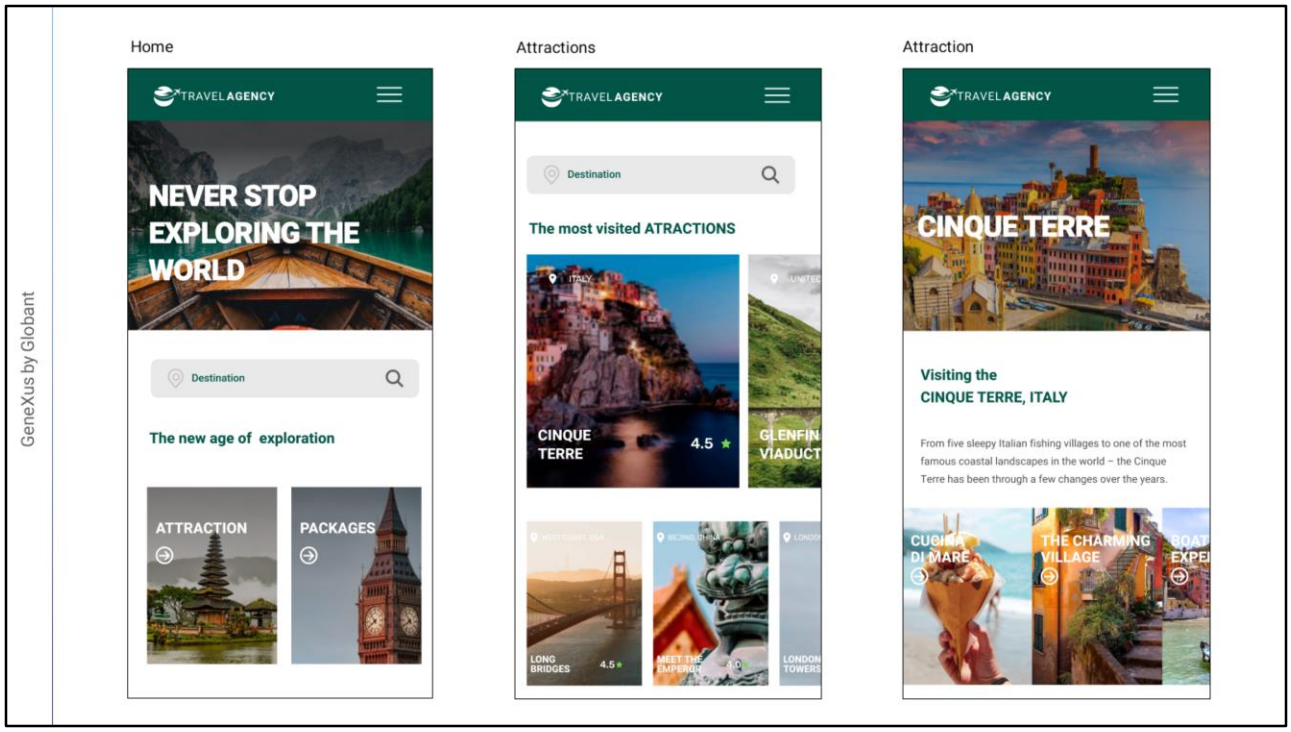
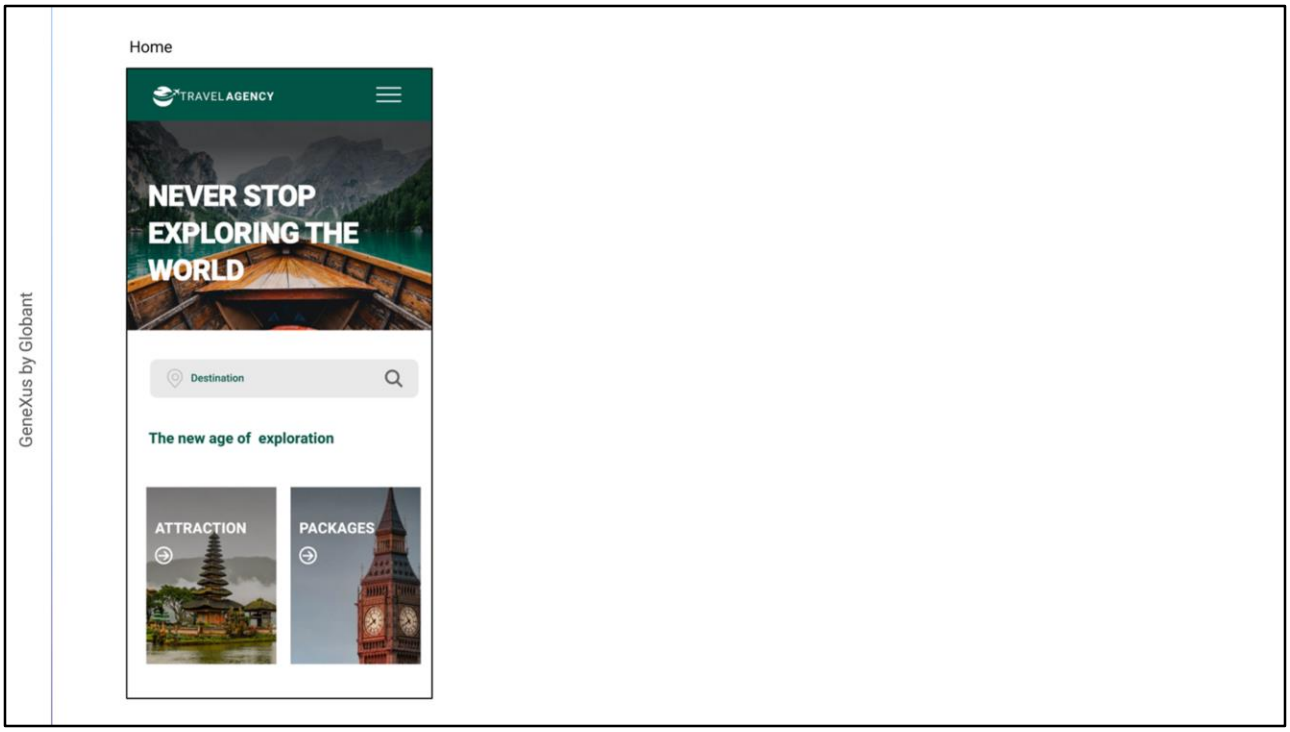
In this video, we will see how to import into our GeneXus application the design of entire screens, including colors, fonts, and so on, made by an application design specialist using Figma, their usual work tool.

