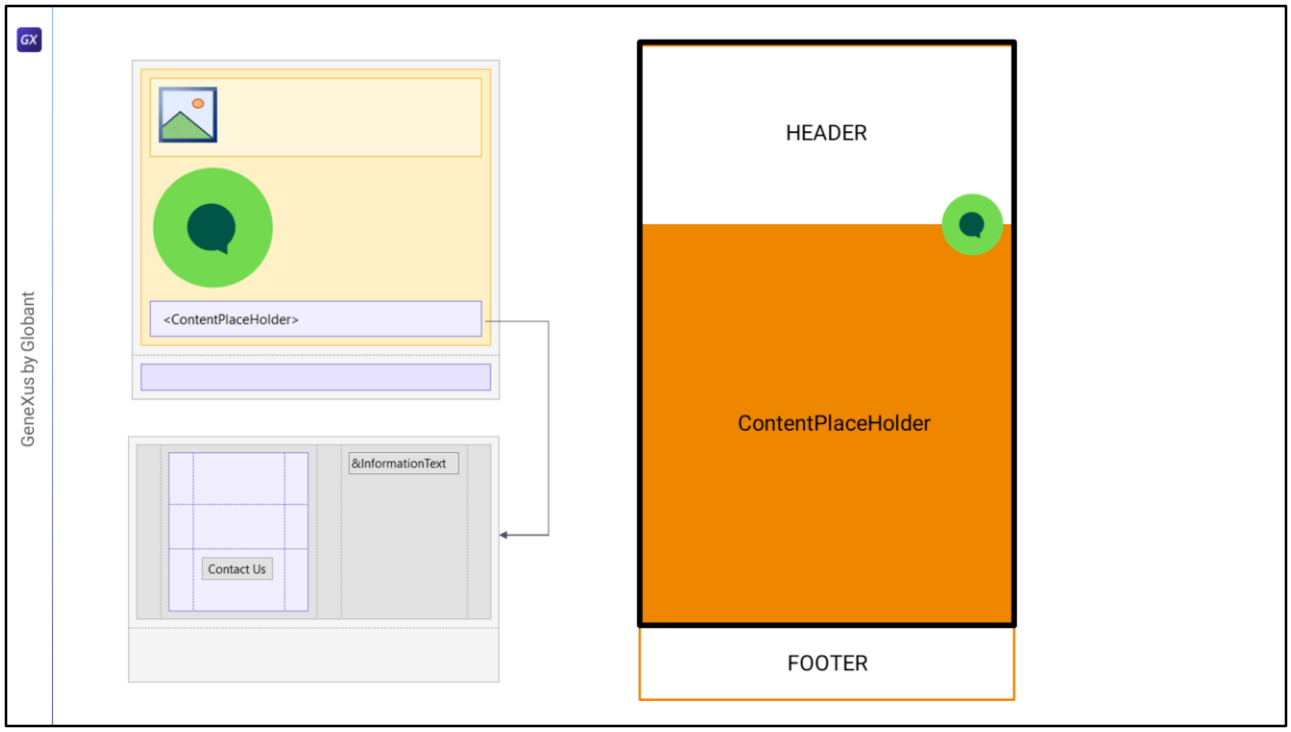
GeneXus by Globant

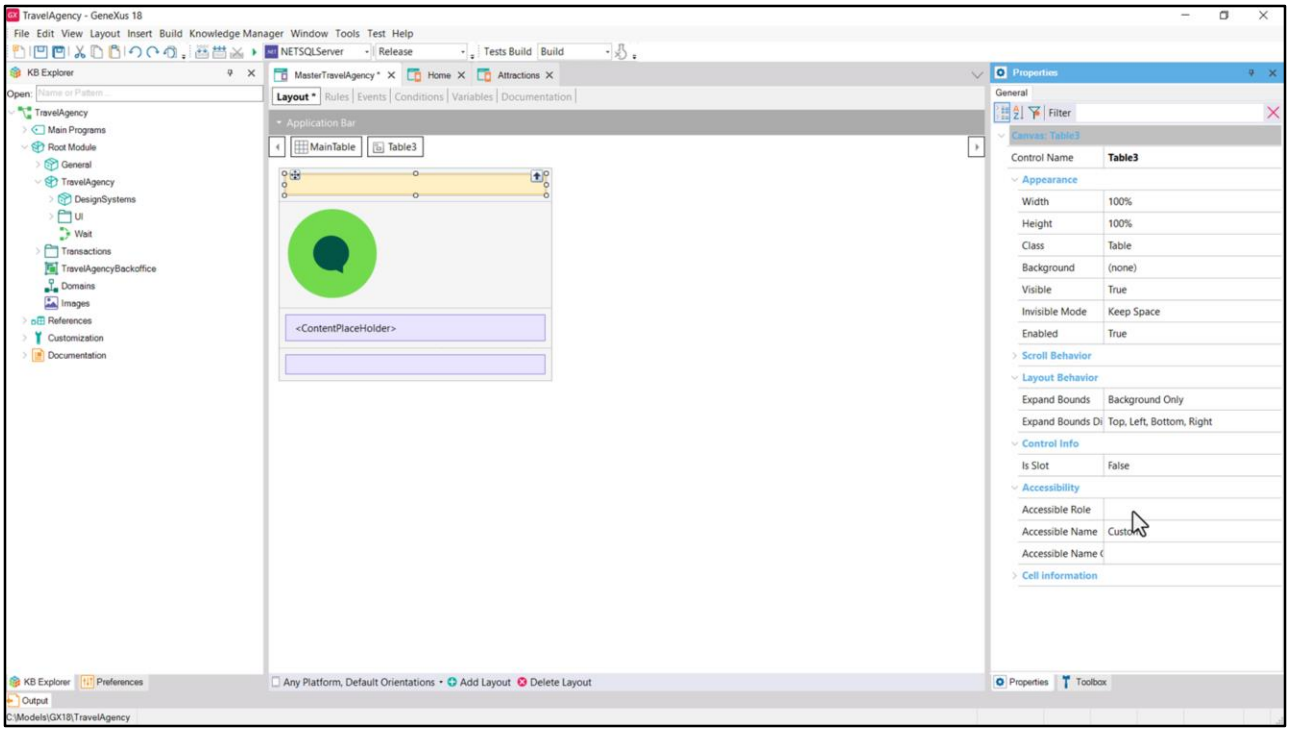
Header in GeneXus: background image and chatbot button

Cecilia Fernández

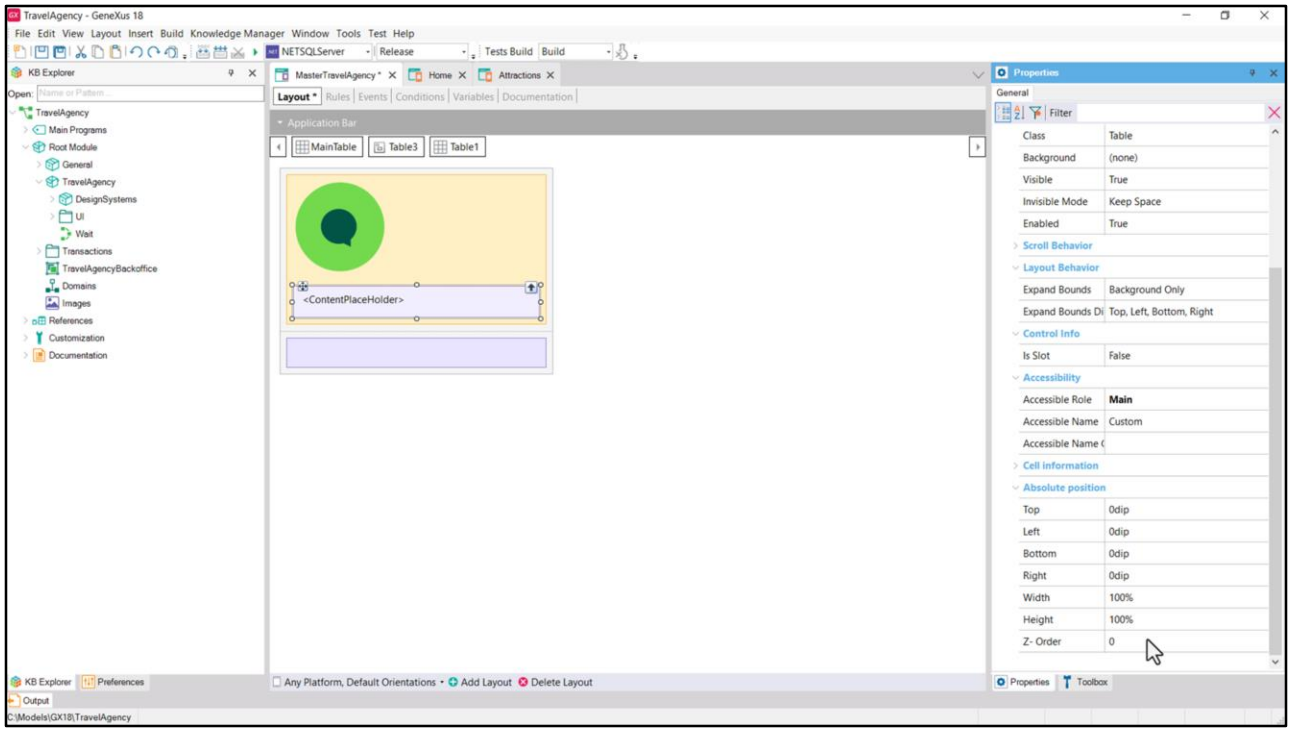


En el video anterior analizamos a nivel conceptual dos posibles implementaciones para el Master Panel.

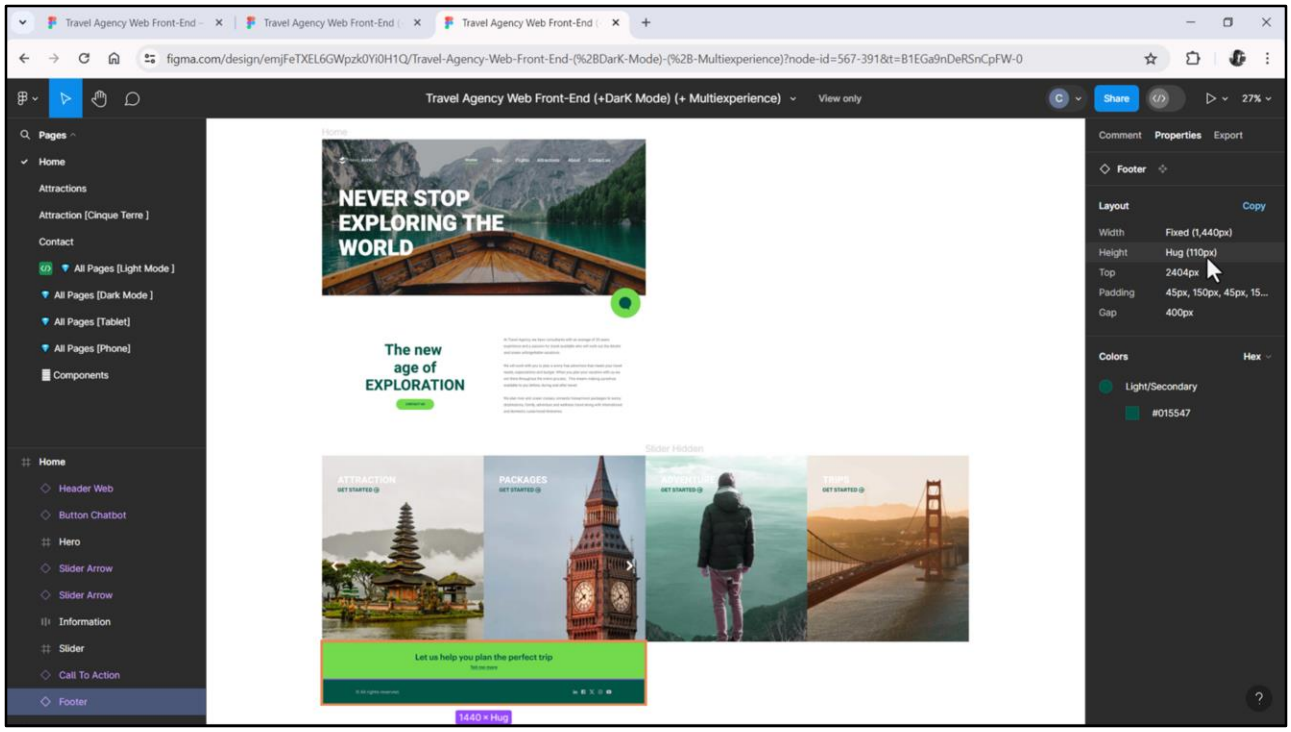
Vayamos a implementar la segunda solución, que fue la que nos pareció conceptualmente la más apropiada.



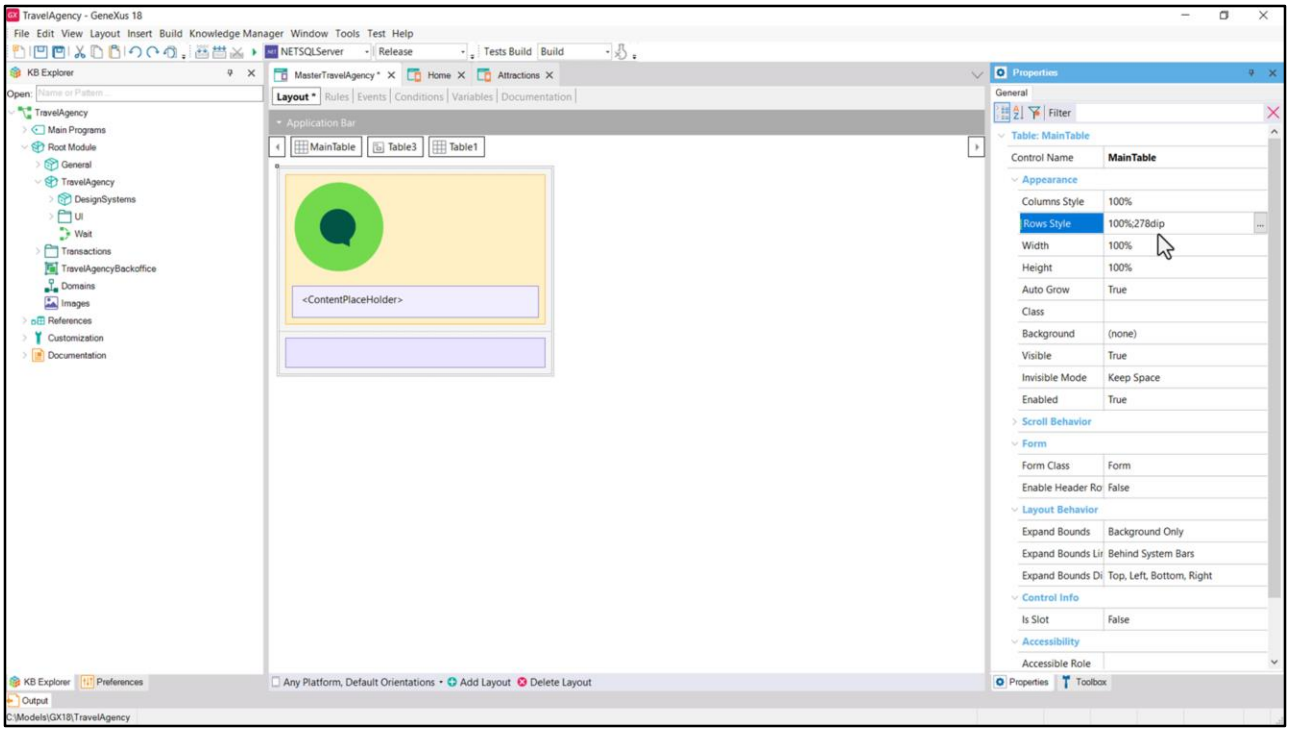
Aquí tenemos la tabla Main. Tendremos que insertar un control canvas en la primera fila, veamos sus propiedades. A esta no le vamos a colocar Role porque tendrá que agrupar al Header y al ContentPlaceHolder.



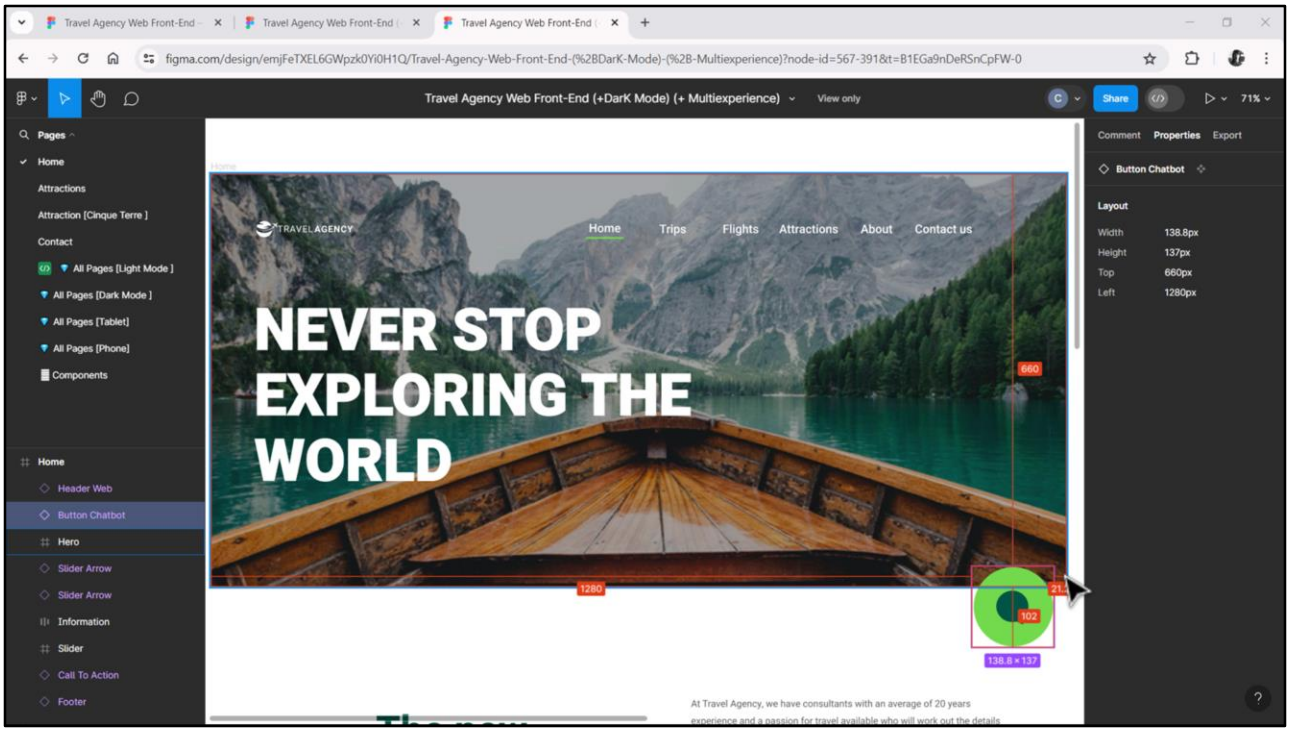
Arrastremos dentro tanto el botón del chatbot, y vemos cómo aparecen ahora las propiedades Absolute Position…
Como a la tabla con el contentplaceholder y vemos que también aparecen las propiedades Absolute Position…



Ahora la tabla Main nos ha quedado con dos filas. La del Footer debe tener un alto de 168 + 110, esto es, de 278 dips.

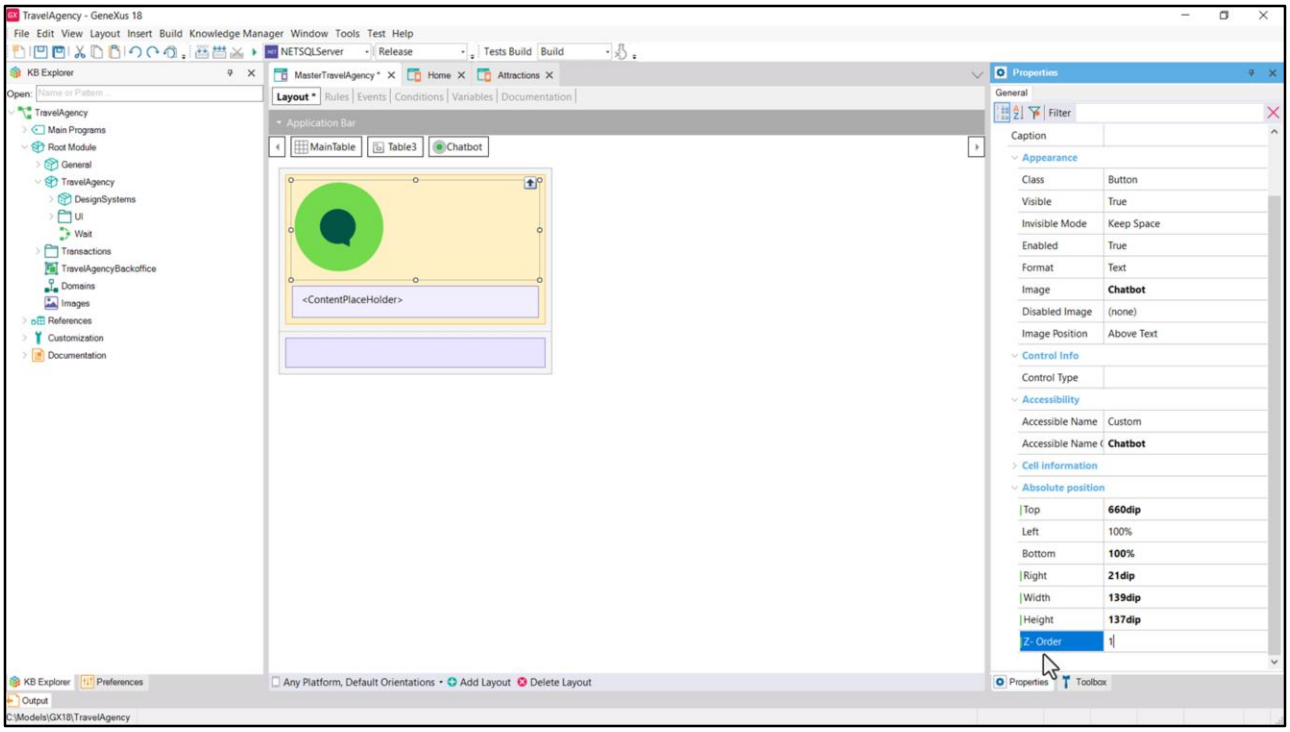


Entonces a la Main table le colocamos en Rows Style: 100% para la primera fila, y 278 dips para la segunda.



