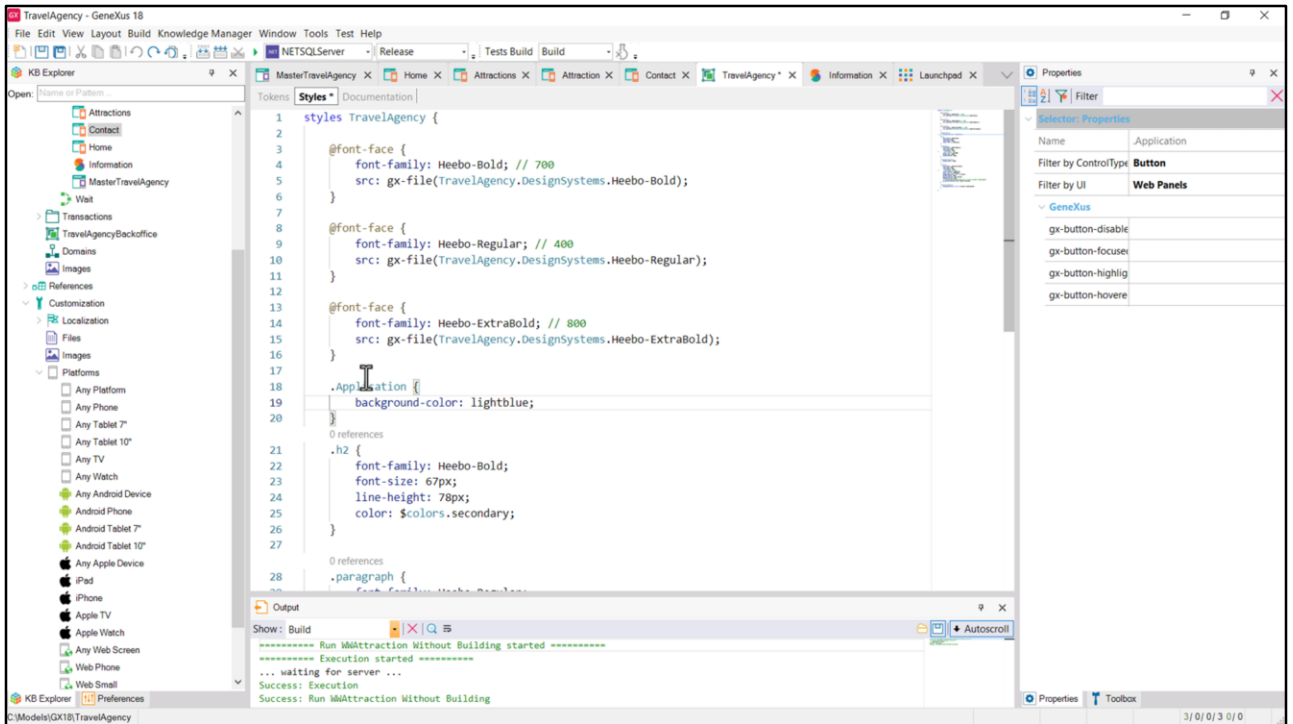
```

element.style {
  flex-grow: 1;
}
Layer GeneXusUnanimoUnanimoWeb
.expandible-container .ww_title-cell {
  margin-inline-start: 0%;
}
Layer
[data-abstract-form] *, [data-abstract-form] > after, no-lib.min...271224374:3
[data-abstract-form] :before {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
Layer user agent stylesheet

```

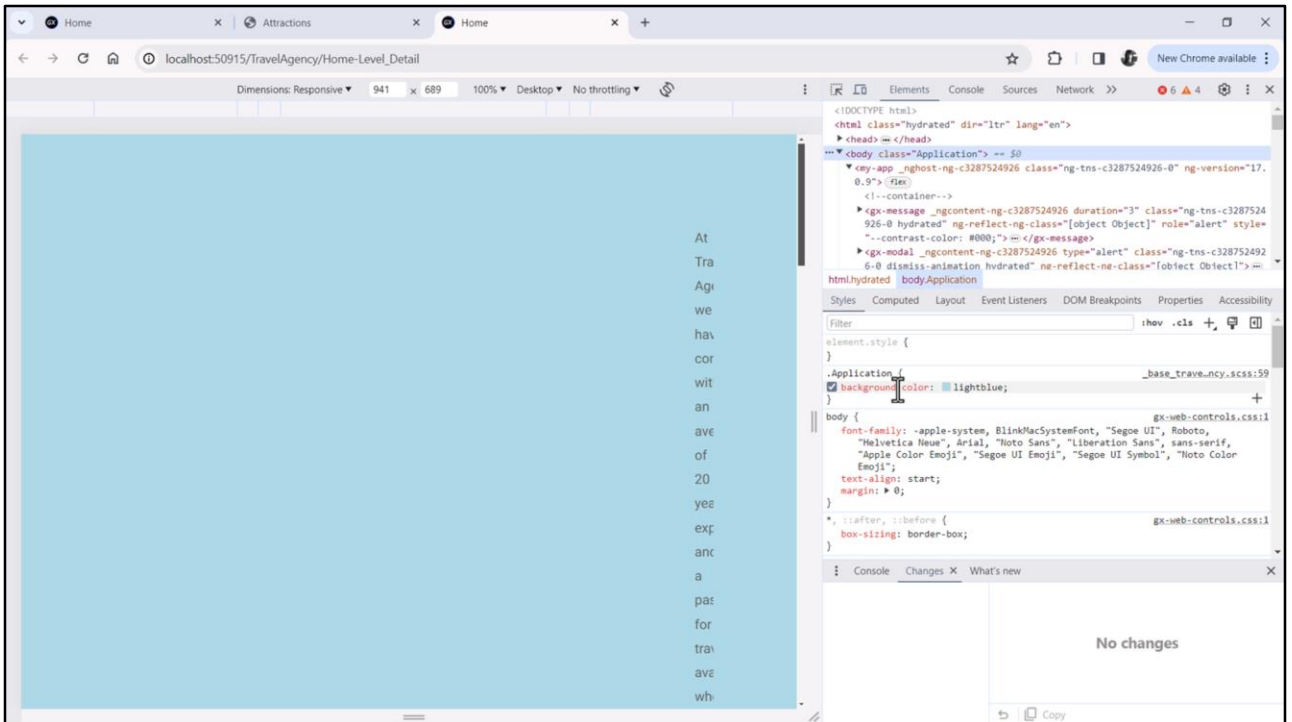
The console also shows the 'No changes' message.

If we were in the Web Panels world, as for example in the backoffice, the body tag of the HTML would not have the Application class but the Form class, so we would have to use this class there, instead of the Application.



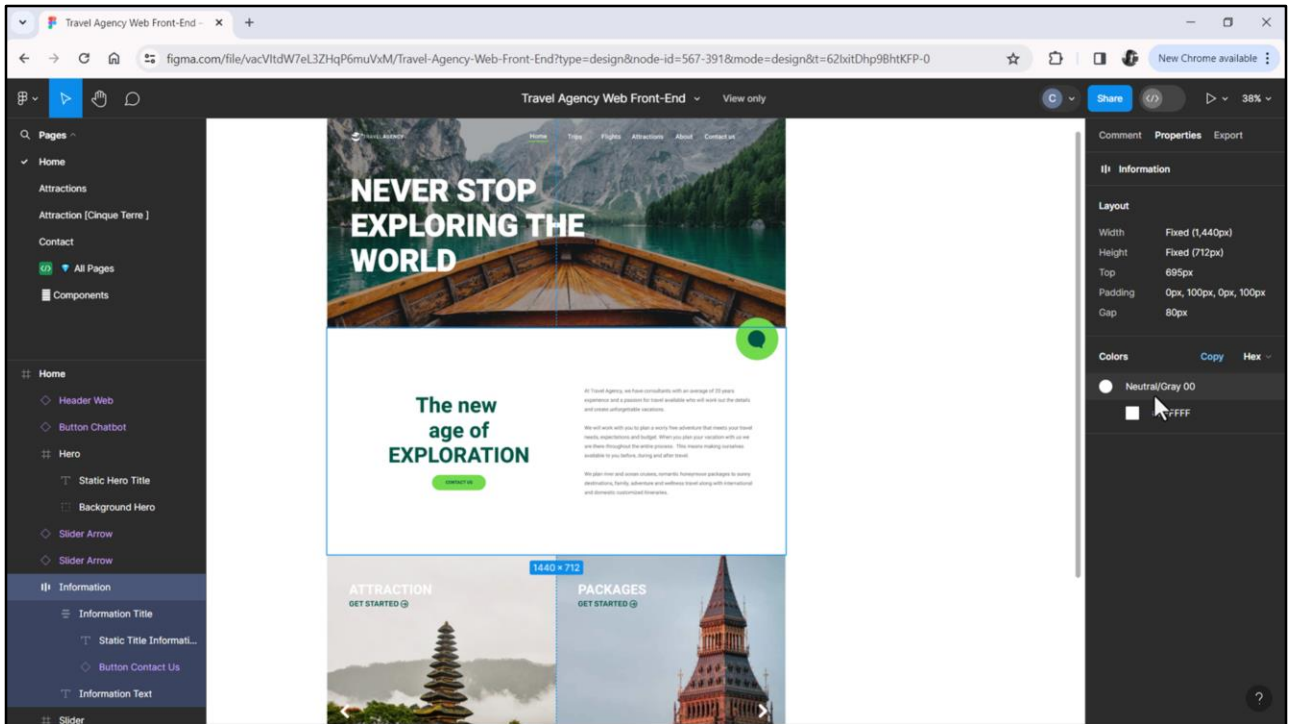
Well, but since we are in the Panels world, because we are implementing for Angular, let's try adding the Application class to our DSO, with the background-color property set to lightblue.

And I execute again.

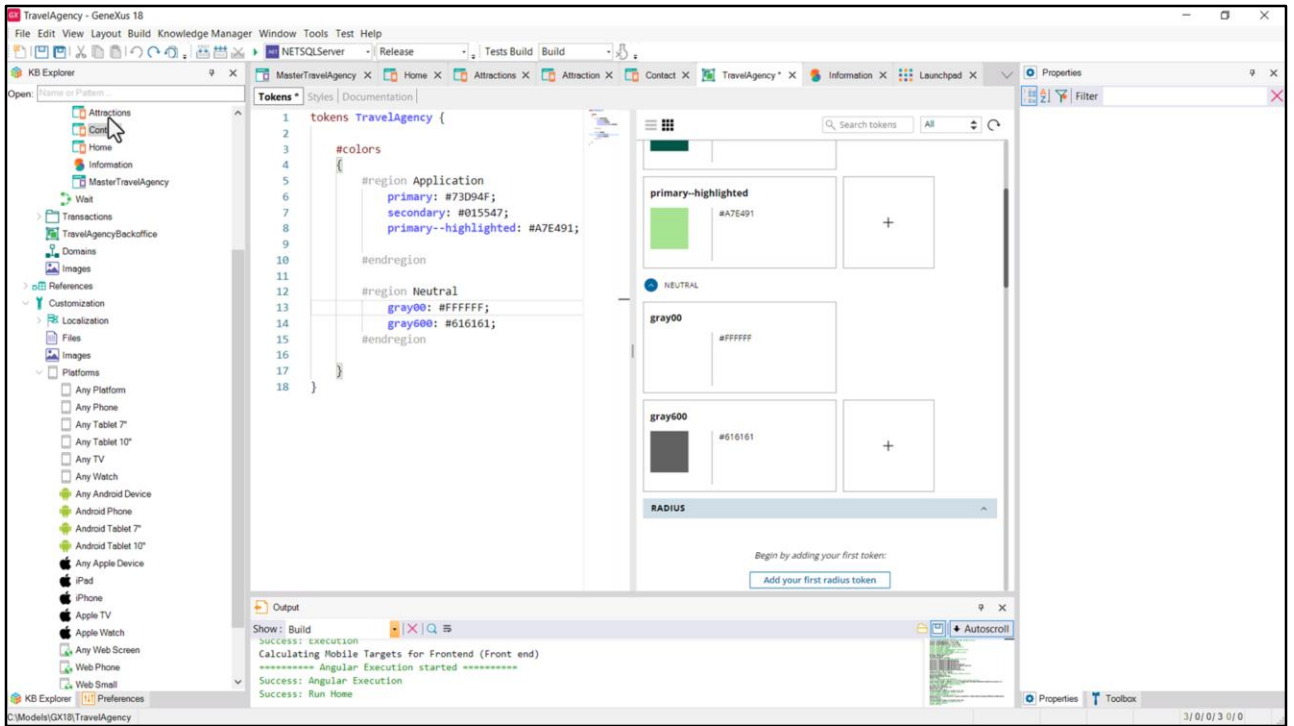


OK, and if we look now... here we see the property.

