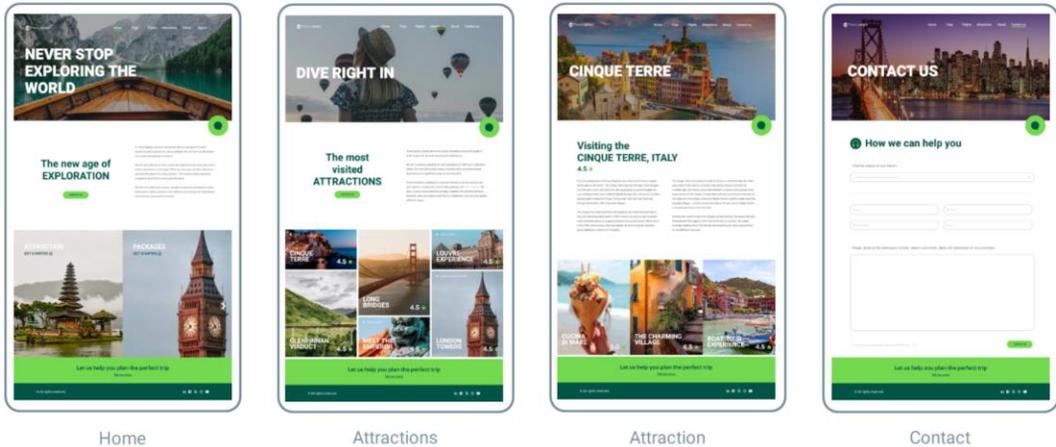


Demo: First steps



Cecilia Fernández

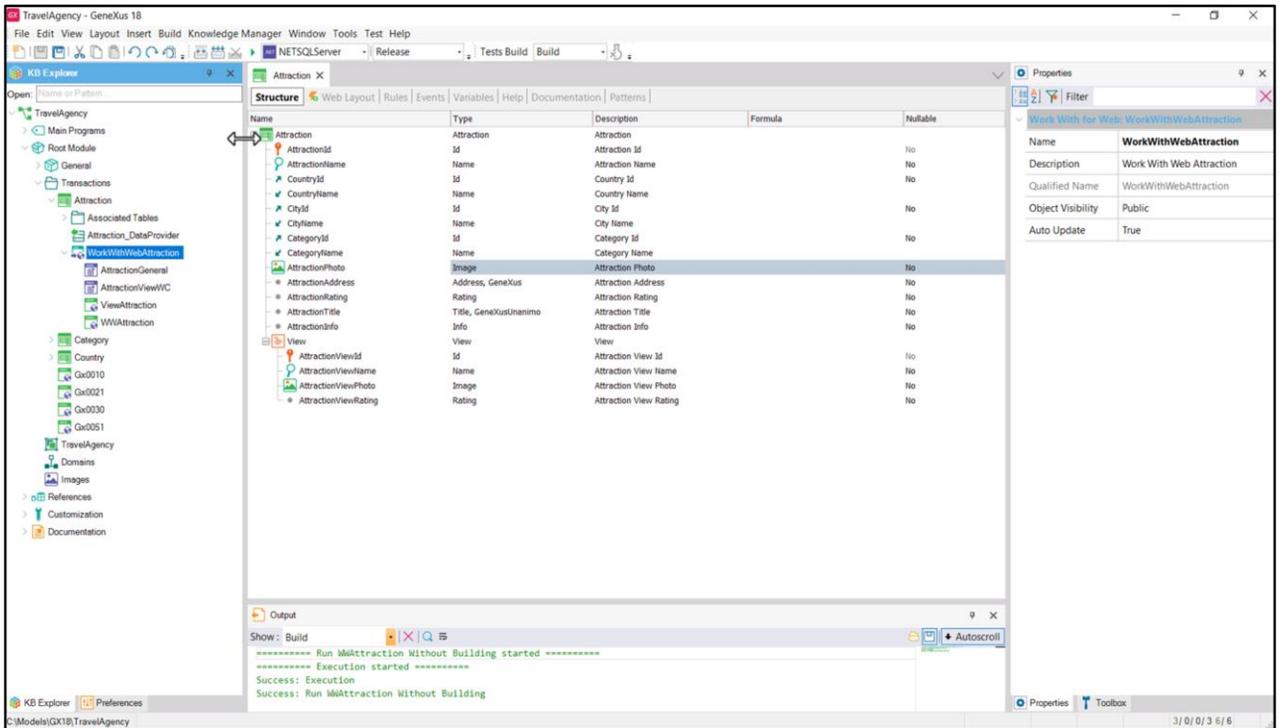
Hola, ¿cómo están?



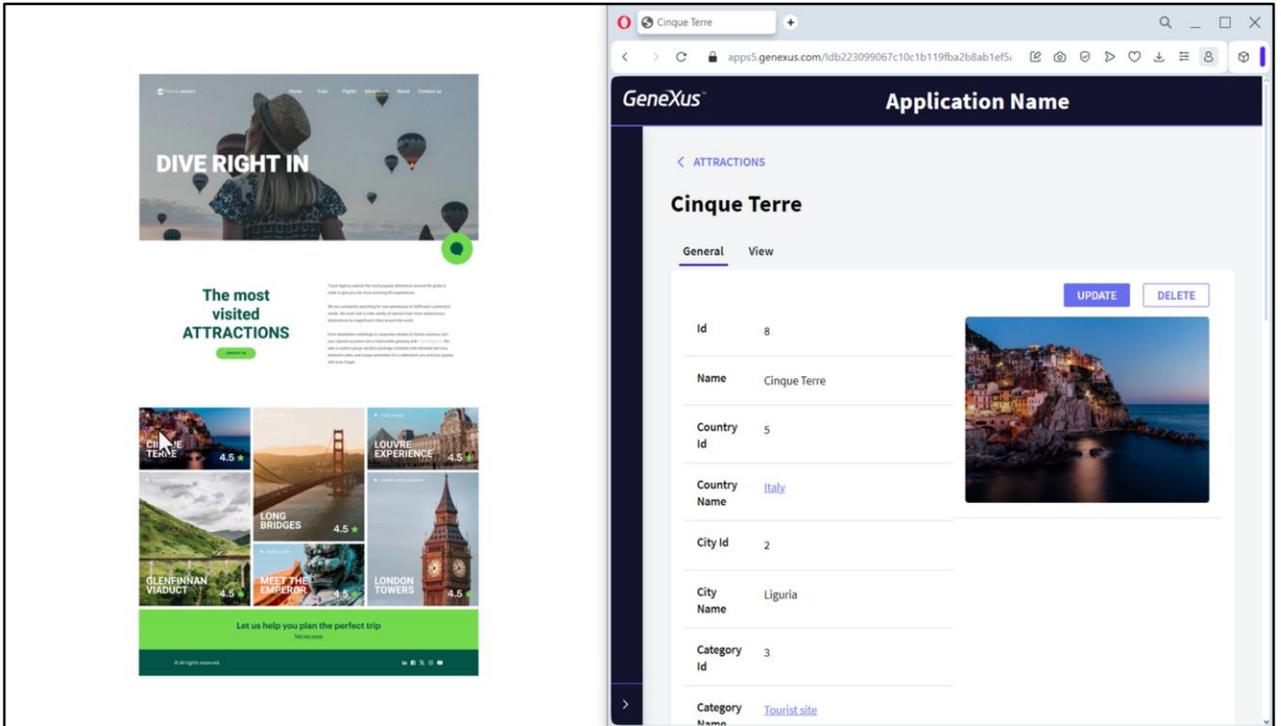
En esta clase vamos a empezar a trabajar en la KB en GeneXus para poder llegar a obtener, unas cuántas clases más adelante, la aplicación web que implementa estas 4 pantallas de las que veníamos hablando, para tamaño desktop.