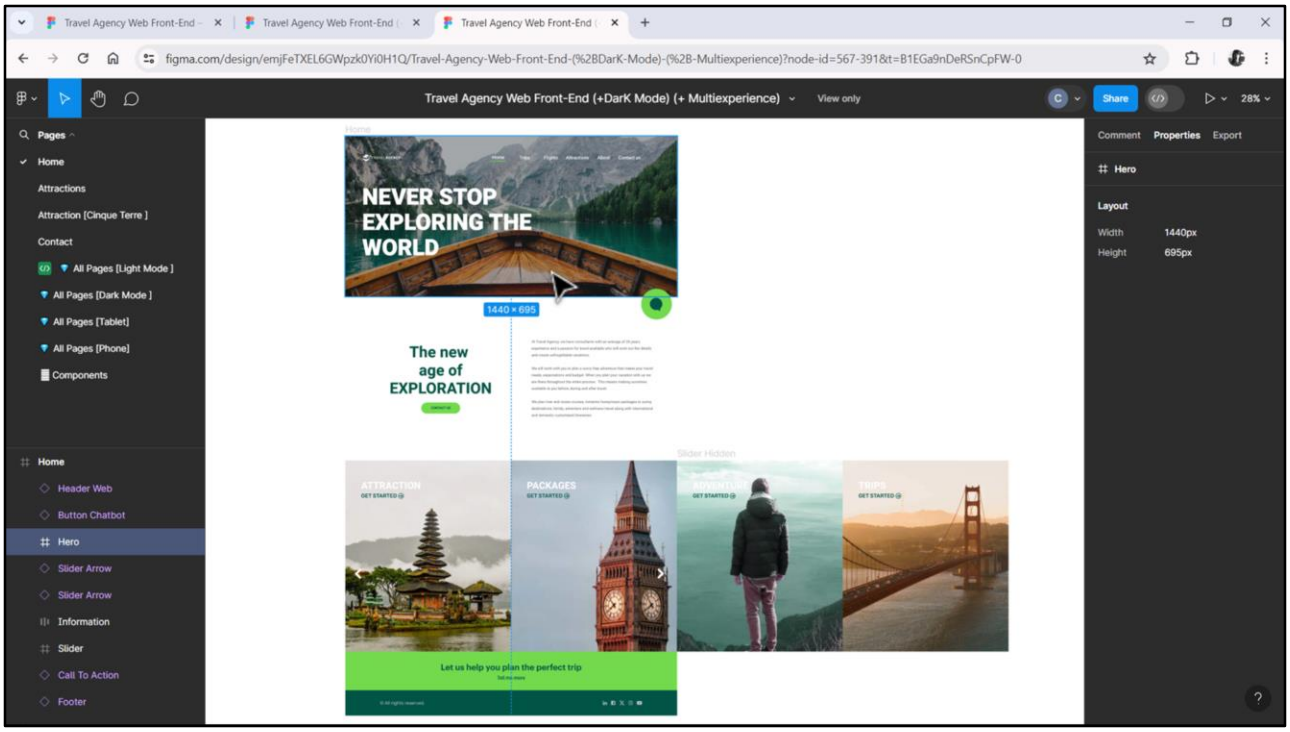
Vamos a darle las posiciones absolutas a los dos controles que tenemos por el momento.

Del botón del chatbot teníamos que su ancho era de 139, y estaba a 21 de la derecha…
Y su alto era de 137 y estaba a 660 de arriba. Entonces…

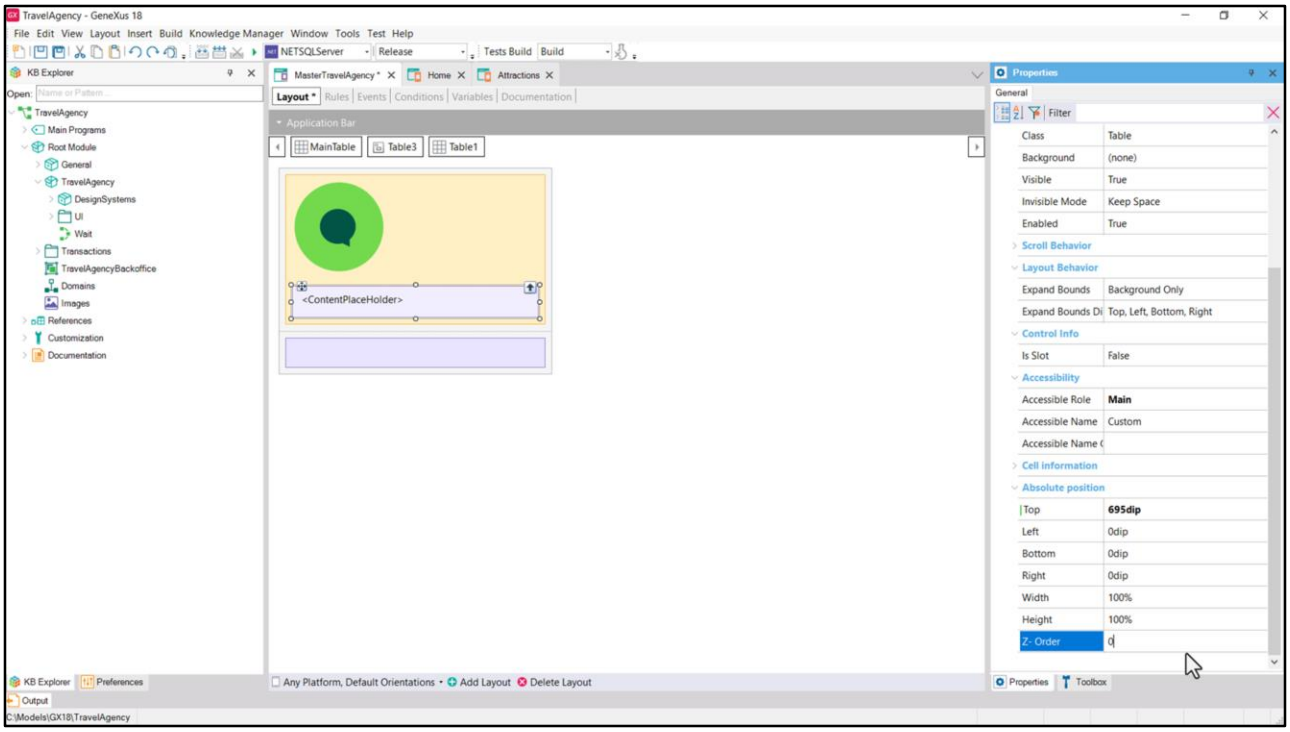


Veamos que si fijamos la Right en 21 y Width en 139, automáticamente la Left queda en 100%. E igualmente si fijamos la Top en 660 y Height en 137, Bottom queda de 100%.

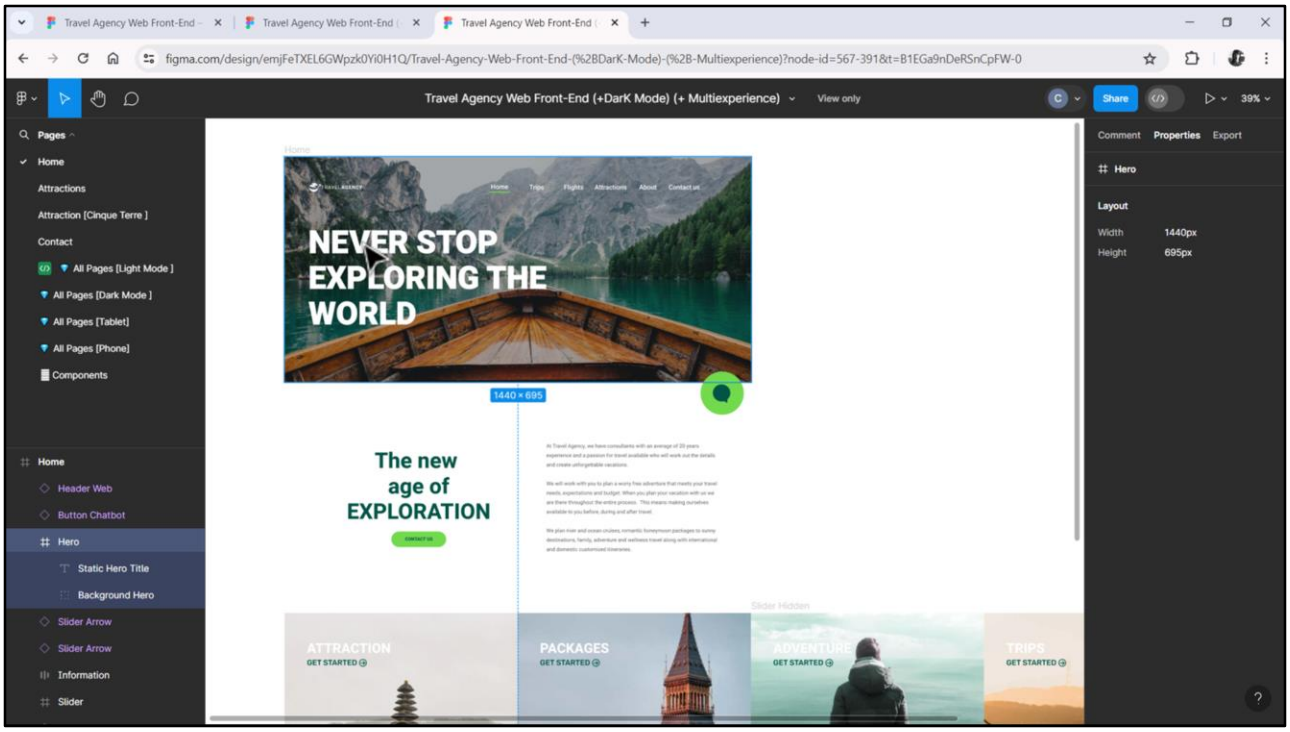
Como este botón es el que tiene que quedar en la capa de más arriba, le colocaremos 1 para el Z-order, porque el contentplaceholder y el Header (que aún no insertamos) pueden estar en la misma capa 0, la de más abajo.



Ahora debemos especificar el posicionamiento absoluto de la tabla con el contentplaceholder. Sabemos que debe empezar aquí, es decir, a 695 dips del Top del canvas… y que se pegue al resto de los bordes del canvas.

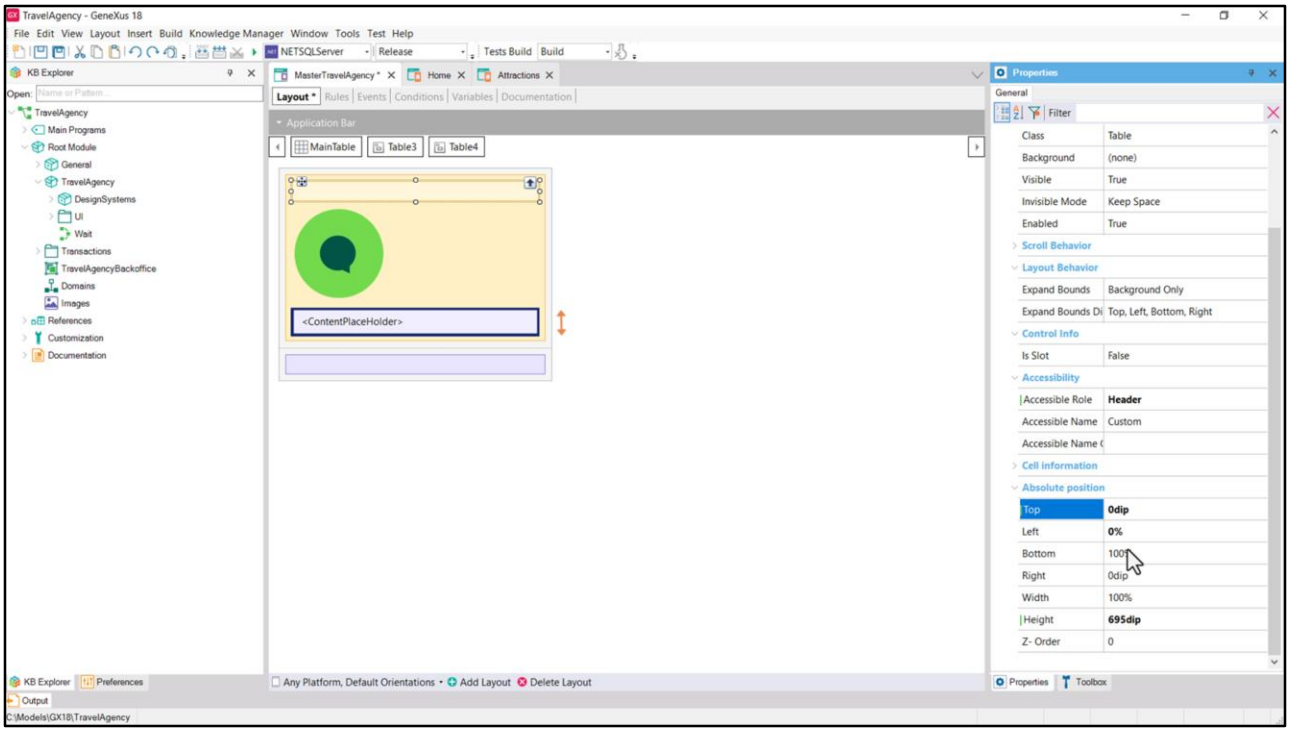


Es decir: Top 695, y lo demás tal como está: 0 de izquierda, 100% de ancho y 0 de derecha. 100% de alto y 0 de Bottom. Y Z-order también 0.



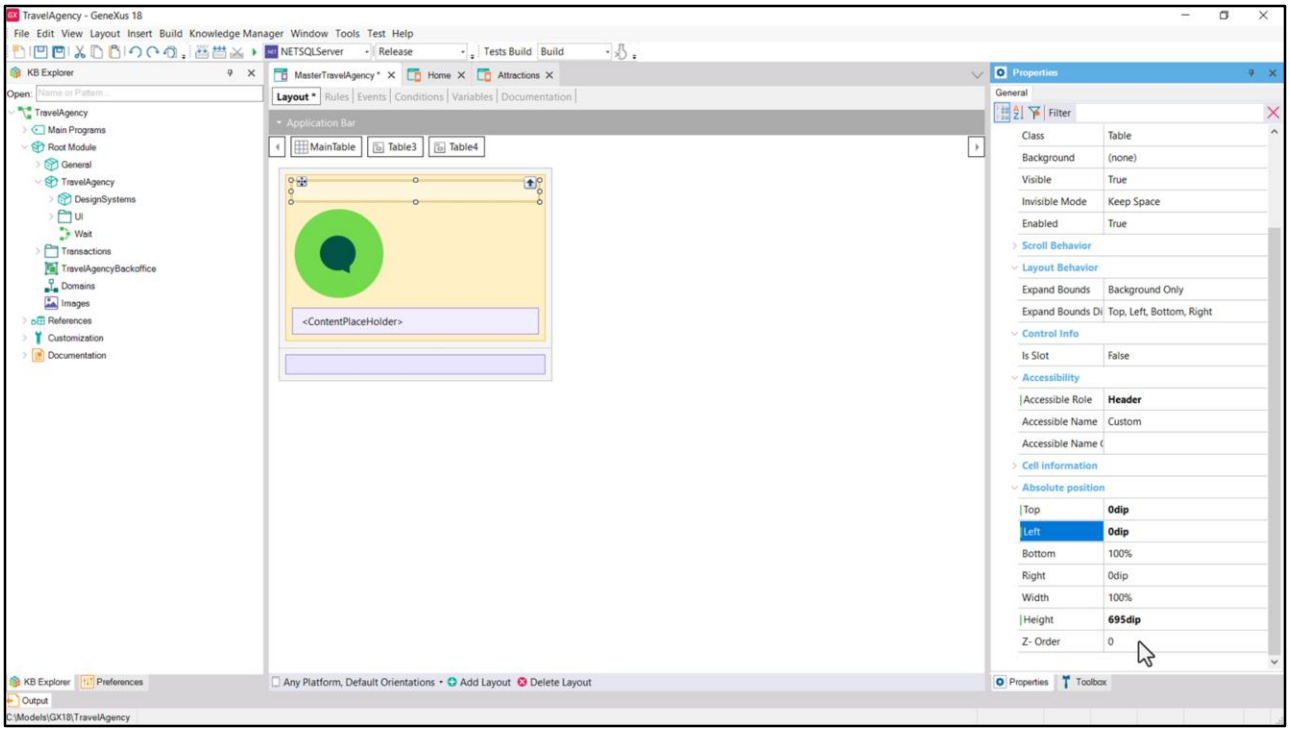
Ahora nos falta implementar el Header propiamente dicho, que contendrá tanto la imagen de fondo, como el menú, el loco, y este texto.

Necesitaremos otro canvas para superponer todo esto. No puede ser el mismo en el que está el botón y la tabla con el contentplaceholder porque necesitamos poder asignarle Role Header para la accesibilidad. Así que necesitamos un contenedor aparte.

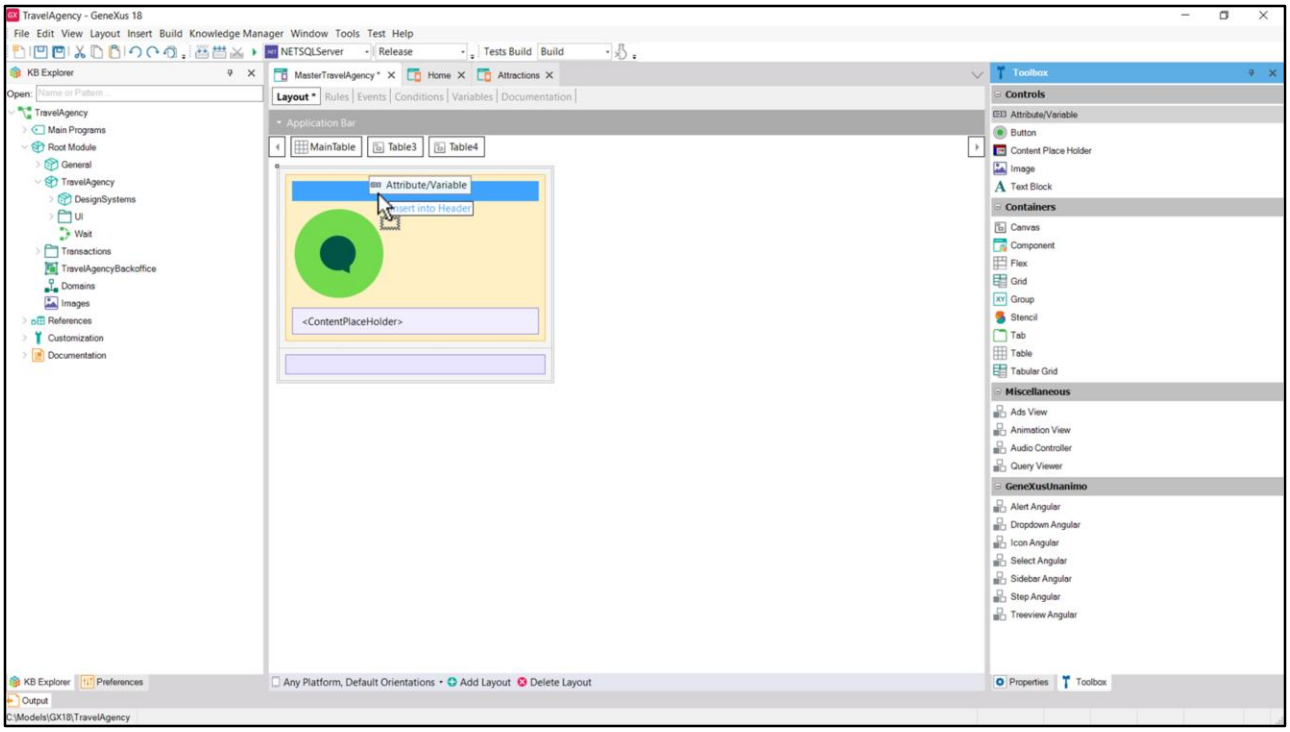


Insertamos, entonces, otro Canvas, al que le llamaremos Header y le colocaremos en Accessible Role el valor Header.