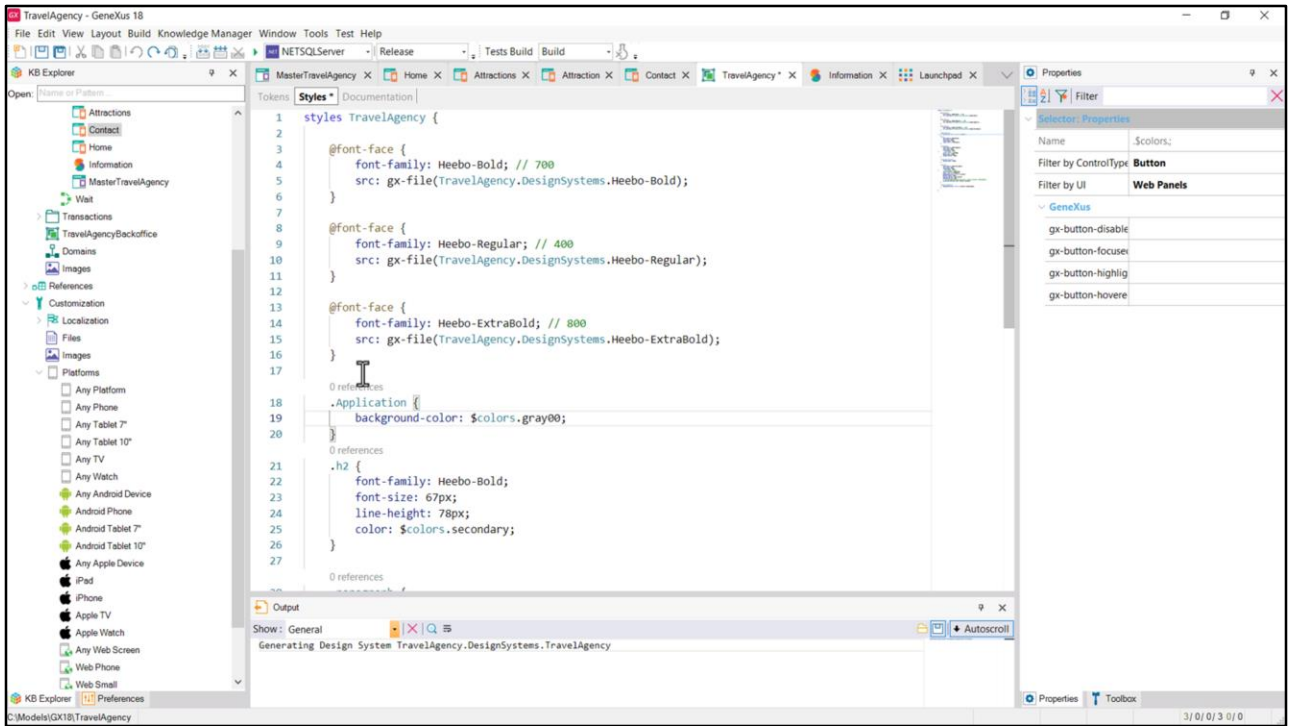
An important clarification: We are using something of low level that is not part of GeneXus to model something of our Design System. A basic feature of the construction of an HTML file is that after the <head> tag comes the <body> tag that appears only once per HTML. So we are taking advantage of the fact that for Frameworks like Angular an Application class (or Form if we are not using frameworks) is associated with that body tag, to give a general style to the screens of our application, using one of those two classes, but if in the future the Application class were no longer associated with the body tag, all this would stop working. The same would happen if for Web Panels the Form class were no longer associated with the body tag. We will have to keep an eye on this because maybe in a future GeneXus upgrade some of this will be modified, or some special property will be provided for this, or who knows what else. We go back...



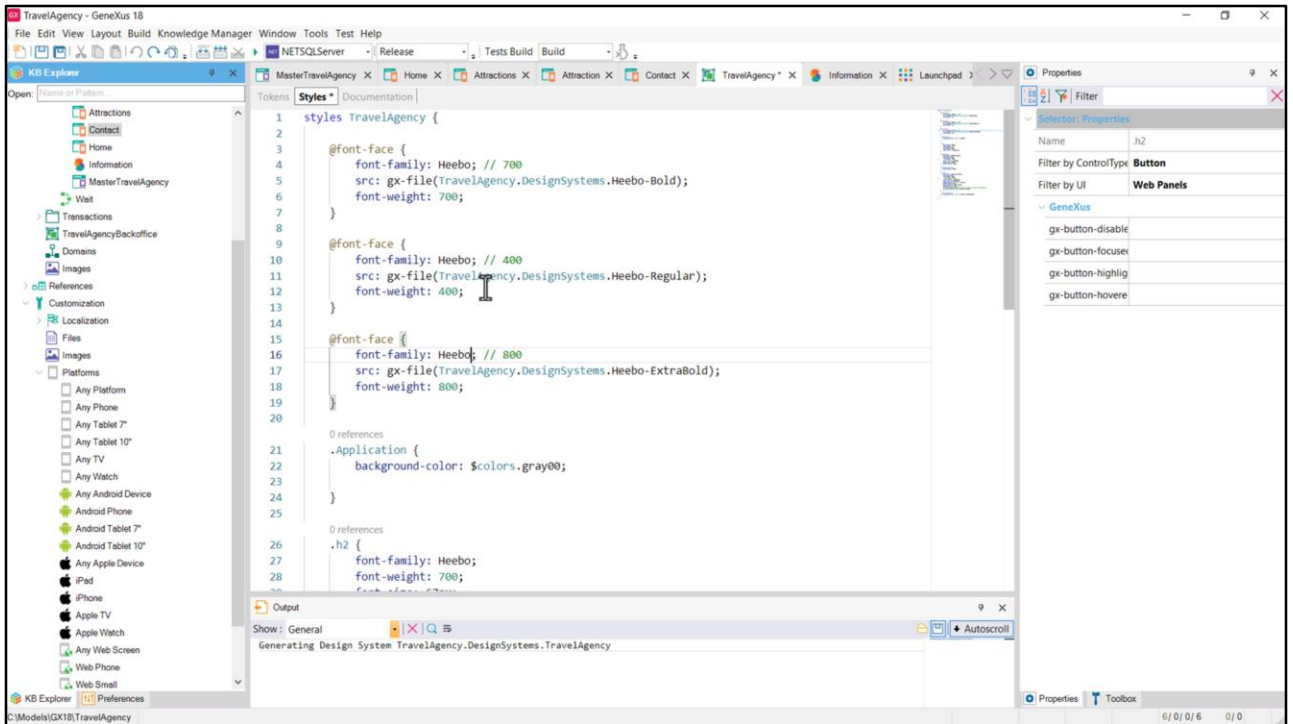
We hadn't paid attention to the background-color because we hadn't noticed it, since the background color is white, and that is the default, but to be able to change it later we will need to explicitly set the value in the property, to change it easily in the future. So, first let's add a gray00 color token...



That's it.



And now let's change the value of the property for the token.



We could take advantage of this class that will be applied as the root of all the objects with an interface of the Panels world to define there the Heebo font family and not have to repeat it everywhere.

You may wonder how, if we have never repeated the font family so far? If for each of the 3 classes we have, there is a Heebo, yes, but it is different! Remember that what made them different, which was their weight, we had decided to express it together with the family.

But we could have kept both separate. So...

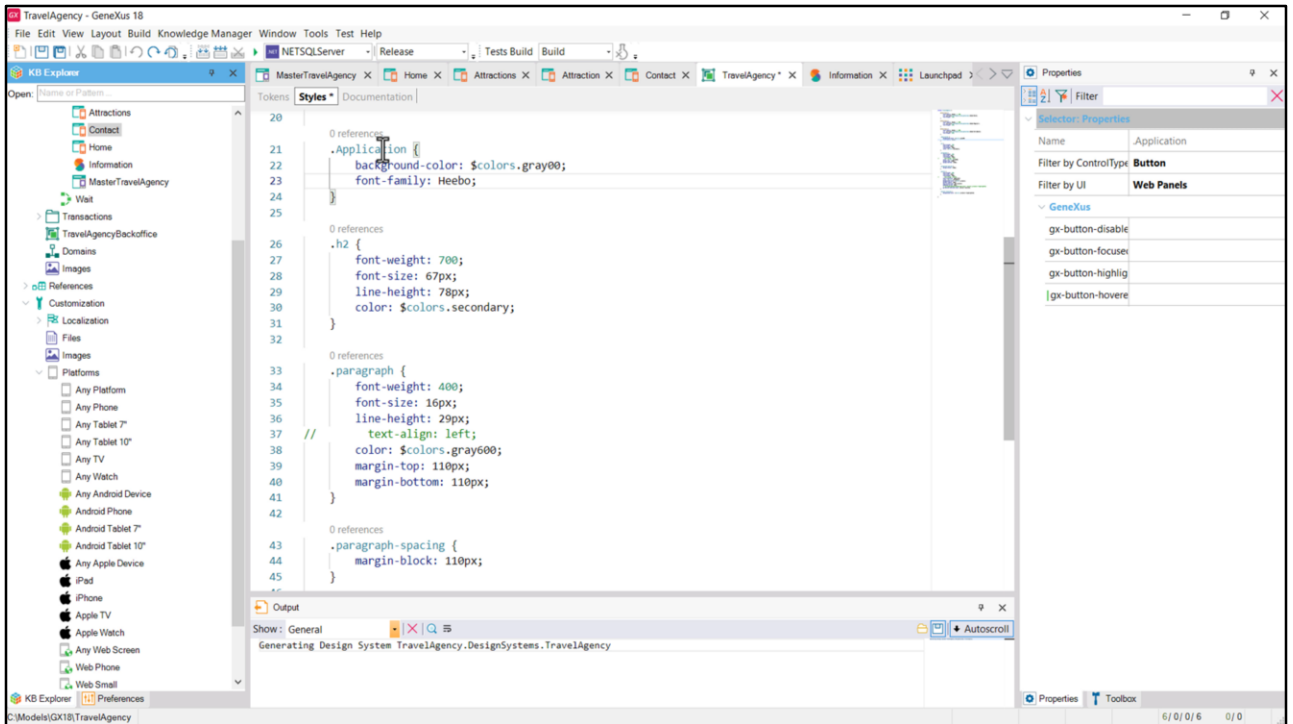
I remove here the name that distinguished it, leave the generic Heebo... which is the one I also place here...

I will remove from all of them what differentiated them by name. Now all three will be called Heebo, and what will make them different will be their weight.

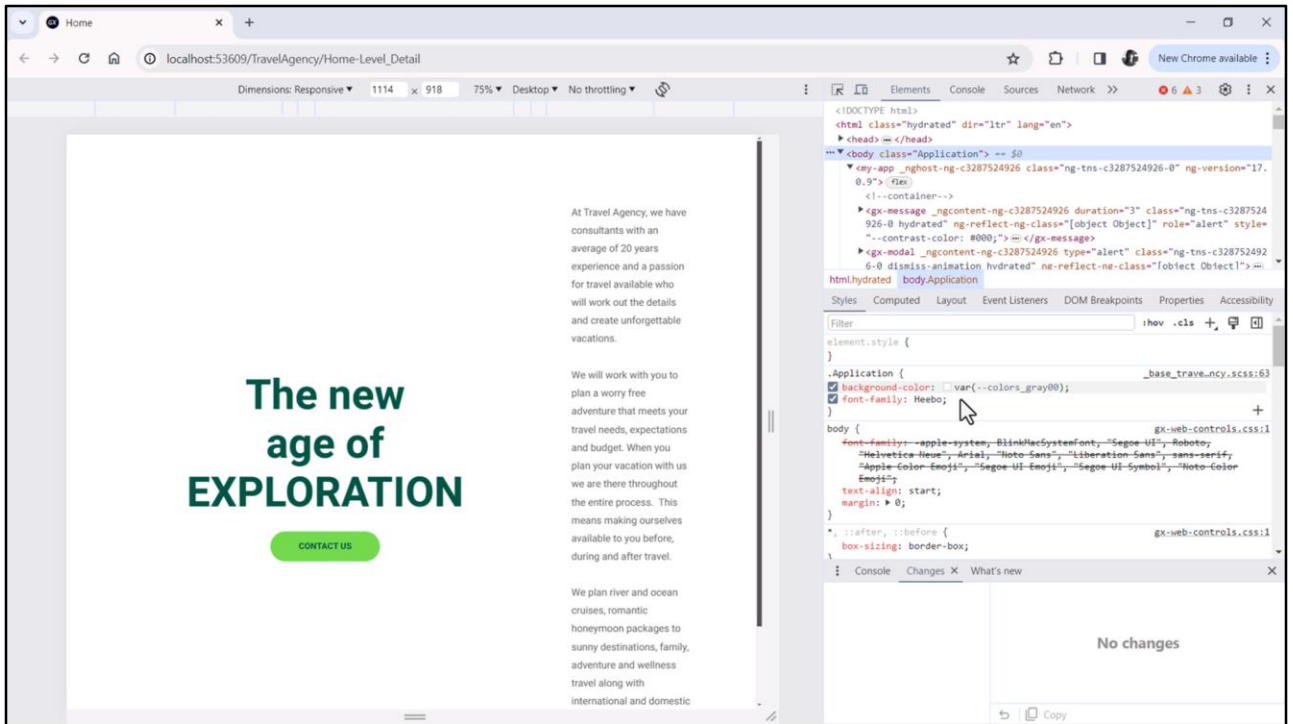
So in class h2 I say it is the 700 weight Heebo; in paragraph I will say it is the Heebo too, but with 400 weight; and in button we will say it is the 800 weight, Heebo too.

So to identify it now we will specify these two dimensions, because the difference between the Heebo fonts will be determined by weight.

With these two dimensions we know which of the font files will have to be used.

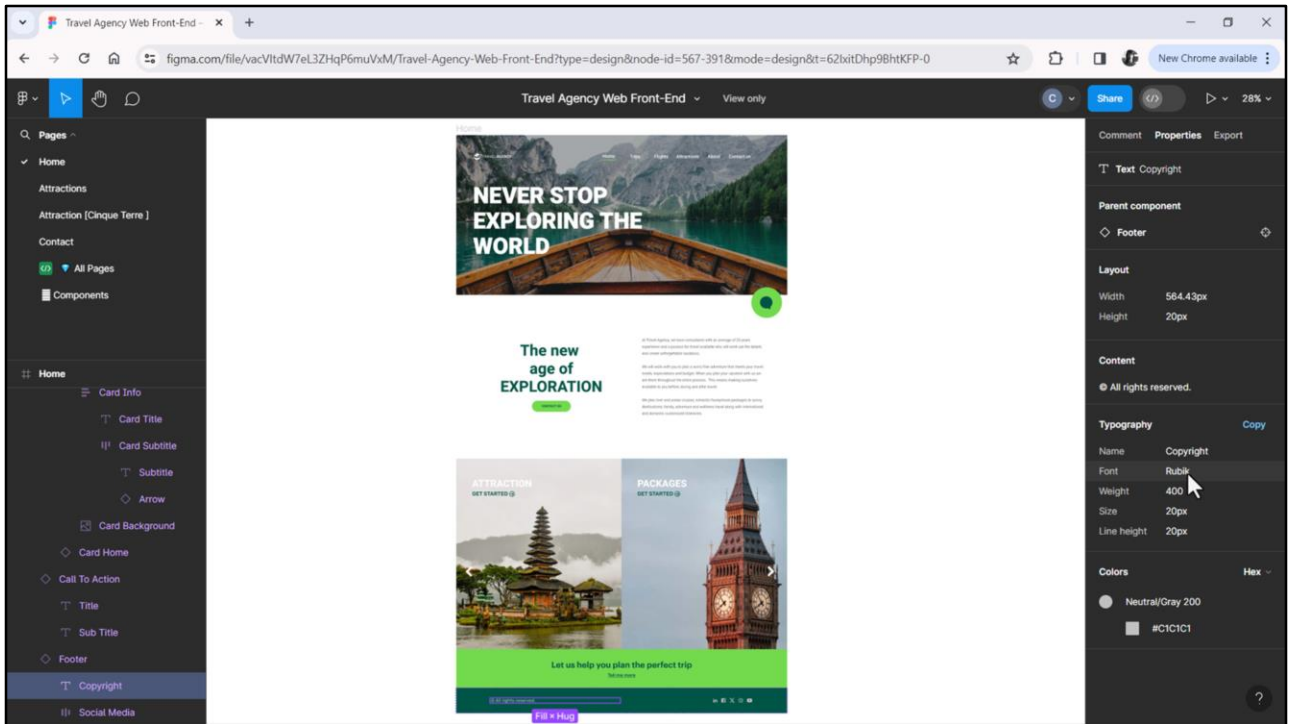


Then, since the font family is repeated in all the classes that we have so far, I could take the opportunity to specify that font family as the default, and remove it from here, here and here... and place it inside the Application class. This way, as this class is the root of each object with an interface, this will be the default font family; that is to say, the one that will apply if nothing else is specified in the classes.

