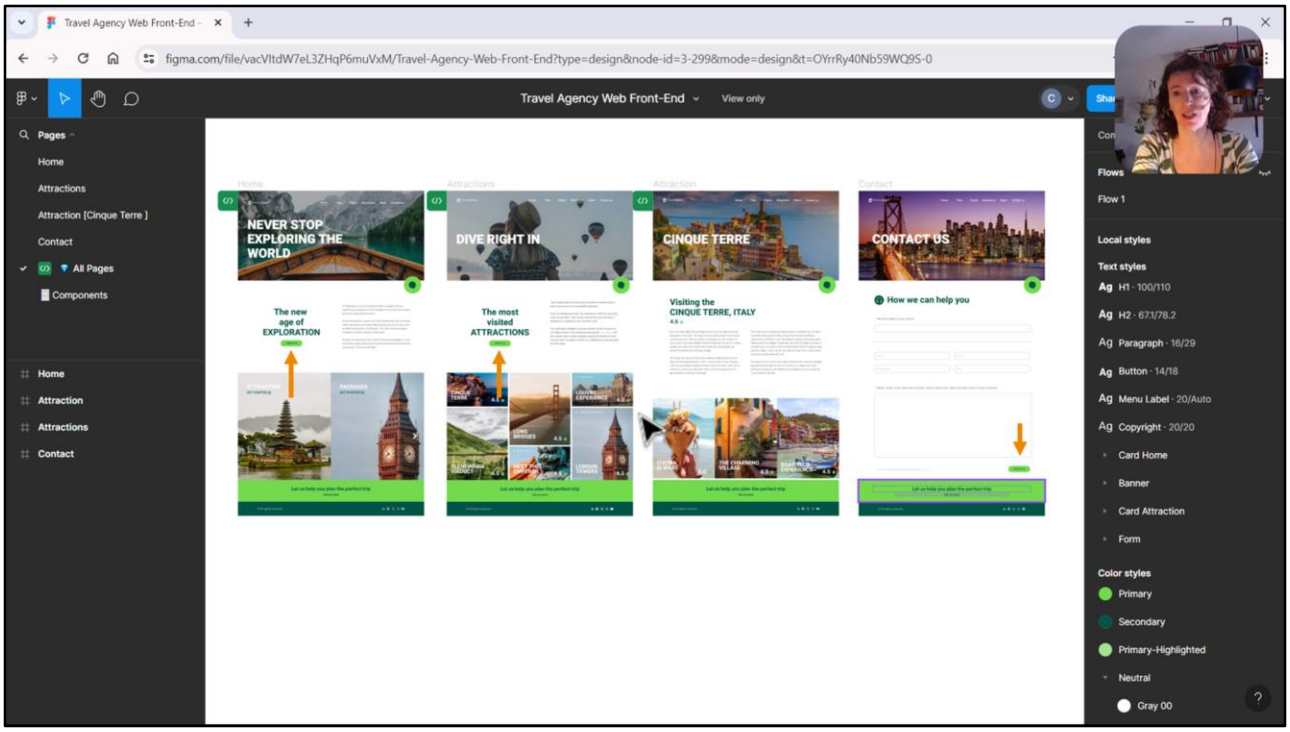
GeneXus by Globant

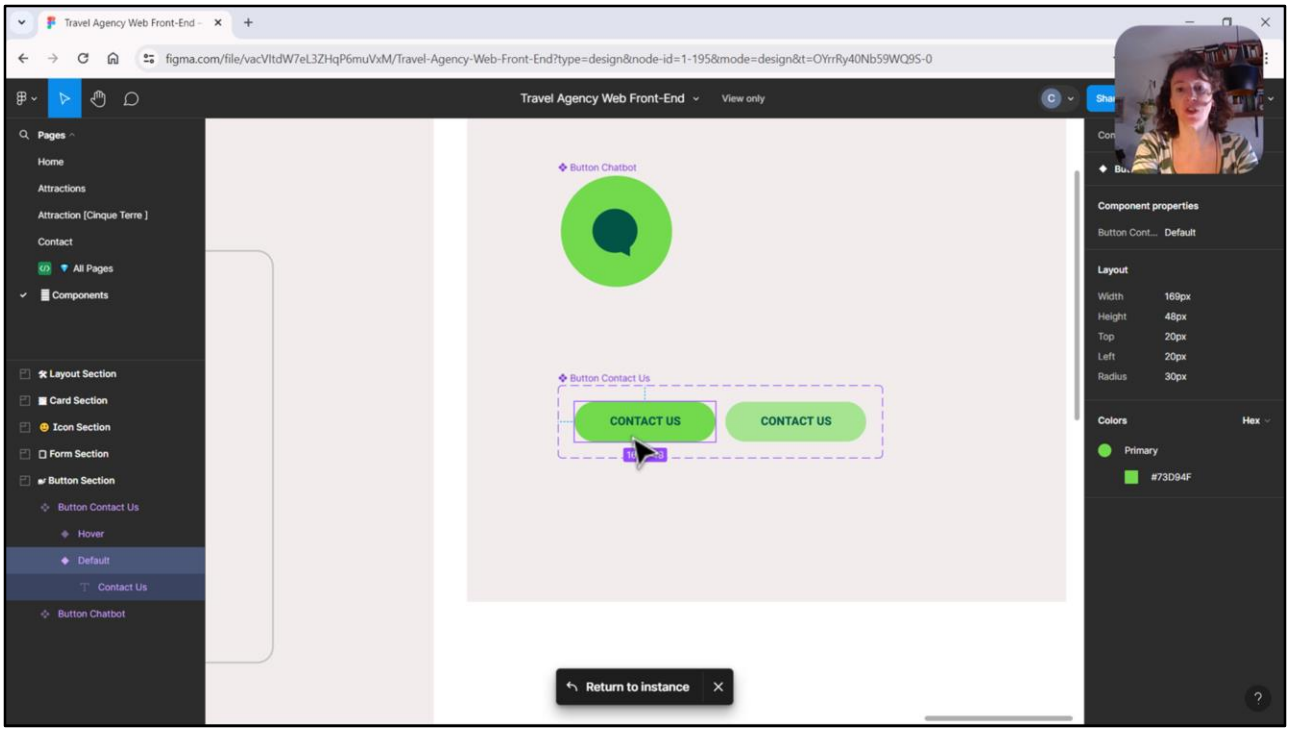
First Layout in GeneXus. Style (cont.)

Button style with behavior

Cecilia Fernández