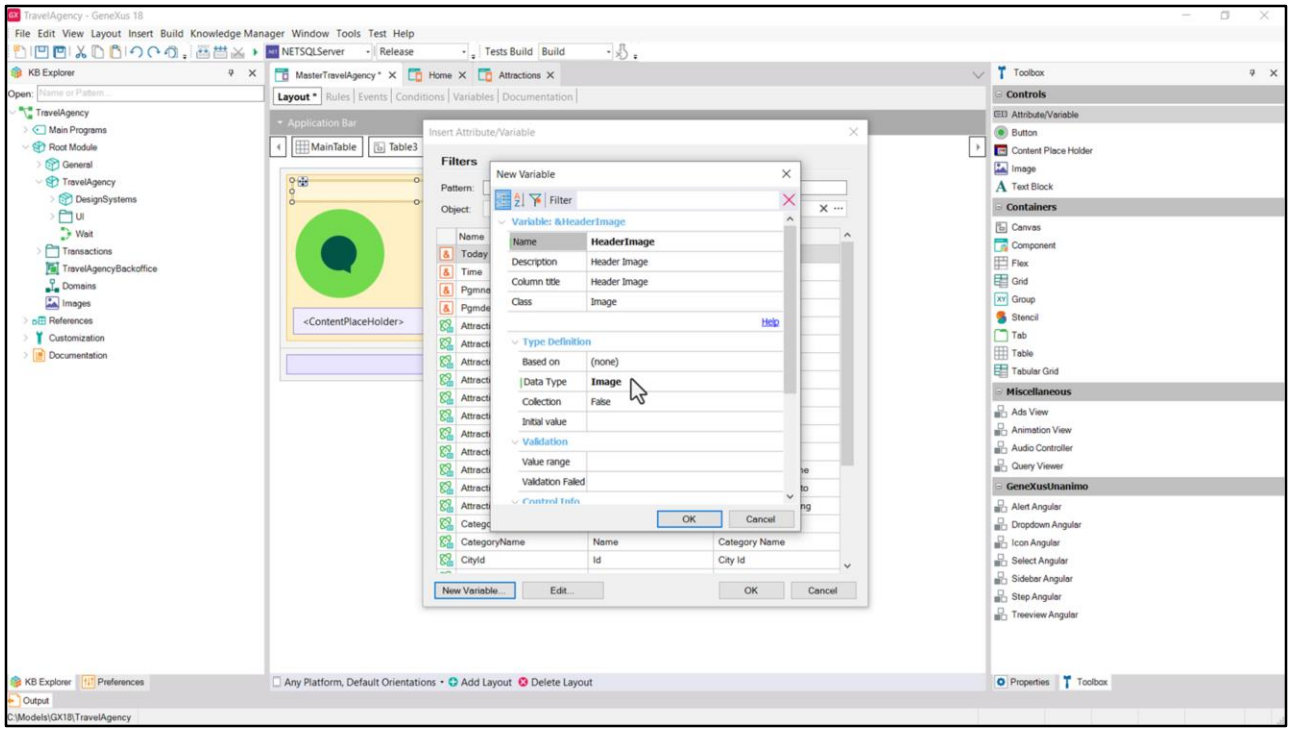
¿Y cuál será su posicionamiento absoluto respecto al canvas exterior?
Primero que nada, su alto será de 695 dips, que era el alto de la imagen de fondo. Queremos que esté pegado al Top, así que 0 dips de arriba, lo que lo deja de Bottom al 100% restante (que va a corresponder al alto del contentplaceholder).



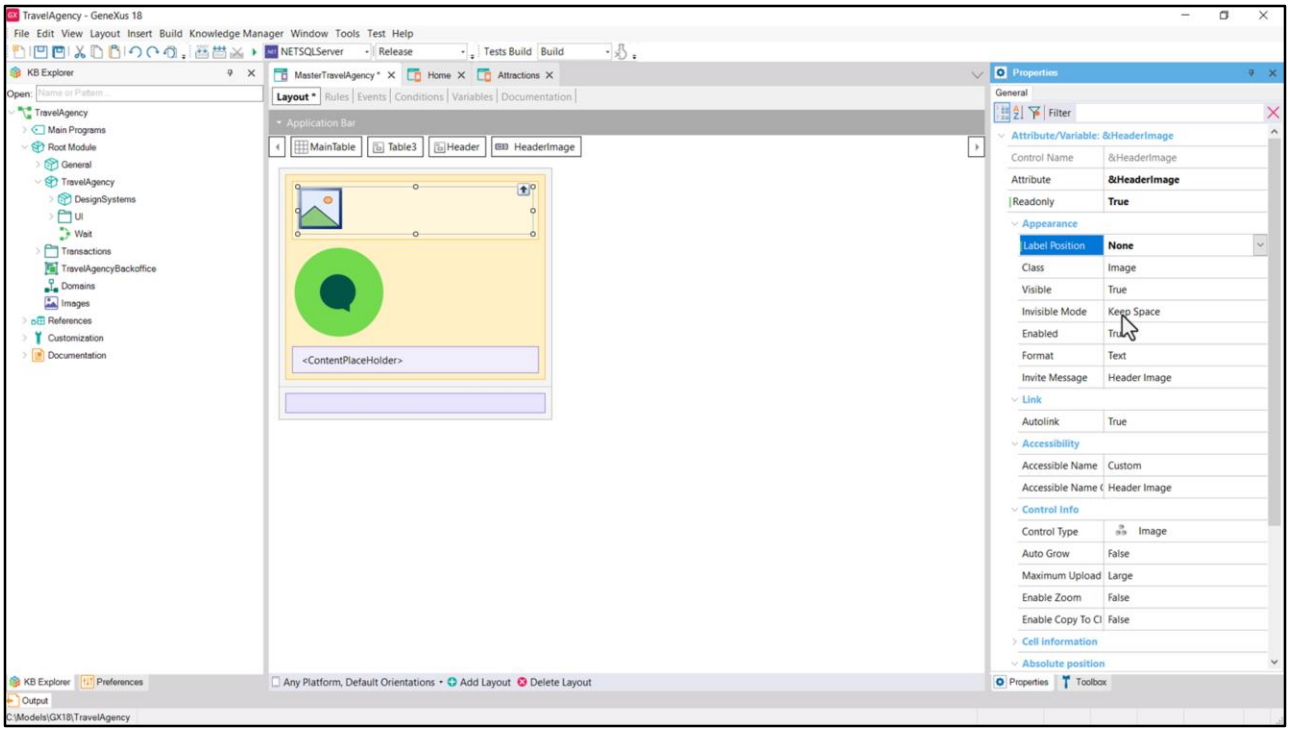
Y de izquierda y derecha quedará también a 0 dips, es decir, pegado, ya que su ancho será del 100%. La propiedad Z-order, como ya analizamos, quedará con el valor 0.



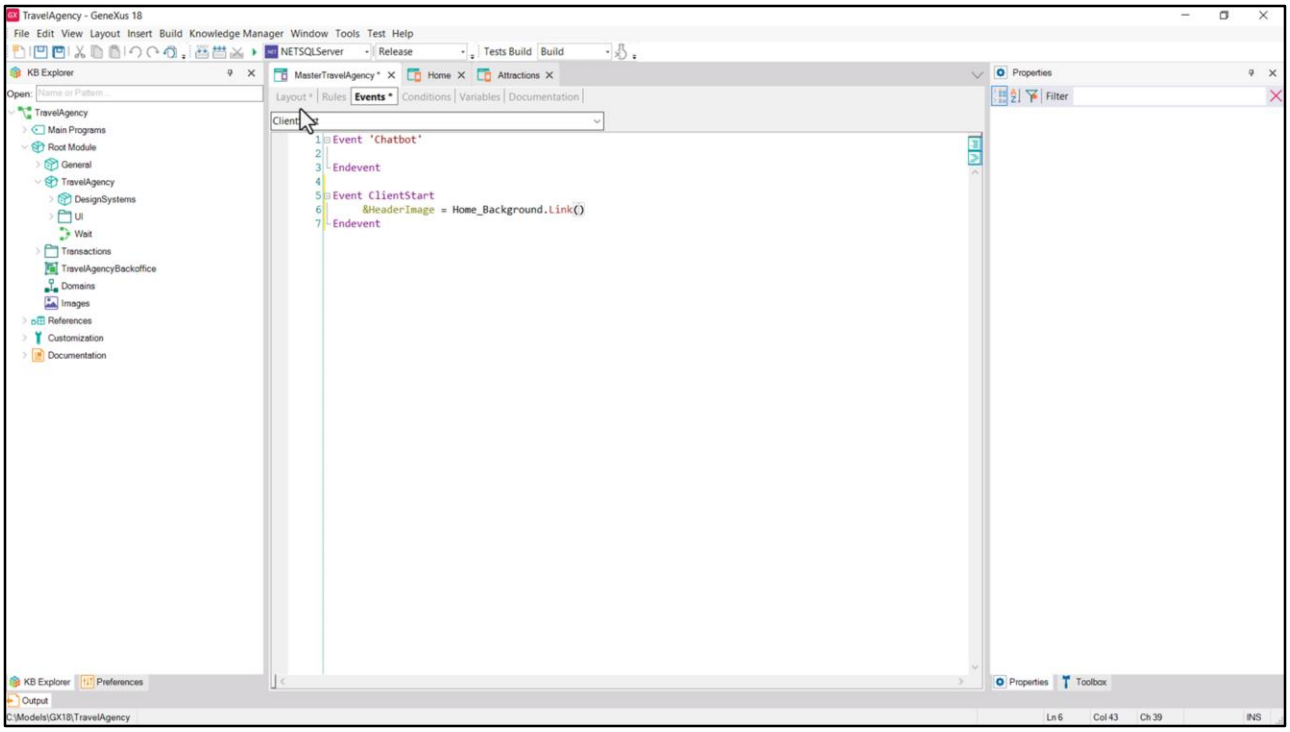
Ahora tengo que implementar la imagen de fondo, esa a la que solemos llamar Hero. Tengo dos opciones: o utilizo un control Image o utilizo un control variable que contenga a la imagen. Utilizaré la segunda alternativa, dado que esta imagen va a variar dependiendo del panel que se esté cargando en el Contentplaceholder en cada oportunidad.



Voy a llamarle HeaderImage y va a ser del tipo de datos Image.

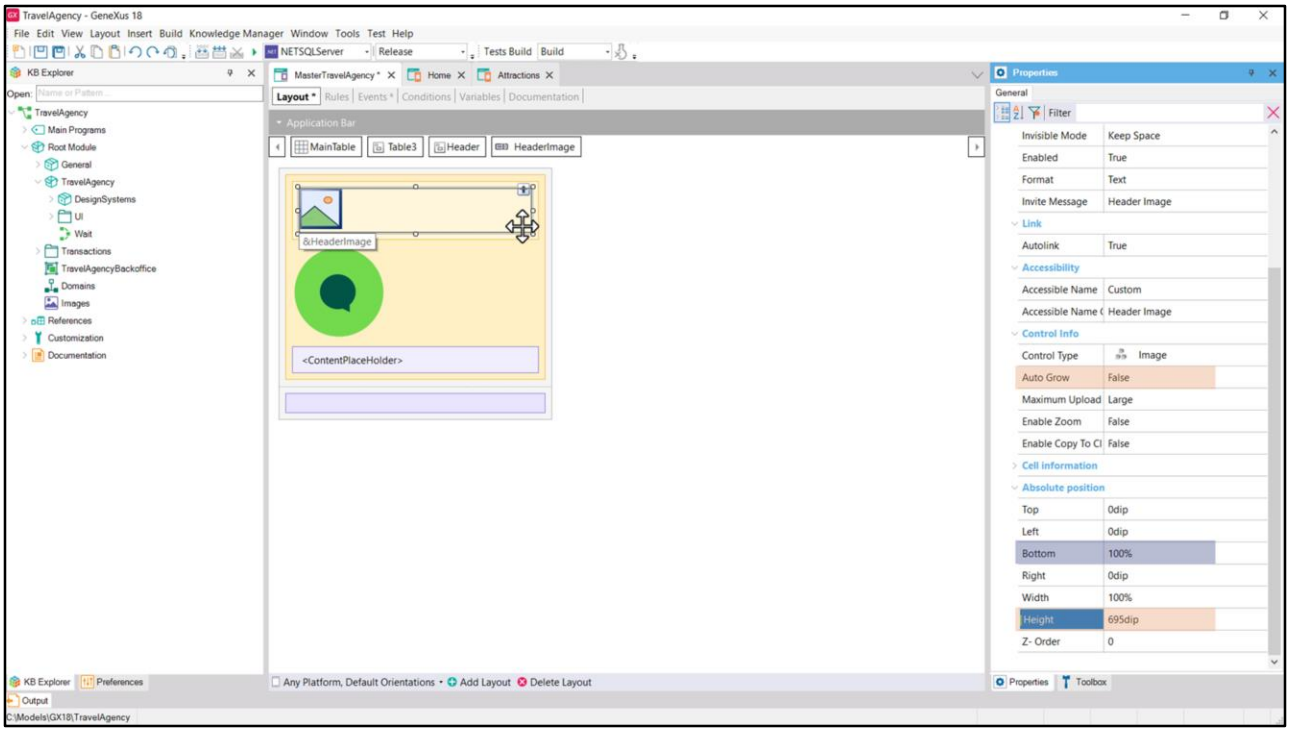


Querré que sea readonly y que no muestre su etiqueta.



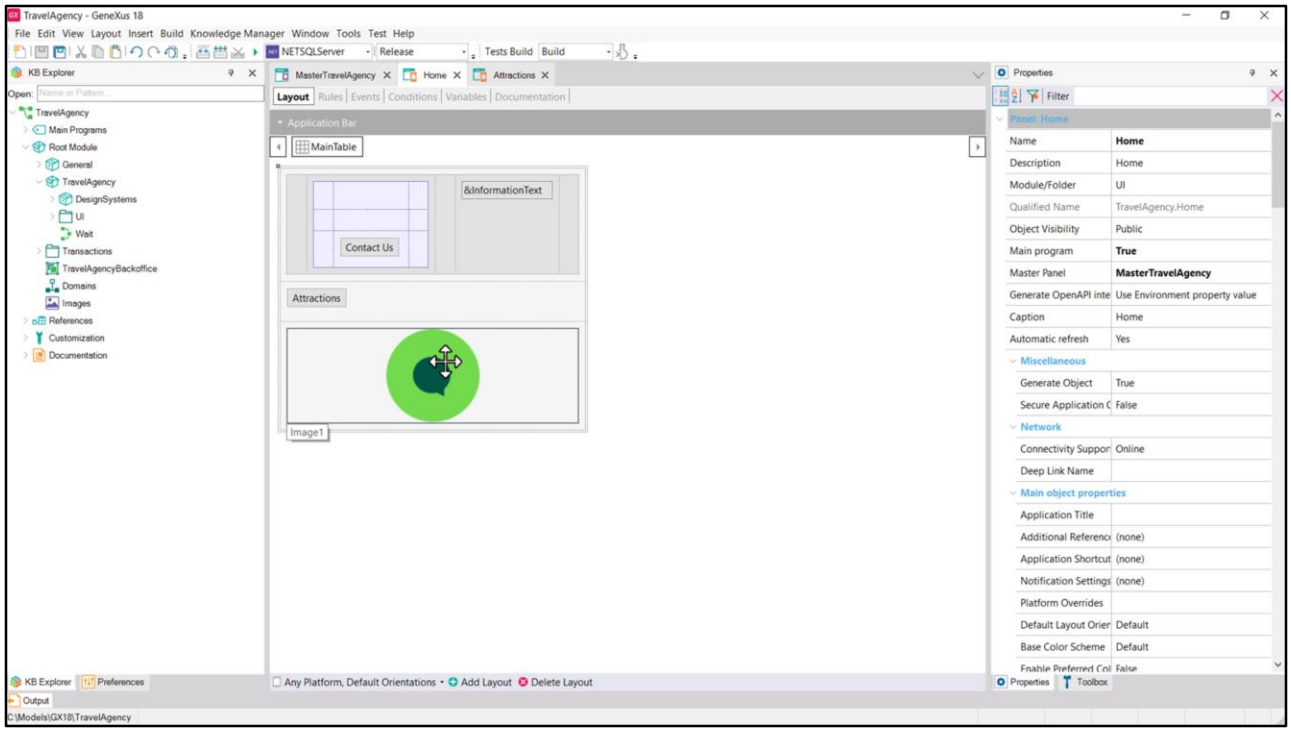
La cargaré provisoriamente en el evento ClientStart, a partir (ctrl-o) del objeto imagen que ya habíamos insertado en la KB en la etapa de preparación.. Y al que habíamos llamado así… Utilizo el método Link.

Después tendremos que hacer variar esta asignación dependiendo de quién se esté cargando, pero eso lo veremos más adelante. Ahora vamos a dejarla fija.

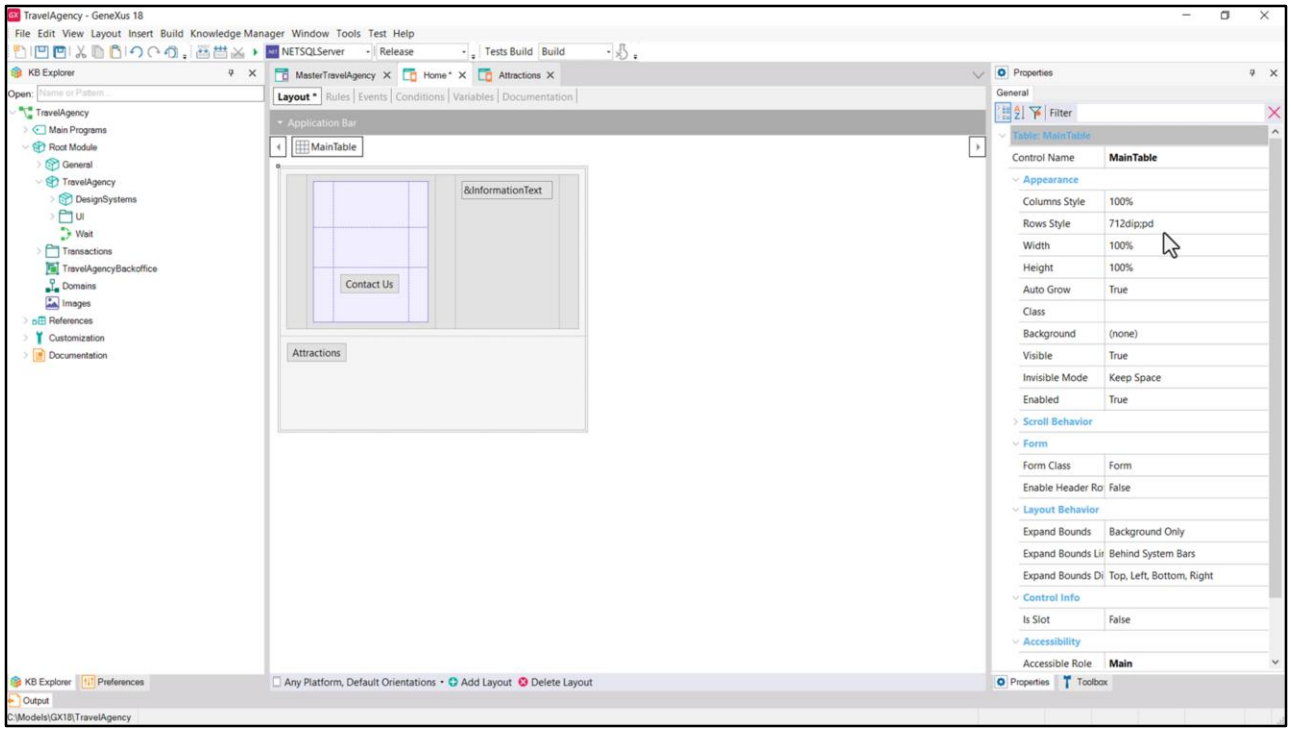


Lo siguiente será indicar el posicionamiento absoluto de la variable en relación al Canvas Header. Diremos, entonces, que su alto será de 695 dips. Y dejaremos en False el Auto Grow. Las demás propiedades las dejamos como están, porque querremos que la imagen se pegue al top y a los bordes laterales. El Canvas, como dijimos en el video anterior, tiene internamente Auto Grow en true, por lo que cualquier control que desborde lo hará expandirse hacia abajo. Pero en este caso solo tenemos la imagen, y fijada a 695 dips, por lo que este bottom de 100%, cuando sea calculado al cargarse la página, será de 0 dips. Es decir, la imagen debería pegarse al canvas por los cuatro costados.