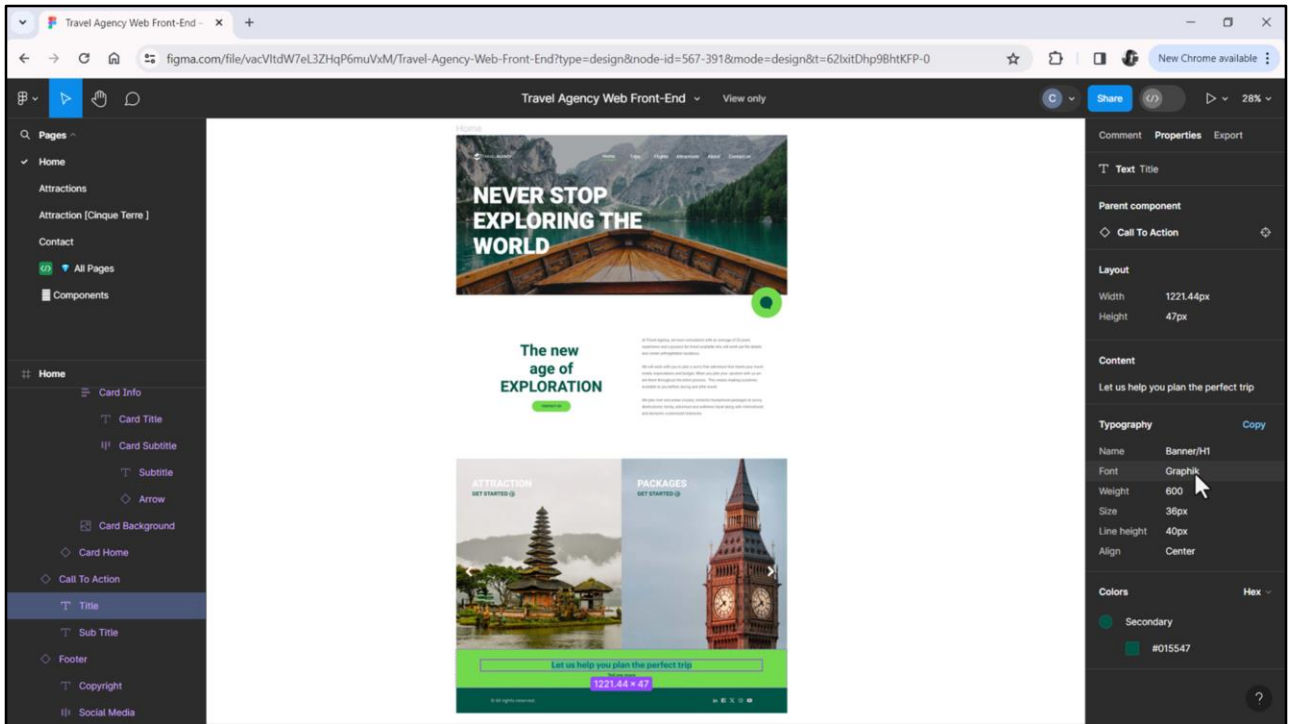


If we try this now... we'll see the white background again, according to this token, but otherwise, we won't notice the difference, obviously.

But if we inspect it... we see there the two properties inside the Application class, and we don't notice any difference from how we had implemented the font aspects before. So it's another way of accomplishing the same thing.

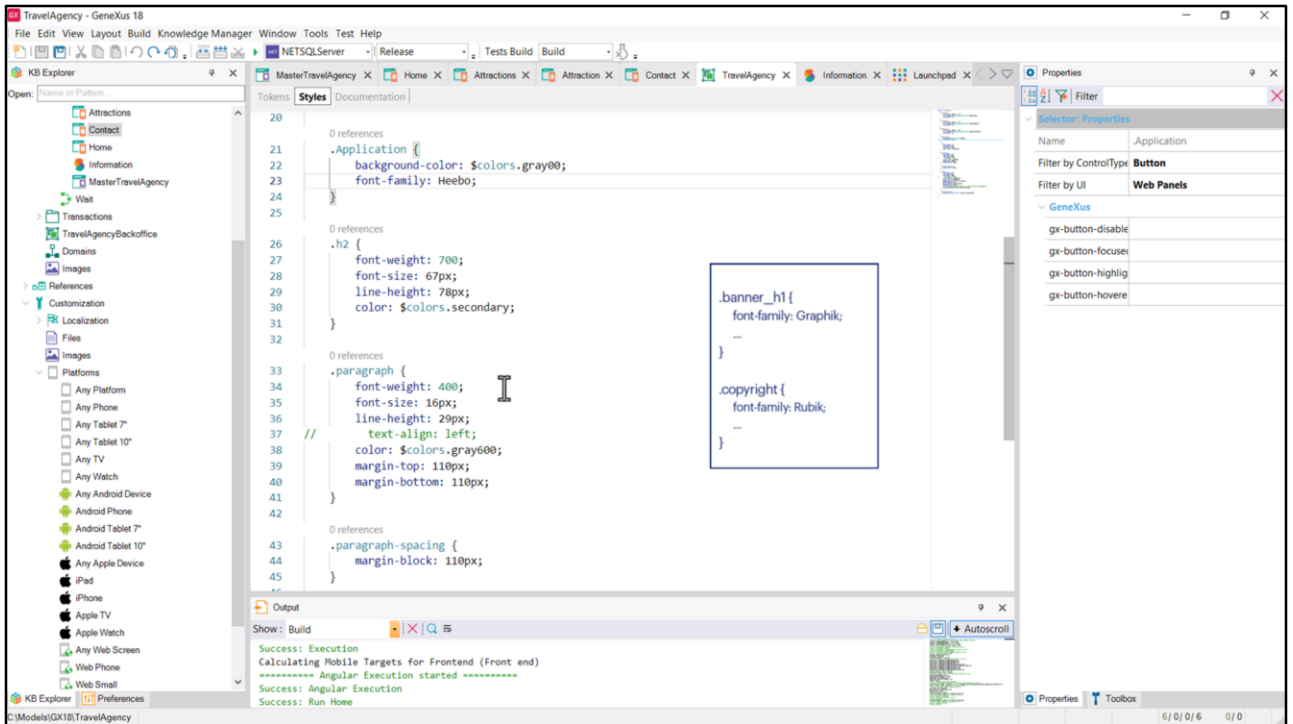


On the other hand, if we come to Figma to explore the typographic styles that our designer created for the screens, and we analyze, for example, the copyright one, we see that it's not using the Heebo font but this Rubik one.



And for this other text, the main text of the Banner, the Graphik font is used.

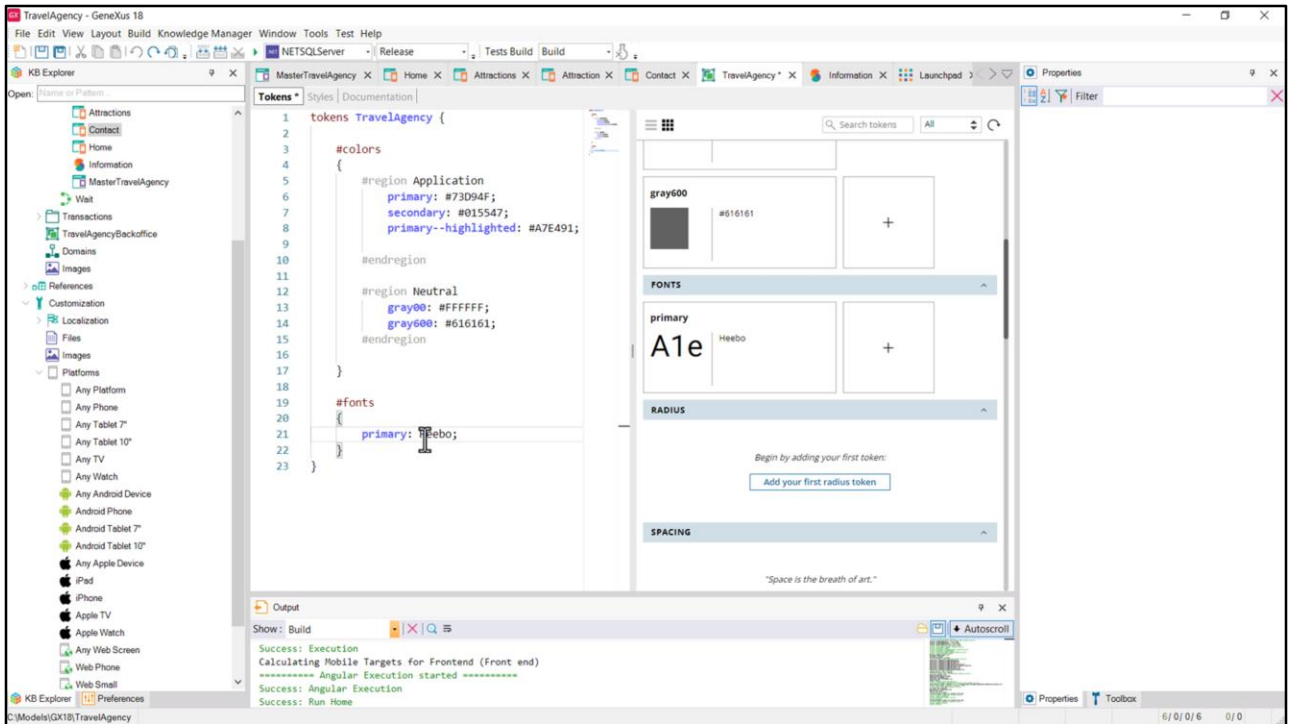
We can conclude that Heebo will not be the only font family. There will be two others.



What we know is that the Heebo font will be the main font of the application, the primary one, and that only in a couple of exceptions other font families will be used.

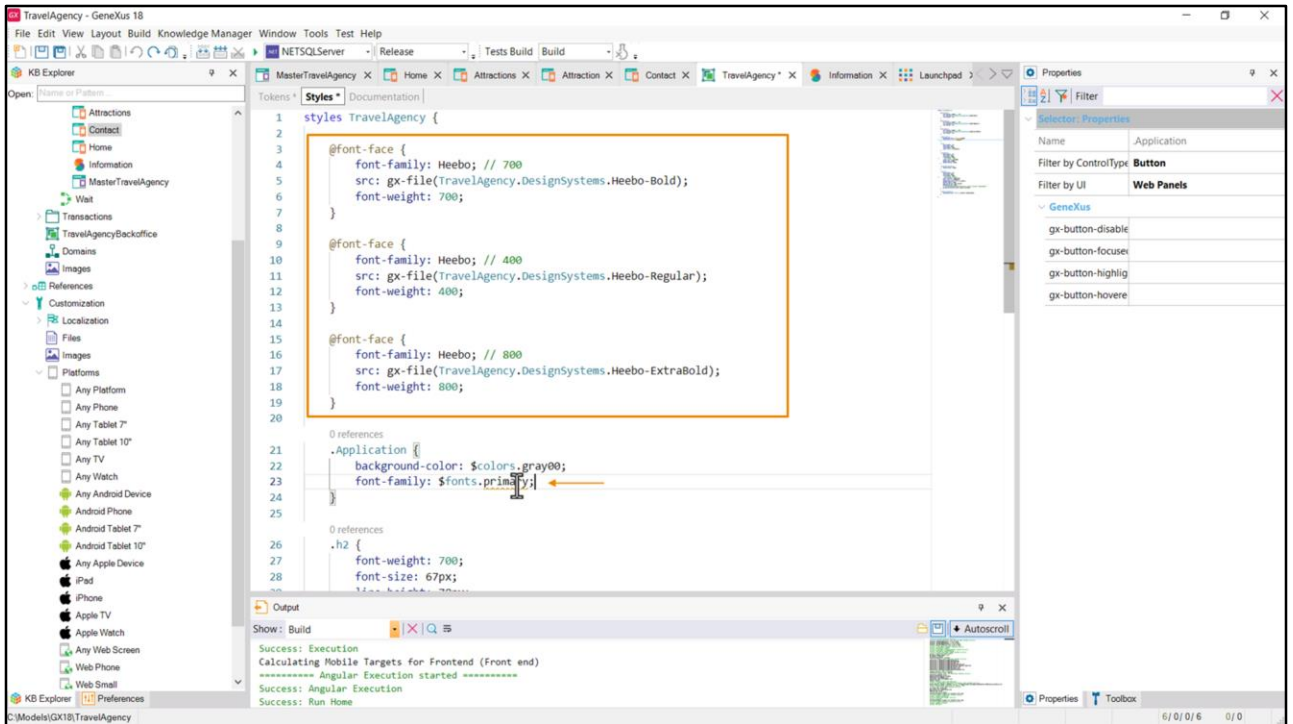
For the classes that use these other families, those of the texts that we have just seen in Figma, it will be enough to specify the family. For one case it will be Graphik, for the other Rubik, and in this way they will stop using this Heebo default, which was the default since we placed it in the Application class.

What we can do to better conceptualize everything, given that we are going to have 3 different font families, and this one is the main or primary one, is to create a token for each one, to abstract them according to their function.