Después de eso veremos cómo hacer variar la aplicación de acuerdo al tamaño de pantalla, para que pueda ejecutarse y visualizarse perfectamente en dispositivos de tamaño teléfono o de tamaño Tablet, siempre hablando de la aplicación web. Y luego, más adelante aún, sobre el final del curso veremos cómo hacer que la aplicación en sí misma se ejecute de forma nativa en esos dispositivos.

Bueno, así que vamos a empezar por ir a GeneXus...

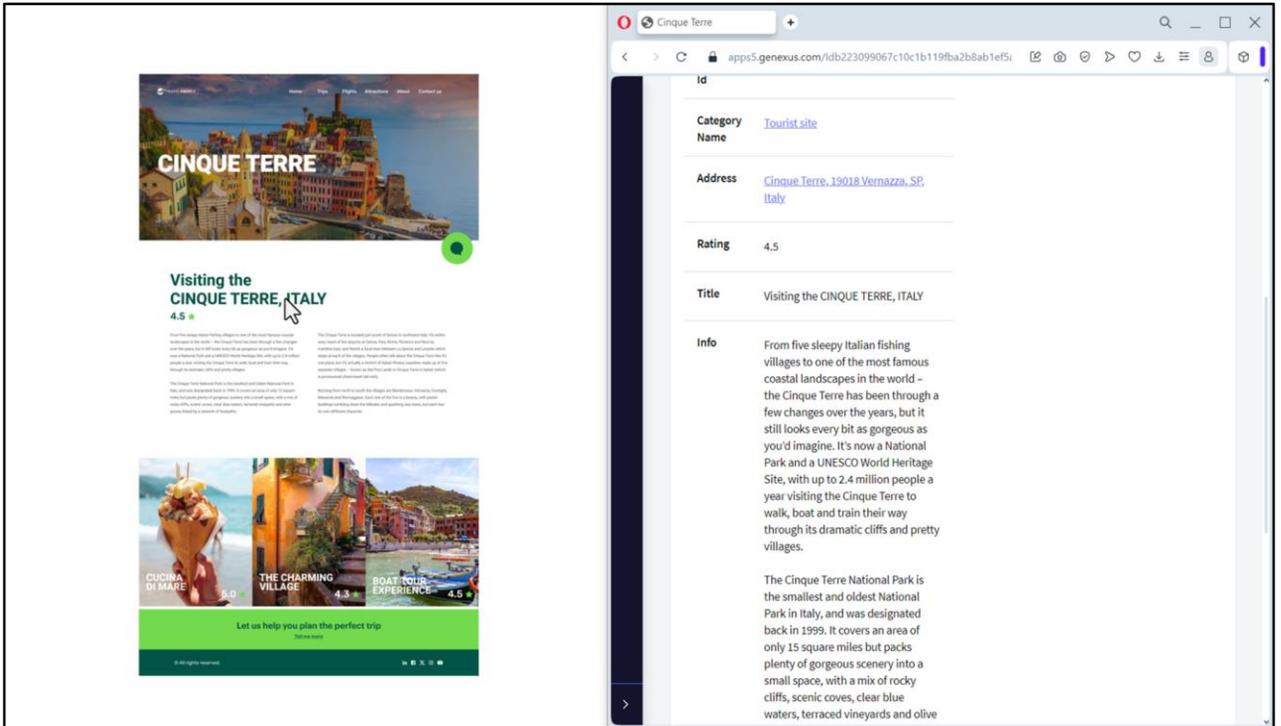


...y observar que creé una KB de nombre TravelAgency, en la que dentro de un folder, Transactions, creé las tres transacciones que de manera simplificada registrarán los datos en el Backoffice. A cada una la poblé de datos a partir de un data provider y le apliqué el patrón Work With Web.

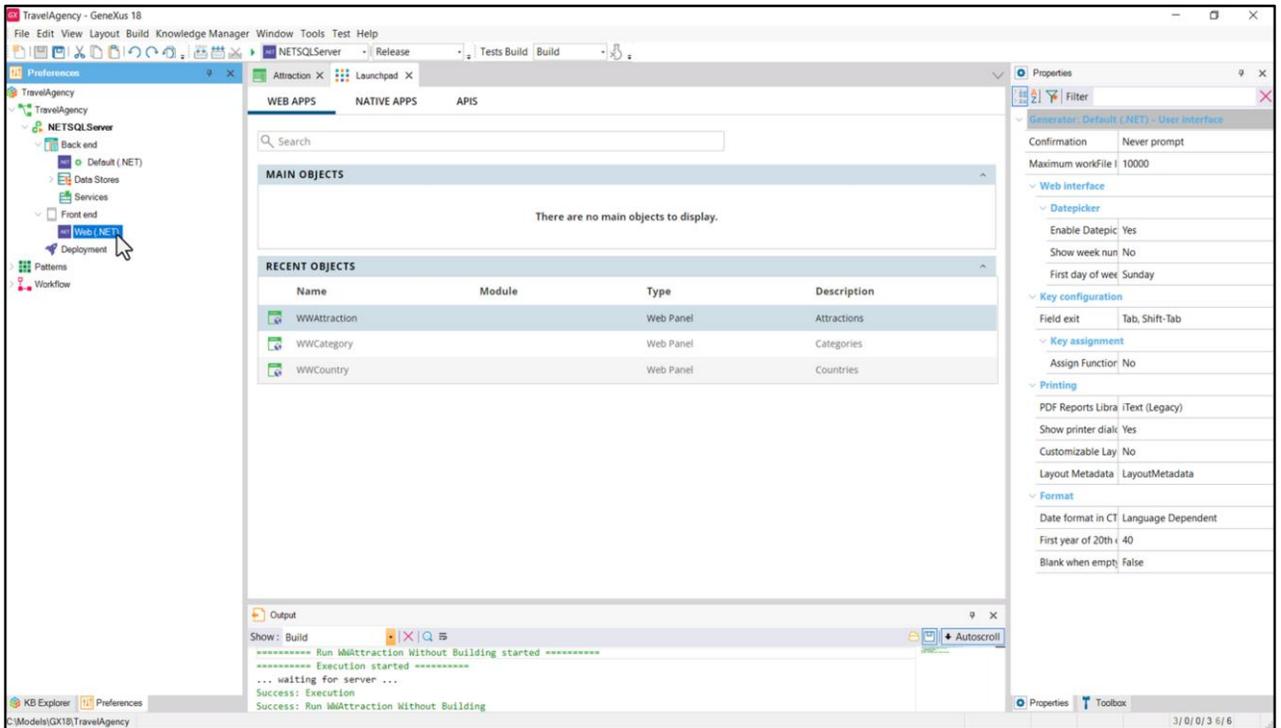


Así, si ejecuto... vemos el Work with de las atracciones turísticas.