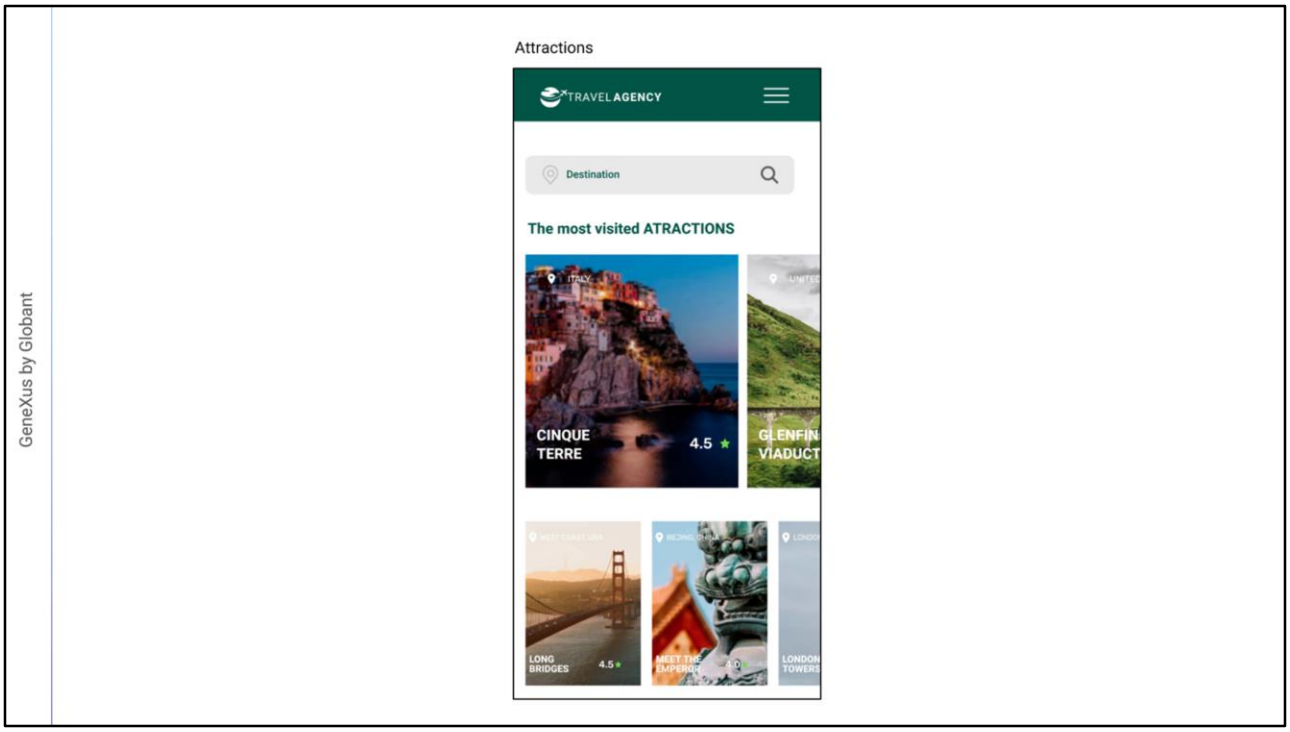


We have told the designer that we need an initial screen that will be the starting point for the rest of the screens in the application, another screen to see the list of available tourist attractions, and a third screen where you can see the details of a tourist attraction.

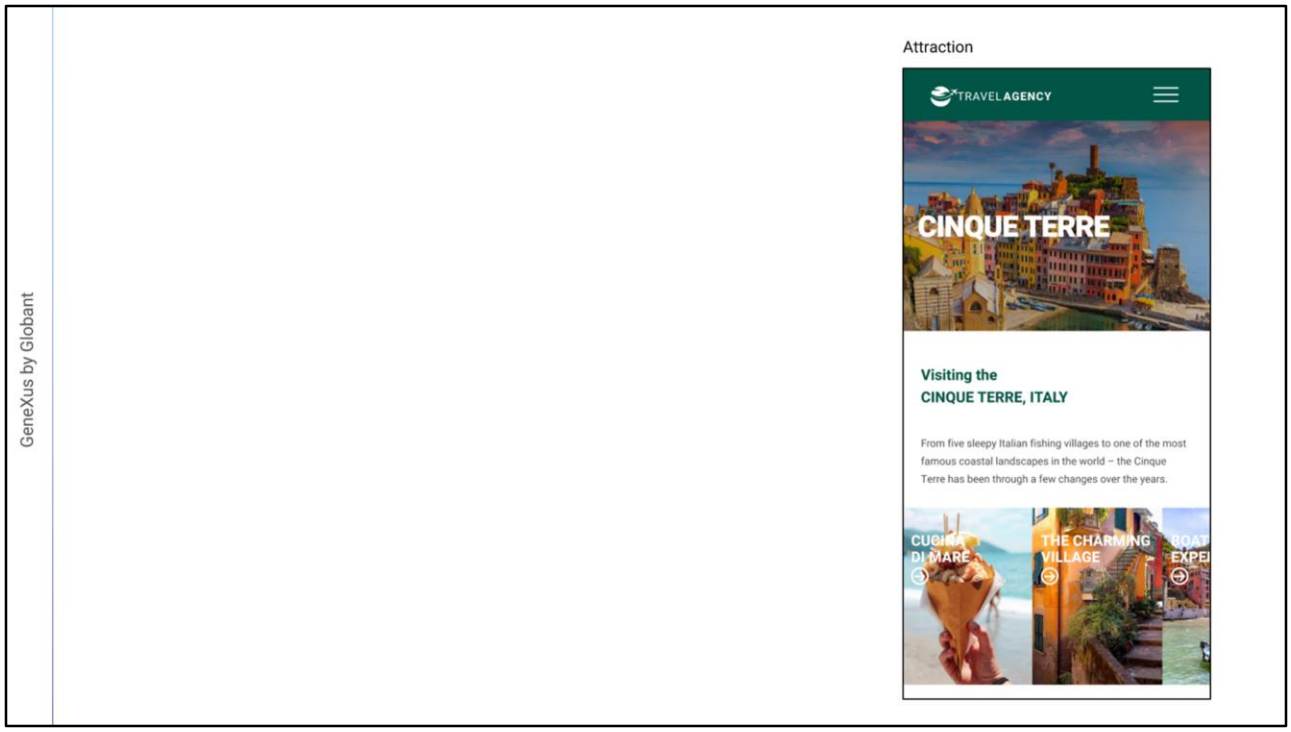
These are the screens made by the designer.



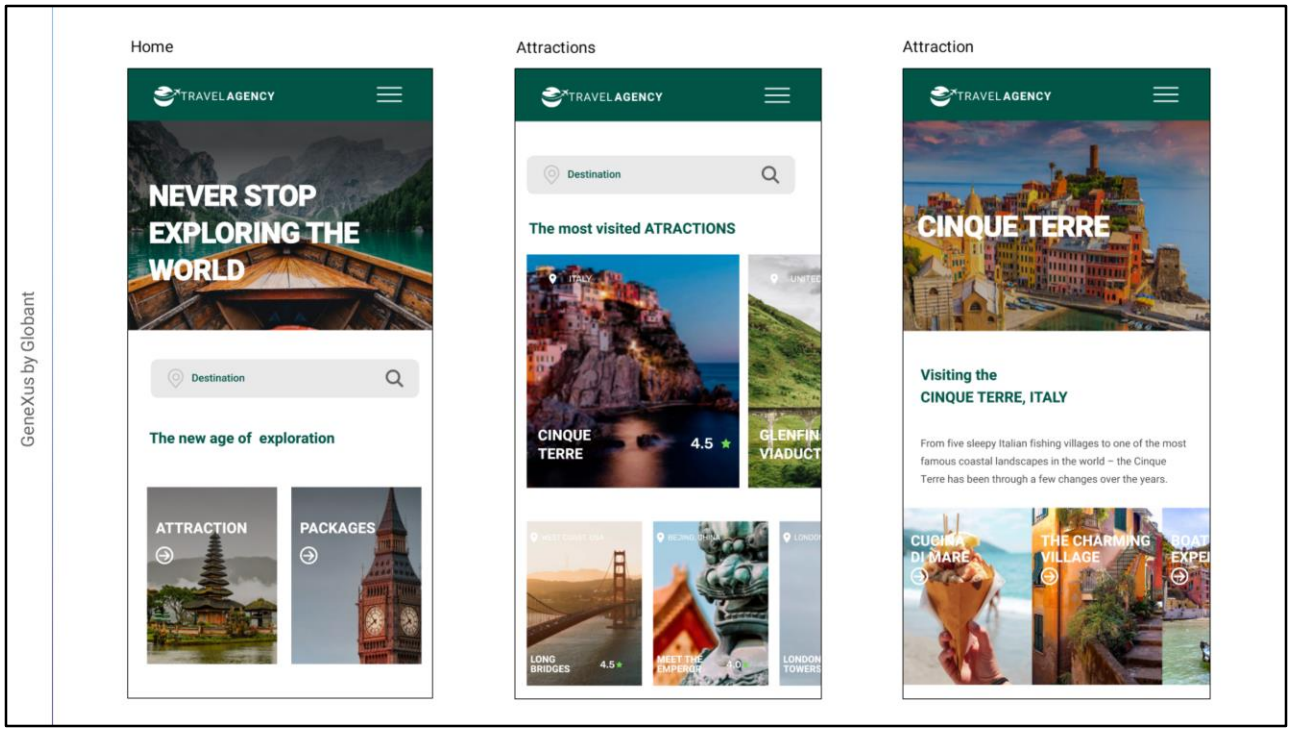
The one on the left is the main screen of the application, and shows at the top the application logo and a menu of actions. Further down, there is a background image with a highlighted title on it, then a bar where the user can search for destinations available in the travel agency, and below the different screens that can be navigated.