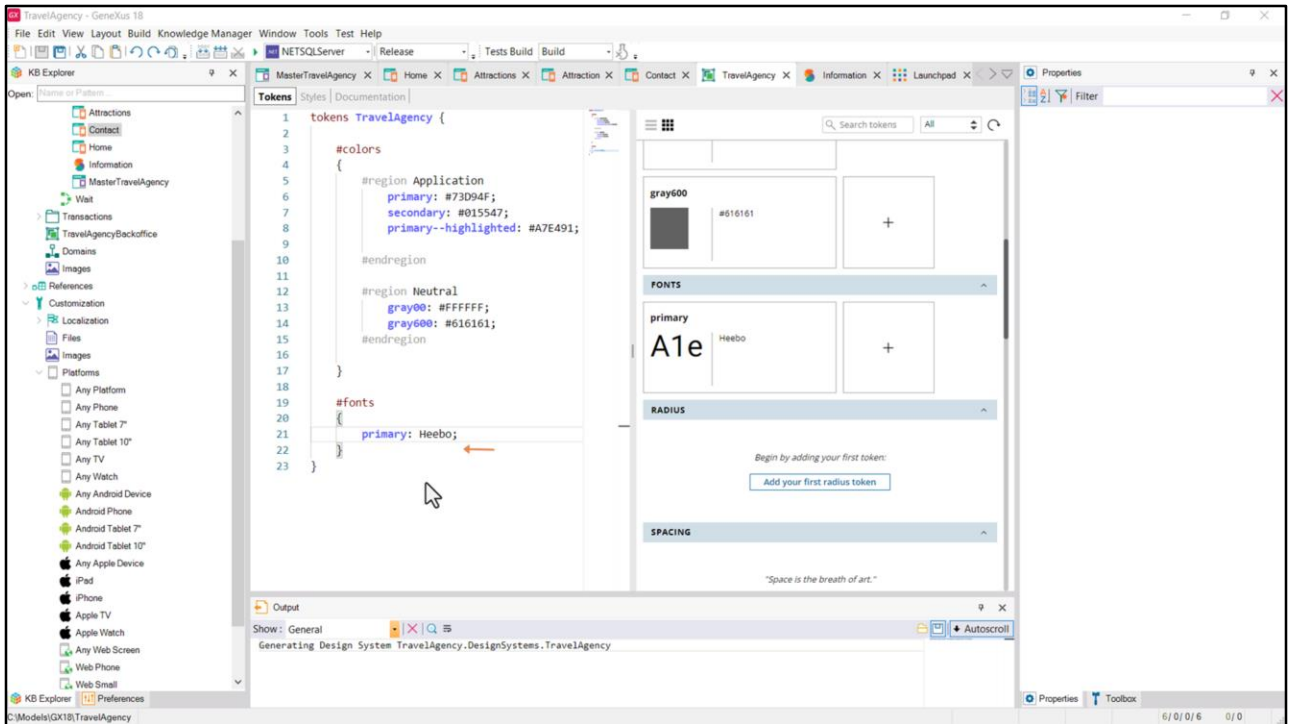
Tokens are not only color tokens. We can define tokens of other types. For example, for fonts.

From this editor we can add a new token of this type, fonts. I will call my default font primary, which is Heebo. This name here will correspond...

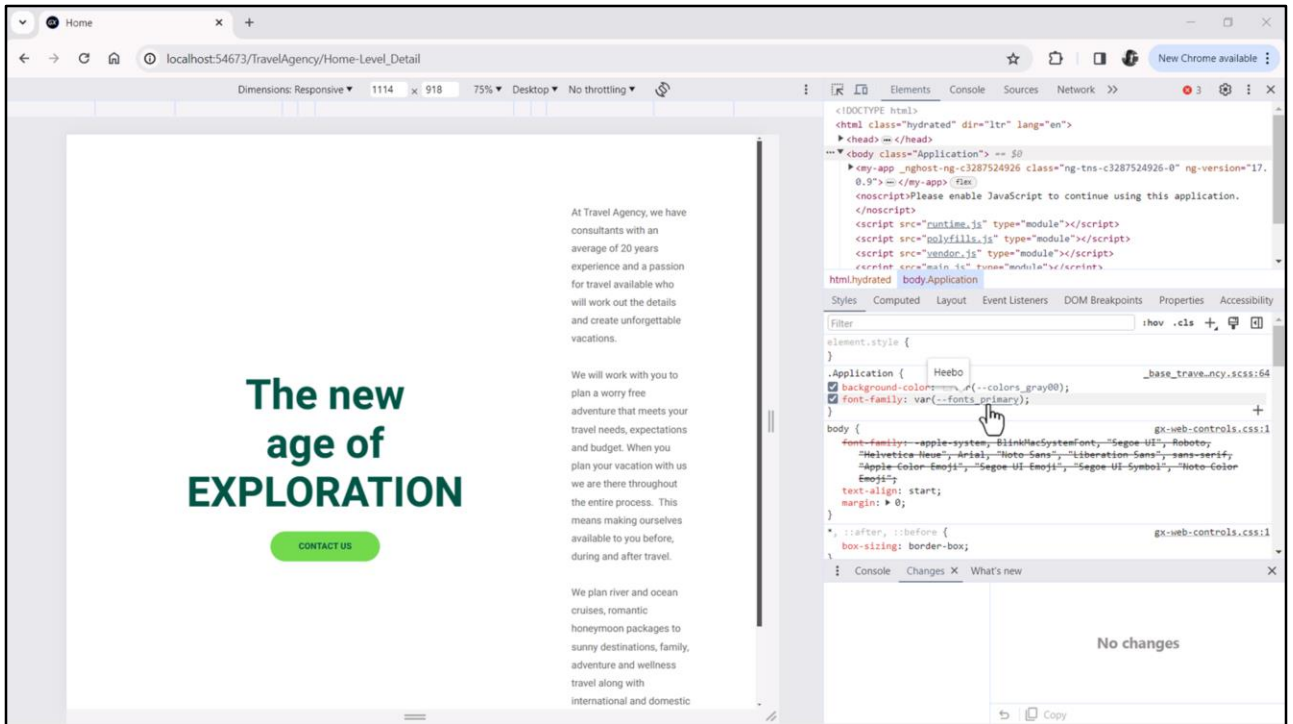


...to these three entries.

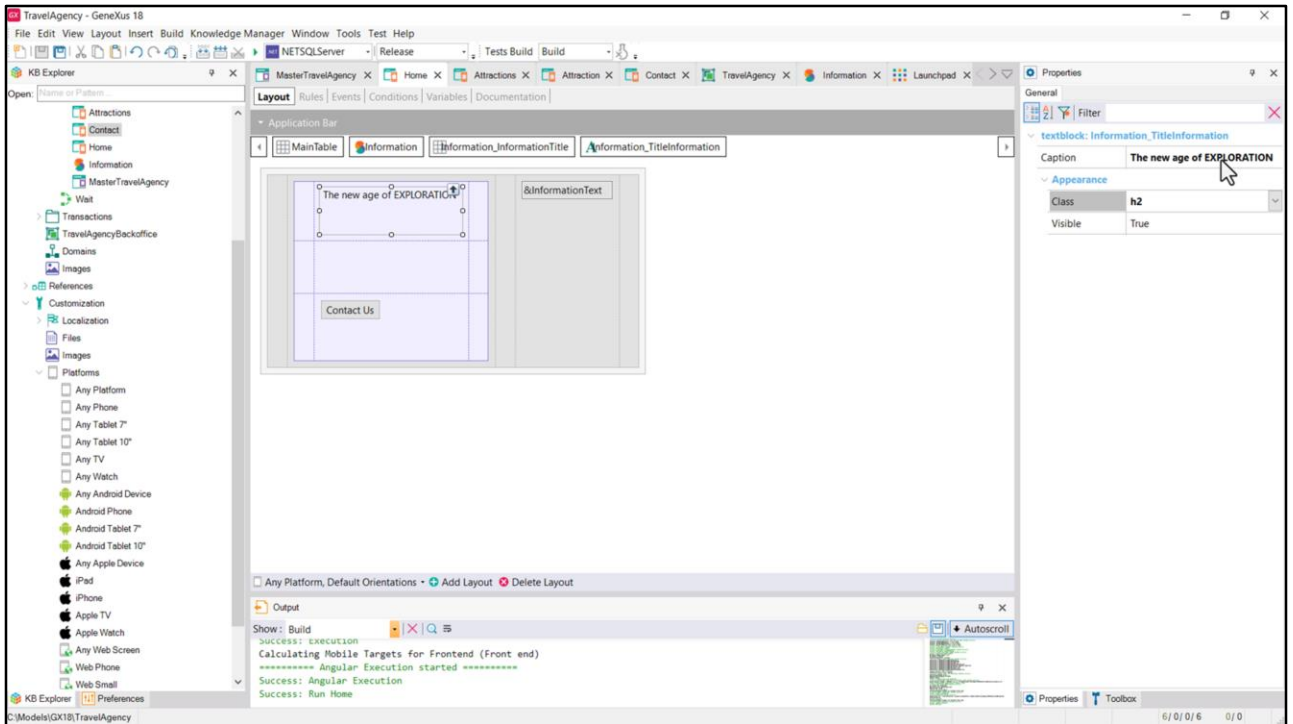
From now on in the classes I can use the token instead of the specific value. That is to say, the default font family will be the one indicated in the primary token. Primary for fonts and not for colors. They are not confused because they are in different categories.



Let's save. Of course, when we need to use the other font families, it will be convenient to create tokens here for them. In this way, it will be quite easy to change the font families of the application without having to go through the styles tab to modify in each class the font that is being used. And this is the great advantage of tokenizing.

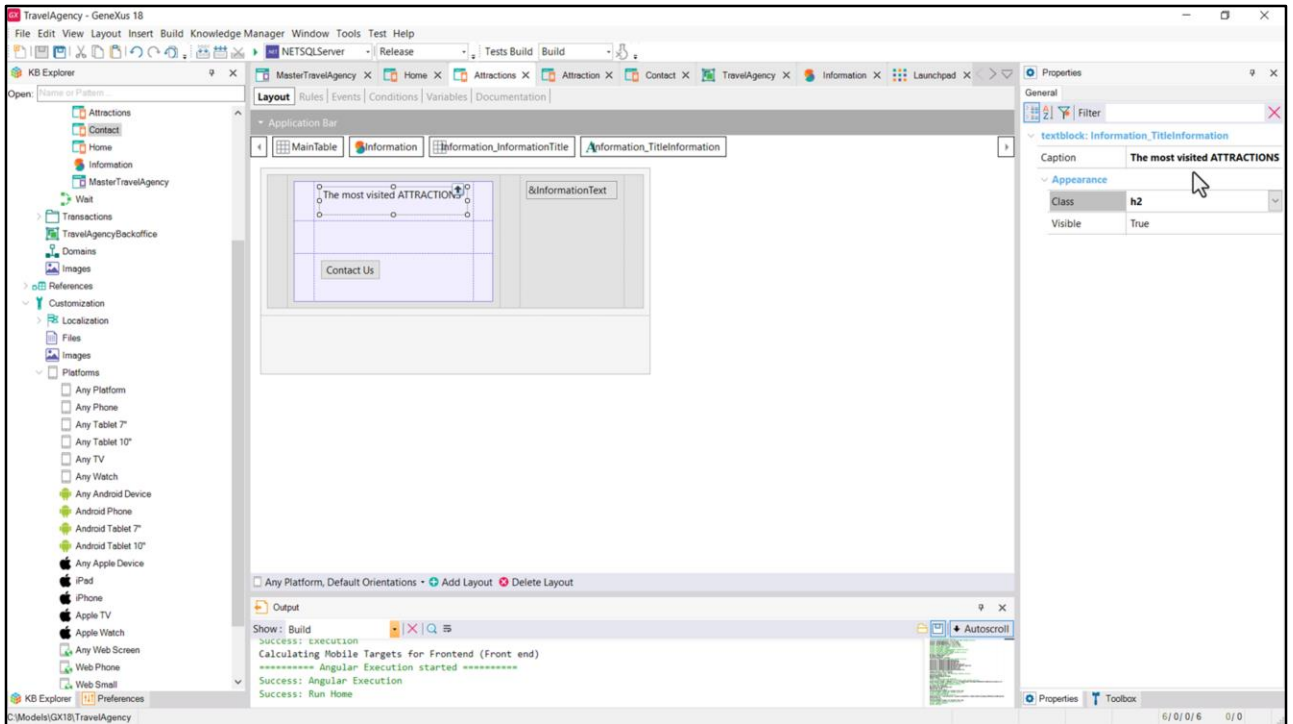


If we want to test it to make sure that everything goes well.... It seems that yes, everything is fine. Note here in the Application class how the token of the font family looks like.



Well, now I would like to discuss something else that was overlooked. And it has to do with the Caption that we assigned to this text block control, but in the Home panel.

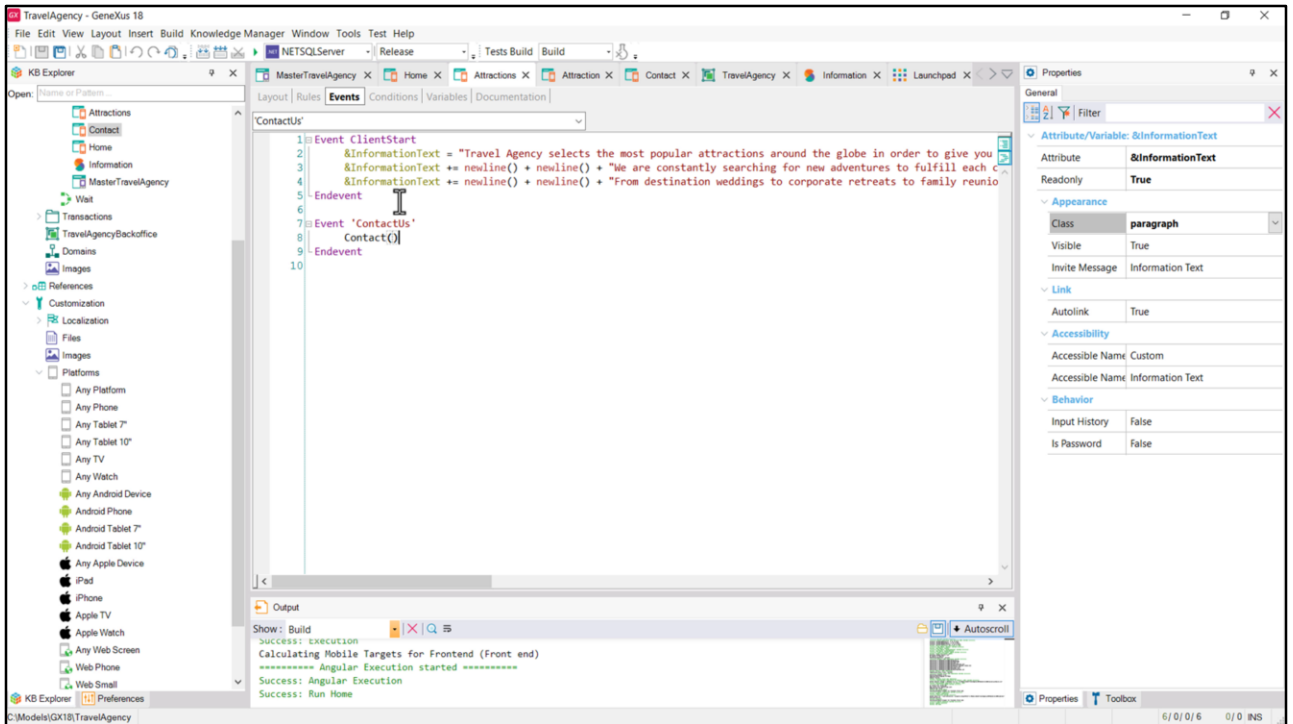
We had started by statically assigning this caption to the textblock instantiated in the Home panel...