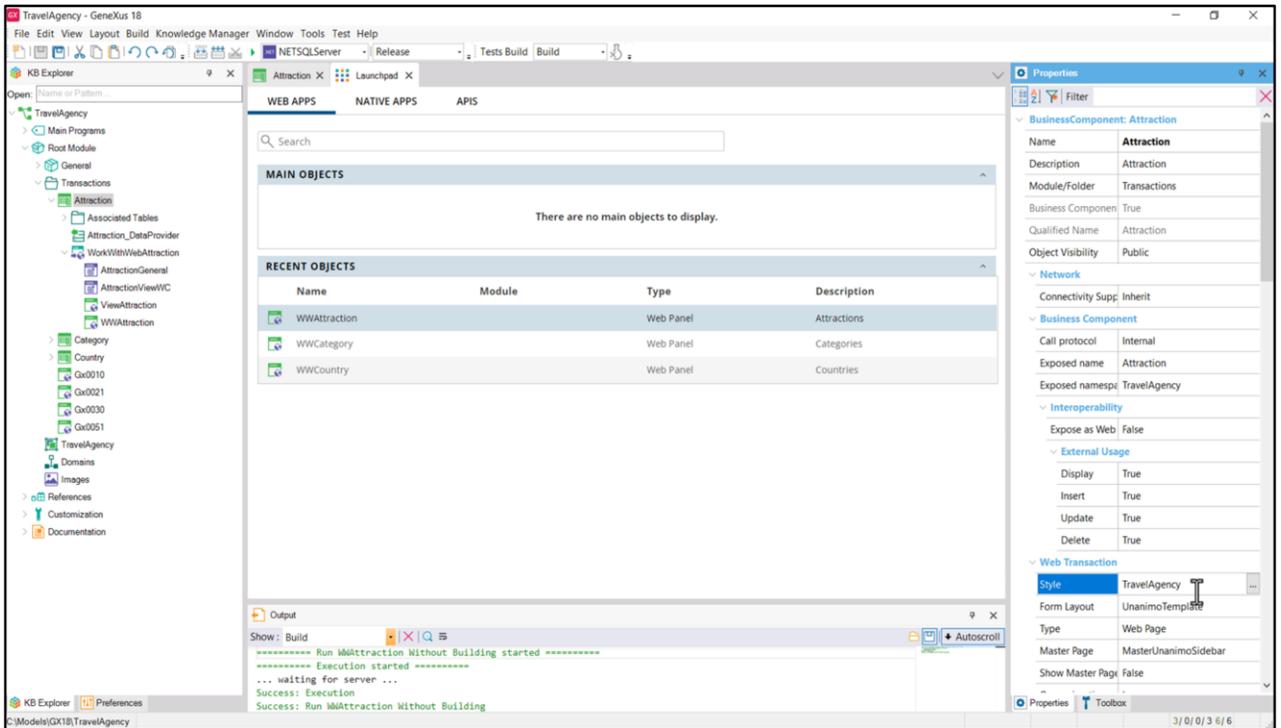
Y si elijo esta, vemos acá la foto que se utilizará para la aplicación Customer-facing (la que estamos queriendo implementar), el nombre de la atracción, el rating...



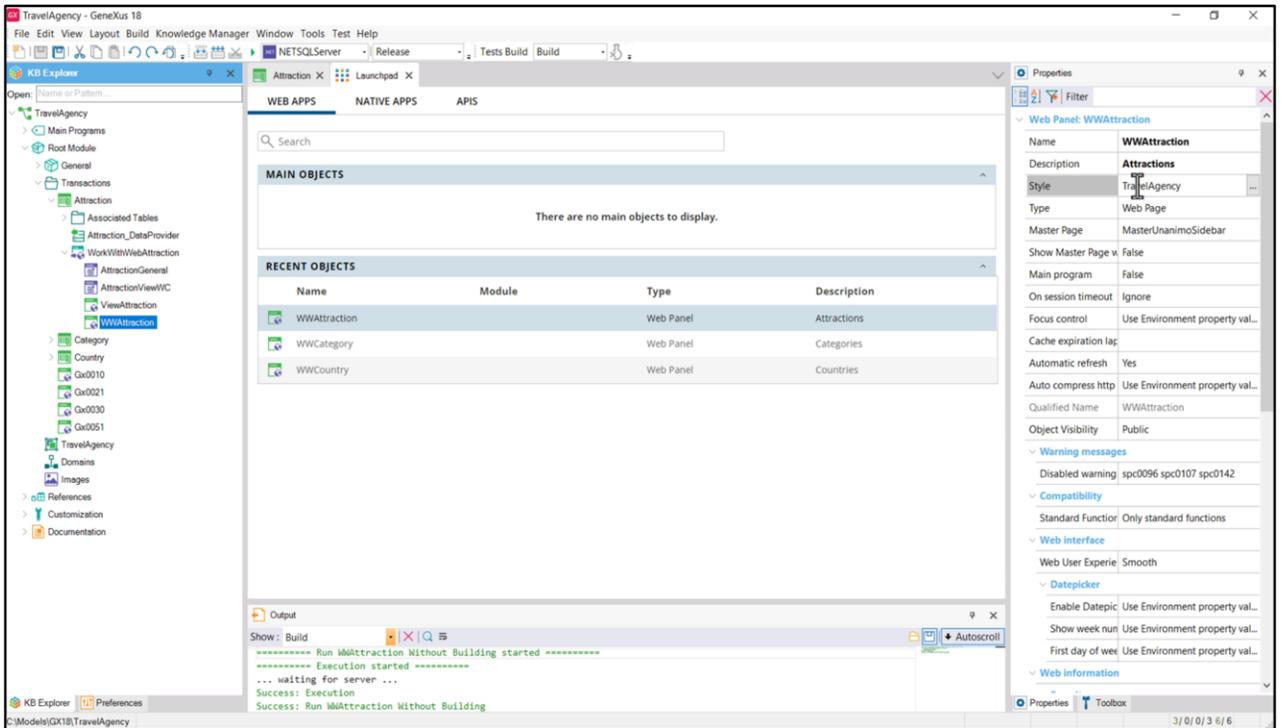
...el título que se verá en la página de la atracción, la información que se listará en esa página, y si vamos a la solapa View, acá vemos esta información relativa a la atracción turística que se va a utilizar para mostrar en un carrusel, en esta página, que se podrá luego scrollear.



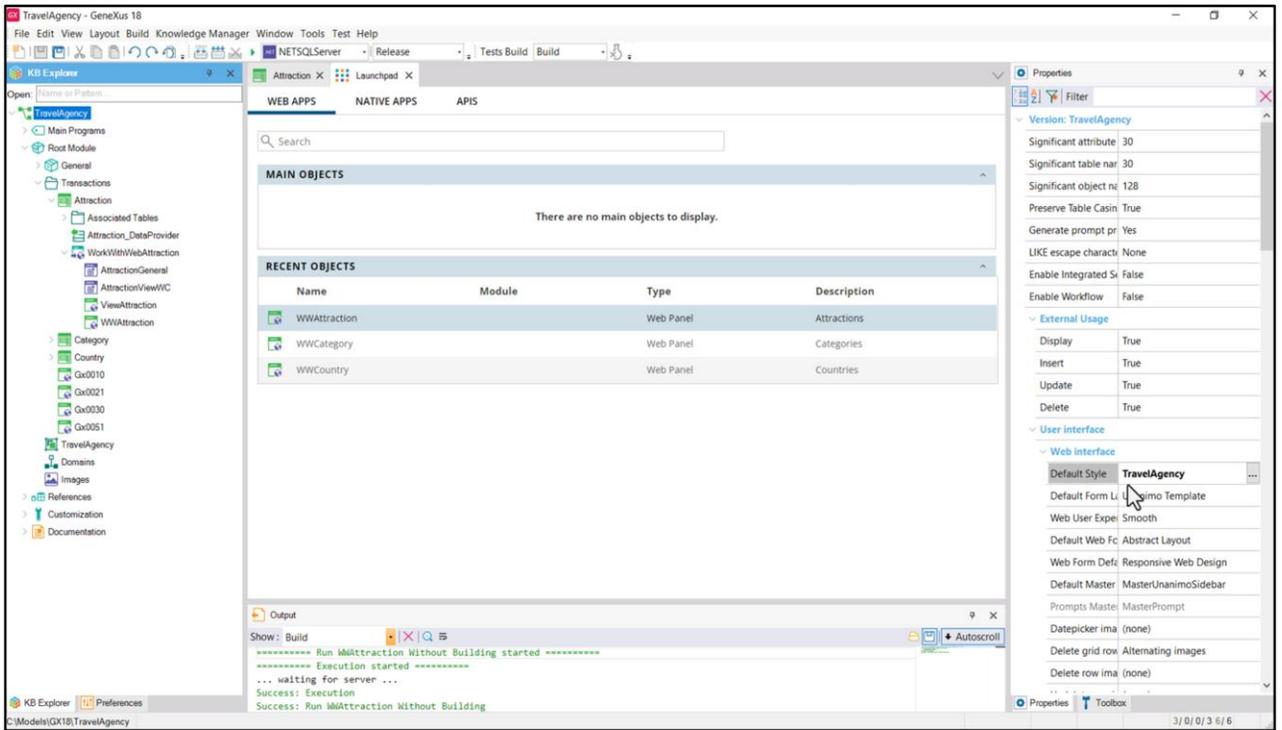
Esto que estamos viendo en ejecución es el Backoffice que sabemos que de forma predefinida utiliza para su Front end el generador Web (.NET).