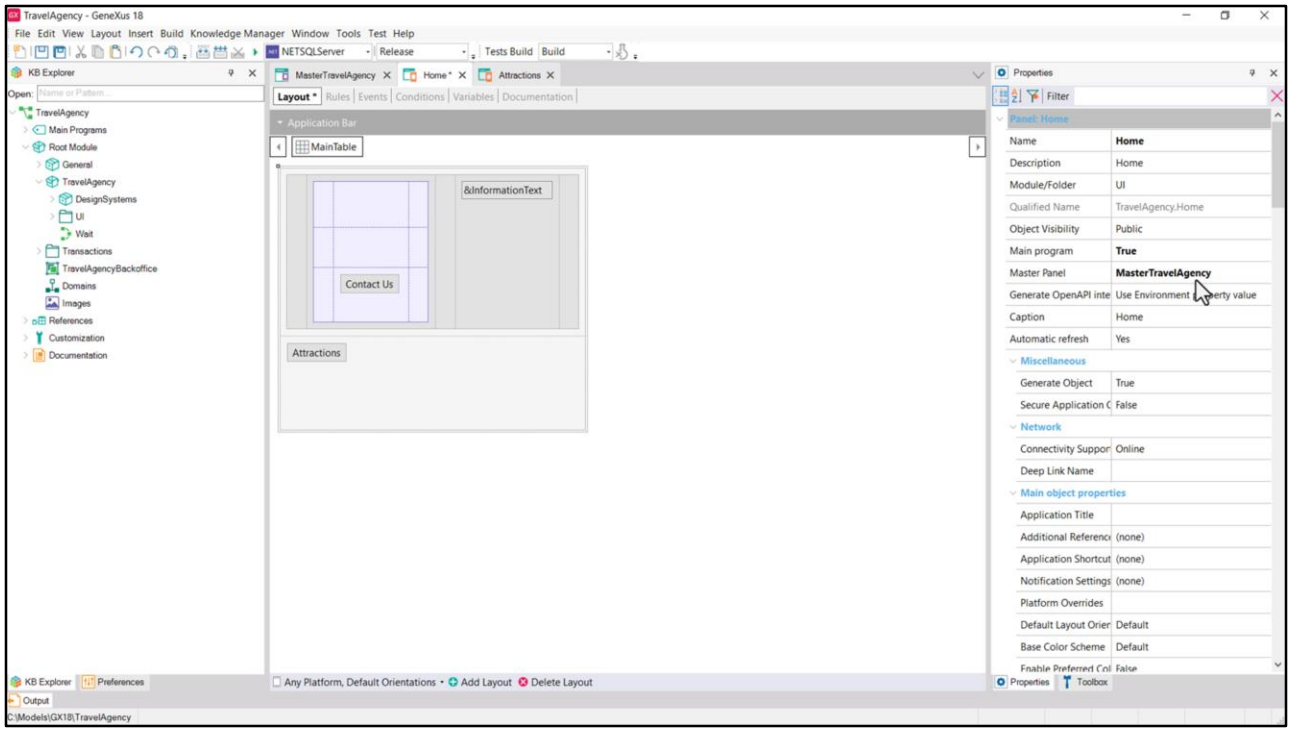
Grabemos el Master Panel.



En el Panel Home habíamos colocado para mostrar ya no me acuerdo qué cosa esta imagen.
La quitamos.
El botón para llamar a Attractions por ahora lo dejamos.

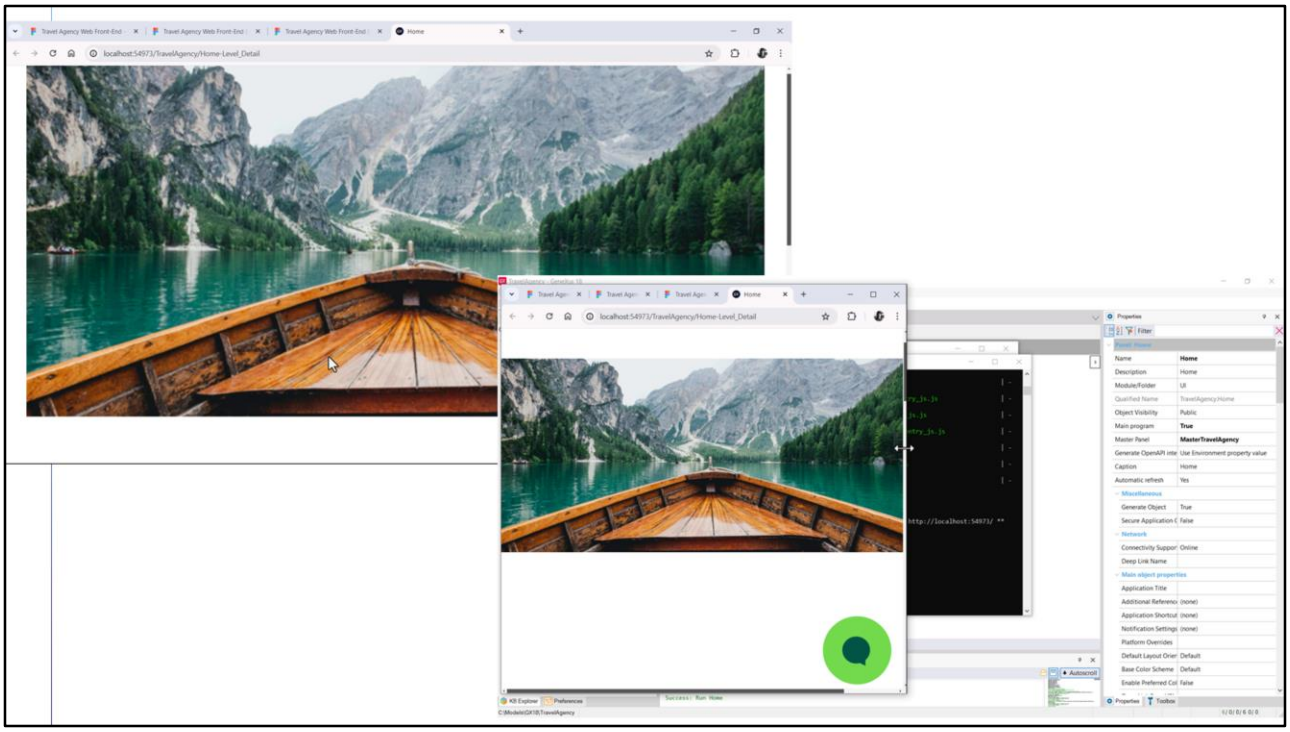


Y Rows Style nos queda como la teníamos antes. Con esta solución no necesitamos hacer aquí ningún cambio.

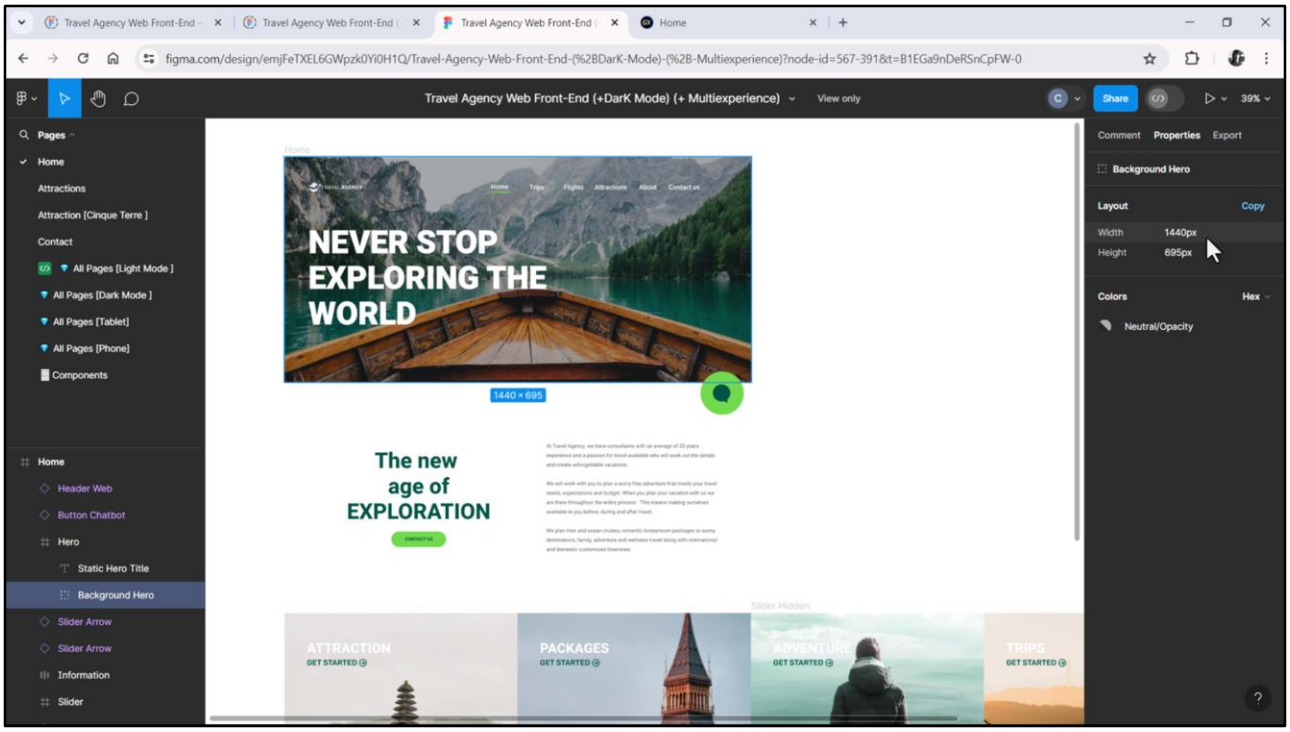


Vemos que Home tiene como Master Panel al que estuvimos trabajando.

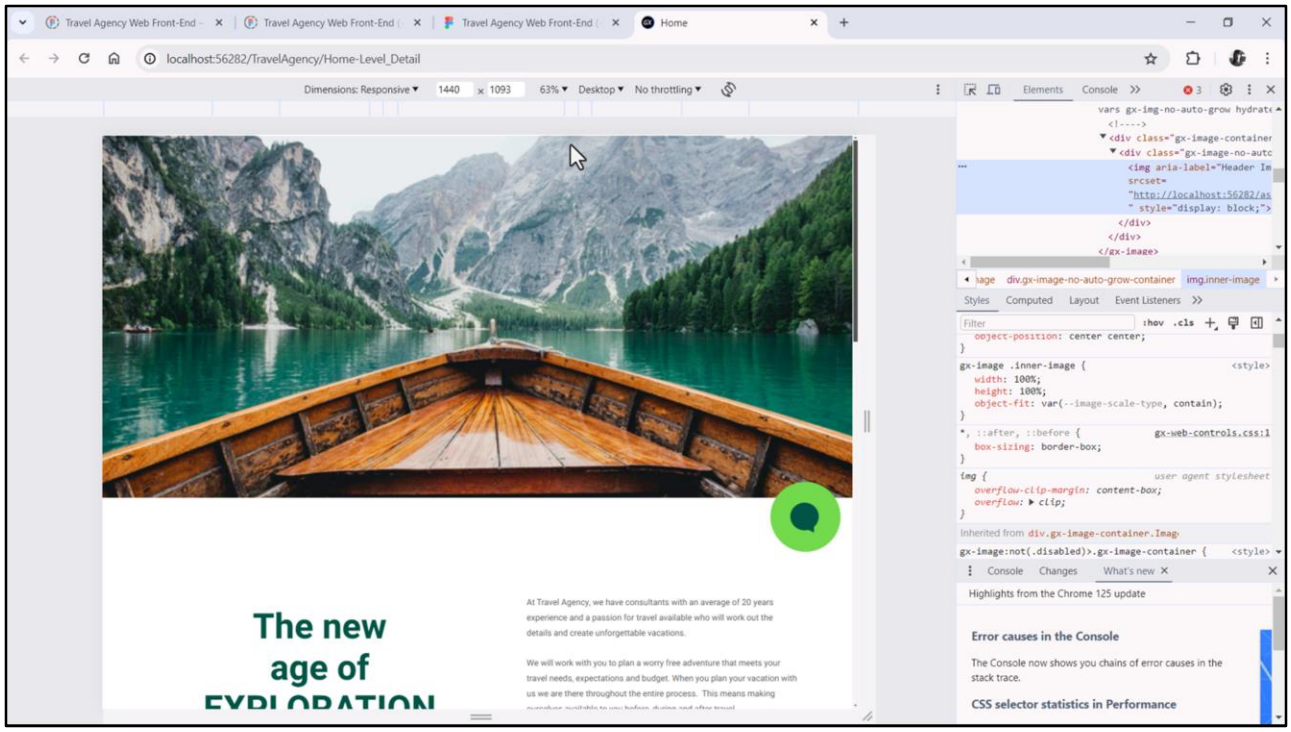
Grabemos y ejecutemos este Home.



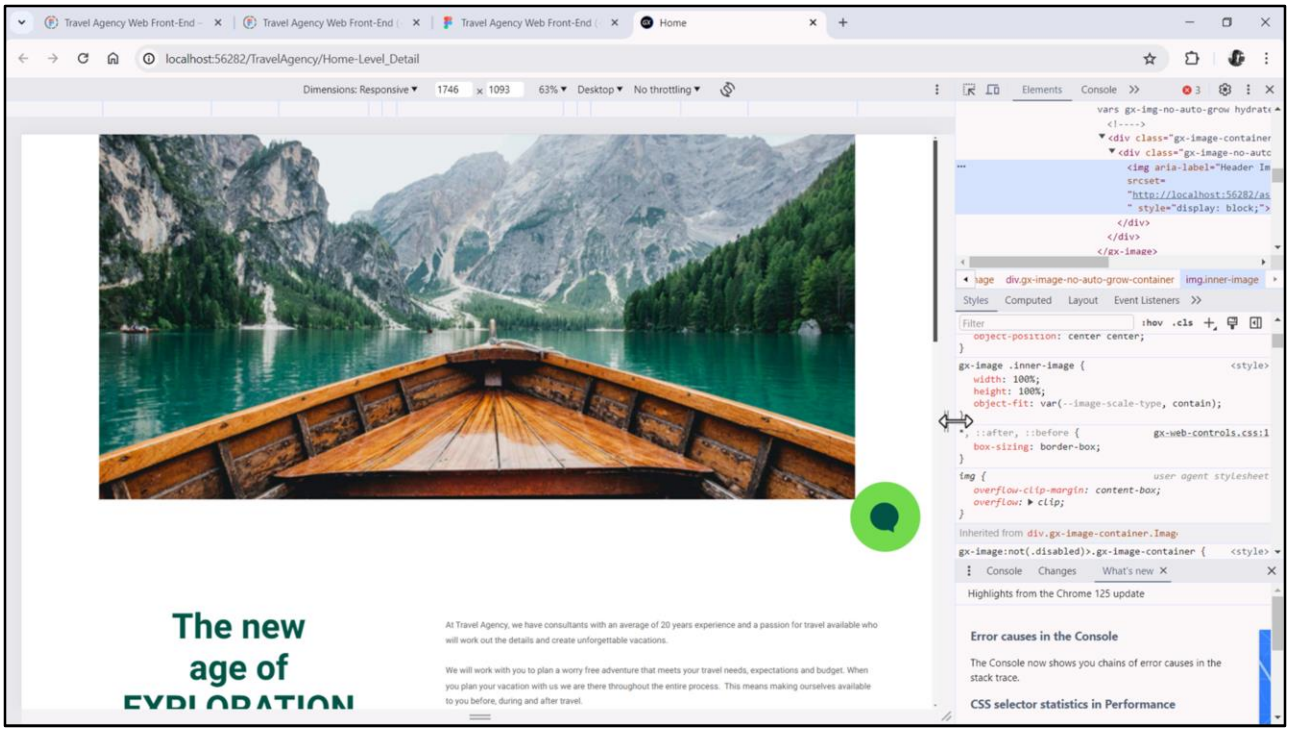
Aquí vemos el panel Home, pero algo no está bien con la imagen. No se está pegando a los bordes.
Y si por ejemplo comprimimos el ancho del navegador vemos que la imagen se achica para mostrarse siempre completa. En cambio el botón del chatbot siempre está donde dijimos, a la distancia de 660 dips de la posición top. ¿Pero la imagen? Vean lo que está haciendo.



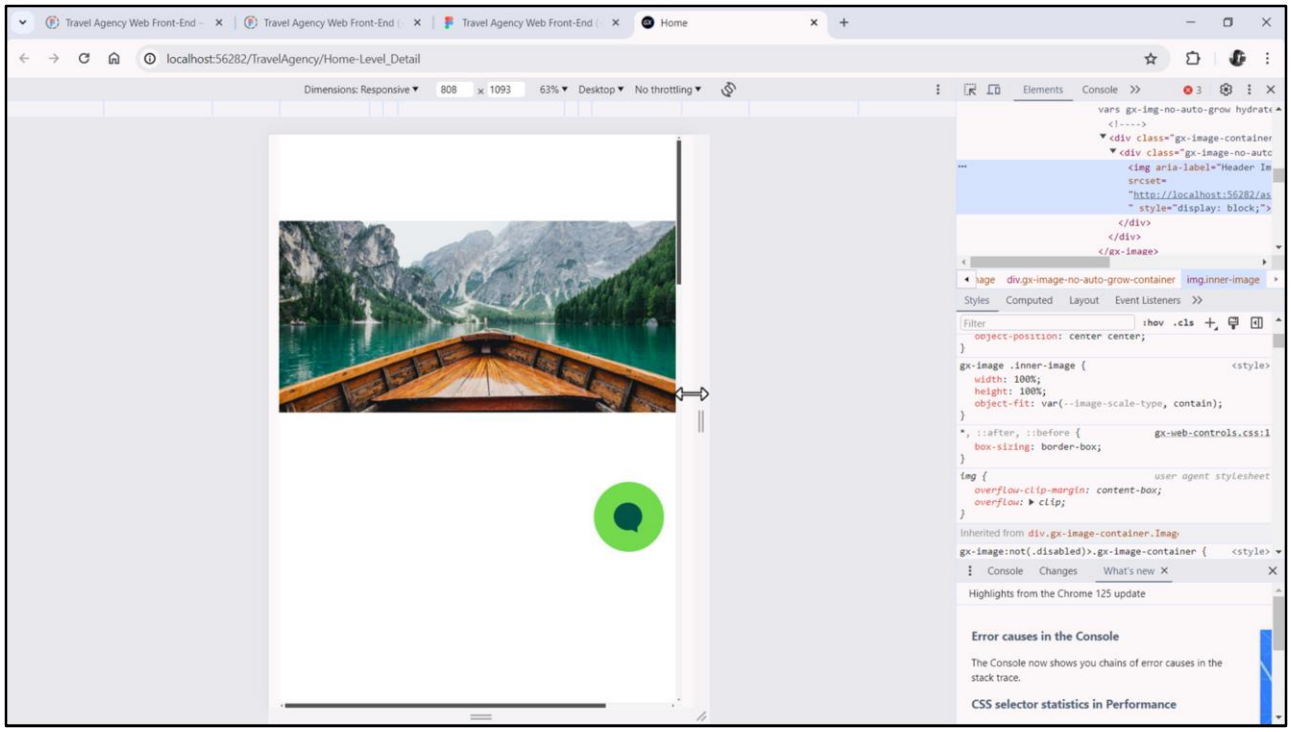
La imagen que insertamos en la KB tiene de ancho 1440 píxeles y de alto, no 695 píxeles, porque la habíamos exportado de Figma, ¿recuerdan? Con sus 3 densidades.



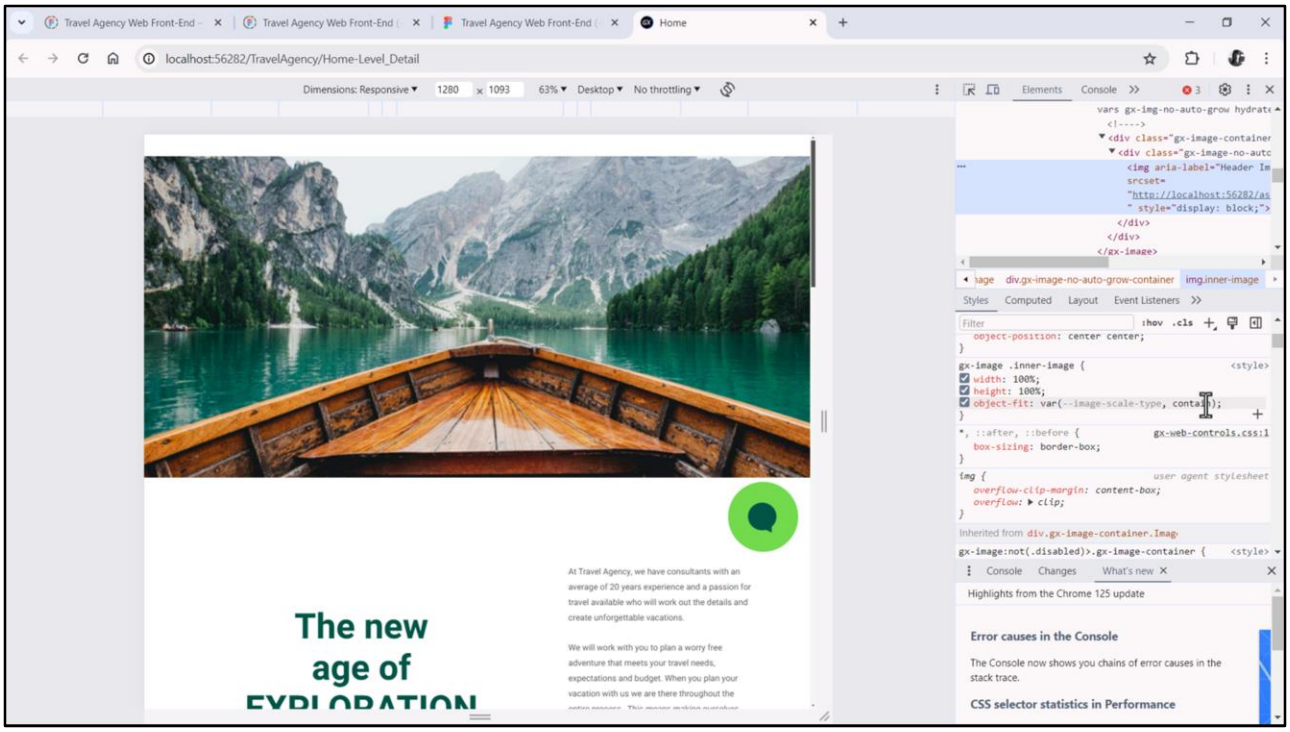
Si la inspeccionamos en Chrome, vemos que cuando el ancho de pantalla es de 1440, sólo ahí se ve como queremos.