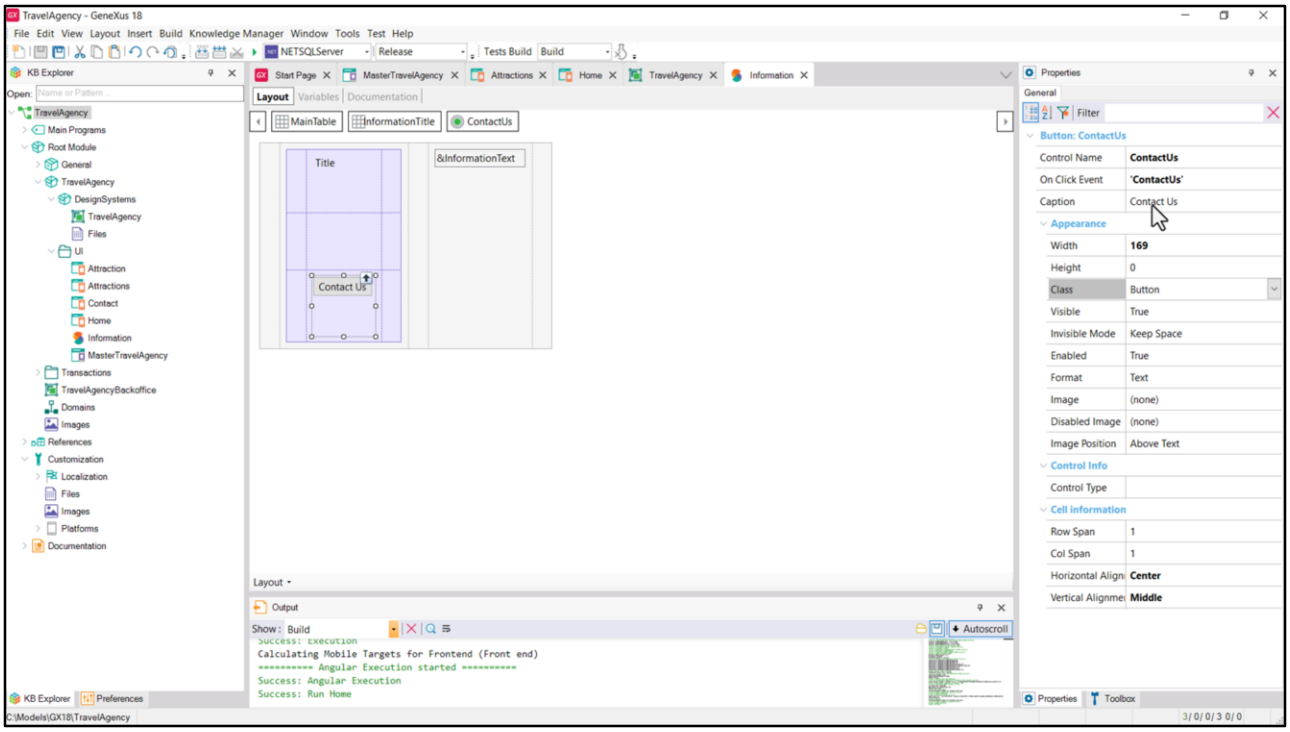
Bueno, en este video vamos a enfocarnos en el botón que aparece en estas pantallas, en estas tres pantallas: la Home, en la de atracciones y en la de contacto.