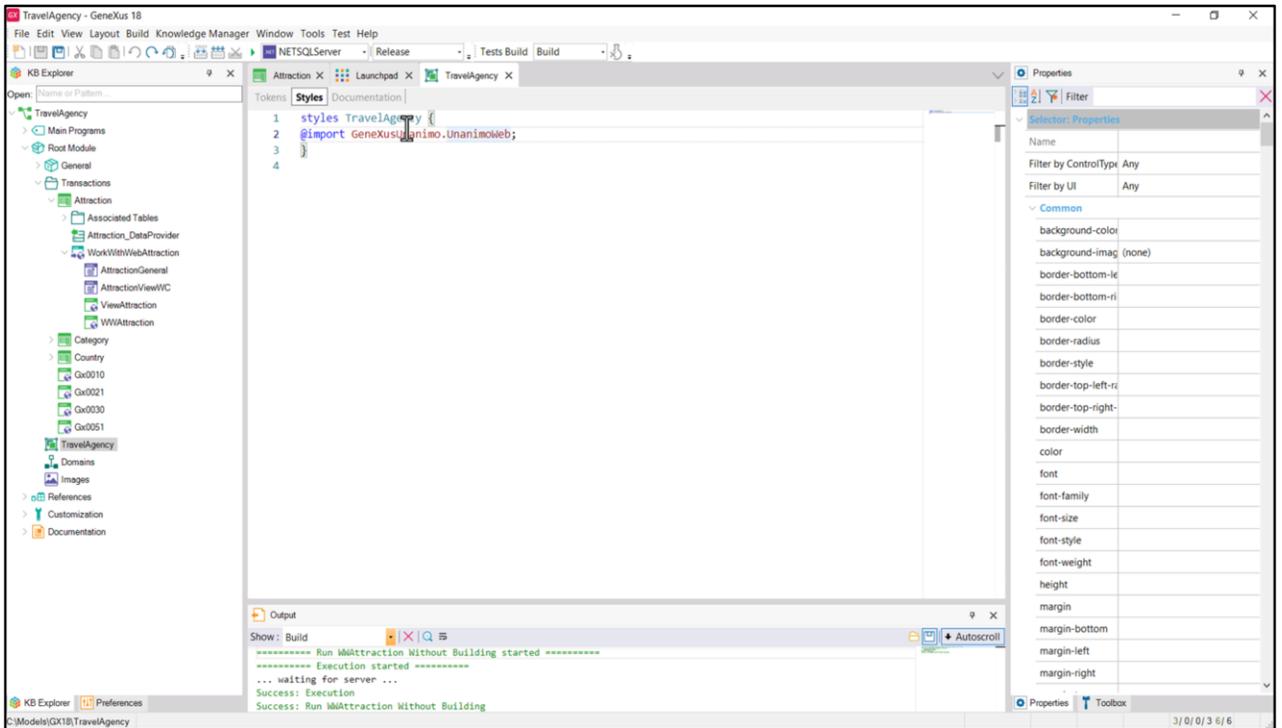
¿Y cuál es el objeto Design System que comanda el estilo de la User Interface de este Backoffice? Si vemos las propiedades de la transacción Attraction, vemos que la de nombre Style indica el objeto de nombre igual a la KB.



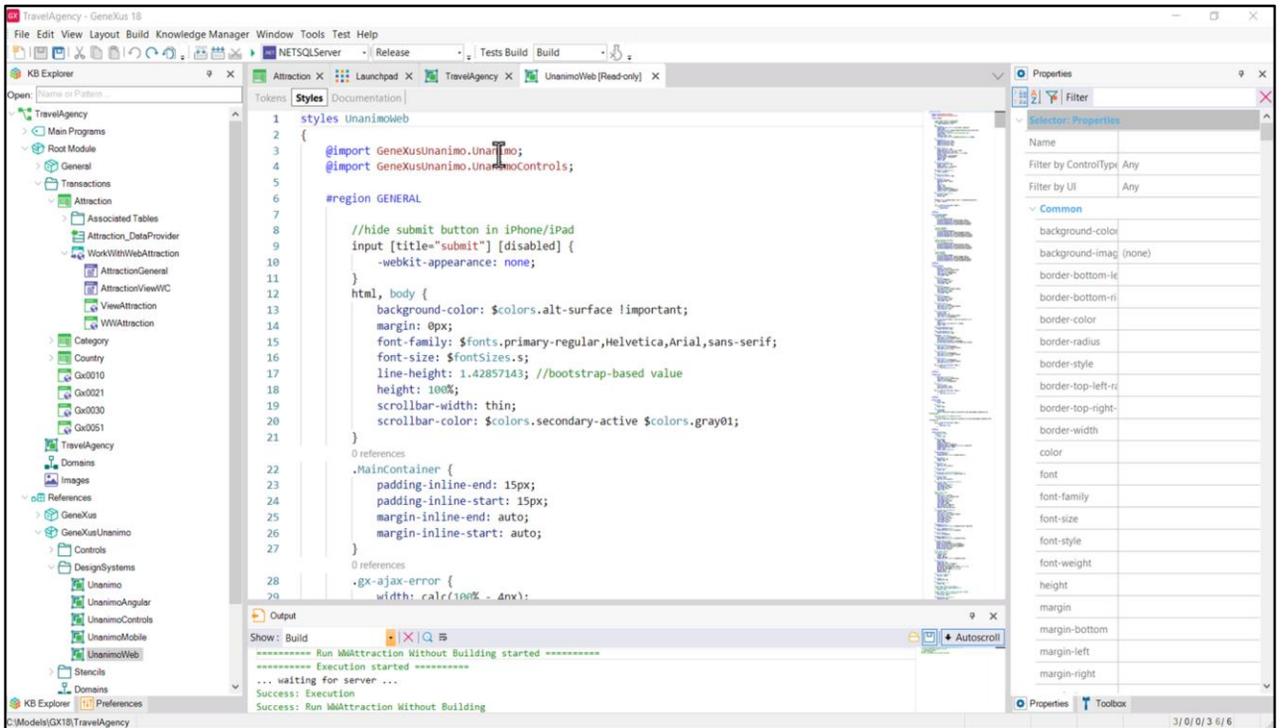
Si vamos a las propiedades del Web Panel WorkWithAttraction vemos para Sytle el mismo objeto Travel Agency. Lo mismo para el View, y lo mismo será para las demás transacciones y Web Panels y objetos con interfaz que se creen para el Backoffice.