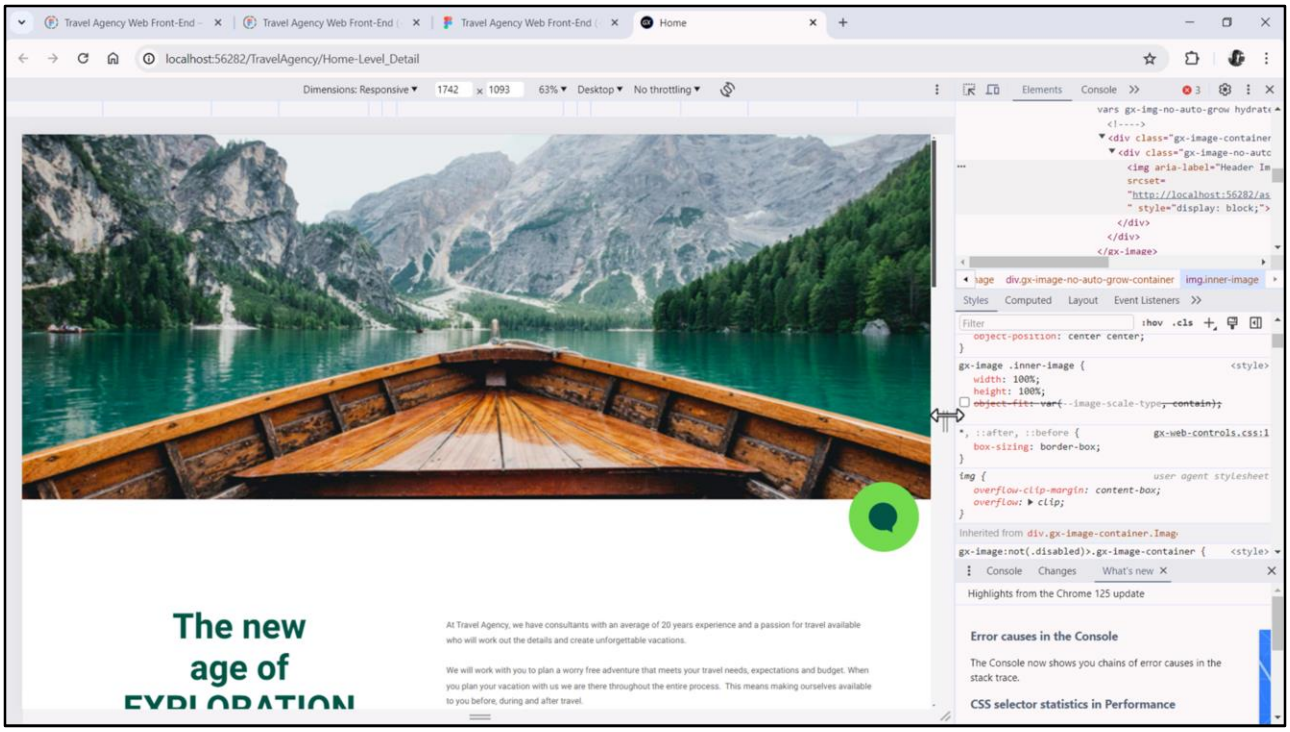
En cambio si ensanchamos queda fija, de su tamaño exacto.



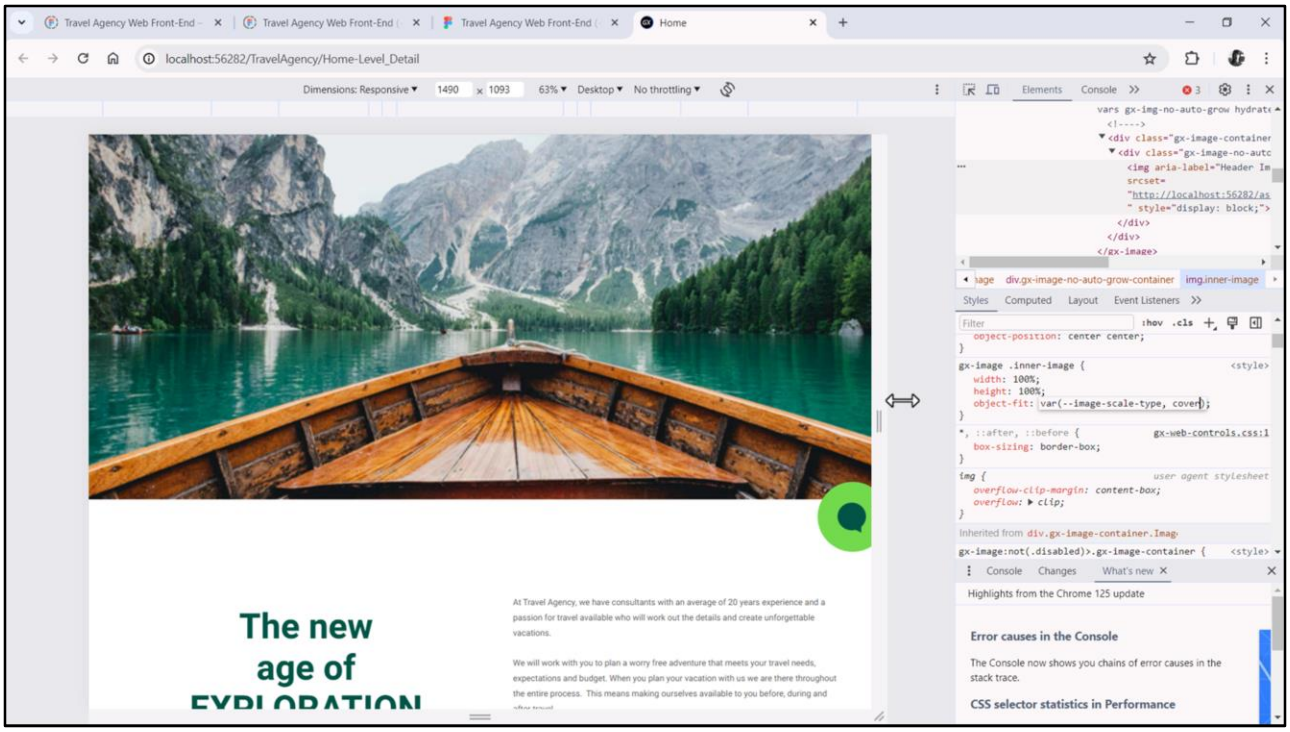
Y si disminuimos, en cambio, empieza a disminuir proporcionalmente de modo de entrar completamente en el espacio que tiene.



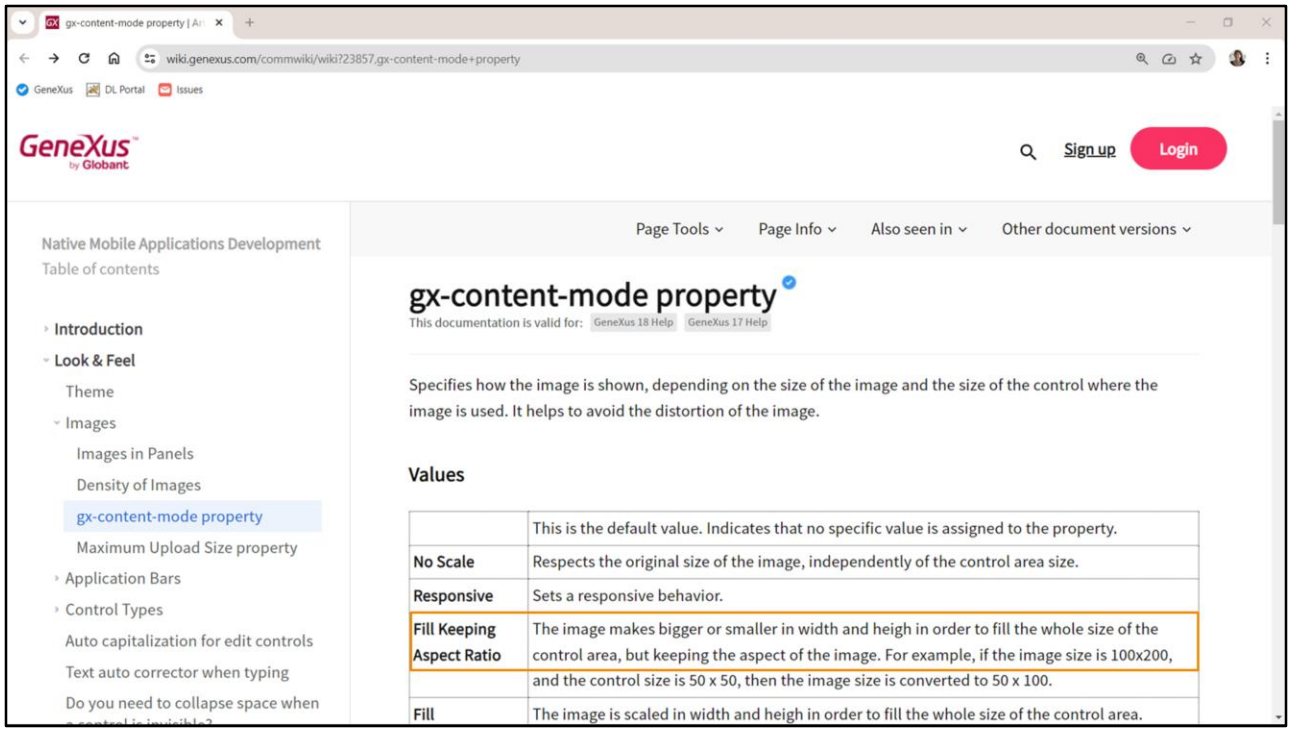
Lo que está pasando tiene que ver con esta propiedad CSS, object-fit, que está asumiendo este valor contain.



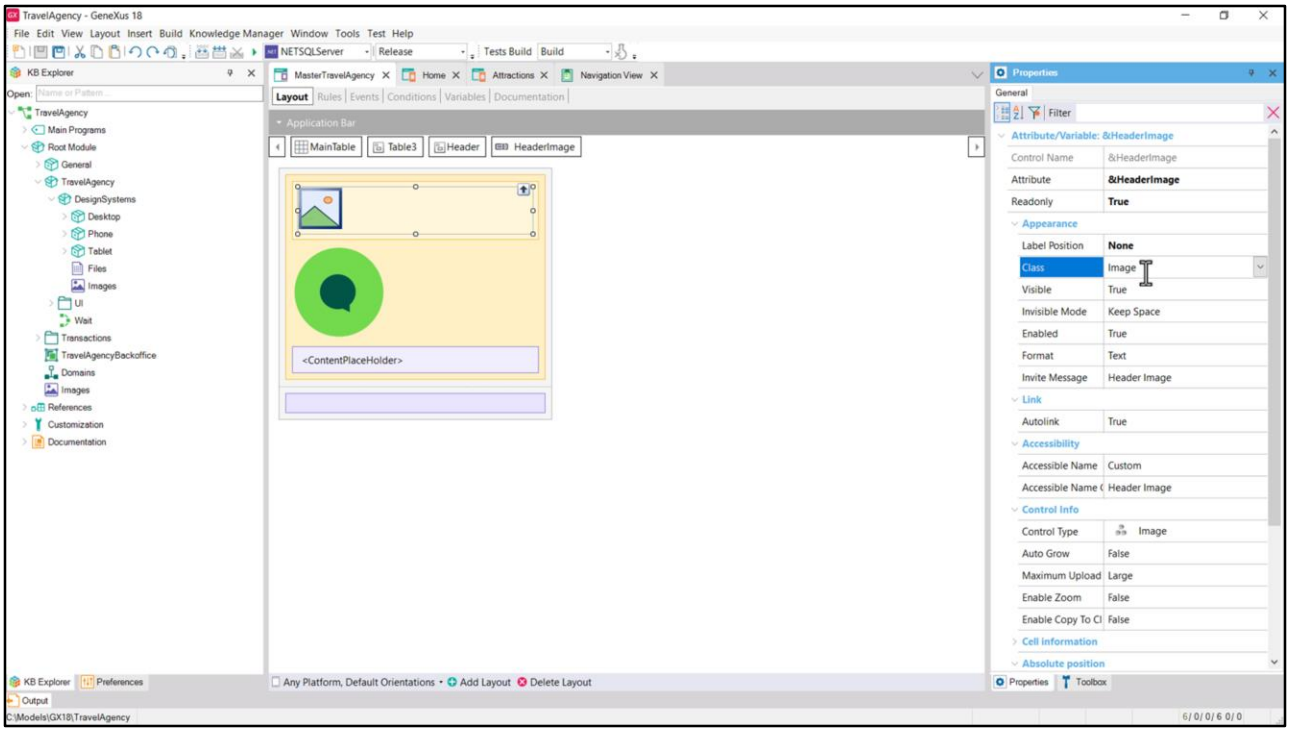
Veamos qué sucede si la quitamos… se estira o comprime de modo de ocupar todo el contenedor. Esto no es lo que queremos.



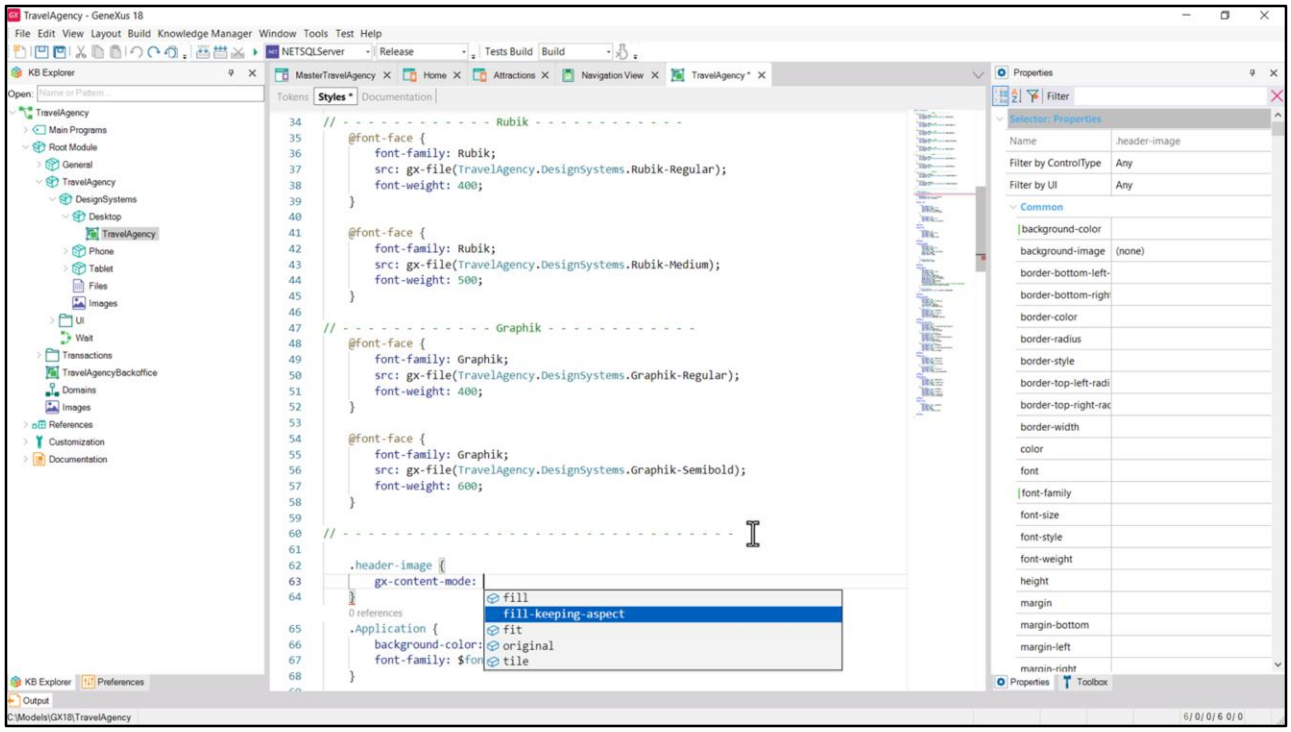
Querremos el comportamiento de cover… Es decir, que se estire o comprima para ocupar todo el contenedor, pero proporcionalmente, por lo que habrá de recortar alguna porción de la imagen, evidentemente.



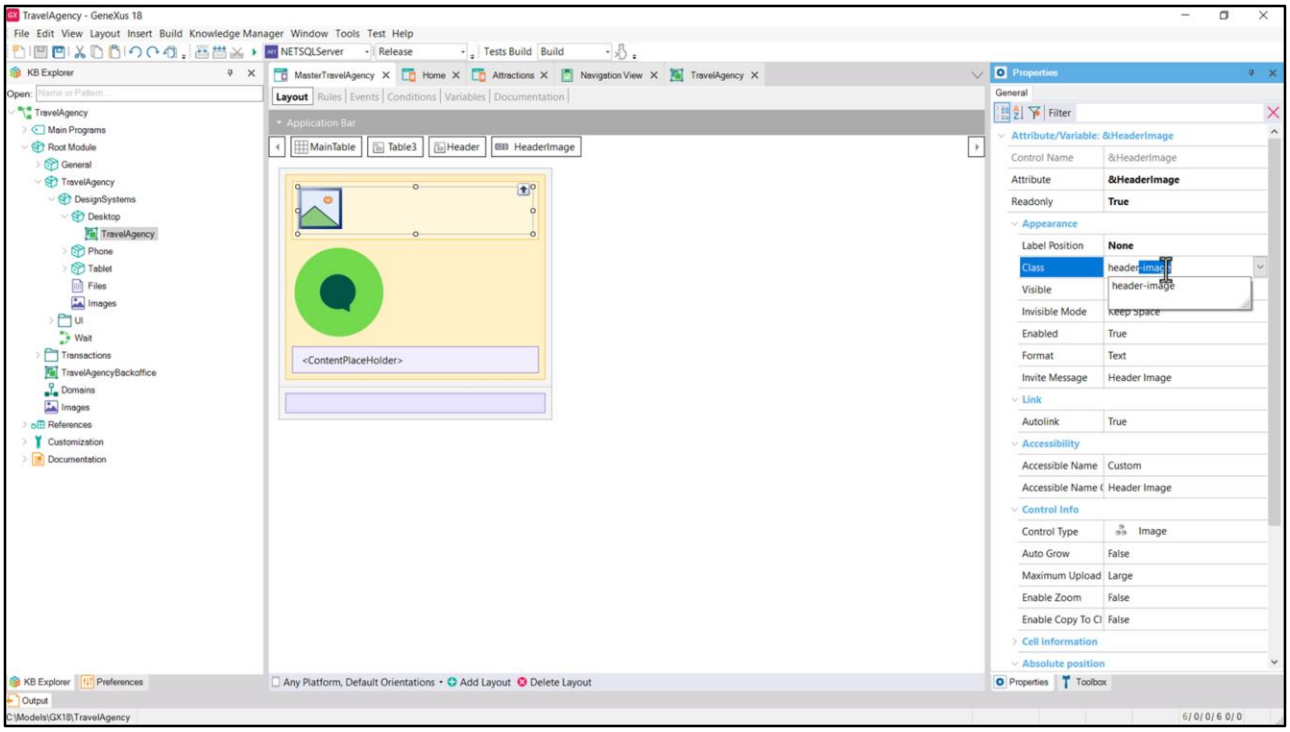
Esta propiedad CSS en GeneXus tiene otro nombre, para que sea transversal también a aplicaciones nativas. Y es la **gx-content-mode**, que aquí vemos explicada en el wiki de GeneXus, y vemos que aplica tanto a Android como a Apple como a Angular. Nosotros querremos este comportamiento: que rellene pero respetando las proporciones.