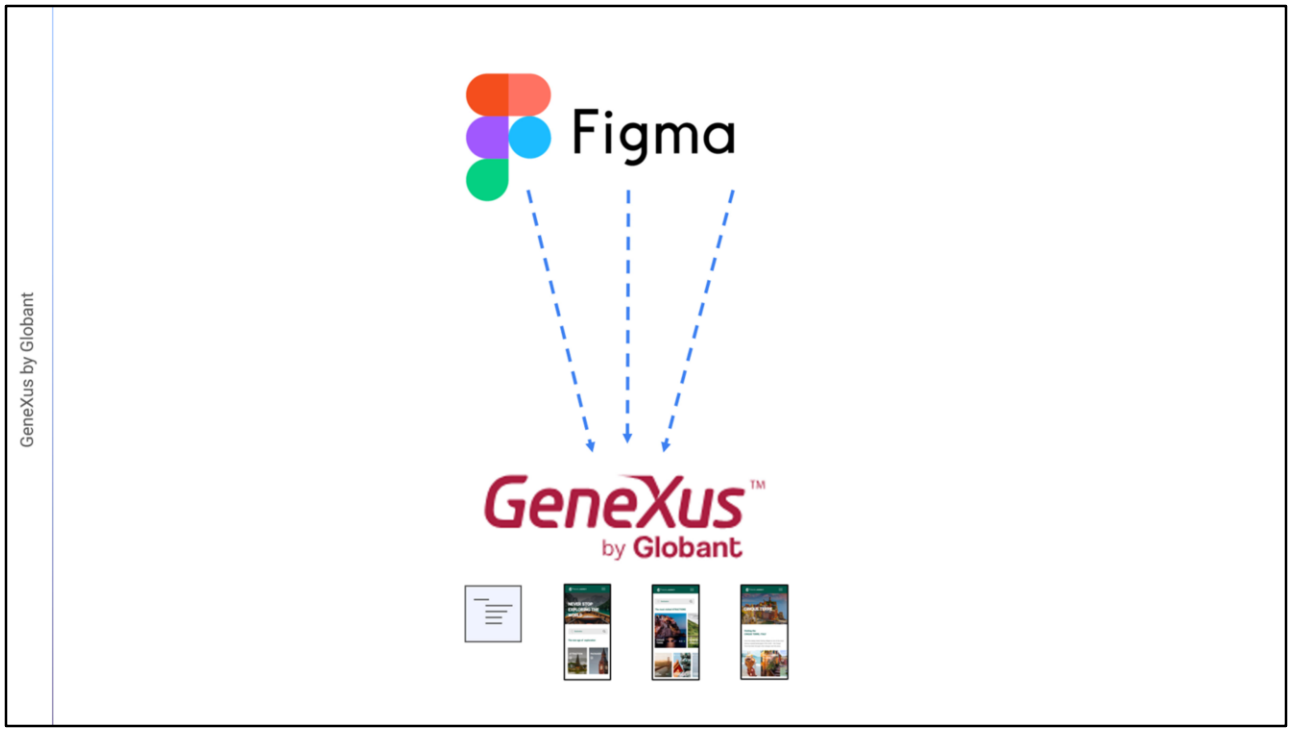
At the top of the screen in the middle is the logo, the menu, and the destination search bar; in the central part there is a grid with the most popular tourist attractions. Further down, there is another grid with the rest of the tourist attractions that the agency offers to visit.



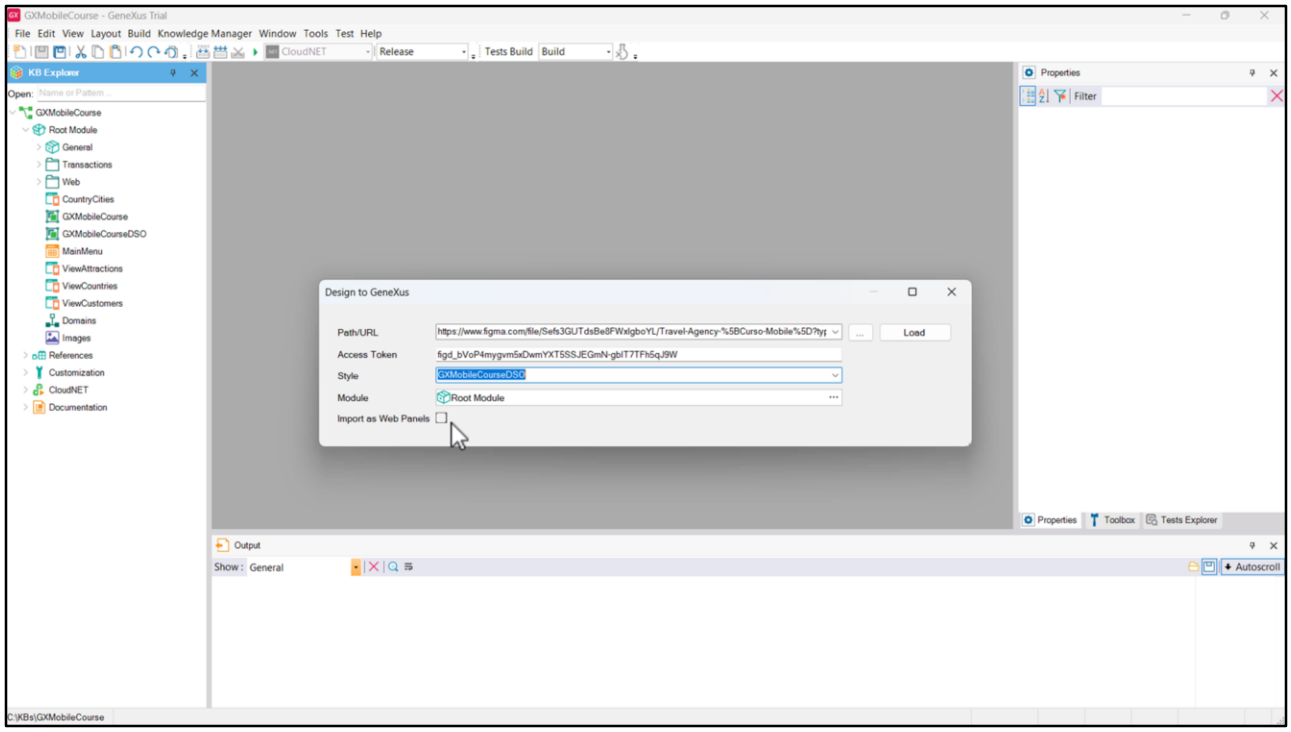
The third screen shows the details of a selected attraction –in this case, Cinque Terre– where once again we see the logo and menu at the top, then the image of the attraction as the background of the highlighted title indicating the name of the attraction, below it the description, and finally a grid where the user can access screens with more information related to the location of the attraction.



We can see that the designer has made a good job, creating a consistent look across the screens with a clear, clean, and attractive display of information.