...and to the textblock instance in Attractions we had also statically assigned this caption at the control property level.

Since this is static information configured in the properties of the control itself, when the panel is loaded for the first time in the browser, this information is already available and can be rendered immediately on the browser screen.

Then, also on the client, the ClientStart event will occur, and if it doesn't have to invoke any service on the server, it is entirely executed there.



The ClientStart event is where we load the content of this variable, remember? Let's look how we assigned a value to it in the Attractions panel.

For the variable we had no other option but to load it by code, because for variables we don't have a property that allows us to enter content.

In this case, as its content was also static, known from the beginning, these assignments will be executed immediately, in the client. Therefore, when we load the page in the browser.... we will see that the text caption and the variable text are loaded at the same time, without any flicker or delay or anything like that.

GeneXus for Angular Course

training.genexus.com/en/learning/courses/genexus/v18/angular/material/events-in-a-panel-6105516

GeneXus DL Portal Issues

GeneXus training Learning Certifications Universities Academic Partners Help Login Try GeneXus

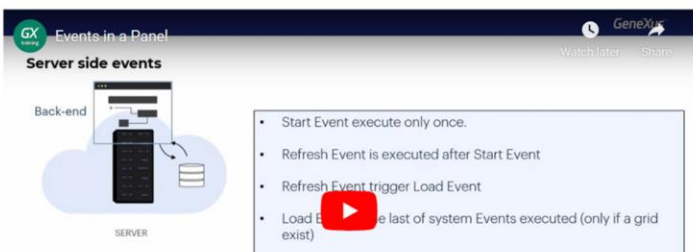
GeneXus for Angular course

Version: GeneXus 18

Events in a Panel

Whether it is a web application generated in Angular, or an application generated for mobile devices in Android or iOS, we use panel objects to implement it. For that reason, the events available in this object are mostly valid for both platforms. Knowing how the events of this object work, both those that are executed on the client and on the server, is essential to program the behavior of our application.

Total length of videos: 5.5h



The screenshot shows a video player interface. On the left, there is a diagram titled "Server side events" showing a "Back-end" server connected to a "SERVER" cloud. On the right, there is a list of events:

- Start Event execute only once.
- Refresh Event is executed after Start Event
- Refresh Event trigger Load Event
- Load Event is the last of system Events executed (only if a grid exist)
- Grid with Base Table Load is executed as many times as

On the far right, there is a table of contents for the video:

Introduction

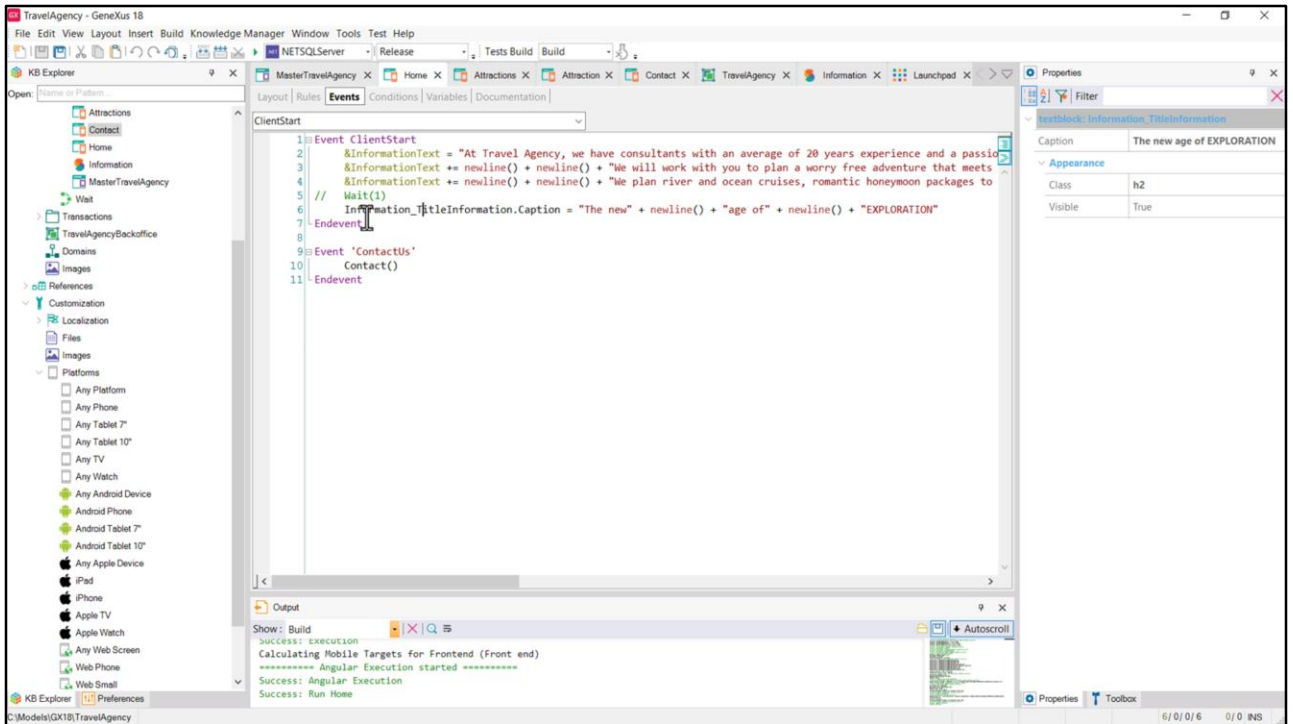
- Course Introduction
- Architecture of an Angular application
- First steps with Angular

Logic and behavior

- Data loading logic on a Panel screen
- Determination of base tables in a Panel
- Events in a Panel
- Panel with multiple grids

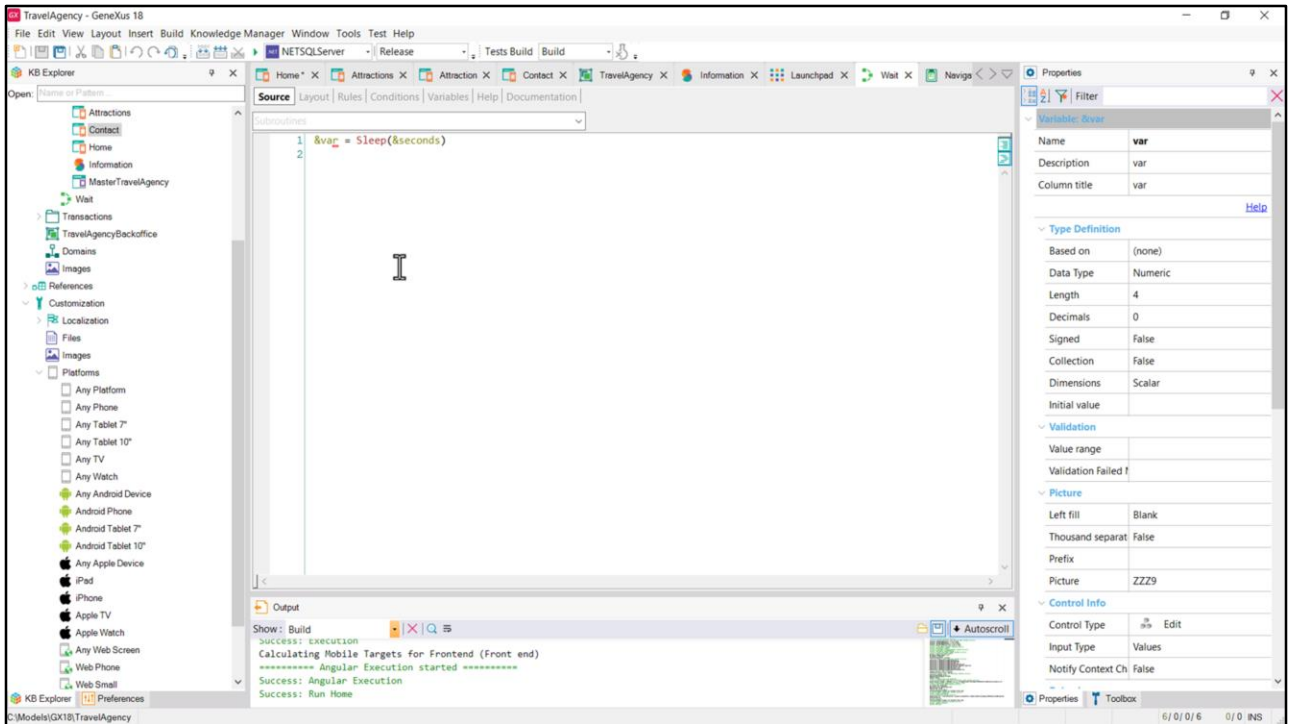
Before moving on I recommend you to review – or watch for the first time if you haven't done it before – this video of the Angular course on the GeneXus training site (it also explains the events for native applications).

The events that are triggered on the client and those that are triggered on the server and their order are explained there.



Going back to our work, for the text block in the Home panel we had to assign the caption by code, to be able to divide it in 3 lines, do you remember? Here I only changed the order of the assignments (first came the textblock one and then those of the variable but now I inverted them)... and I placed this, that since it is commented for now it doesn't matter.

When we implemented this we didn't take into account, once the assignment was added here, to remove the caption set at the level of the control in the property. So, we have two caption assignments.

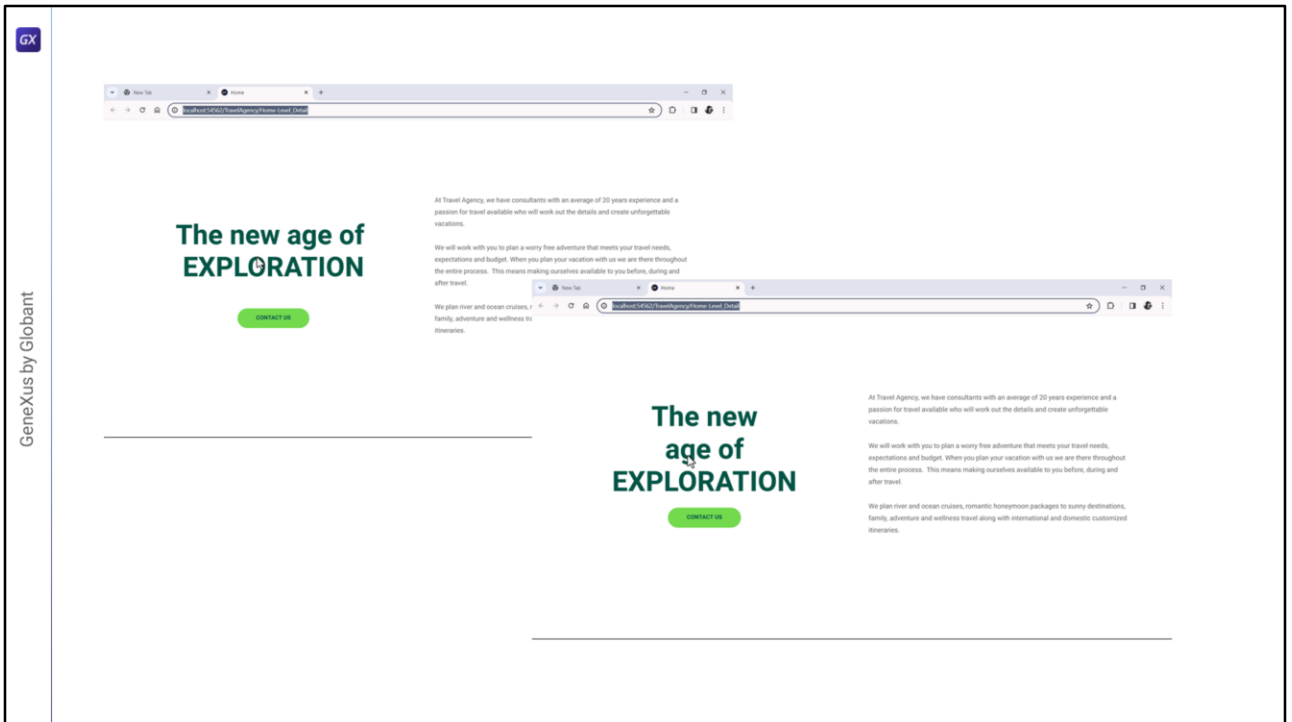


To clarify the effect that this can have, I added (I uncomment it) this invocation to a procedure after loading the variable information but before loading the caption of the textblock.

This invocation will be made to a procedure that I created earlier, and because it is a procedure it will be executed in the server.