Esto lo vemos en la propiedad Style de la versión de la KB.



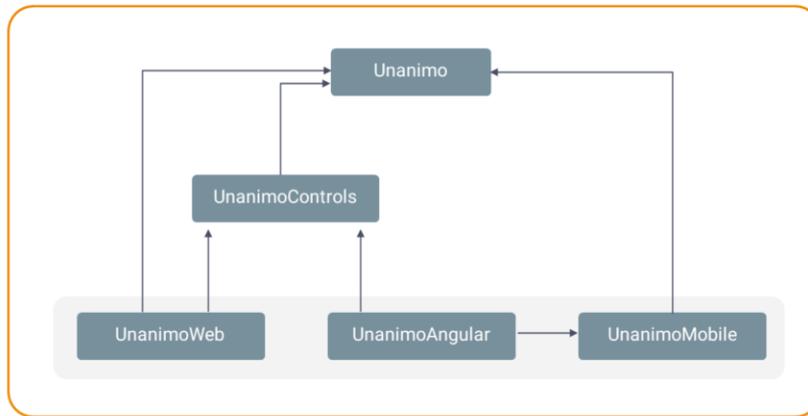
Este objeto DSO, recordemos, es el único que se crea por default en toda KB, al momento de crearla y que asume su mismo nombre. Si lo abrimos, vemos que tiene vacía la solapa de Tokens, pero en la de Styles tiene esta regla, que lo que está diciendo es que importe todos los tokens y styles de este otro Design System Object, que se encuentra en el módulo GeneXusUnanimo.



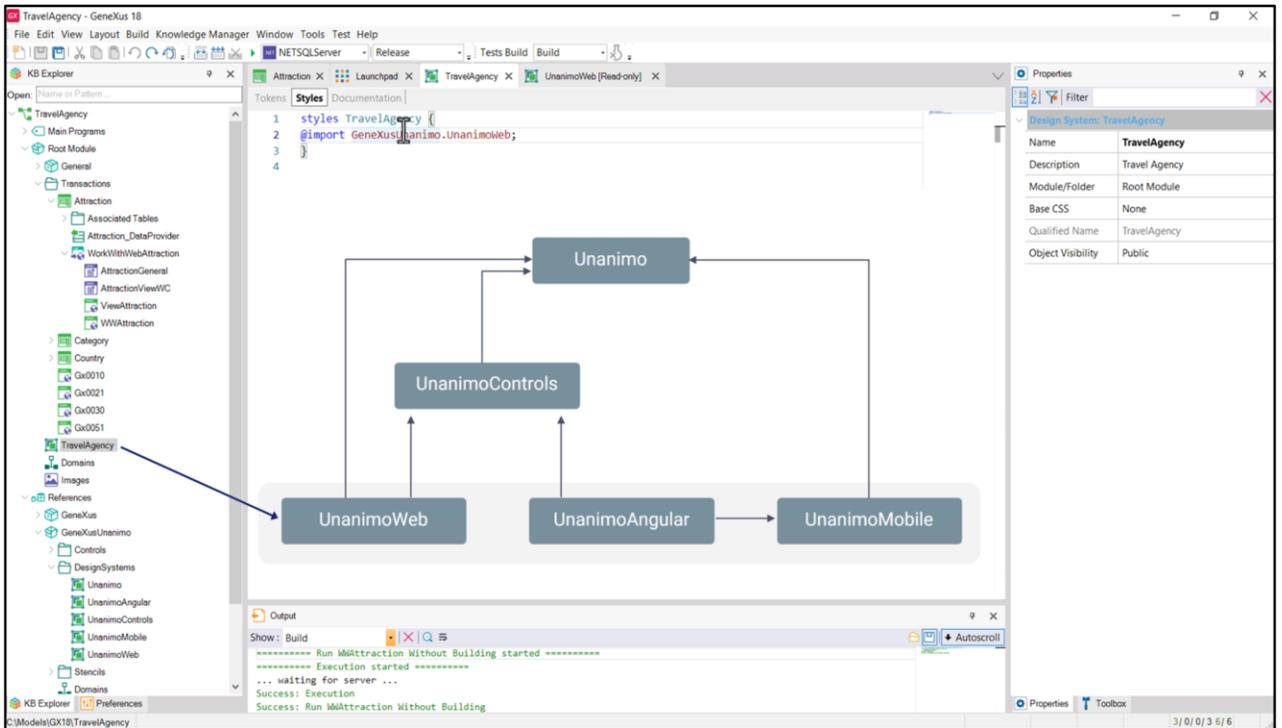
¿Y dónde estaba ese módulo GeneXusUnanimo? Venía instalado en forma predefinida en toda KB, accesible bajo el nodo References.

Si lo abrimos vemos que es read-only, pero, además de sus definiciones específicas de tokens y estilos, está importando otros dos DSOs, estos, que están en el mismo módulo.

Así que se construye sobre estos otros dos.



Y por otro lado, si abriéramos estos otros que vemos aquí, lo que llegaremos a ver es este árbol, donde el UnanimosMobile solo importa el Unanimos, y el UnanimosAngular importa al mobile y también al UnanimosControls.



Si el Unanimoweb será el que se utilizará por Default en el Frontend Web, pues, como vimos, utiliza un DSO con el mismo nombre que la KB que importa a éste...

The screenshot shows the Genexus 18 IDE interface. The main workspace displays a diagram with the following components and relationships:

- Unanimous** (top center)
- UnanimousControls** (middle left)
- UnanimousWeb** (bottom left)
- UnanimousAngular** (bottom center, highlighted in orange)
- UnanimousMobile** (bottom right)

Arrows indicate dependencies: UnanimousWeb, UnanimousAngular, and UnanimousMobile all depend on Unanimous. UnanimousControls also depends on Unanimous. UnanimousAngular has a dependency on UnanimousMobile.