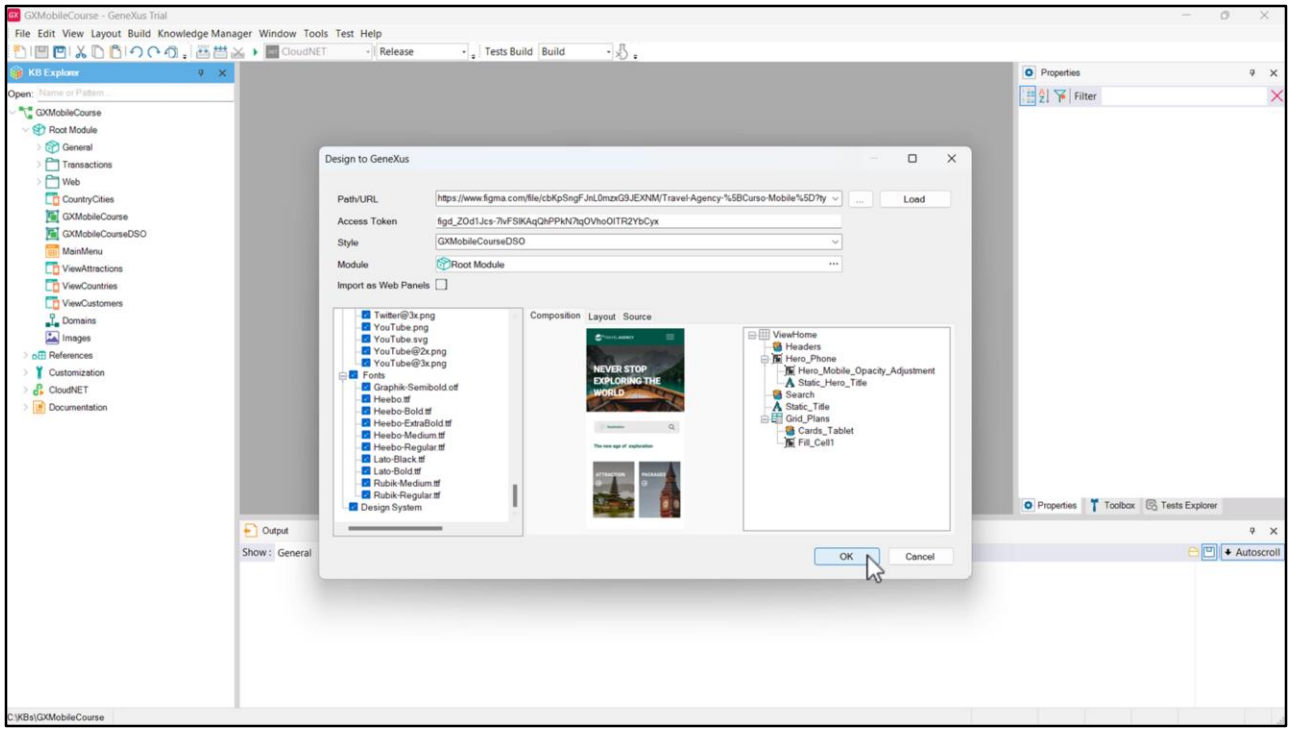
GeneXus allows us to import this file created in the designer's tool directly into our KB and all the design elements will be automatically created in a DesignSystemObject; even the corresponding GeneXus objects will be created with all the necessary controls for the application to work, with the screens we saw before.



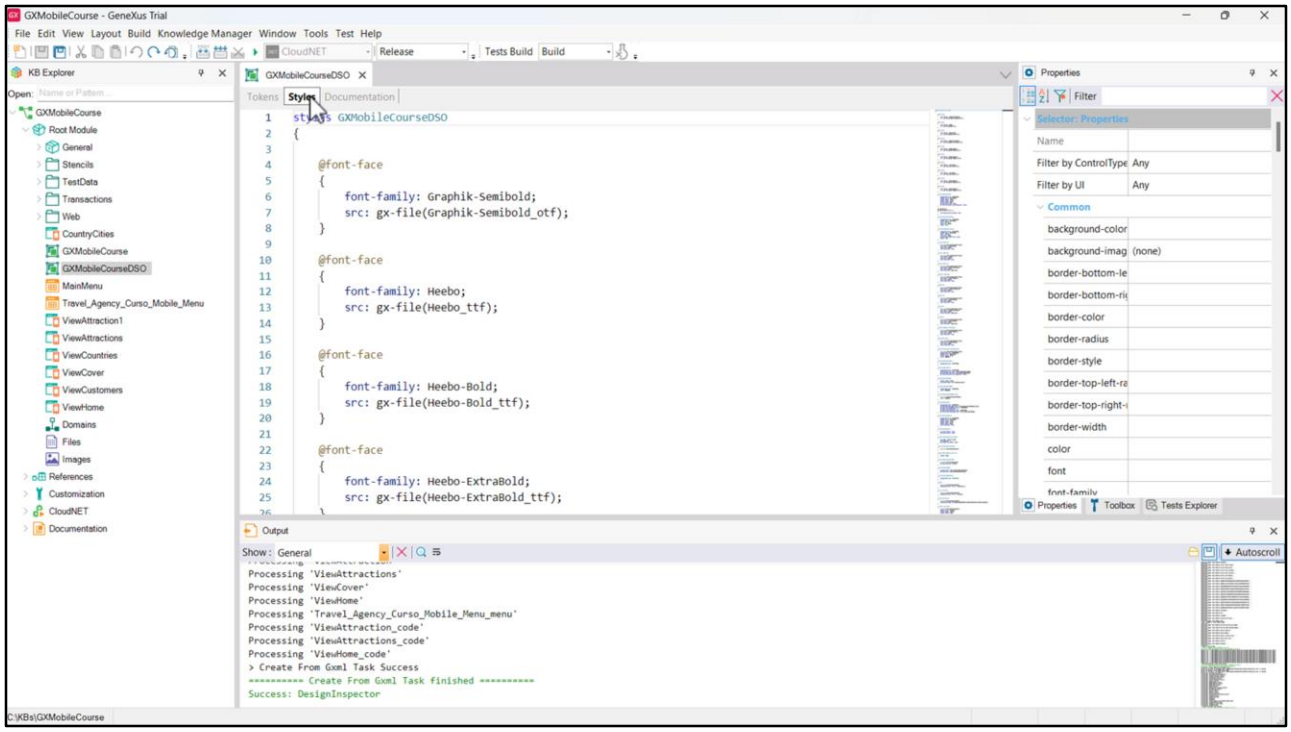
To import the design sent by the designer, we go to Tools/Application Integration and choose the Design Import option.
In this dialog box, we must enter two pieces of information provided by the designer to access the design: the Figma* URL that contains it, and then the Access Token** in the text box that is displayed. The Access Token is a code that gives access to all the data of this design in the Figma account. (For this example, the designer created a public account so we can all try importing the design into GeneXus using the same token).

We add the name of the Design System object where we want all the design definitions to be stored (GXMobileCourseDSO), and leave the Import as Web Panels checkbox cleared, because we want Panels to be created and not Web Panels.

*** Figma URL:** https://www.figma.com/file/Sefs3GUTdsBe8FWxlgboYL/Travel-Agency-%5BCurso-Mobile%5D?type=design&node-id=1%3A624&mode=design&t=z5pXsGs4BAxDZhMD-1
**** Access Token:** figd_bVoP4mygvm5xDwmYXT5SSJEGmN-gbIT7TFh5qJ9W