Justamente debido a estas repeticiones es que Chechu eligió modelar este elemento como un componente. Y entonces acá lo vemos en la página de componentes.

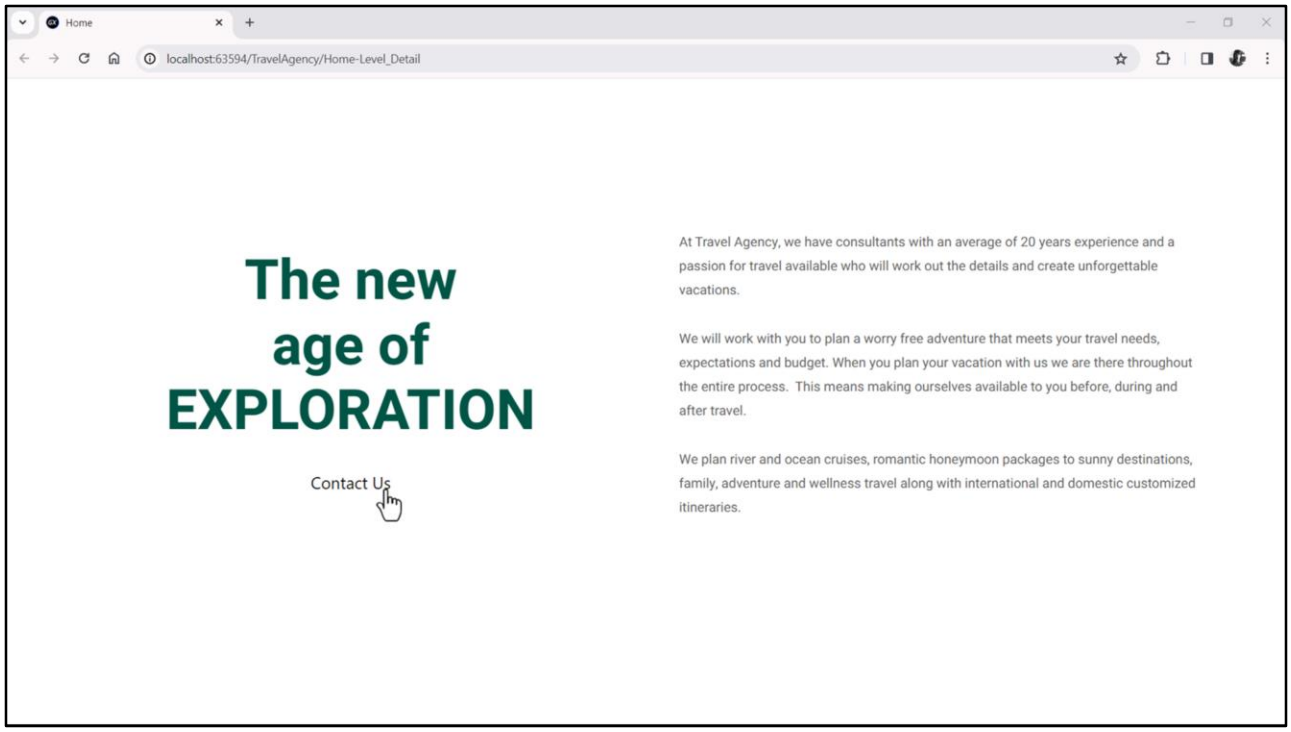
De hecho acá estamos viendo una variante del botón, que es la variante default, que representa cómo se va a ver el botón en todos nuestros layouts. Pero también Chechu modeló la variante Hover, que es cuando se hace hover sobre el botón.