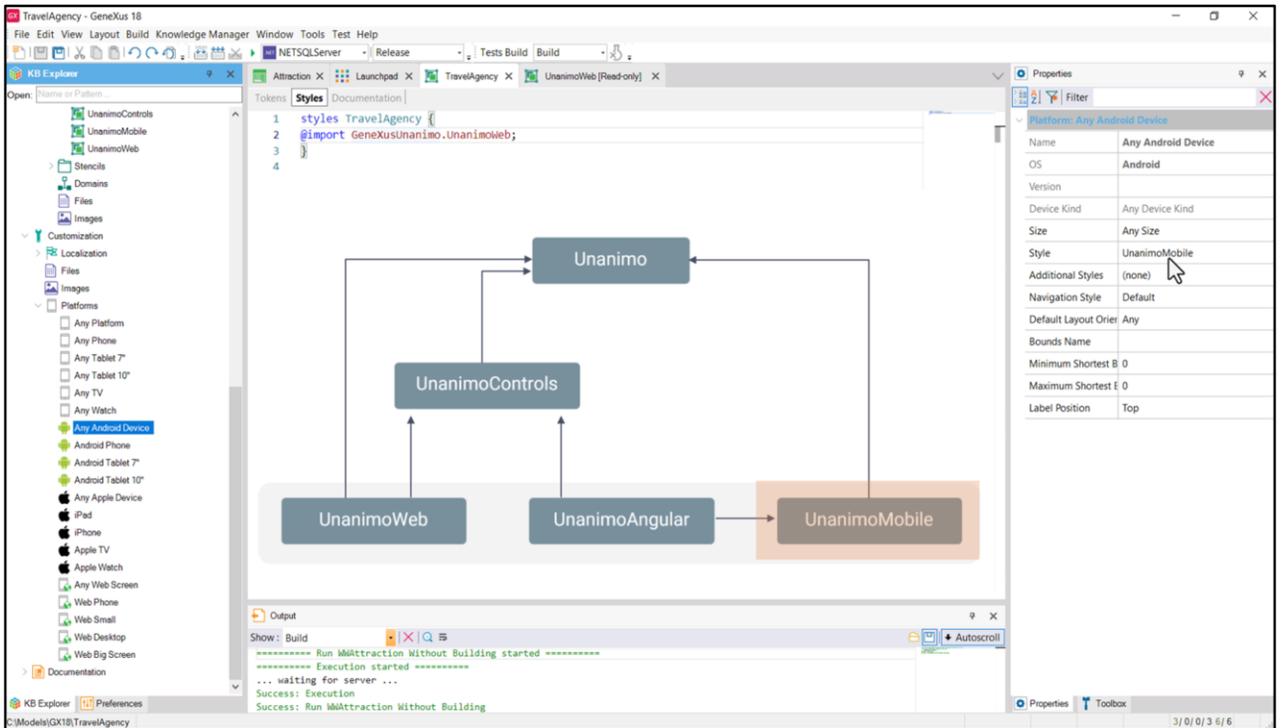
The Properties panel on the right shows the following settings for the selected component:

Platform: Any Web Screen	
Name	Any Web Screen
OS	Web
Version	
Device Kind	Any Device Kind
Size	Any Size
Style	UnanimousAngular
Additional Styles	(none)
Navigation Style	Default
Default Layout Orier	Any
Bounds Name	
Minimum Longest B	0
Maximum Longest E	0
Label Position	Top

The Output window at the bottom shows the following log:

```
Show: Build
===== Run MAttraction Without Building started =====
===== Execution started =====
... waiting for server ...
Success: Execution
Success: Run MAttraction Without Building
```

... claramente el UnanimousAngular será el default para el Frontend también Web pero Angular...



...y el UnanimoMobile será el default para Frontend Mobile (para toda plataforma, tanto Android, como Apple).

The screenshot shows the Genexus IDE interface. On the left, the 'KB Explorer' shows a project structure for 'TravelAgency' with sub-projects for 'NETSQLServer', 'Back end', 'Data Stores', 'Services', 'Front end', 'Web (.NET)', 'Web (Angular)', 'Deployment', 'Patterns', and 'Workflow'. The main editor displays a 'Styles' document with the following code:

```
1 styles TravelAgency {
2   @import GeneXusUnanimo.Unanimoweb;
3 }
4
```

In the center, a diagram illustrates the architecture. A box labeled 'Unanimo' is at the top. Below it is 'UnanimoControls'. At the bottom, there are three boxes: 'UnanimoWeb', 'UnanimoAngular', and 'UnanimoMobile'. Arrows indicate dependencies: 'UnanimoWeb' and 'UnanimoAngular' depend on 'UnanimoControls', and 'UnanimoWeb', 'UnanimoAngular', and 'UnanimoMobile' all depend on 'Unanimo'. 'UnanimoAngular' also has an arrow pointing to 'UnanimoMobile'.

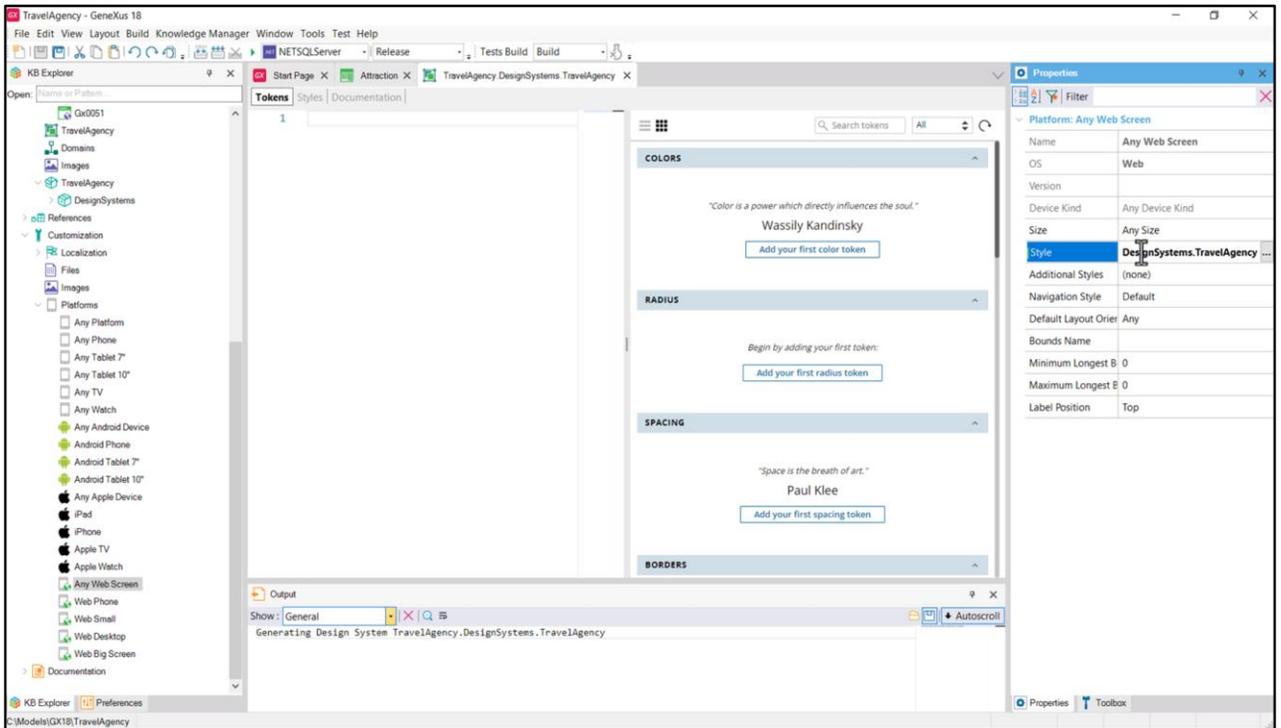
On the right, the 'Properties' panel is open for the 'Generator: Frontend (Front end)'. It shows various settings:

- Name: Frontend
- Main Platform: Angular
- Dynamic Services UI: False
- Services URL: https://apps5.genexus.com/ldb2...
- SSL Pinning Pin Set
- Smart Devices Cache: On
- Visual Testing Report
- Generators:
 - Generate Android: False
 - Generate Apple: False
 - Generate Angular: True
- Angular Specific:
 - Build Mode: Prototype
 - Run Target: Default
 - Services URL Mod: Host Relative
 - Setup Command: npm update
 - Run Command: npm start
 - Deploy Command: npm run deploy

At the bottom, the 'Output' window shows the following log:

```
Show: Build
***** Run MAttraction Without Building started *****
***** Execution started *****
... waiting for server ...
Success: Execution
Success: Run MAttraction Without Building
```

Como no tenemos aún ningún Front end para estas plataformas no podemos ver en acción estos DSOs. De todas formas nosotros no queremos utilizarlos. Es que vamos a querer construir la aplicación Customer-facing que tendrá otro sistema de diseño, el que nos diseñó Chechu, el que veníamos analizando, por lo que crearemos el DSO para Angular (que es la plataforma en la que nos vamos a centrar buena parte del curso) desde cero. Y luego, sobre el final, veremos el caso nativo.