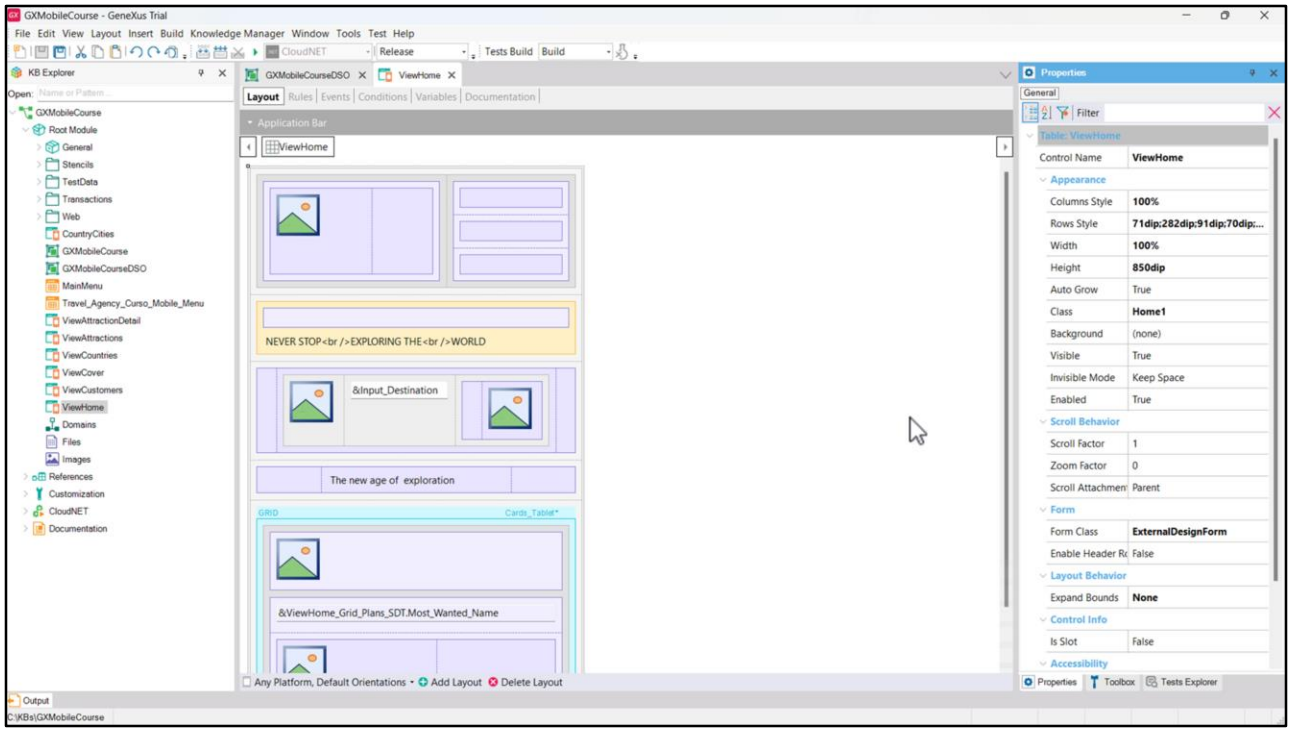


We click on Load, and if we click on any of the Panels that will be created by the import, we see a preview and confirm that they correspond to the design that we validated. On the right there is information about the controls that it will contain.
It will also import Stencil objects, the images to run the Panel with fixed data that will allow us to check that the functionality is the desired one (later on we will have to replace this data with the data stored in our application's database), fonts, and the definitions that will be imported to the selected DSO.
We agree, so we press OK.



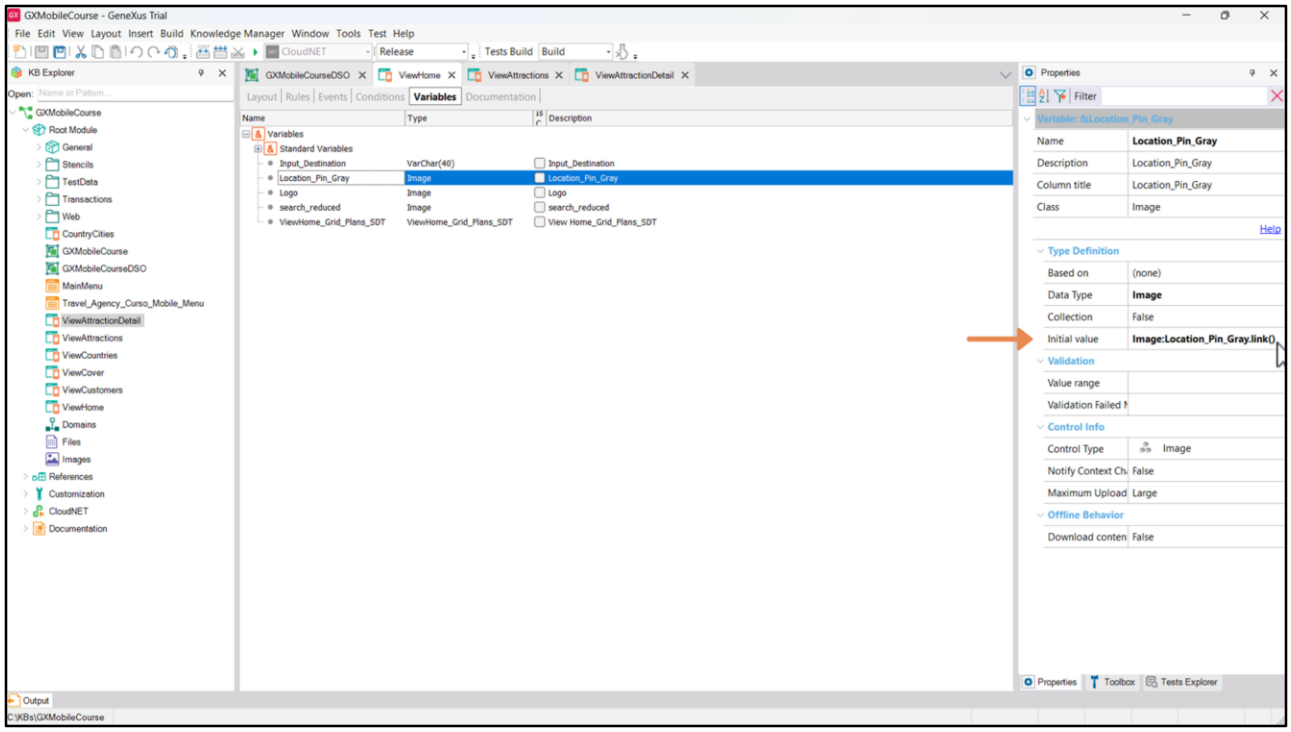
In the Output window, we see the progress of the import and the result without errors.
In the KB Explorer, two folders were created: Stencils, containing several Stencils used to encapsulate the design in the application and Test Data, containing Data Providers and SDTs to load fixed data, which will allow us to test the application to check if the design looks as we expected.

If we open the GXMobileCourseDSO Design System object, we can see that the Tokens and Styles that define the controls' appearance on the screen have been added.

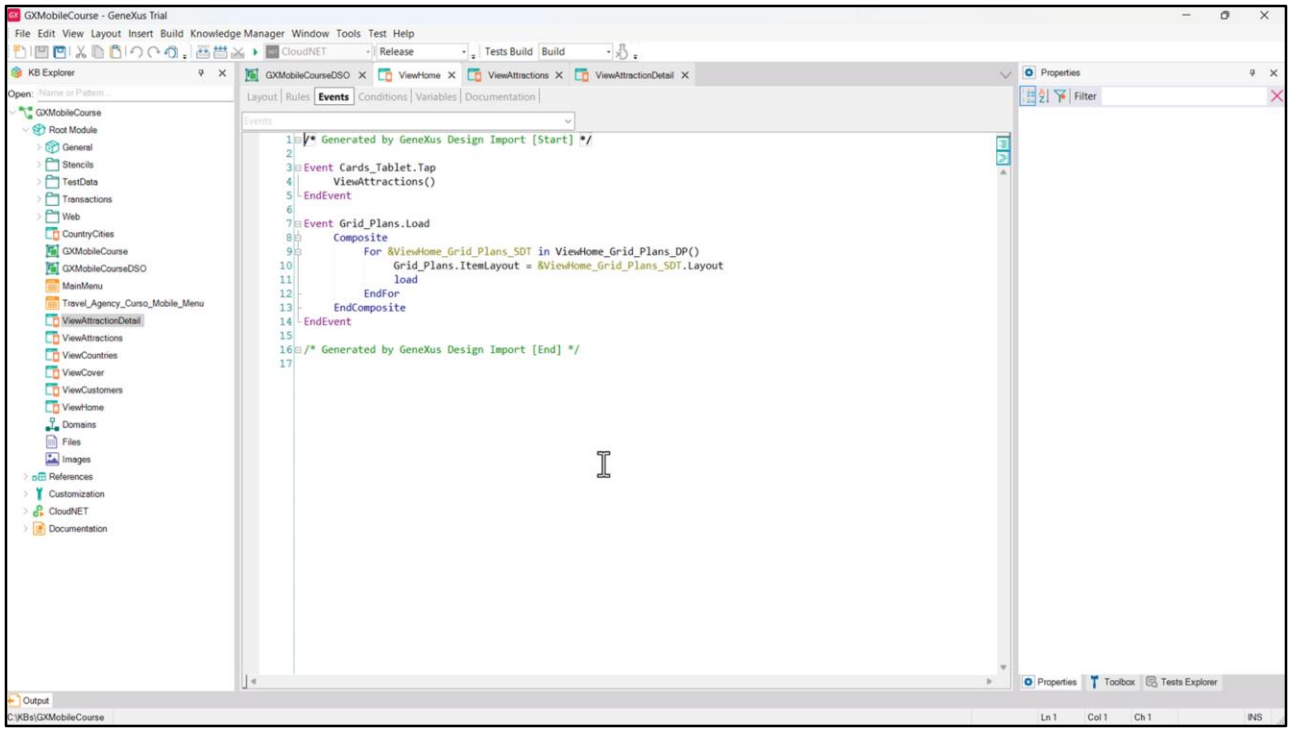


At the bottom we find that the Panel objects we had seen in the import wizard screen were automatically created, such as the one on the home page (ViewHome), the one that shows the attractions (ViewAttractions), and the one that shows the detail of an attraction (ViewAttraction1, a name we will change to ViewAttractionDetail).