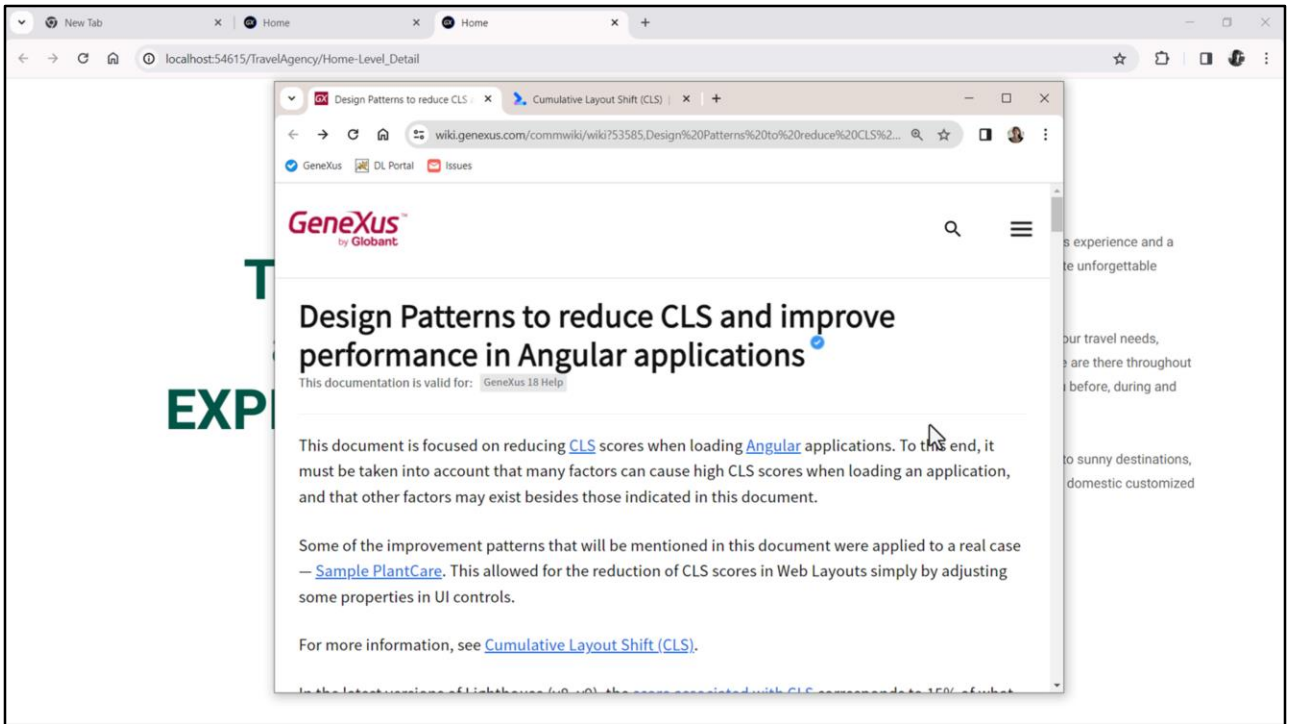
So, this server procedure will have to be executed **as a service** from the client. And the only thing that this procedure does is to call the sleep function, which implements a pause – in this case, of the number of seconds that we send it by parameter, which will be 1. When this time has expired, the procedure ends and the execution continues in the client, in the following statement, executing, then, the assignment of the caption.

To understand what I want to show you, it will be best to execute it.

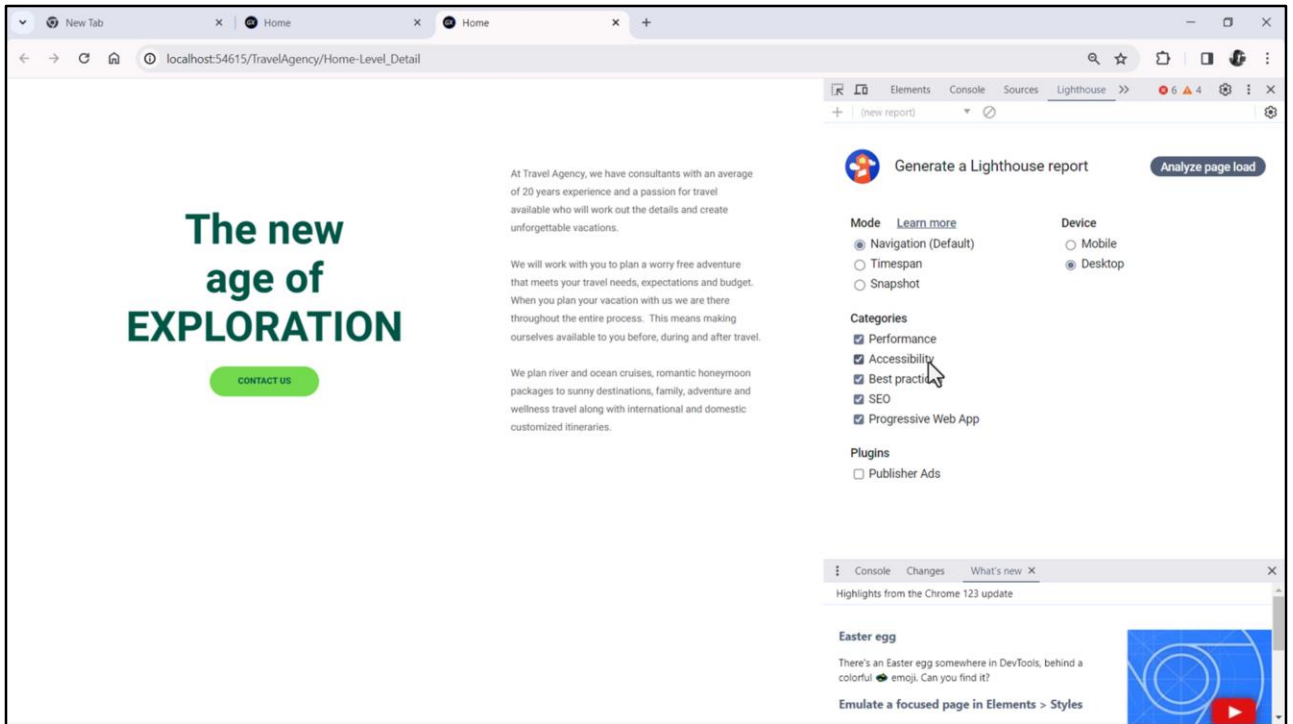


It was clearly seen, wasn't it? If the application behaved like that, with that unexpected change in the layout, the user would surely feel uncomfortable and our application's rating would go down.

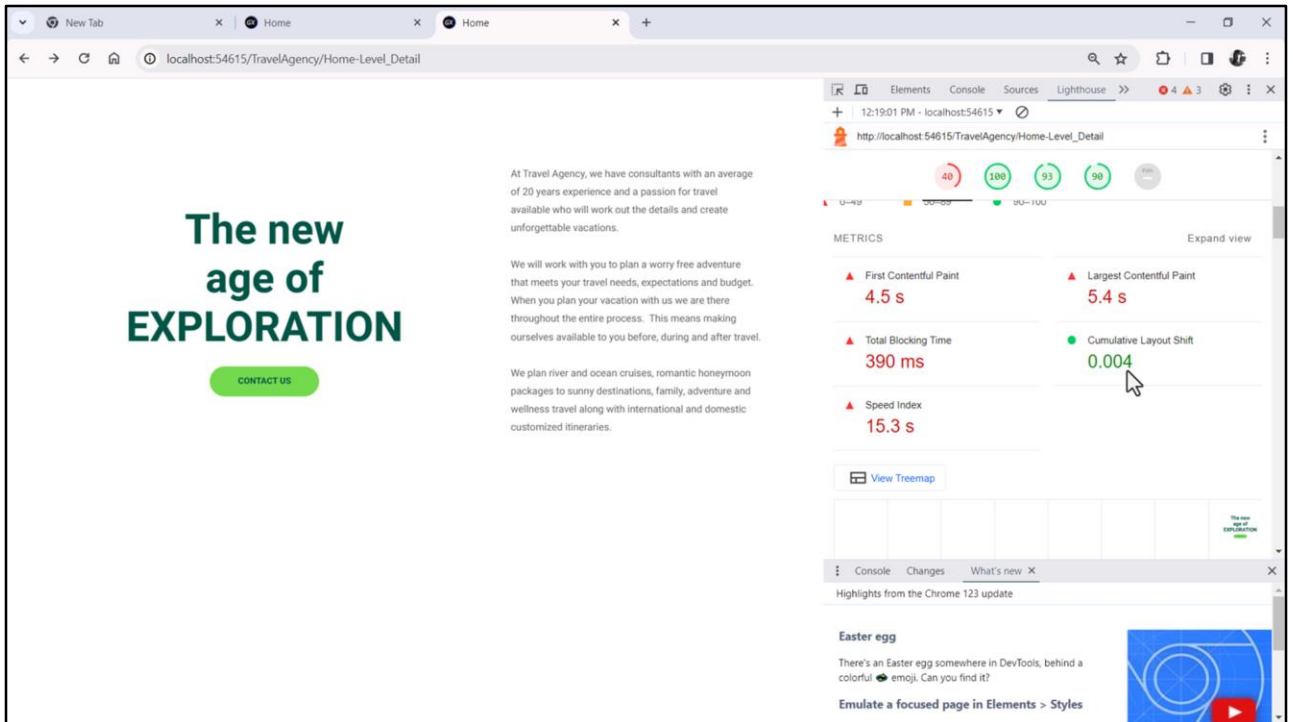
It is not the same if the caption takes a long time to appear than if it changes. The latter is what gives a very unfavorable impression. I placed this procedure so that the change would be obvious to us. I'm going to remove it... and run it again... it was unnoticeable in this run, but we can't be sure that it will always be that way. Also, we can't be sure that in the future we won't add code in the ClientStart event that delays its execution and that therefore causes this undesired effect.



Measuring this type of experience has become so important that the CLS (Cumulative Layout Shift) metric is used to measure these unexpected changes in a page. In our wiki you will find information on how to reduce CLS. The closer this metric is to 0, the better our application will score.



Actually, among the DevTools (I will only show the web, not devices, and zoom out) we have Google's tool, Lighthouse, a performance and audit tool used to improve the overall quality of web pages by evaluating aspects such as performance, accessibility, use of web development best practices, search engine optimization, PWA. The CLS metric will be used for performance evaluation.

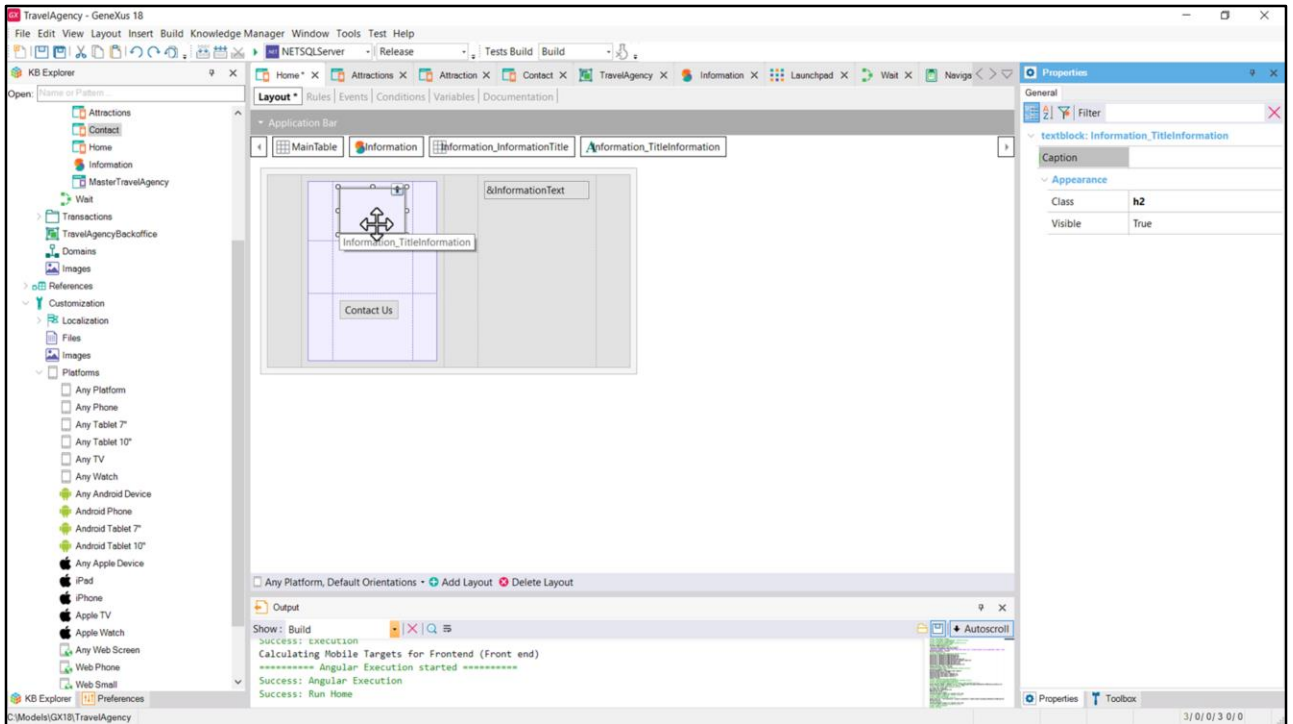


We are going to ask it, then, to analyze the loading of our page for Desktop size... here I accelerated it in the video, because it took 10 times more. We see that it suggests running the same in an incognito window.

You can test it later.

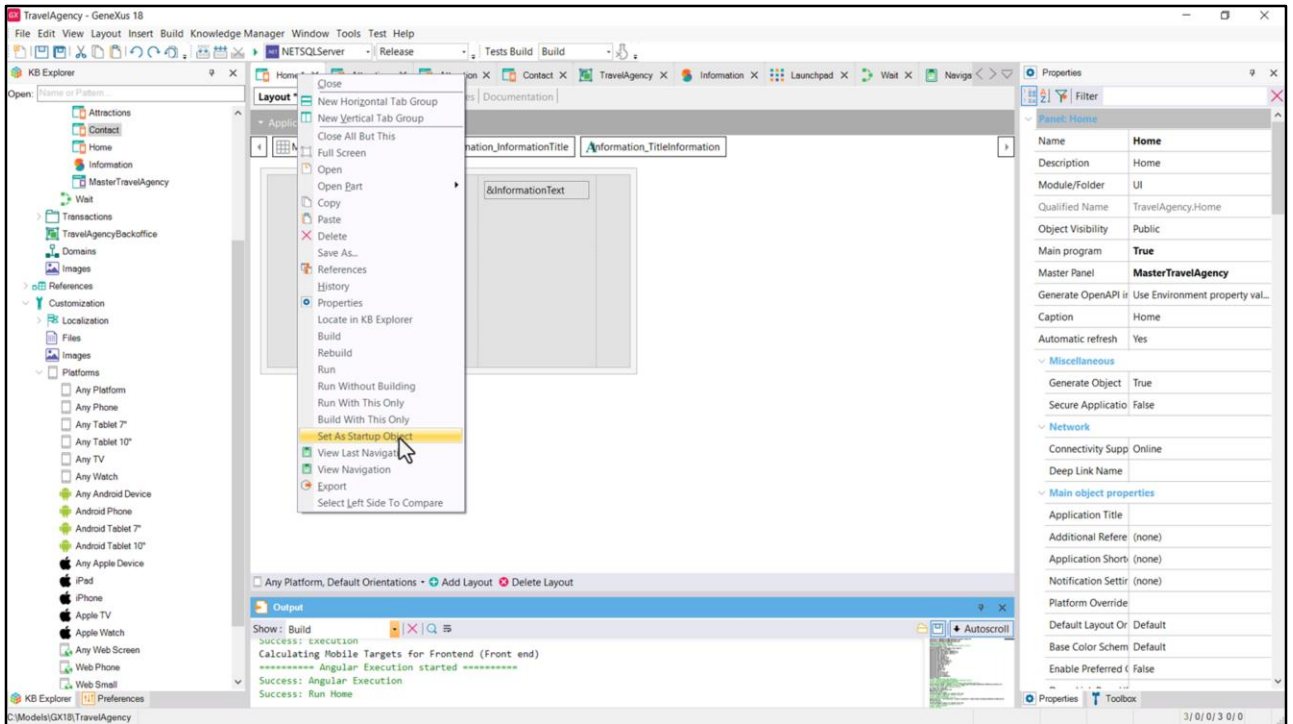
Now I want to show you that it gives measurements in the 5 dimensions, of which Performance is highly relevant. If we go to the disaggregated metrics, we see CLS, which is not 0, and that probably has something to do with this unwanted layout change we have been talking about.