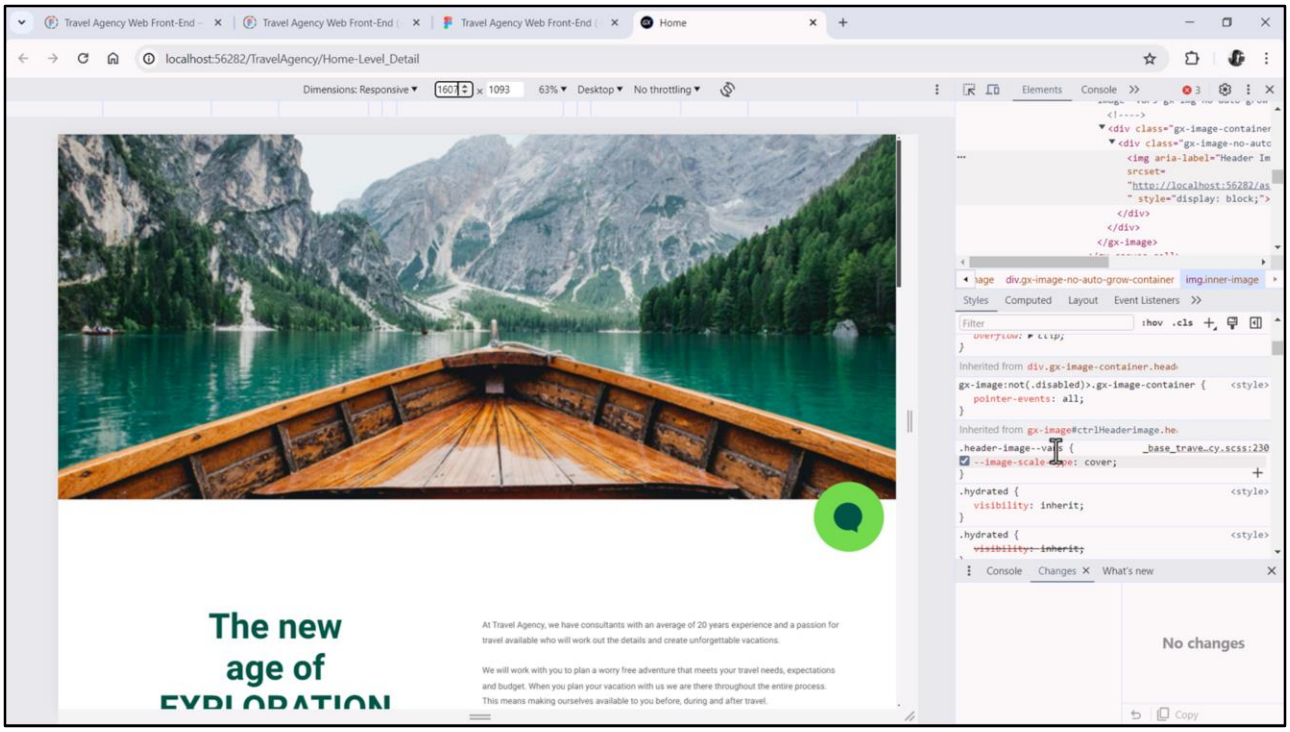
Por tanto, deberemos definir una clase para la imagen, que contenga esta propiedad.



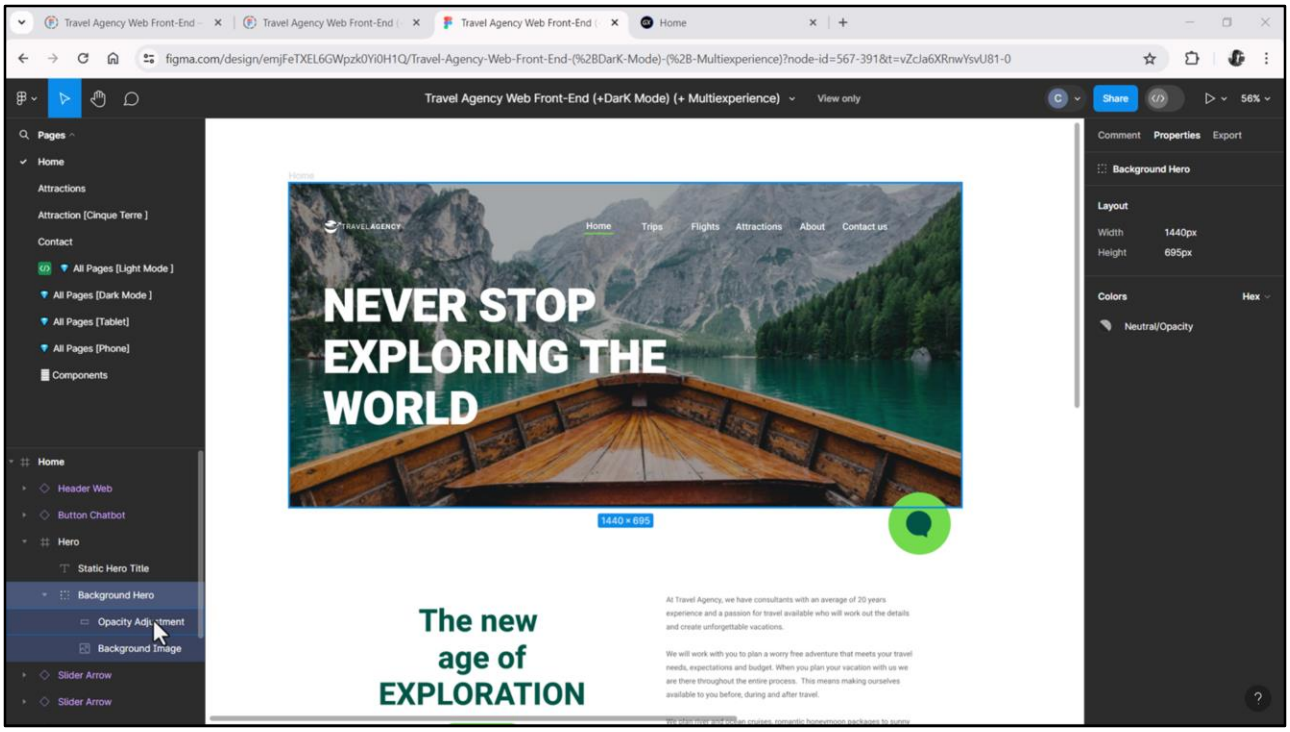
Así que abro el DSO correspondiente, es de Desktop, y en algún lado especifico la clase, a la que llamaré header-image. Y allí especifico la propiedad y su valor.



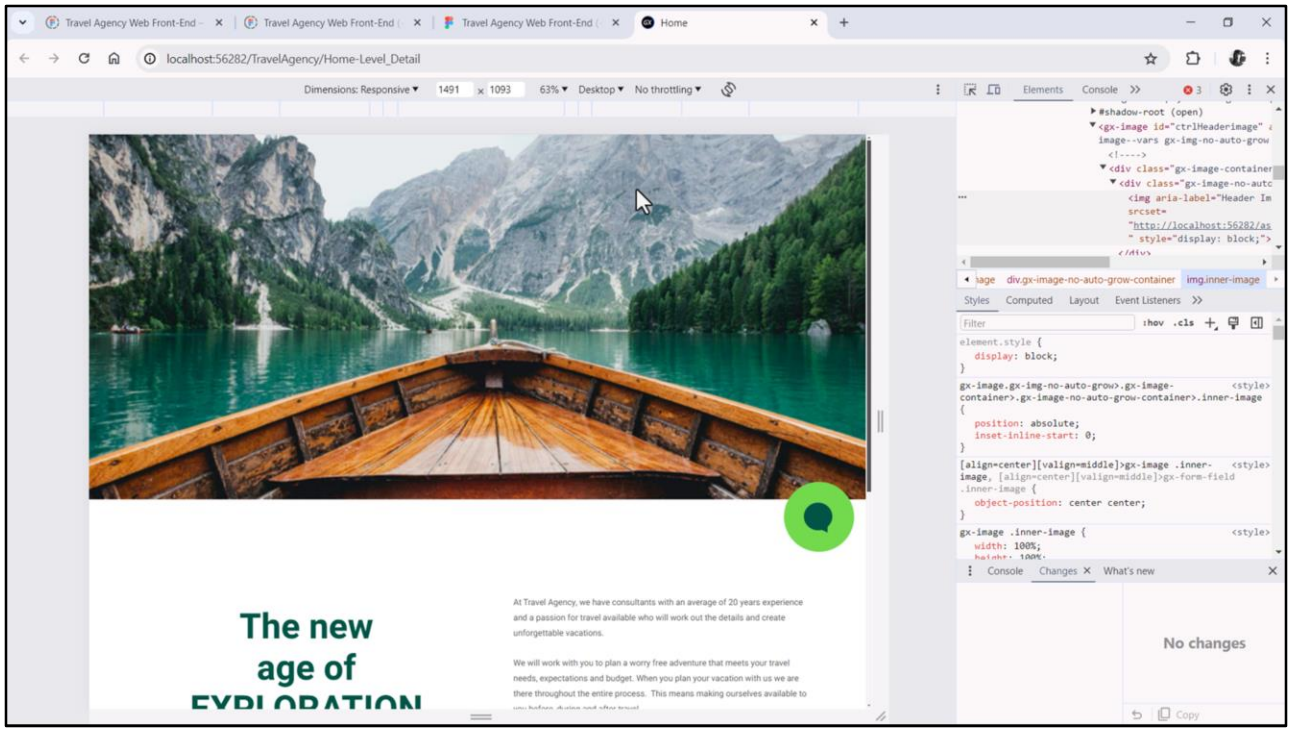
Luego se la asocio al control en el layout del Master Panel. ¿Probamos?



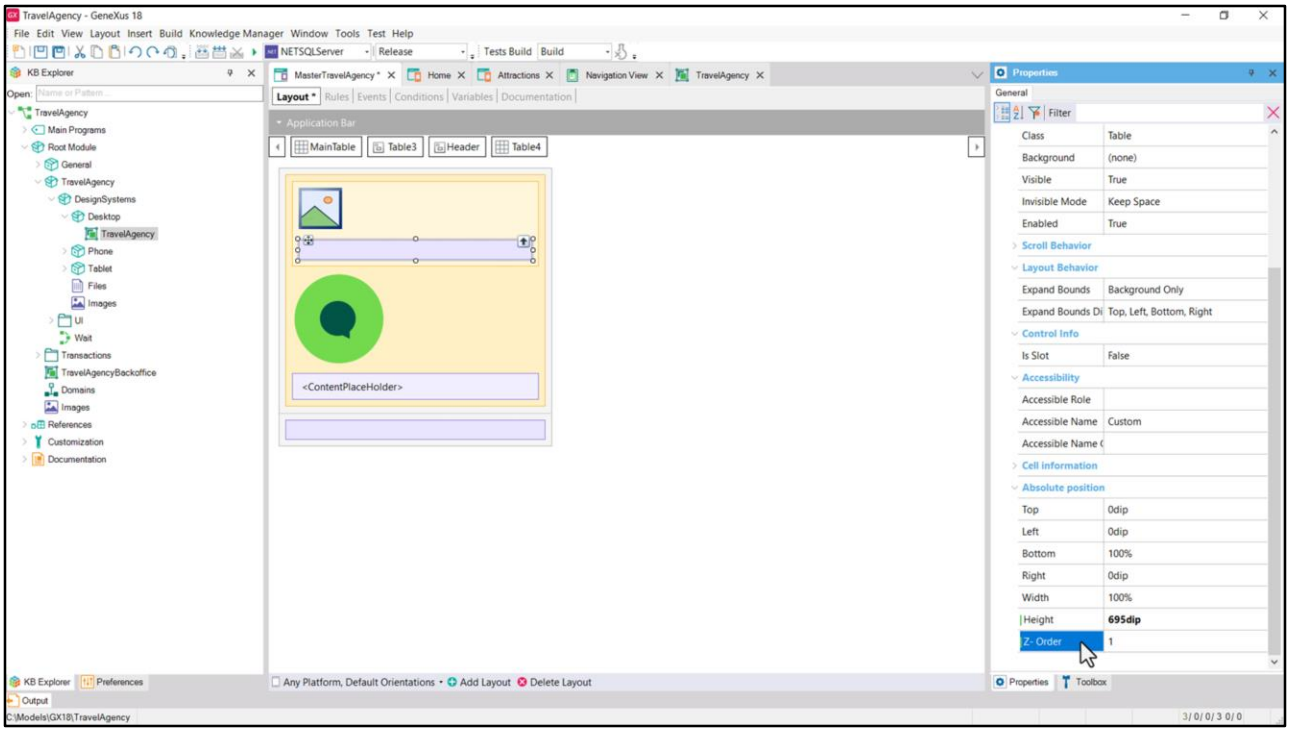
Ahora sí, al expandir o contraer vemos la imagen como queremos. Aquí podemos ver la clase operando, convertida por GeneXus al código necesario. No tenemos por qué entender nada de esto.



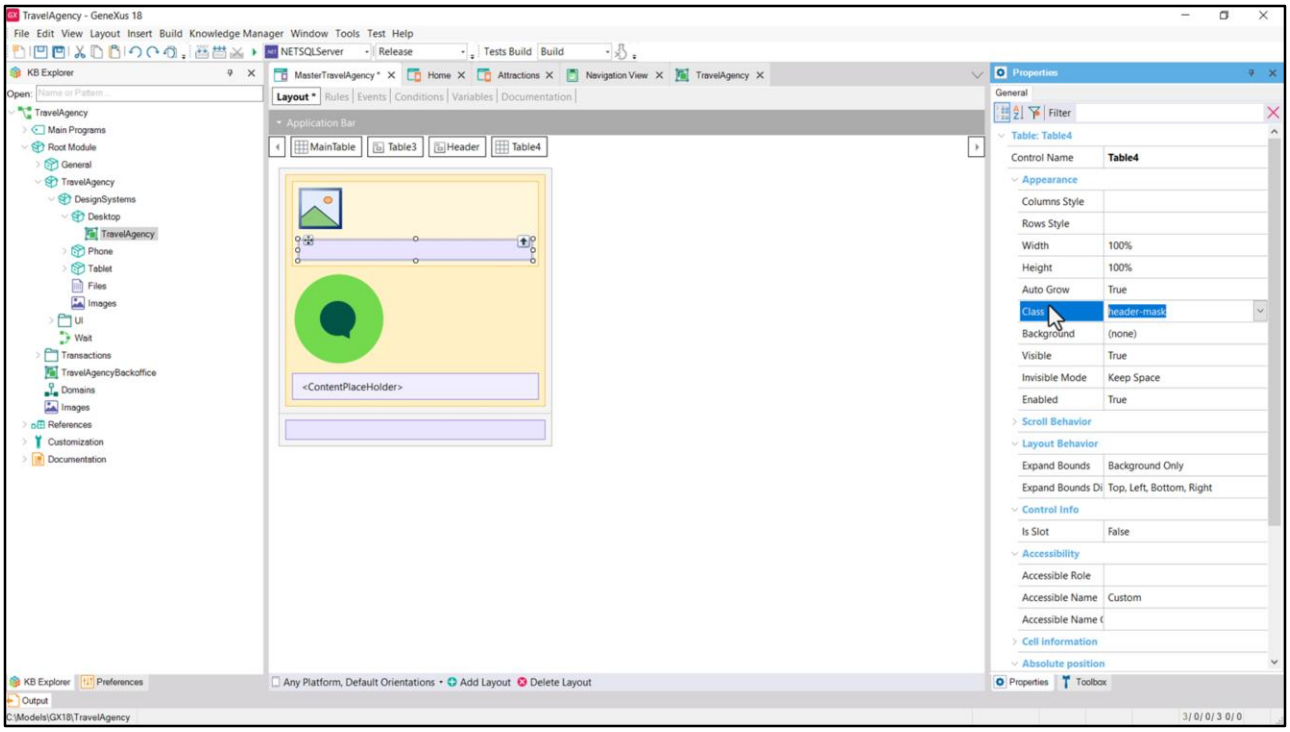
Otra cosa que podemos ya apreciar si vamos a Figma, es que hay una máscara sobre la imagen, para oscurecerla y lograr un mejor contraste de todos estos elementos que se le van a superponer.



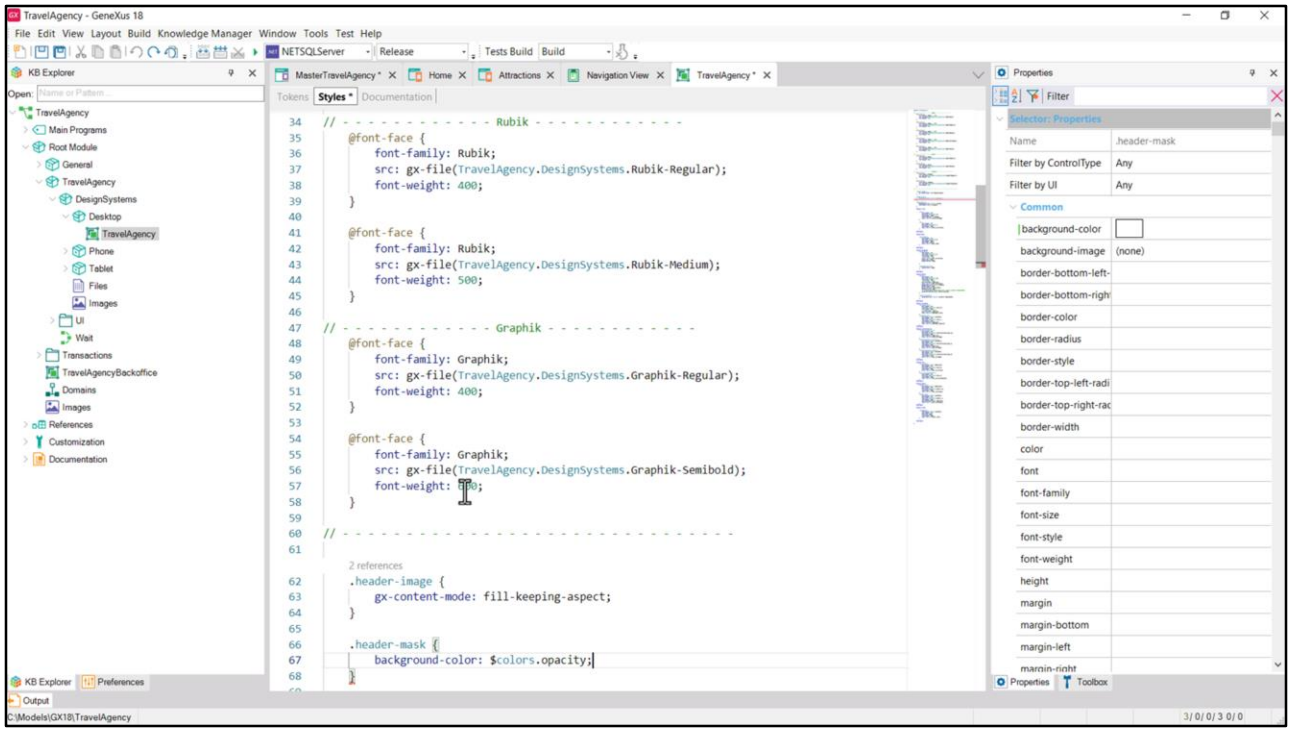
Aquí la vemos más bien clara. Así que necesitamos esa máscara.



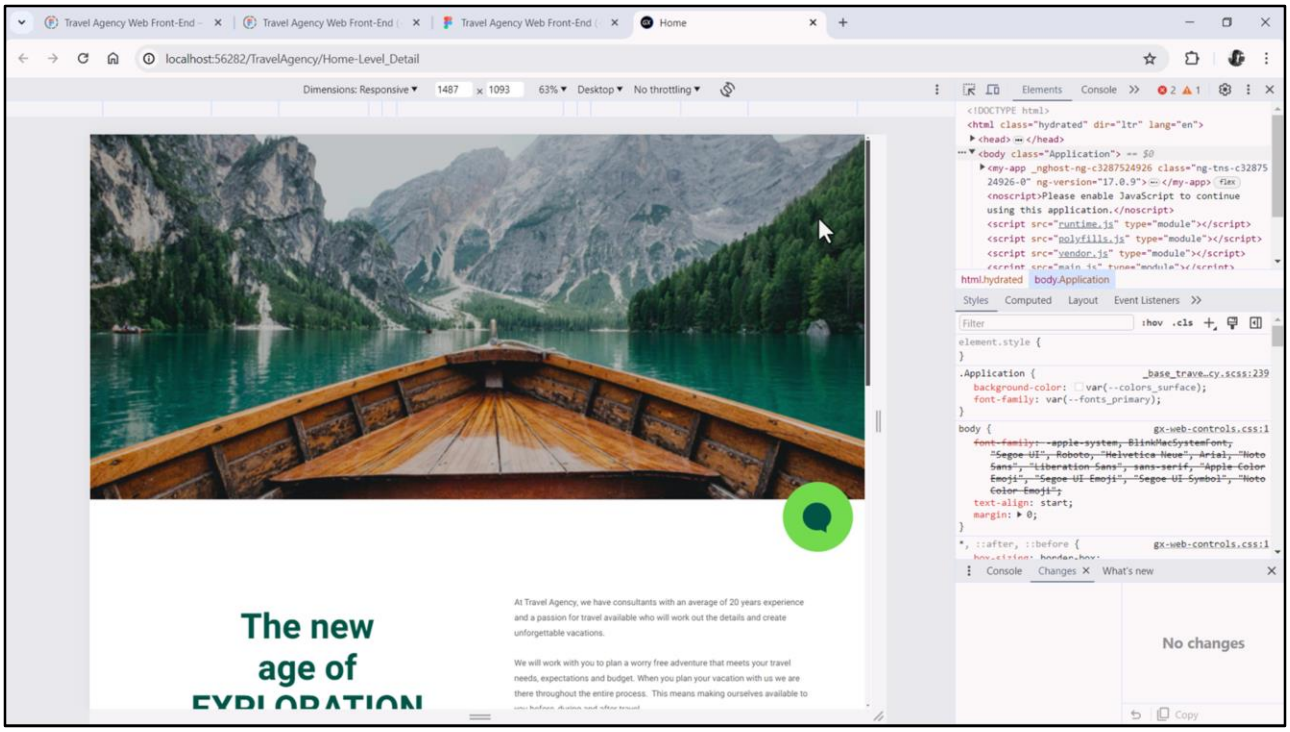
Una manera de implementarla es a través de una tabla vacía, que se le superponga, con exactamente las mismas dimensiones, es decir, 695 dips de alto y el resto pegado a los bordes del canvas, es decir, 0 dips de arriba, 0 de izquierda y de derecha y bottom 100%, que al calcularlo cuando se cargue será también 0, y le podemos dar al Z-order el valor 1 para asegurarnos de que esté encima de la imagen, que tenía 0 de posición z.