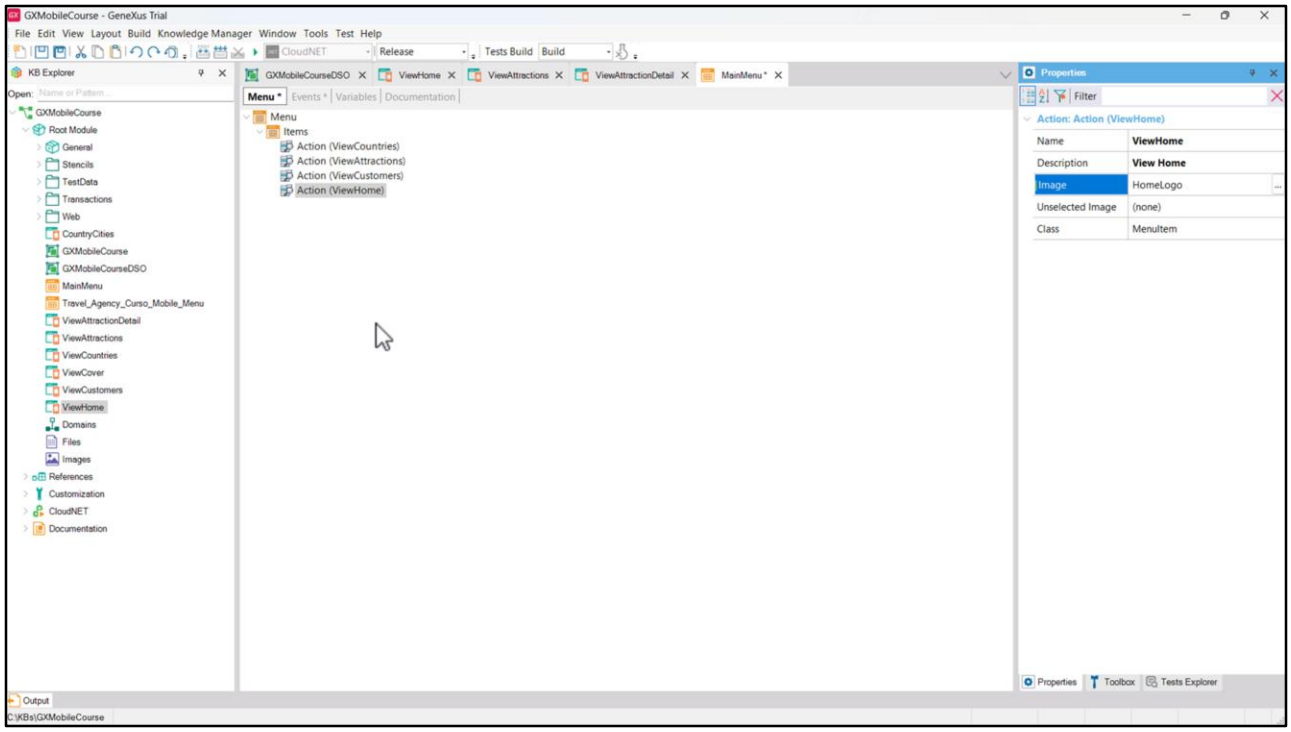
In the ViewHome Panel form, we can see that it already includes all the components for the visual content, such as tables, variables, images, etc., all created automatically in the import process. The same happens with the other imported Panels.



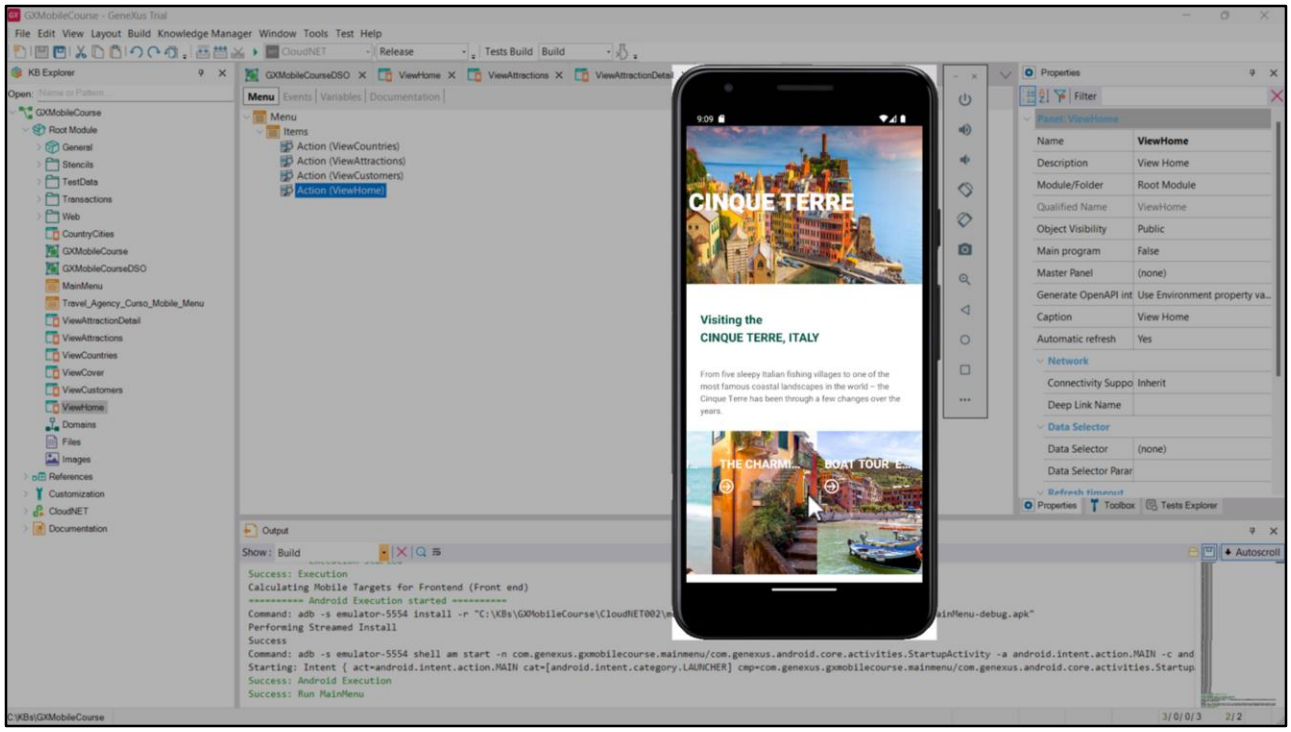
Going back to the ViewHome panel and looking at the variables, we see that the images were loaded using the InitialValue property, making reference to the images that the import process loaded in the KB.



In the events, we see that the interaction between panels is also programmed –since pressing on Attraction invokes the ViewAttractions Panel–, as well as the loading of the Grid with its different options (show in the screen image).