It may seem that it is the least important to take care of, given that all these performance-related metrics are shown in red. However, it is normal that now they are shown like this. At the time of deployment to production we should change one property in the KB preferences, the Build Mode, changing it to Distribution, and this alone will improve this performance metric by at least 30 points. We don't do it in the development stage because it greatly increases times. On the other hand, we must address the CLS metric from the beginning.



What would we have to do to reduce CLS? Remove the caption assignment from here. Leave it empty. What is the problem with leaving it empty? This one. That the presence of the control in the layout is invisible to us, unless we click on it.

It is convenient to consider all these matters from the beginning, to avoid this type of issues later and that's why I wanted to tell you about them in this video, before we go on.

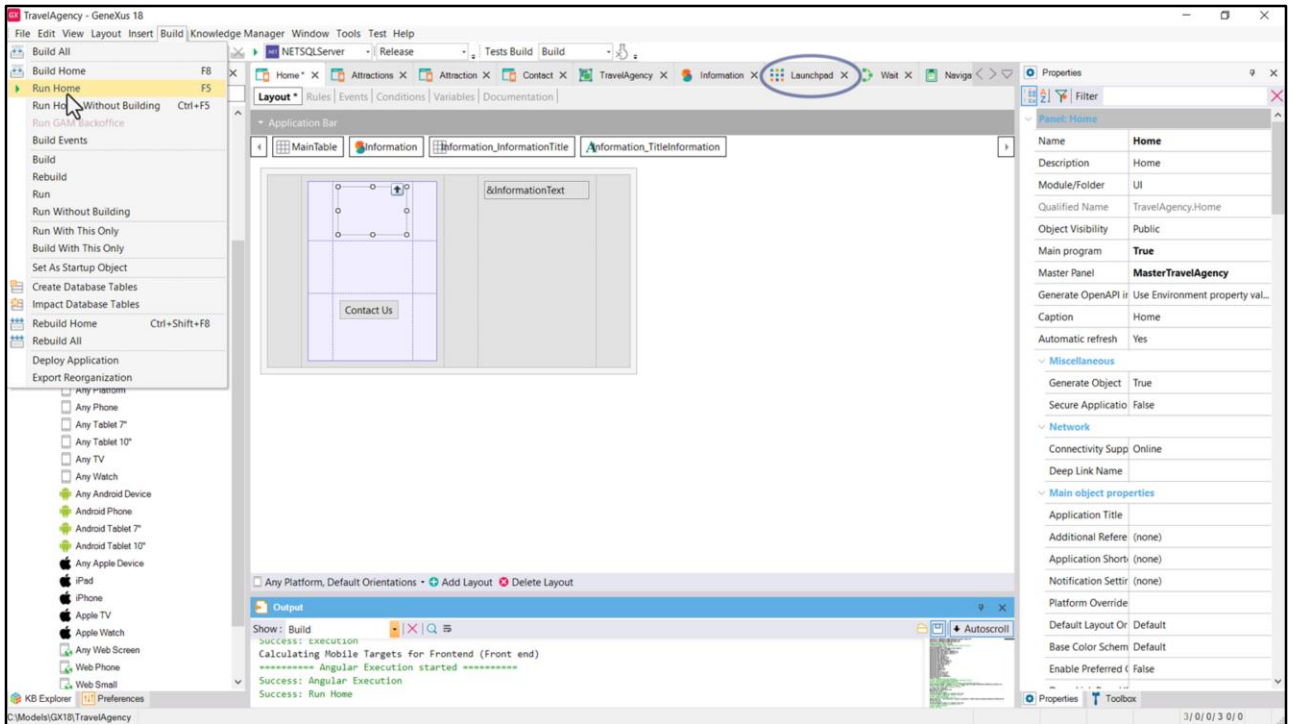


Finally, since I am in the middle of the recommendations section: several videos ago we mentioned that it is very important for us to optimize development times.

That's why we are prototyping locally. I will close, before moving on, the two server windows: the frontend server and the backend server.

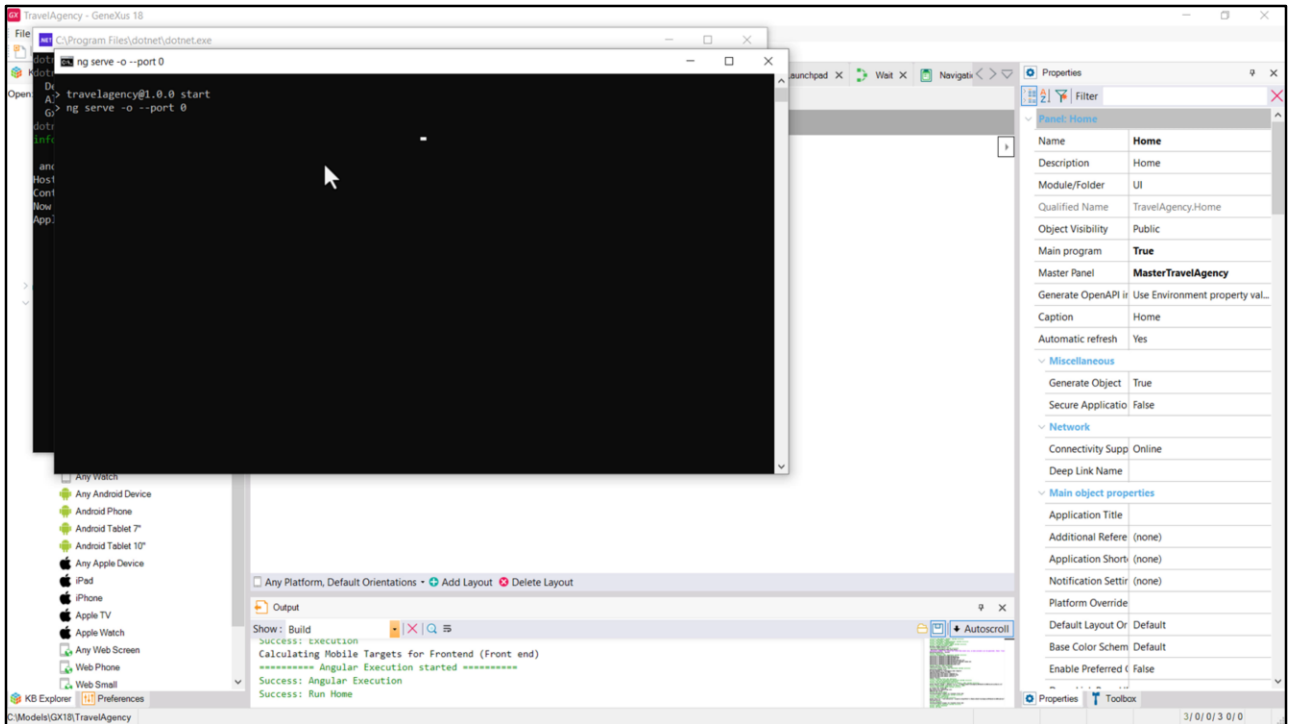
The entry point of our final application is going to be the Home object, and that's why we had configured it as main program and we can run it with the Run option.

Actually, it would have been reasonable to configure it as the startup object, which is what I will do, so that we don't need to select it and click on Run, but simply press F5 to run it.

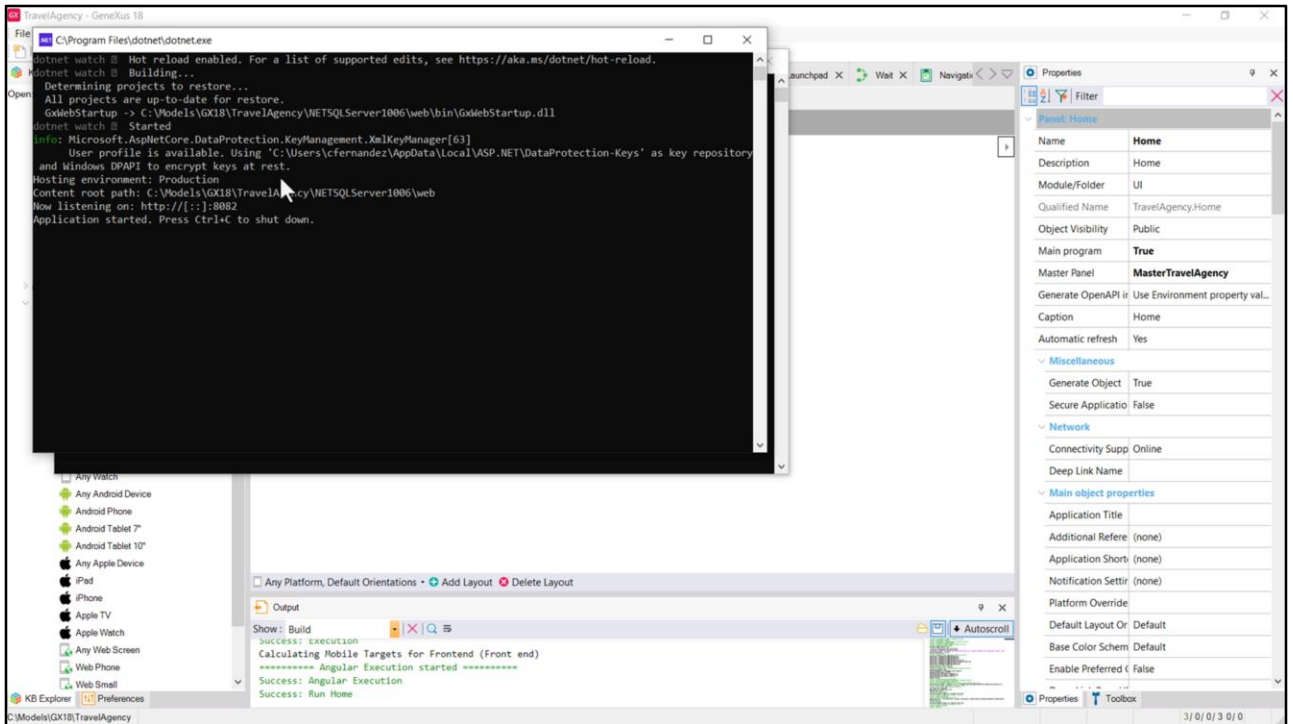


Note that now for F5, instead of offering the Run option of the Developer Menu that opens the launchpad, the Run option of the Home is displayed. Let's run it.

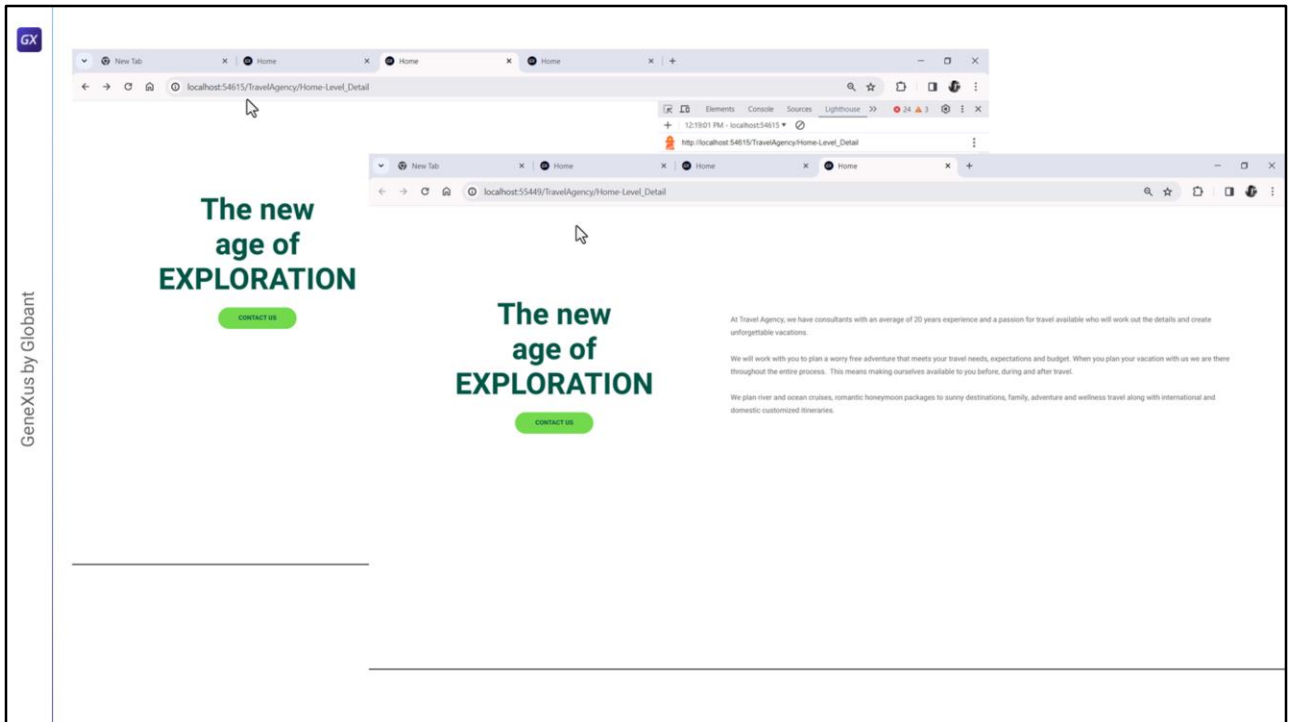
It will specify, generate, and compile the entire dependency tree that starts from the Home object... and then start the backend server, for the services, and the frontend server for the Angular application.