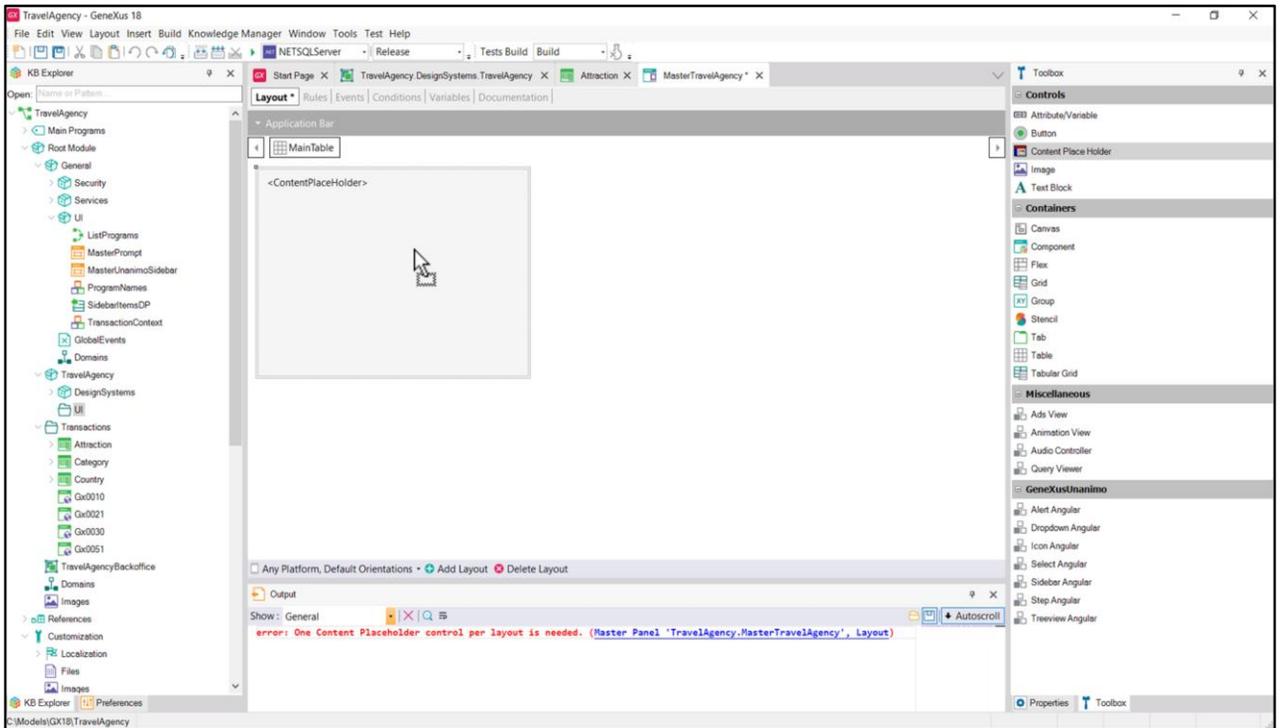


Para dejar bien separados los objetos que serán propios de nuestra aplicación Customer-facing de todos los objetos que vienen default en la KB, y de los del Backoffice, vamos a crear un módulo al que por ahora llamaré TravelAgency.

Y dentro crearé otro módulo, DesignSystems, para colocar los objetos DSs que utilizaremos. Luego veremos por qué serán varios y no uno solo. Ya podemos crear el que comandará el estilo de la aplicación Angular que desarrollaremos en principio para tamaño Desktop. Vamos a llamarle TravelAgency y definir que se utilice este para esa plataforma.

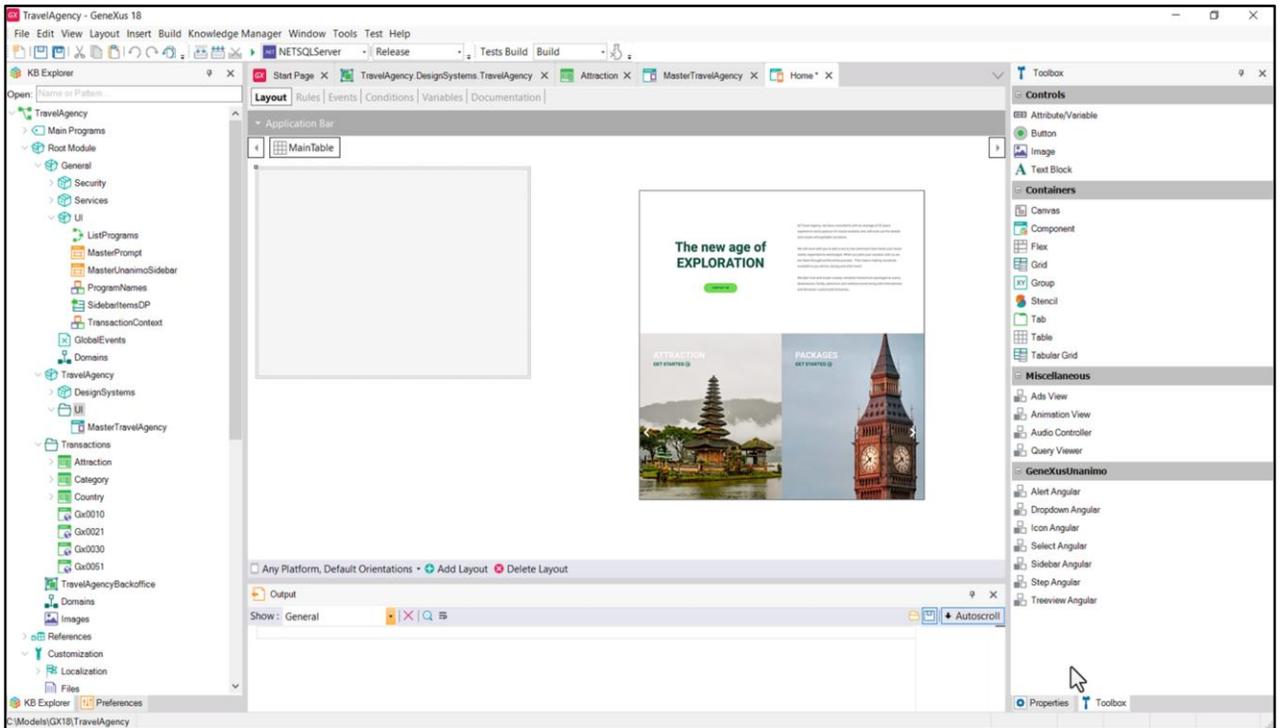
Tenemos ahora dos DSOs TravelAgency, pero elegimos el del módulo DesignSystems que acabamos de crear. Y vemos que todas las plataformas más específicas de la web Angular heredarán ese DSO.

Pudimos llamarle igual a este otro DSO porque lo colocamos dentro de un módulo. Pero vamos a cambiarle el nombre a este otro para que no se nos confundan.