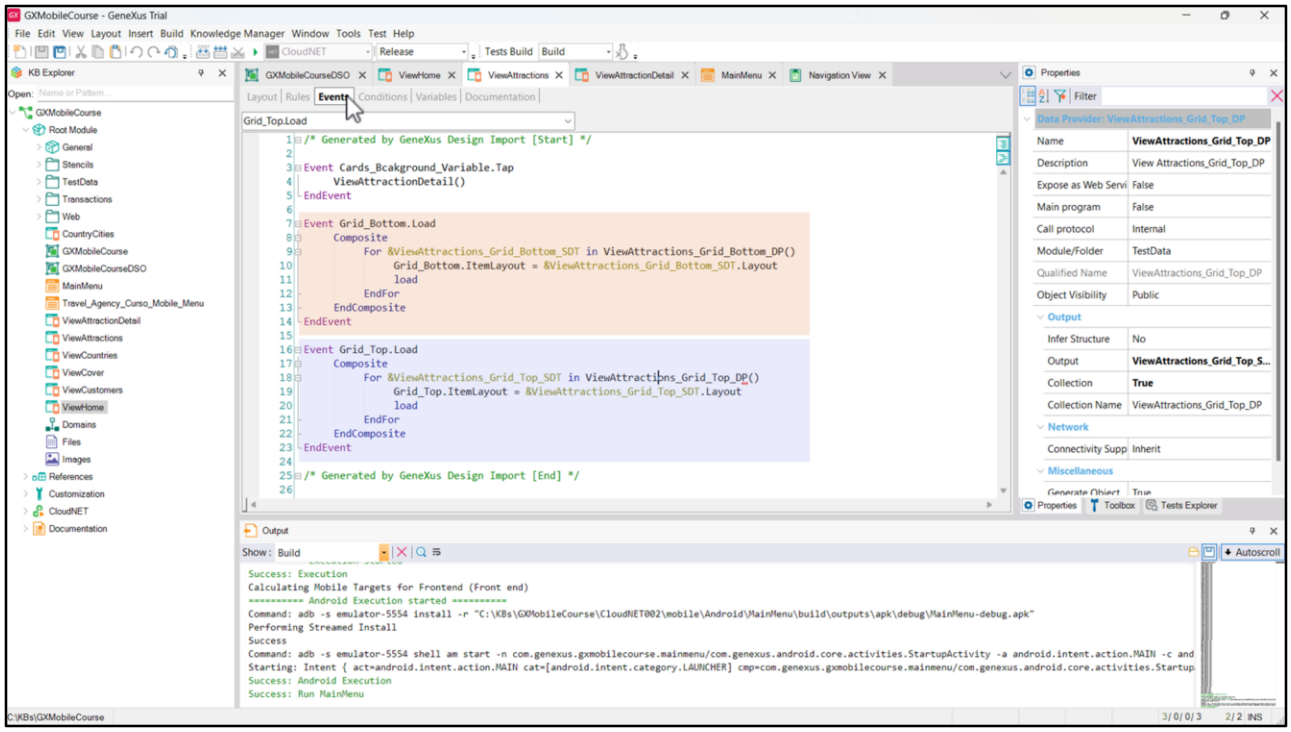


In addition, a menu object was created, which by default would be the main object to be executed. However, since we can access the other screens sent by the designer from the ViewHome panel and our MainMenu has already been created, we simply add its reference under the Items node to keep this menu as a Startup Object.



We run it. If we open the ViewHome Panel, we see that the initial screen is running and looks just as our designer envisioned it.
We click on Attraction and see the list of tourist attractions of the travel agency.
We click on Cinque Terre, and see a page with detailed information, with a polished and attractive design.

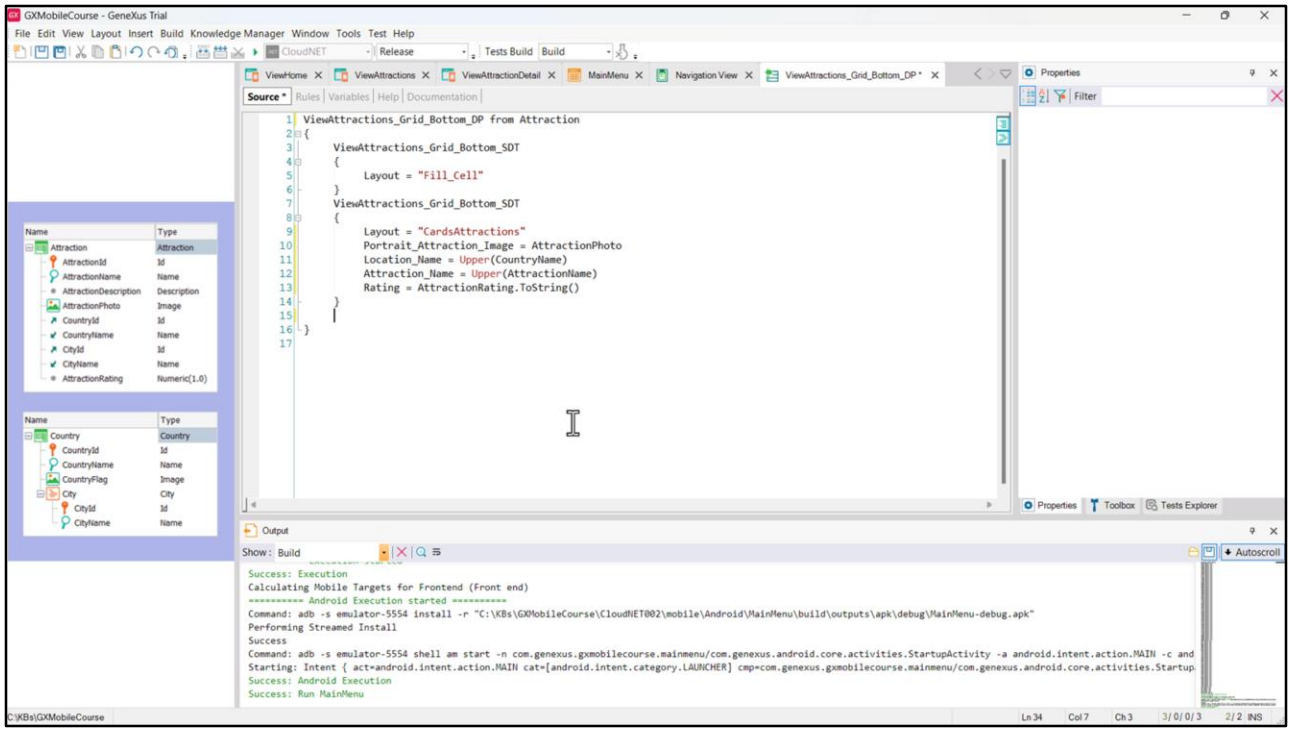
This example showed the development process of an application and how important it is to work as a team with people who have different profiles, each one contributing to the best solution.



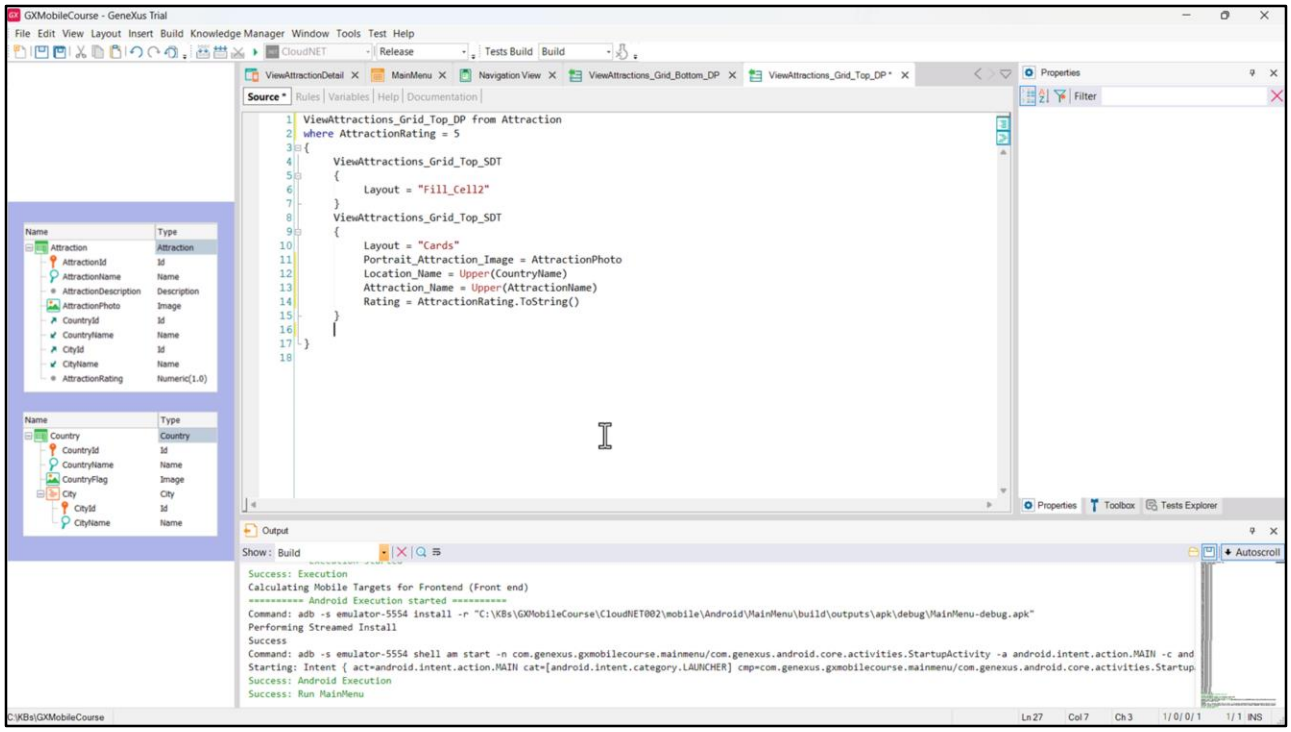
So far we have been looking at fixed test data, which the designers added so we could see the app's behavior with the new design. So, we will make some changes to see the actual data from our database.