This is the one for the Angular application. Every time the server is shut down and restarted again that initialization takes time, and the application is also assigned a different port... this will be clear soon.

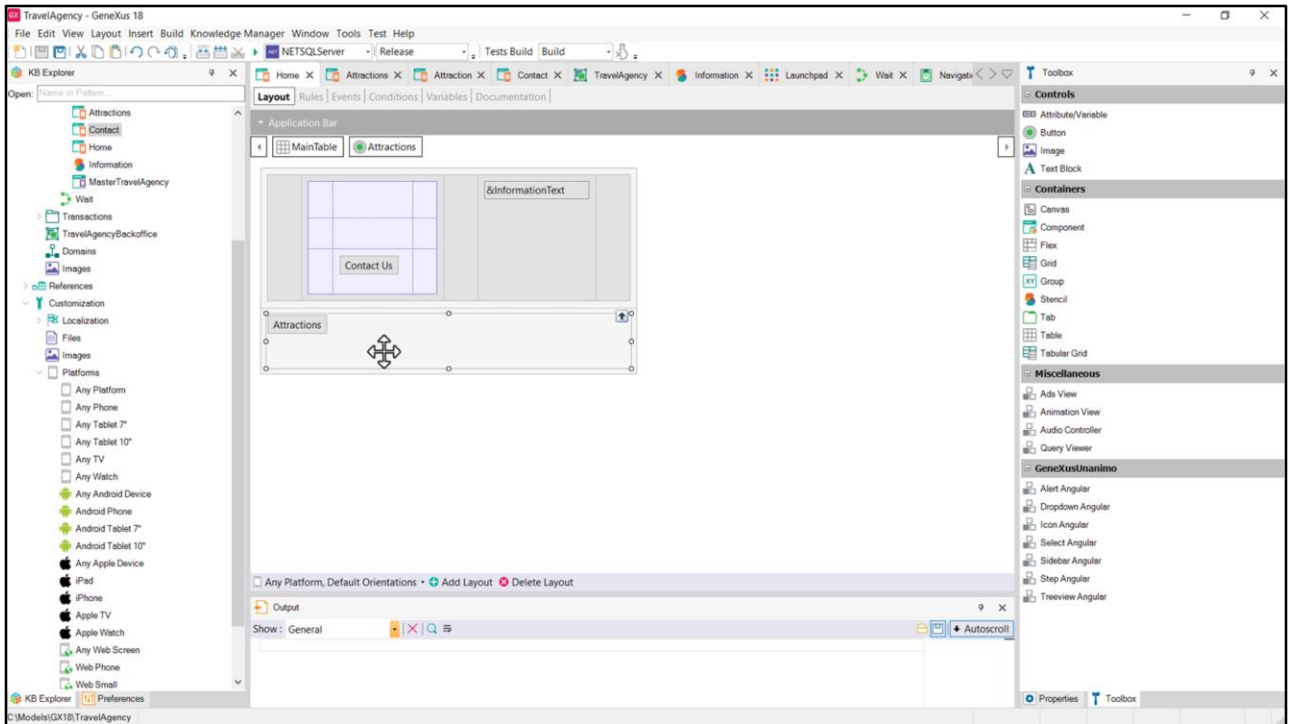


And this one here is the backend server, which is where the services are always located; for example, our Wait procedure, which is no longer invoked now but when it was, it was served from here.



And here we see what I was telling you, it opened in another tab because the port is different from the previous execution and that is why I closed the windows before, so that you could see this clearly.

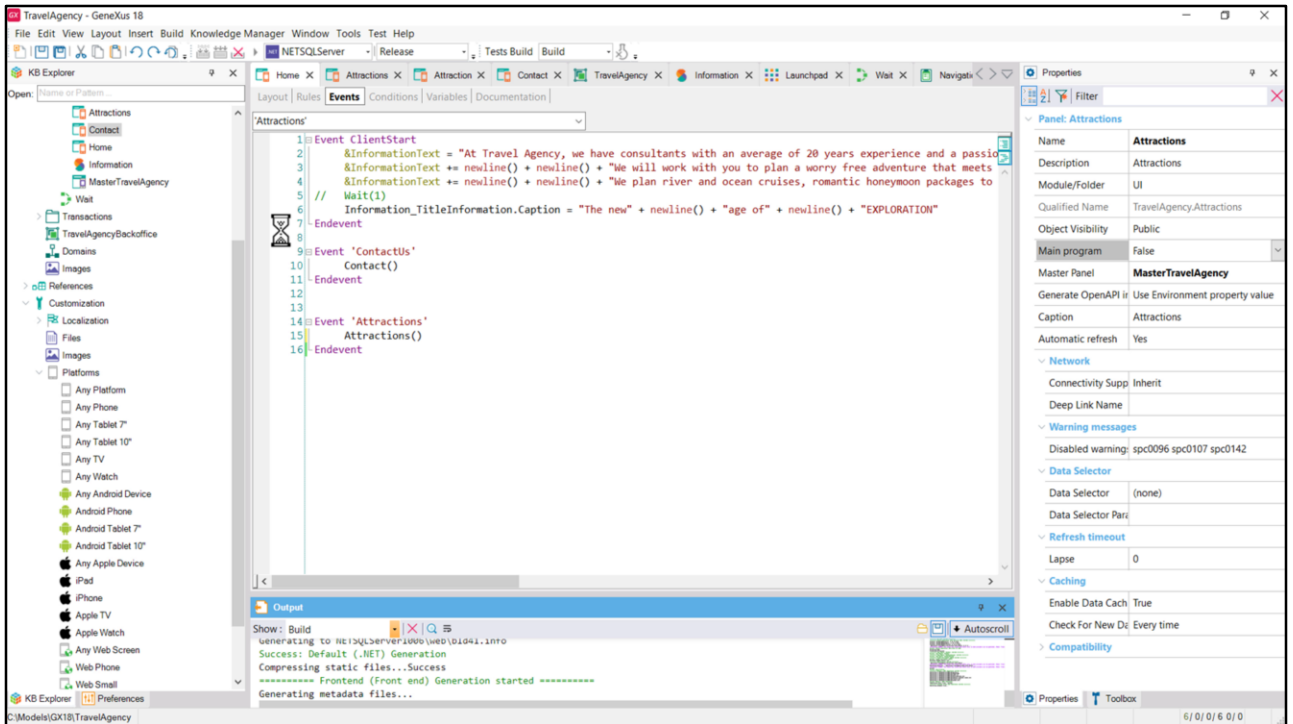
If we were prototyping in the cloud none of this would be obvious to us, but as a trade-off, the times would be longer, since you have to transfer the whole package every time to the cloud server.



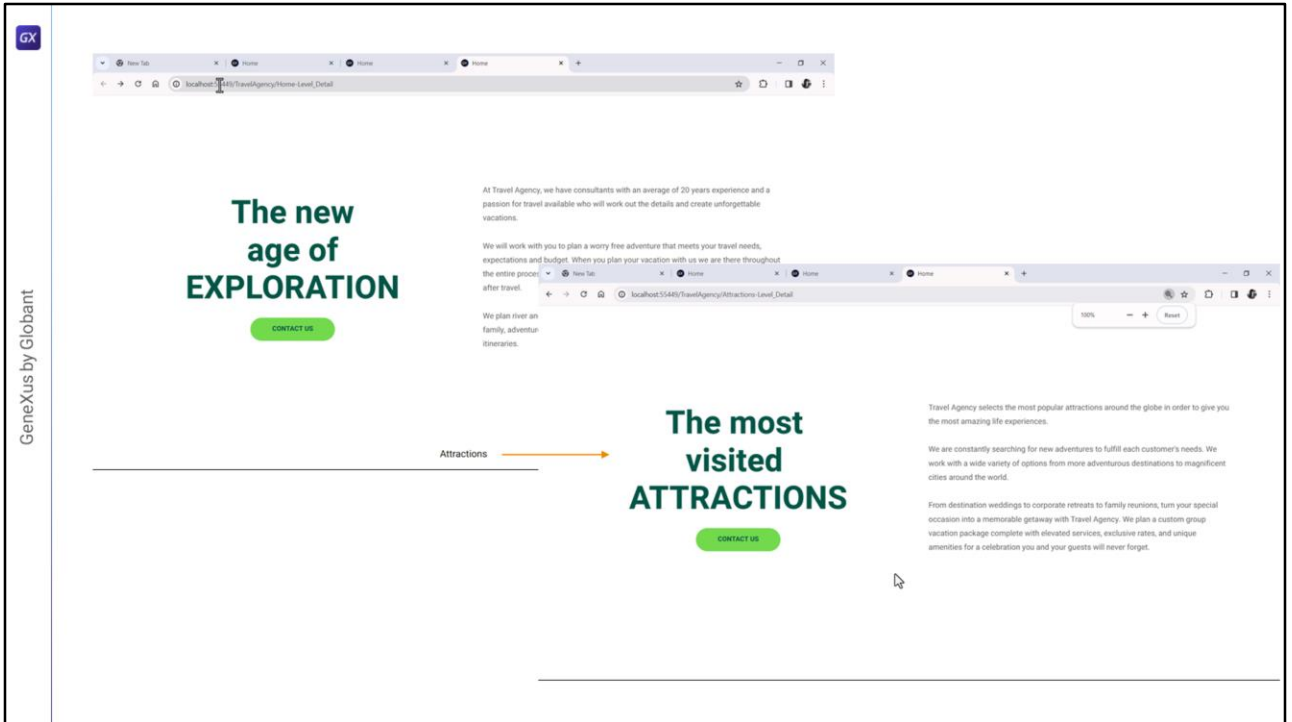
Well, and what is our problem now? That if we want to execute the Attractions panel, as we don't have access to it from the Home panel, the provisional solution that we had found before was to also declare it as main, that is to say, we are generating a second application. From there we can also Run the object, which will compile its entire dependency tree and will have to open another server on another port, so, before pressing Run we have to close the two servers, which will involve the cost of starting up again and installing the package, now Attractions, from scratch. All that equals lost time.

Instead, we now have the servers open for the Home application, so if we make a change to any object in its tree, for example this one... let's add a button with an associated Attractions event that I'll leave empty for now. Then I will call the Attractions panel from here; this way the panel is connected inside the application, even in a provisional way, for prototyping, to optimize our development times. When we implement the menu we will be able to remove this button. So it is convenient for us to be more efficient in terms of development times.

Now we run the Home. It opened in the same tab as before, on the same port and with the button.

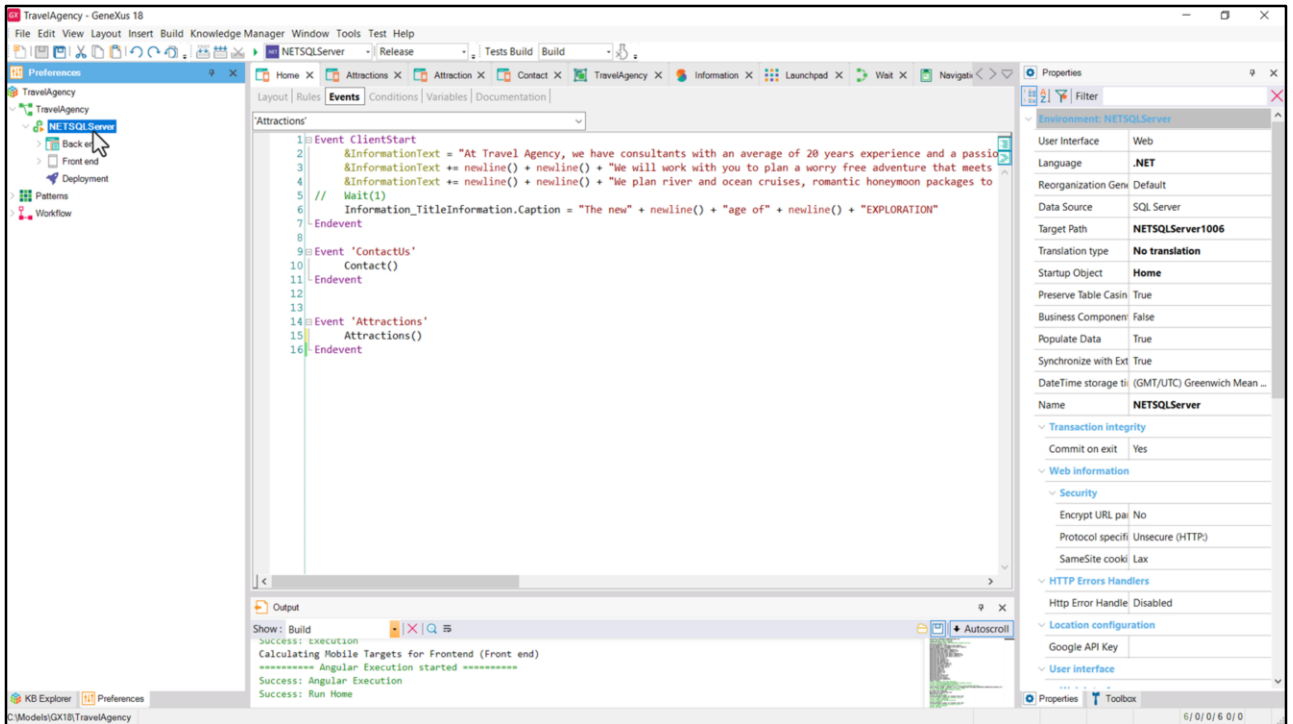


So the recommendation is that we temporarily include the button with the invocation to Attractions and reassign the default value to the Main Program property, which is False. We run it...



...and press the button. Here we have the Attractions panel. We see things this small because we had zoomed out. Let's bring it back to 100%. Good.

Note that we are still in the same port, with the same application.



Finally, remember that the startup object is configured at the level of the active Environment; you can see it here, in case you want to return to the previous situation that was “no startup object” and when you do it, in Build you will see that F5 is associated again with the Run option of the Developer Menu.

Well, that's the end of what I wanted us to see together in this video. I look forward to seeing you in the next one.

GX

GeneXus by Globant

GeneXus[™]
by Globant

training.genexus.com