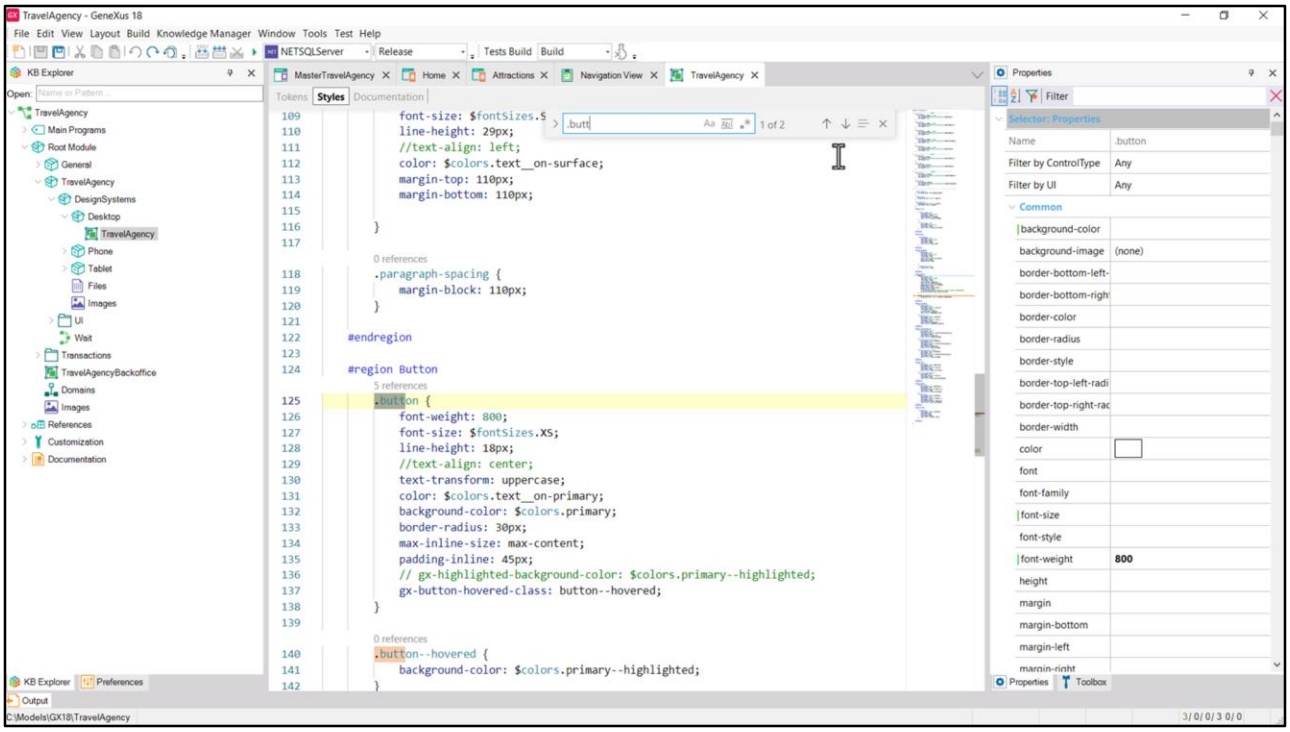
Luego, tendremos que especificar el background color de la tabla a través de una clase. Así que la nombraremos header-mask.



Y la especificamos en el DSO, recordando que teníamos ya el token de color opacity definido. Entonces a la propiedad background-color le asociamos ese token de color.



Si ahora ejecutamos, vemos la imagen con la máscara.

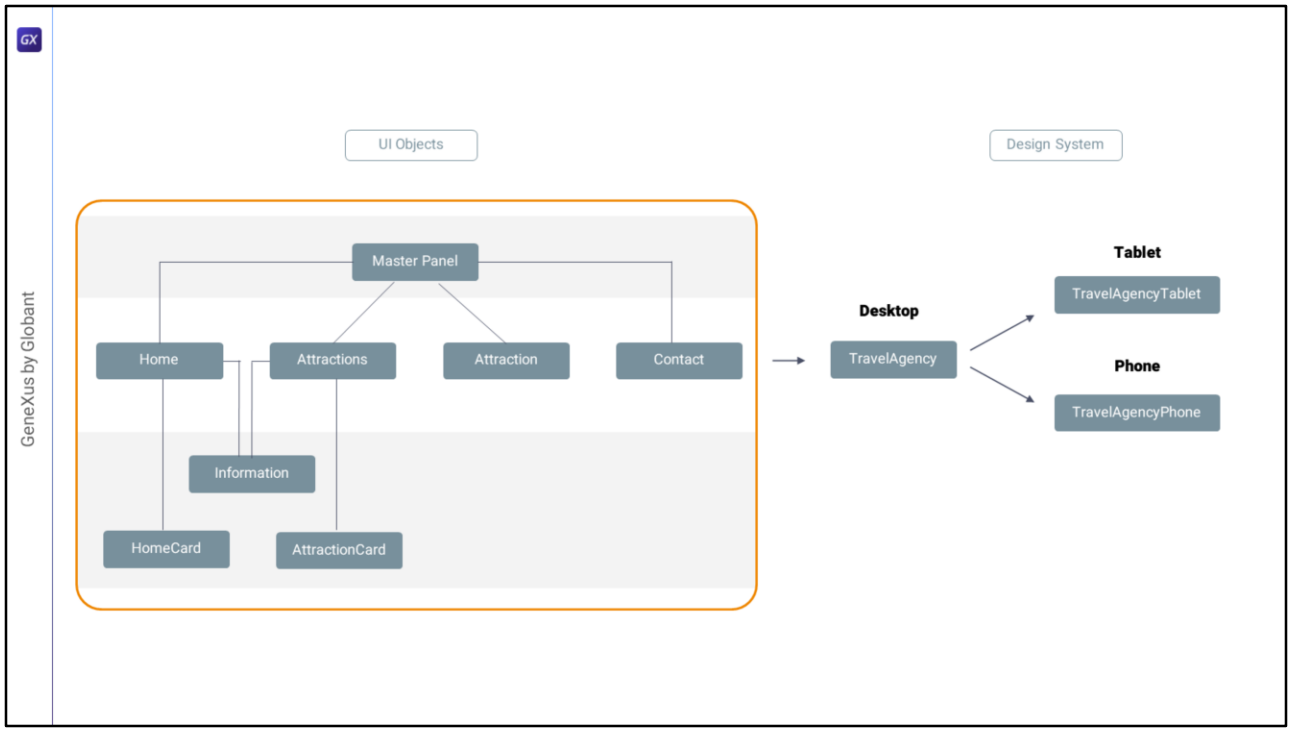


Antes de seguir, no sé si notaron que se no está complejizando el DSO.
Cuando fui a especificar las dos clases que necesitamos crear para el Header: la de la imagen y la que usaríamos para la máscara, las escribí después de las reglas font-face, pero las podría haber escrito en cualquier lado.

Si repasamos lo que teníamos especificado en este DSO eran, además de las reglas para incorporar las fuentes, todas las clases para los estilos tipográficos de toda la aplicación, y además para la del botón de contacto, las que comandaban el estilo del botón.

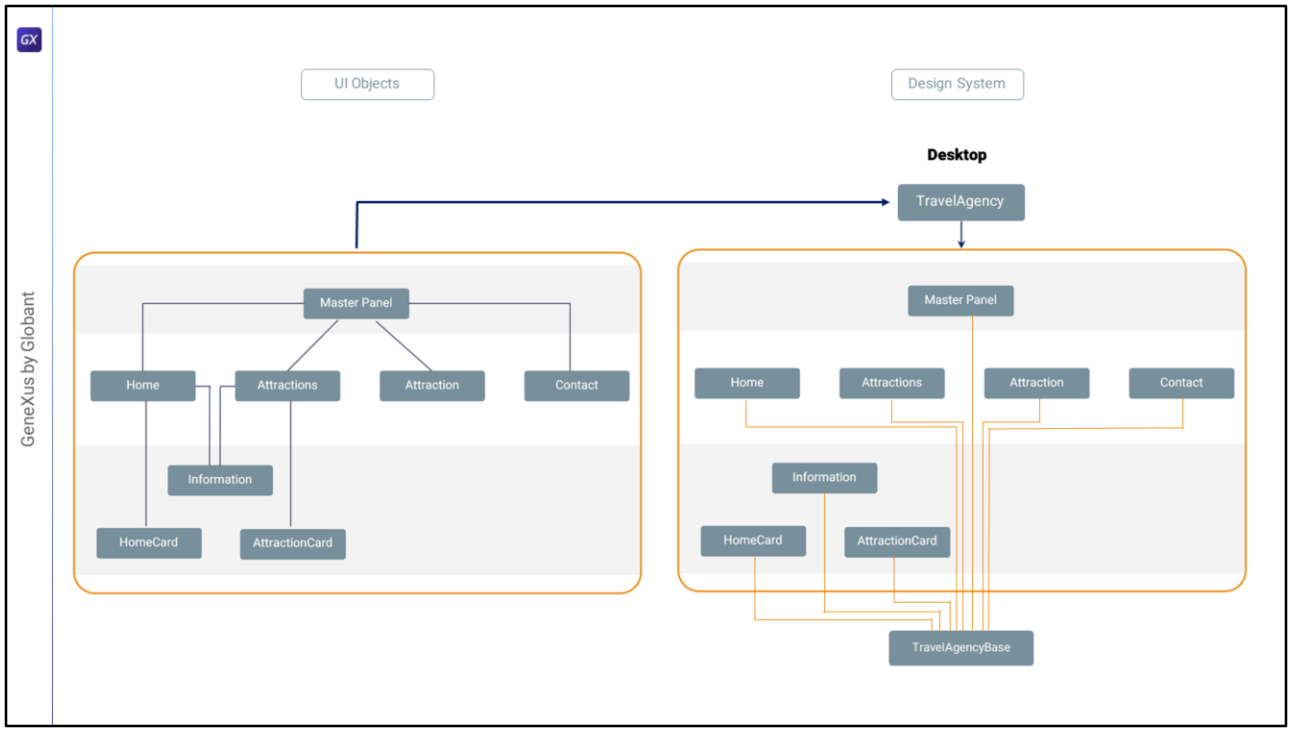
Pero ahora tenemos que empezar a incorporar clases para los distintos controles de todos los layouts. Imaginen que esto va a crecer bastante. El editor nos brinda la posibilidad de hacer búsquedas con control F.

Pero para hacer más manejable y comprensible todo esto, una posibilidad que es la que voy a elegir (después la discuto un poco) va a ser también de algún modo componentizar el DSO.



Me explico: todo lo que habíamos hecho hasta el momento había sido especializar el DSO TravelAgency para lo que será el tamaño Tablet por un lado, y por otro el tamaño Phone.

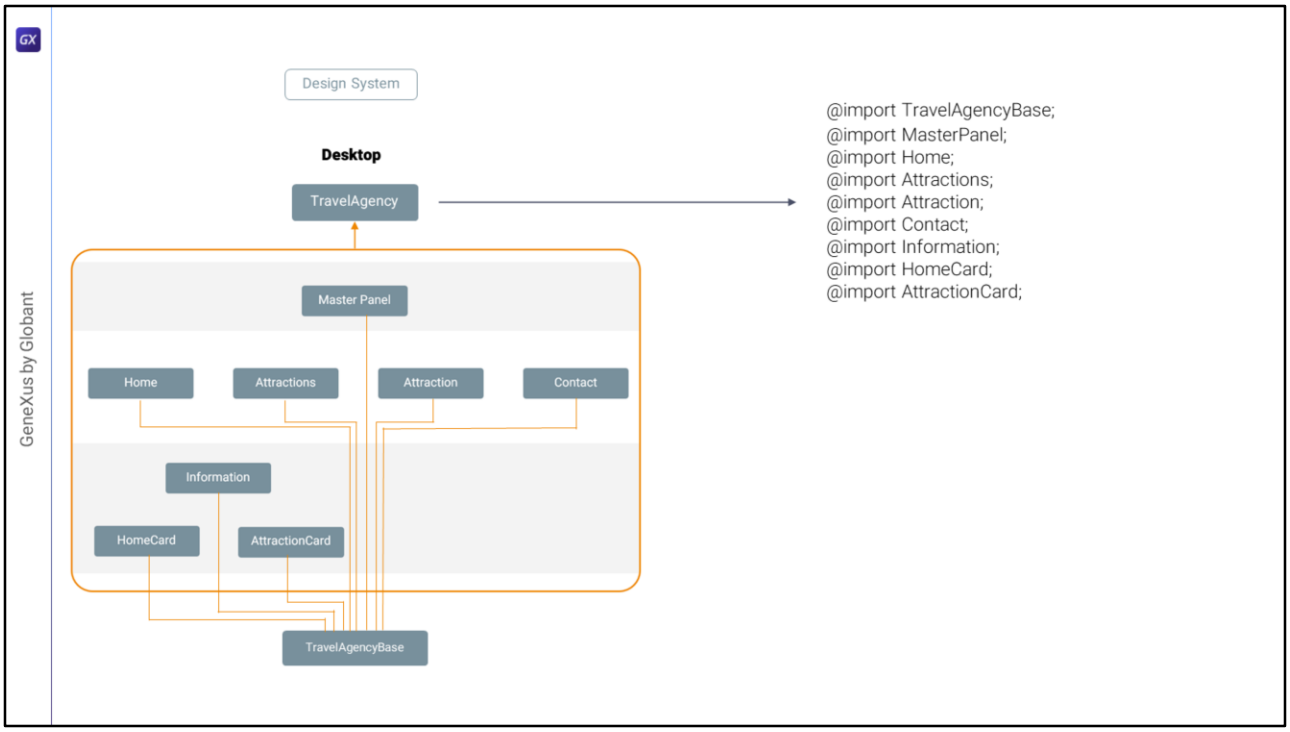
Pero en realidad nada nos impide tener un árbol de DSOs para cada uno y no un único objeto.



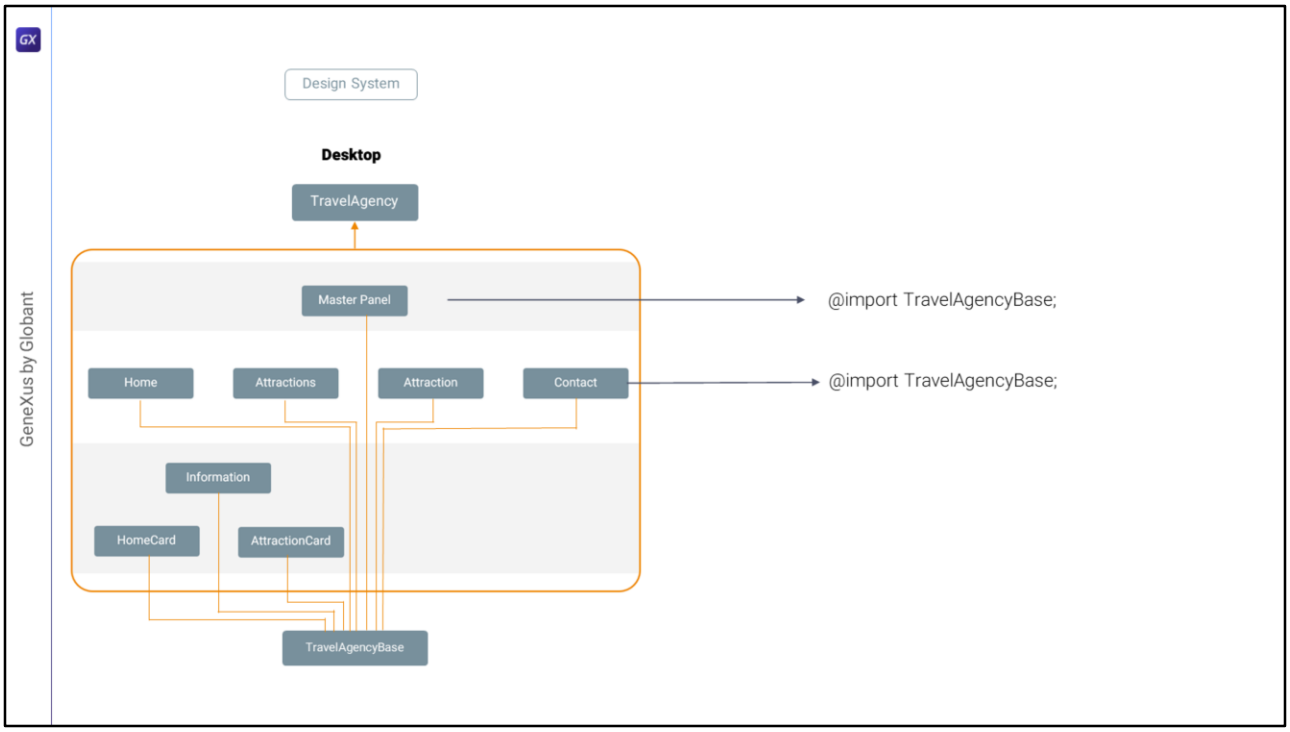
Es decir, pensemos únicamente en el tamaño Desktop… ¿No sería bastante más ordenado, por ejemplo, tener un DSO base, con todas las definiciones transversales a toda la aplicación, como por ejemplo la inclusión de las fuentes, y todo el análisis de tokens que ya hicimos, e incluso, si se quiere, también podríamos dejar allí toda las clases para la tipografía… y luego tener tantos DSOs como objetos, que definan, cada uno, las clases que aplicarán a los controles de su layout.

Cada uno importando las definiciones generales del DSO Base, y especializando lo que necesiten, y definiendo sus clases propias. El criterio debería ser: todo lo que sea común a varios objetos, debería ir en el DSO Base y lo que es absolutamente específico, en el DSO del objeto.

Luego, como no podemos decirle a cada objeto de la UI cuál es su DSO, sino que se trata de un único DSO para toda la plataforma, nos alcanzará con tener el DSO padre, Travel Agency, que lo único que hará será importar todos estos otros.



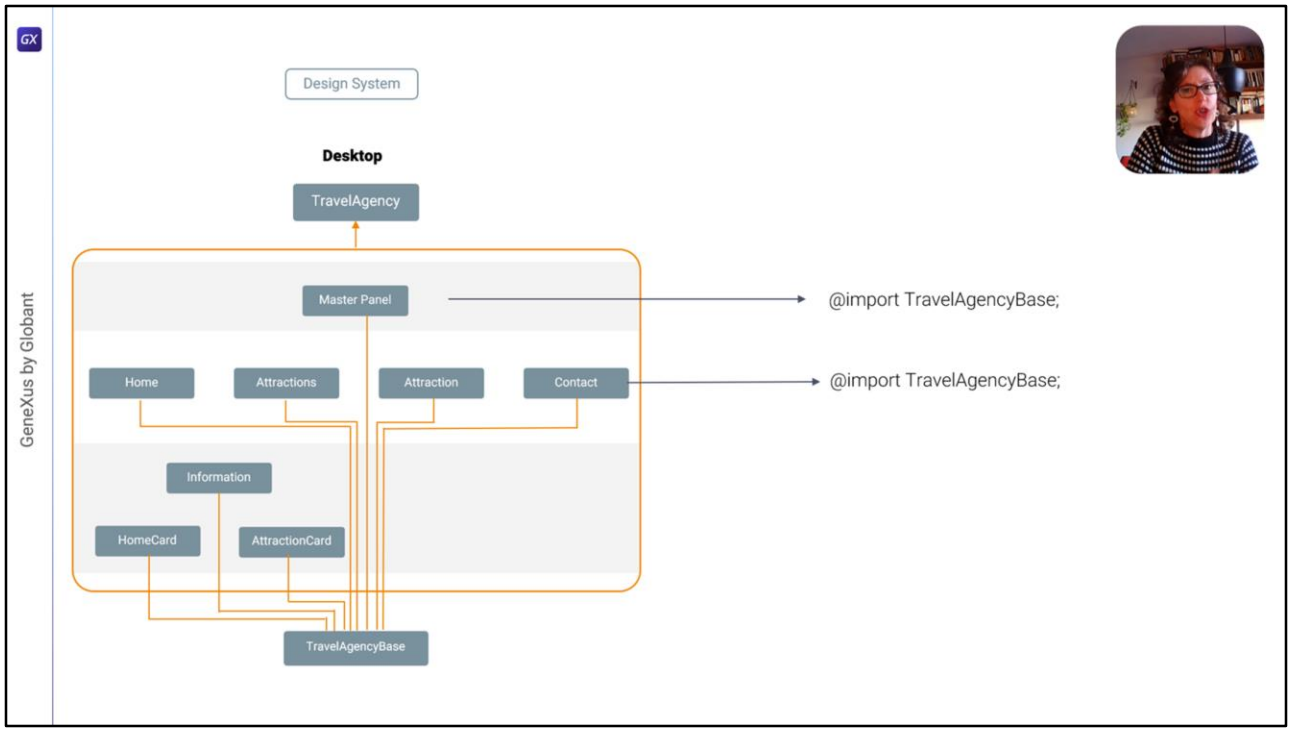
En definitiva, el DSO TravelAgency solamente tendrá tantas reglas import como DSOs para cada objeto hayamos definido, más un import del DSO Base (lo que en principio no sería necesario porque va a ser importado por los demás, pero conceptualmente está bien).