When we go to the ViewAttractions Panel events, we see that the grids are loaded using a variable of SDT type and a Data Provider.

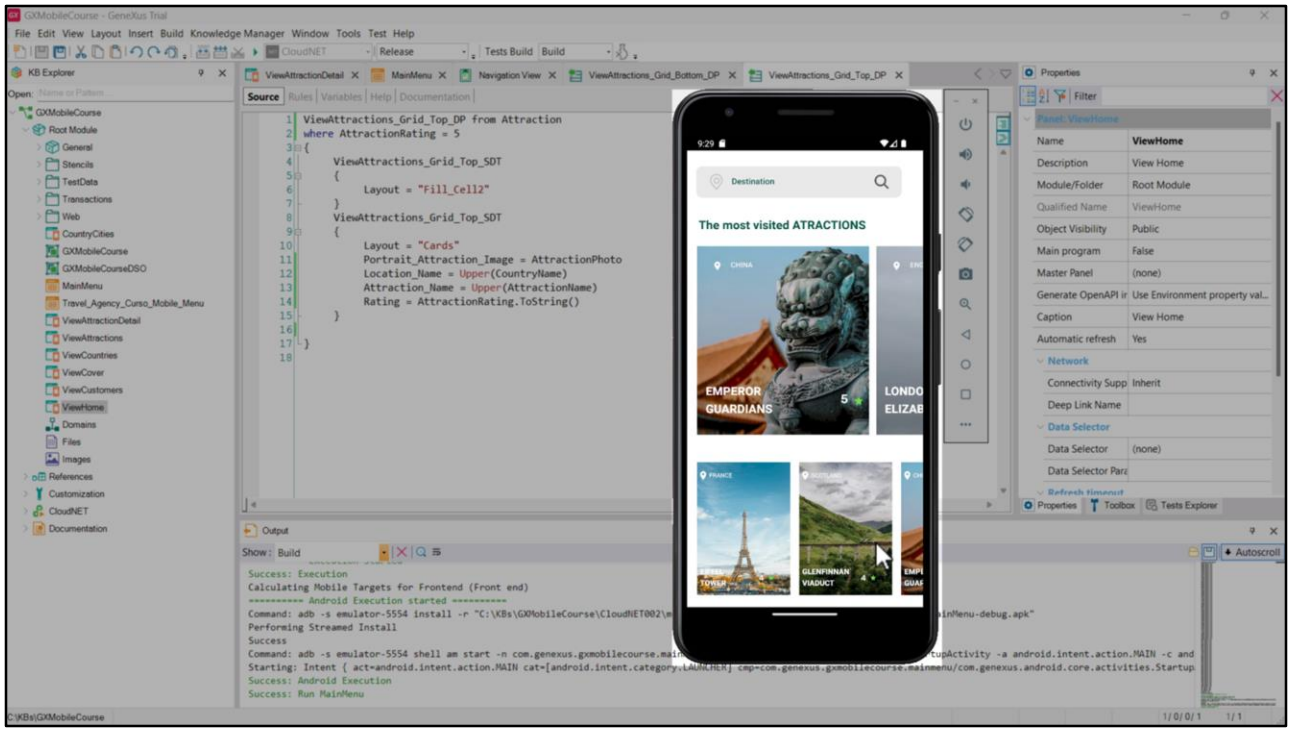
Let's change the loading of the DPs so that instead of using fixed data, it takes the attributes of the Attraction and the Country tables.



Let's start with the one that is loading the lower Grid. We add the clause from Attraction, and load the field Portrait_Attraction_Image with the value of the AttractionPhoto attribute; the field Location_Name with the CountryName attribute; and the field Attraction_Name with the AttractionName attribute. To Rating we assign the value of the AttractionRating attribute, and delete everything unnecessary because it is for loading fixed data. We will leave the specifications of the Layout because it is useful to keep the spacing before and after showing an attraction.

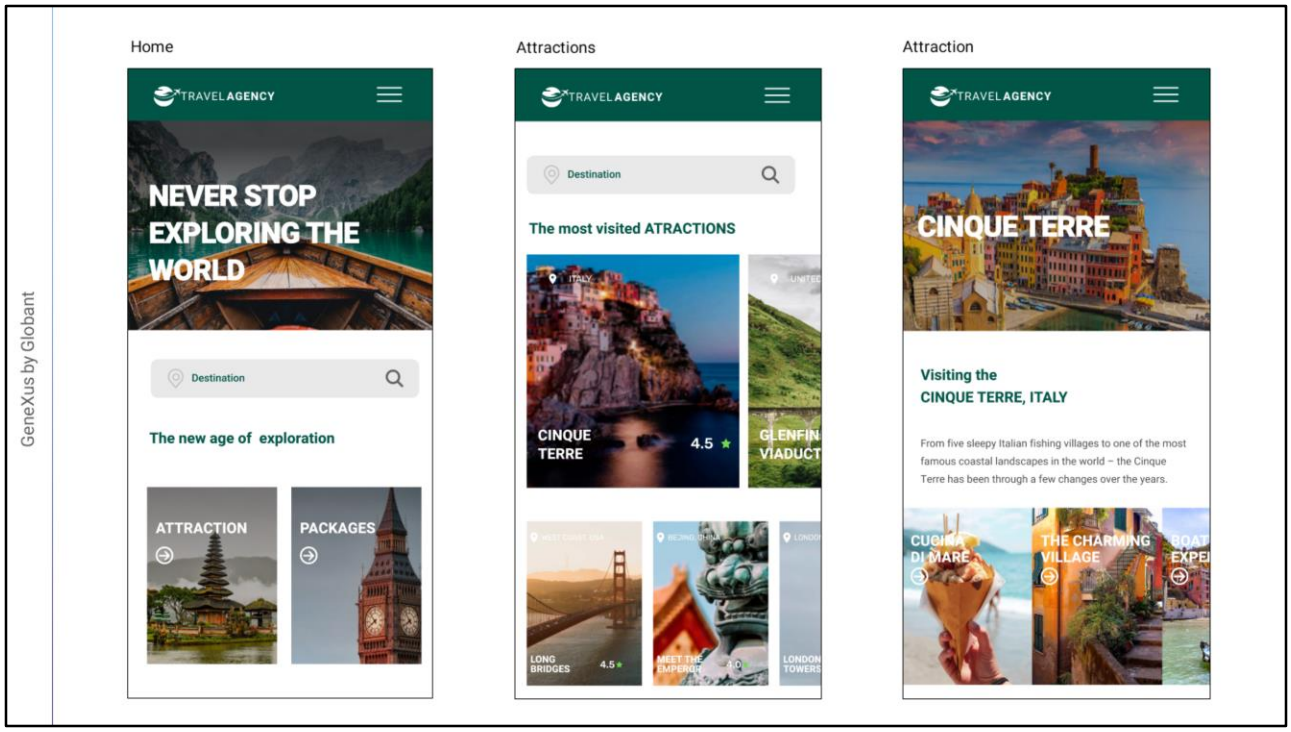


We do the same with the Data Provider that loads the data of the first Grid, but in this one we are going to apply a filter so that only the attractions with a 5-point rating are displayed.



Let's run it to see the changes reflected.
In the attractions, we are running through the attractions that we had loaded in the database because we see, for example, the Eiffel Tower, which was not included in the fixed data; if we paginate, we can see that all the attractions are there.

We are meeting the objective requested by the Agency, which was to show all the tourist attractions, but now the application has a more suitable design.



Of course, if we have a designer in our team, we will never write the DSO manually from scratch, because importing these definitions from Figma is a more professional way to handle the design and we can focus on the functional part of our application.

Integrating the design made by a professional designer to our GeneXus project brings many advantages, since the effort is optimized and each team member contributes according to their profile, doing what they do best.


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