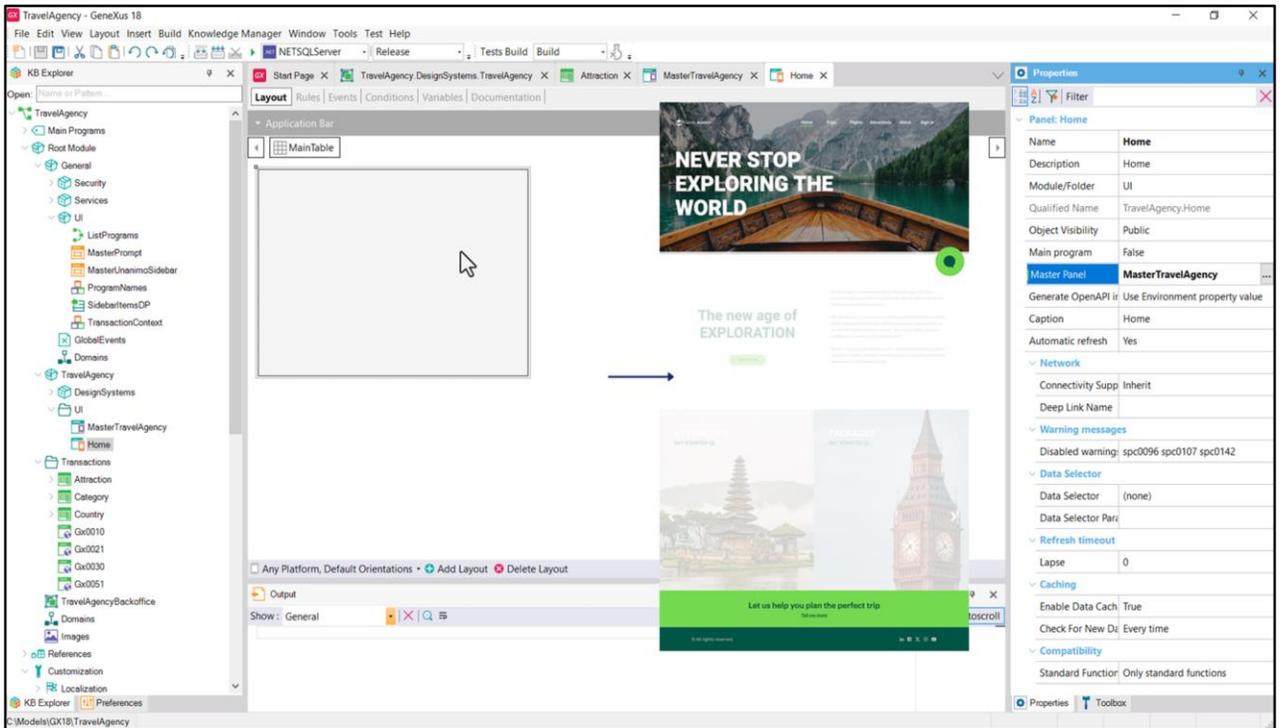


Ahora vamos a crear un folder dentro del módulo para organizar allí dentro todos los objetos con interfaz de usuario.

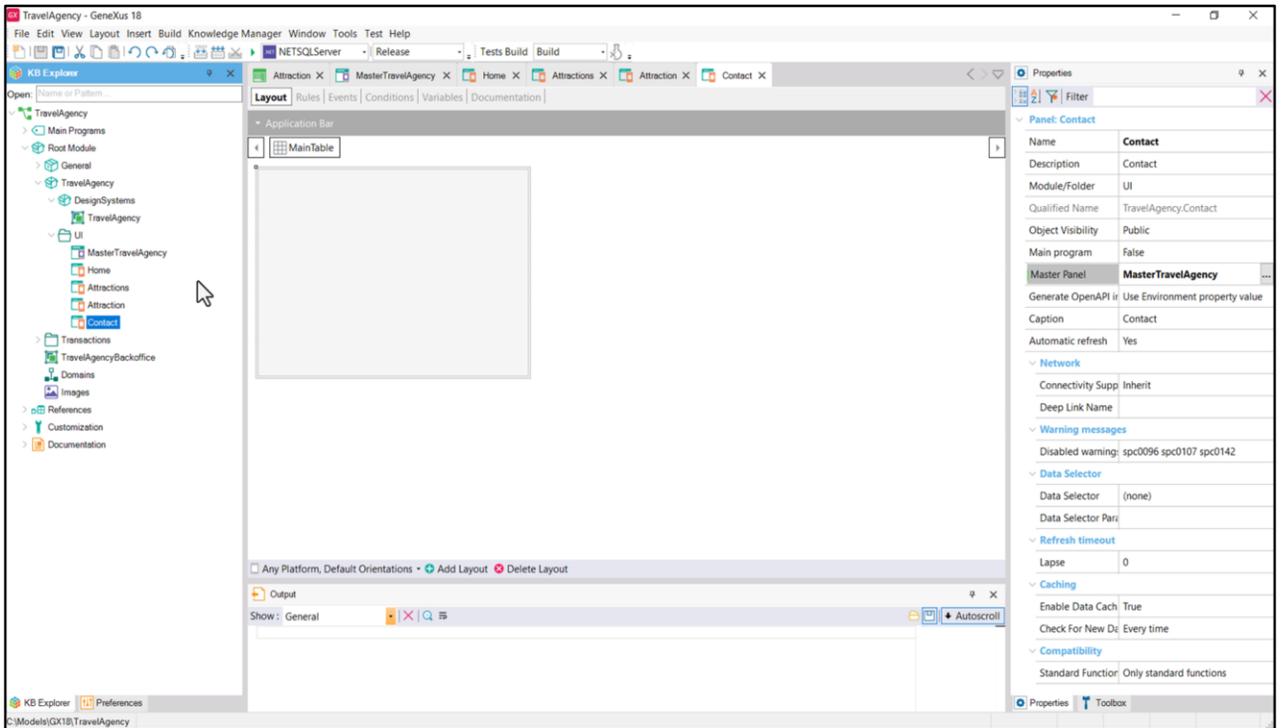
Así que creemos primero que nada el Master Panel, al que llamaré así. Si intento grabarlo vacío me dará este error. Es que es obligatorio colocar el control ContentPlaceholder, que es donde se cargarán los paneles que tengan a éste como su Master Panel. Ahora sí se nos permitirá grabar el objeto.



Vamos a crear entonces el primer panel, Home, que será el que implementará esta parte de la pantalla que vimos en Figma.

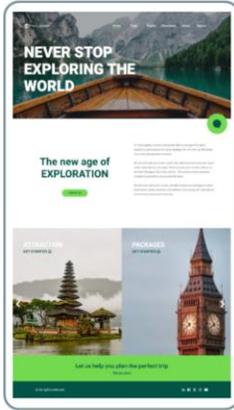


Y le tenemos que indicar que se cargue dentro del Master Panel que recién creamos. Es decir, se cargará aquí dentro...

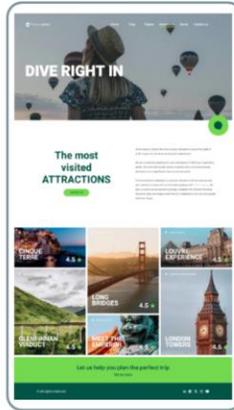


Y le tenemos que indicar que se cargue dentro del Master Panel que recién creamos. Es decir, se cargará aquí dentro...

Bueno, de igual manera crearemos los otros tres paneles, por ahora vacíos. Y con el mismo Master Panel.



Home



Attractions



Attraction



Contact

Bueno, llegados a este punto ya tenemos la KB inicializada para poder empezar a trabajar en cada uno de los objetos e implementar cada una de las porciones de layout. Así que los esperamos en la siguiente clase.

GX

GeneXus by Globant

GeneXus[™]
by Globant

training.genexus.com