Luego, el DSO que da estilo particular a cada objeto seguramente necesitará importar el DSO base, que tiene las definiciones generales. Así el del Master Panel, el de Contact, y todos los demás.

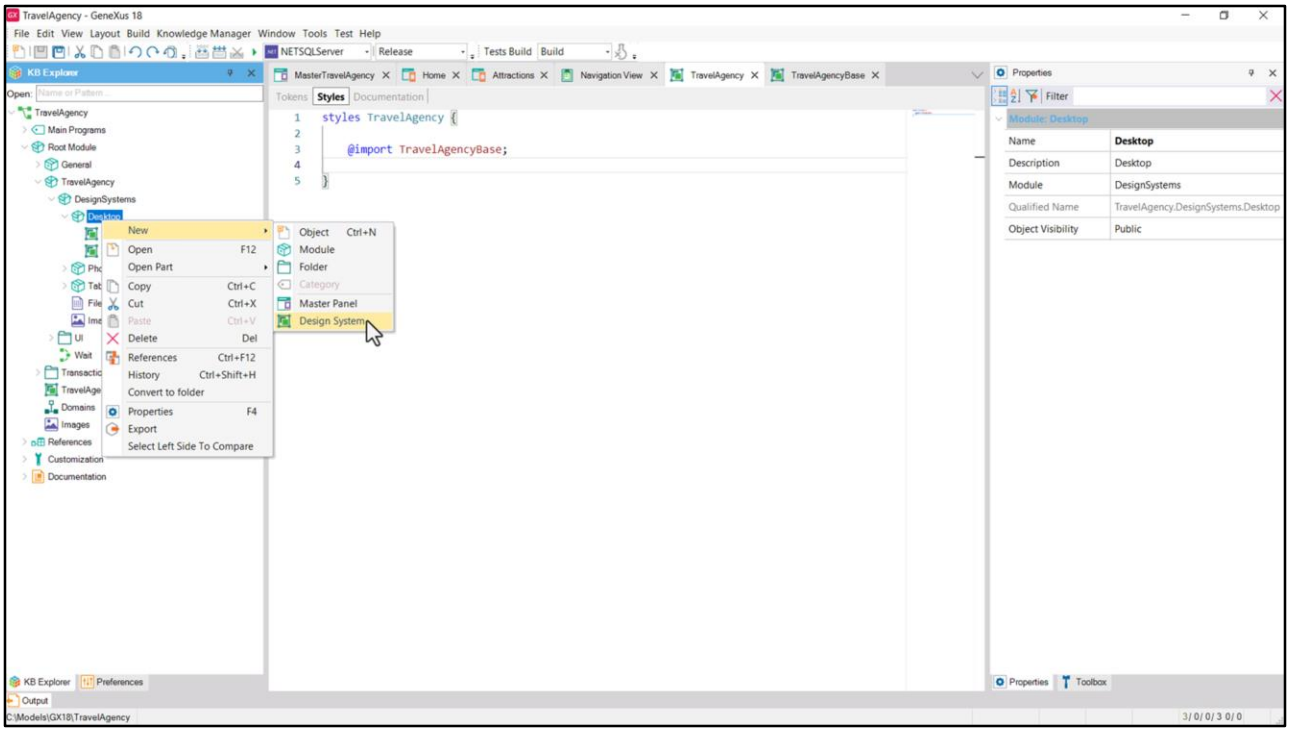
Podríamos pensar que importar el DSO Base aquí, y además en cada uno de los DSOs lo repetirá innecesariamente y penalizará entonces a la aplicación generada, en el caso de Angular, con un CSS enorme. No será así. En el CSS final aparecerá una única vez el DSO Base.

Tendremos que ver hasta qué granularidad llegamos (estoy pensando en los stencils, por ejemplo). Aquí siempre se trata de soluciones de compromiso.

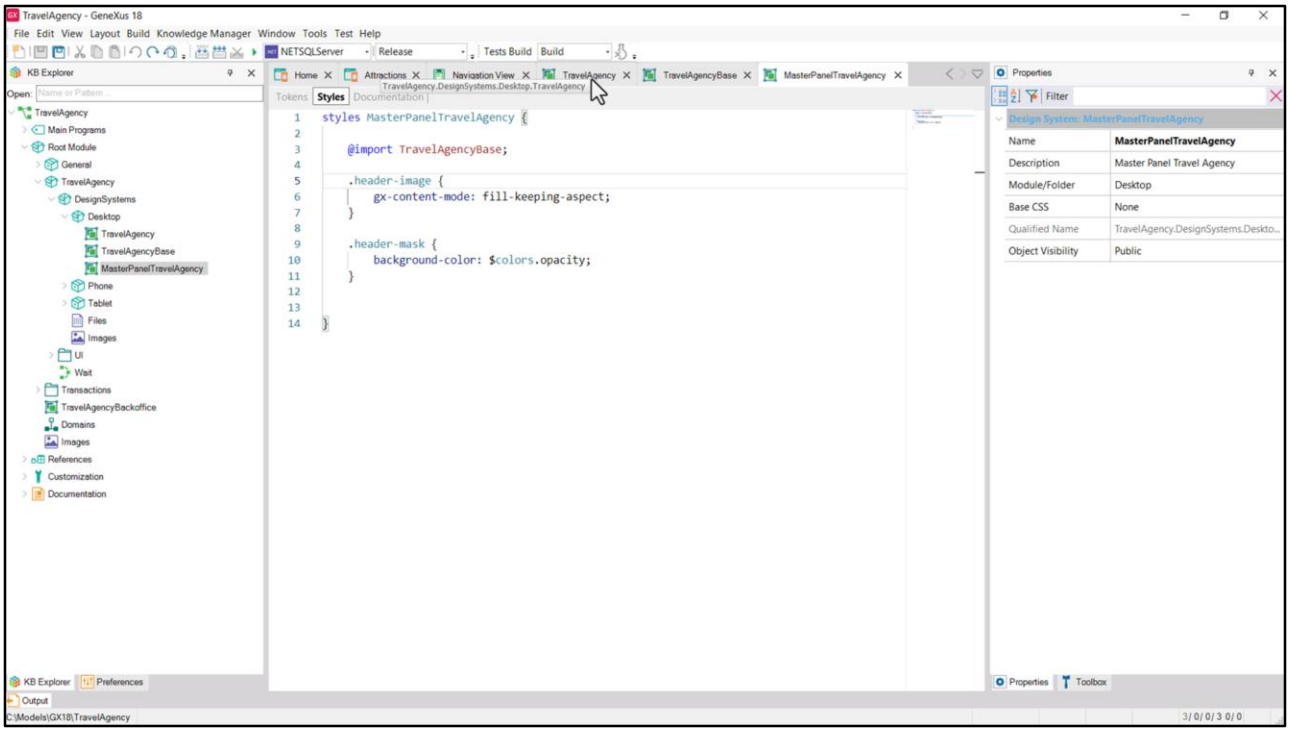


Esta que planteo es una posible solución para enfrentar la complejidad, que no tiene consensos. De hecho la sometí a discusión con varios de los compañeros que están trabajando permanentemente en frontend dentro del equipo de GeneXus y no había mucho consenso.

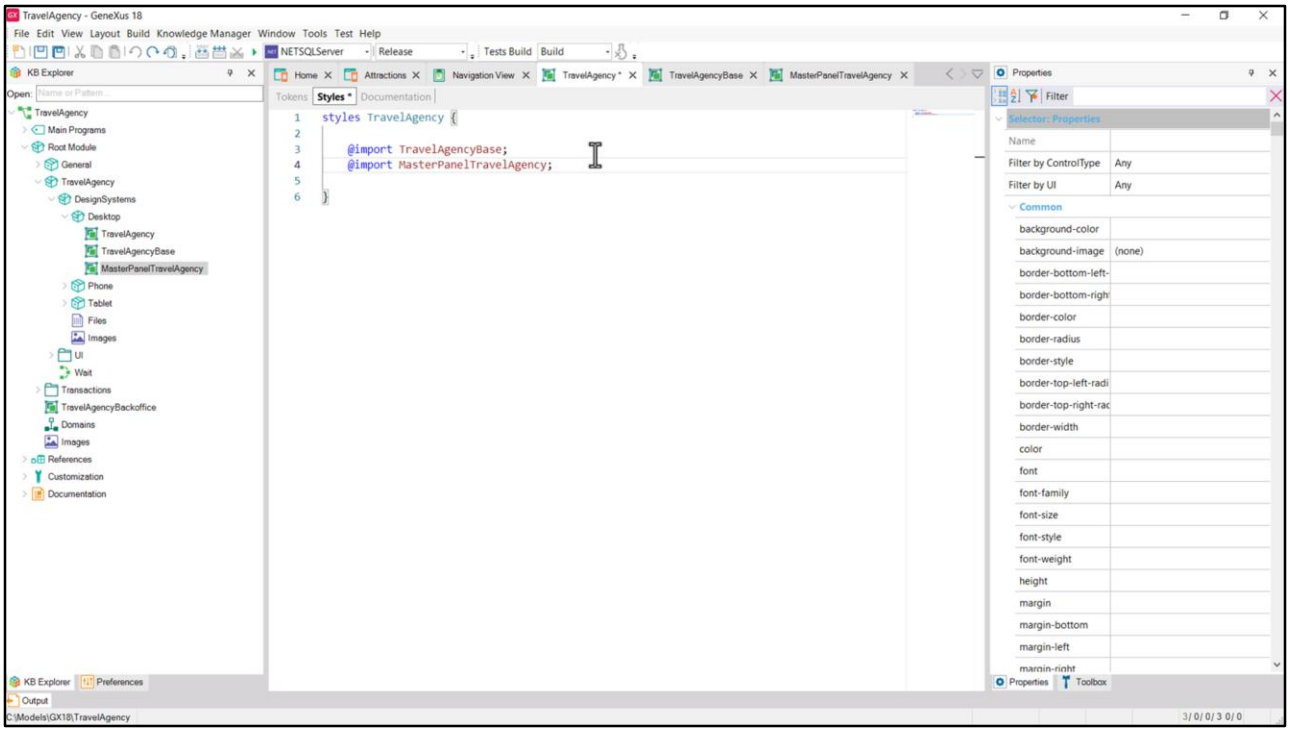
Pero la dejo abierta como discusión, no es este el momento de zanjarla, obviamente, pero se las quería presentar porque es la que voy a empezar a utilizar, y en todo caso más adelante la podremos cuestionar.



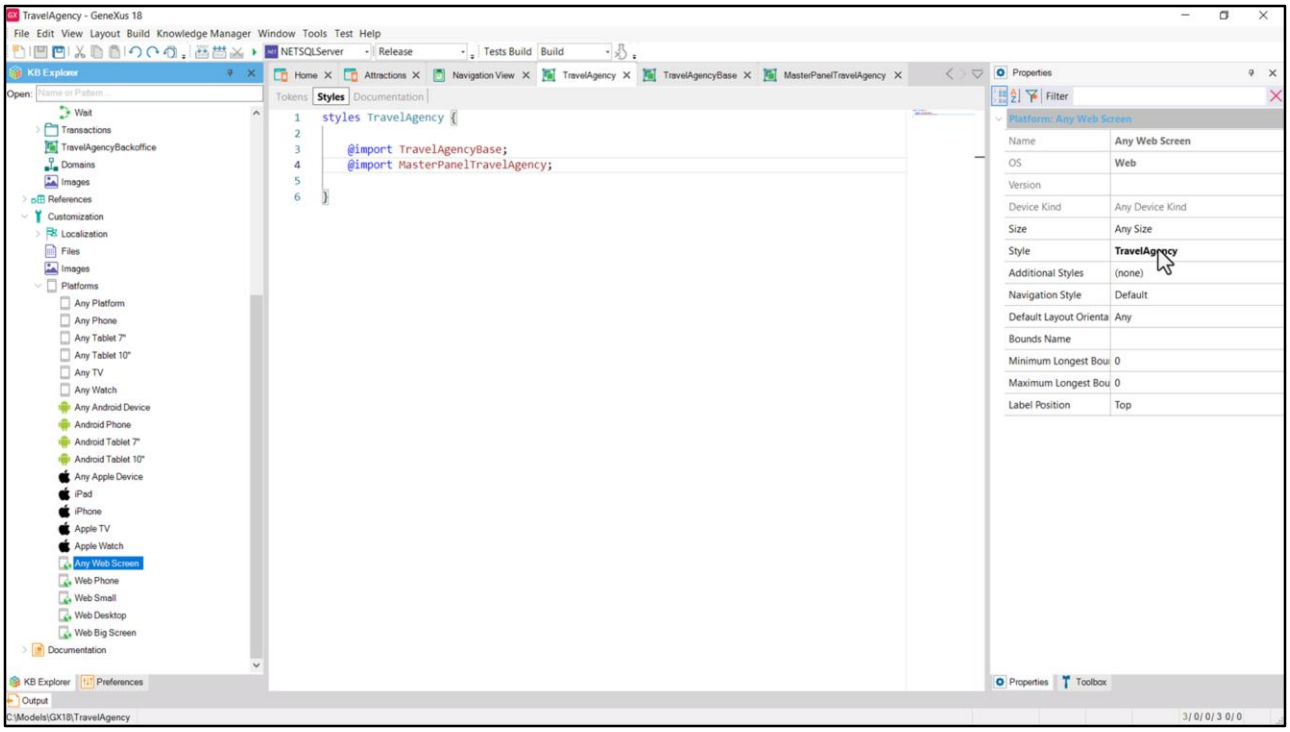
En definitiva, entonces, grabaré este objeto con el nombre TravelAgencyBase.
Al TravelAgency lo vaciaré, y lo que haré será por el momento importar este otro.
Y, por otro lado crearé otro DSO…



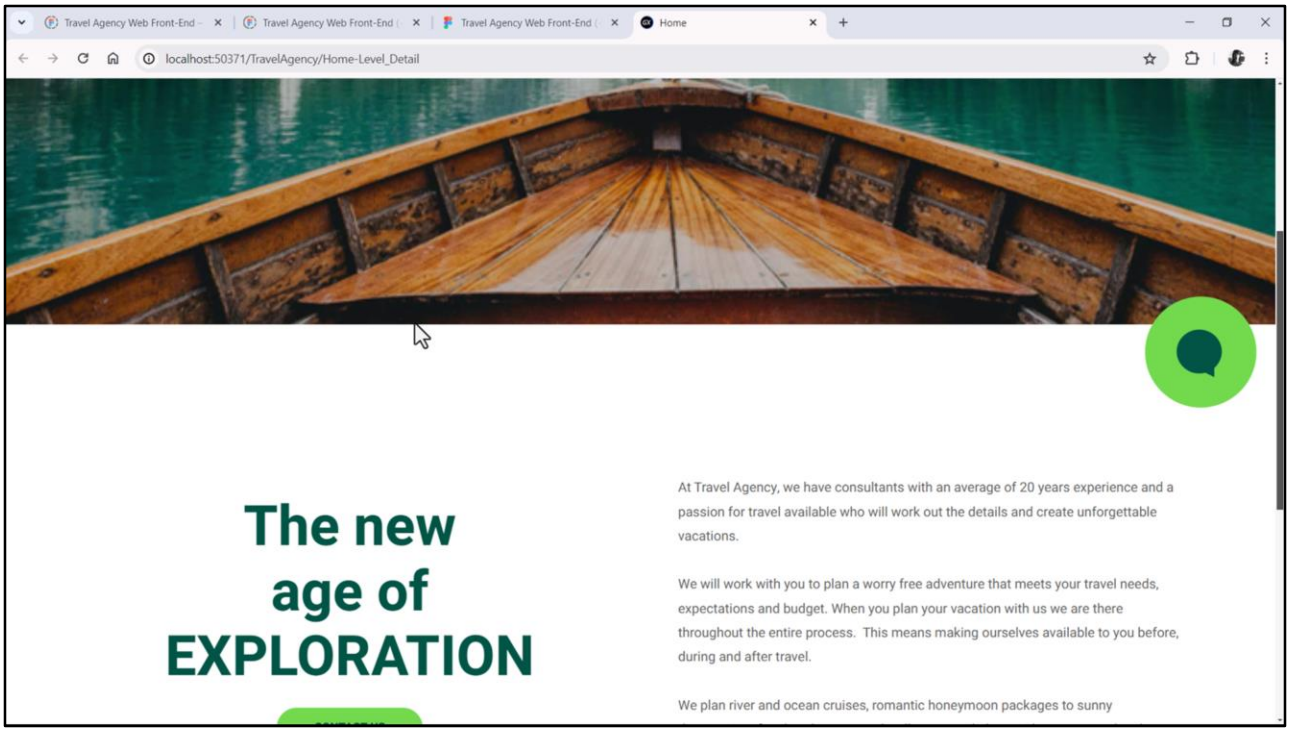
…al que le llamaré igual que al Master Panel. En él importaré el TavelAgencyBase, y moveré de este las clases que especificamos para el Header por el momento.



Y por último importo también en el DSO padre a este otro.

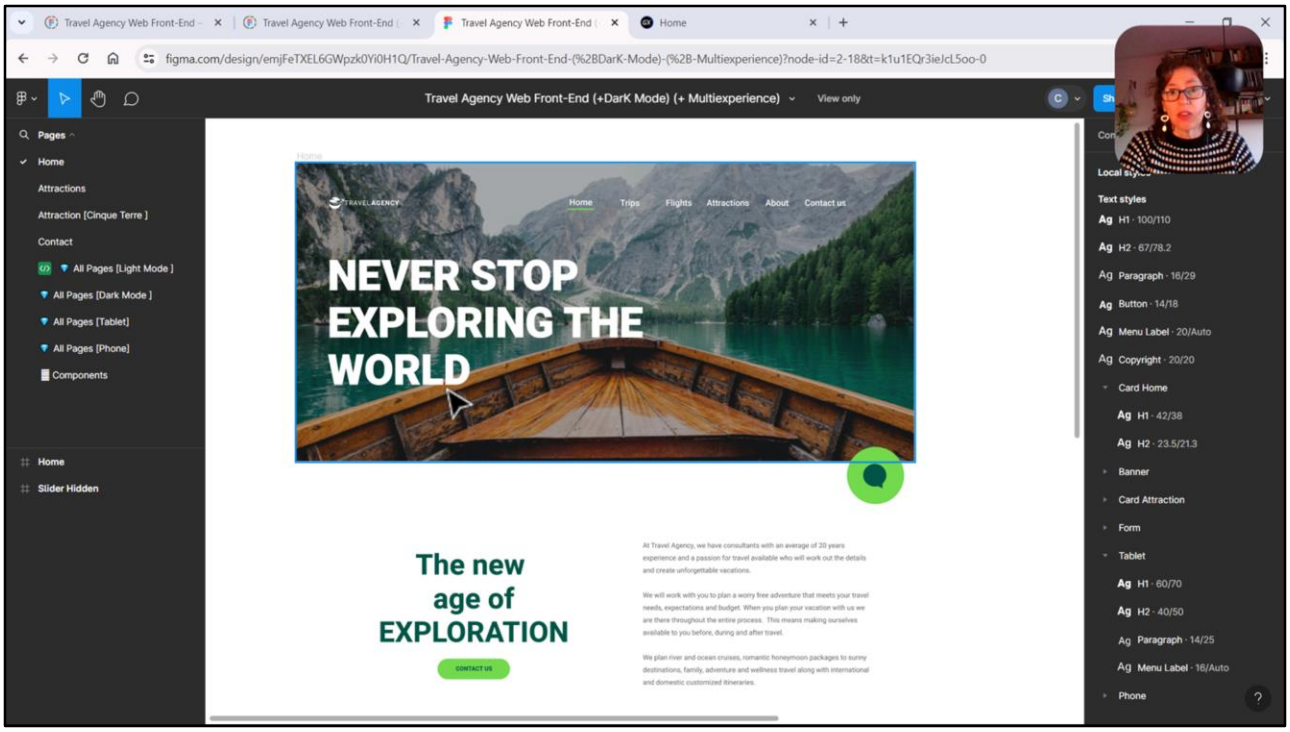


Todo debería seguir funcionando tal cual, dado que la plataforma en la que estamos ejecutando sigue teniendo como su DSO este, TravelAgency, que ahora importa estos otros dos. Por lo que contendrá tanto las clases y tokens del Base, como las de este otro.



Probemos ejecutar para comprobar que se vea todo exactamente igual.

Y sí, se ve todo igual.



Bueno, como este video ya quedó un poco largo, vamos a continuar en el siguiente con el texto que queremos superponer a la imagen, el logo, y el menú.


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