Vamos a empezar por la variante Default.



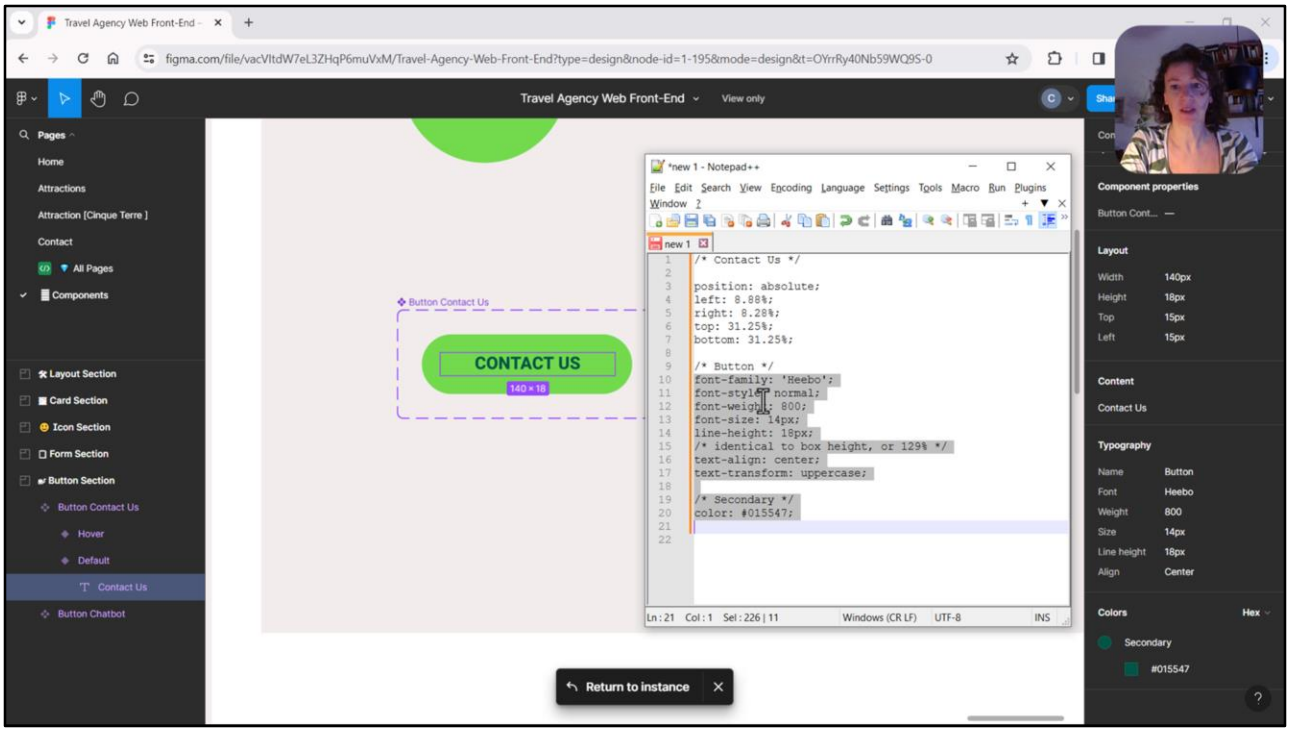
En GeneXus nosotros contamos con el control de tipo botón, que fue el que insertamos en el Stencil, y al que le configuramos este Caption.

Si prestamos atención a las propiedades, vemos que tiene como clase asociada la de nombre Button… aquí vemos el botón en el panel Home.



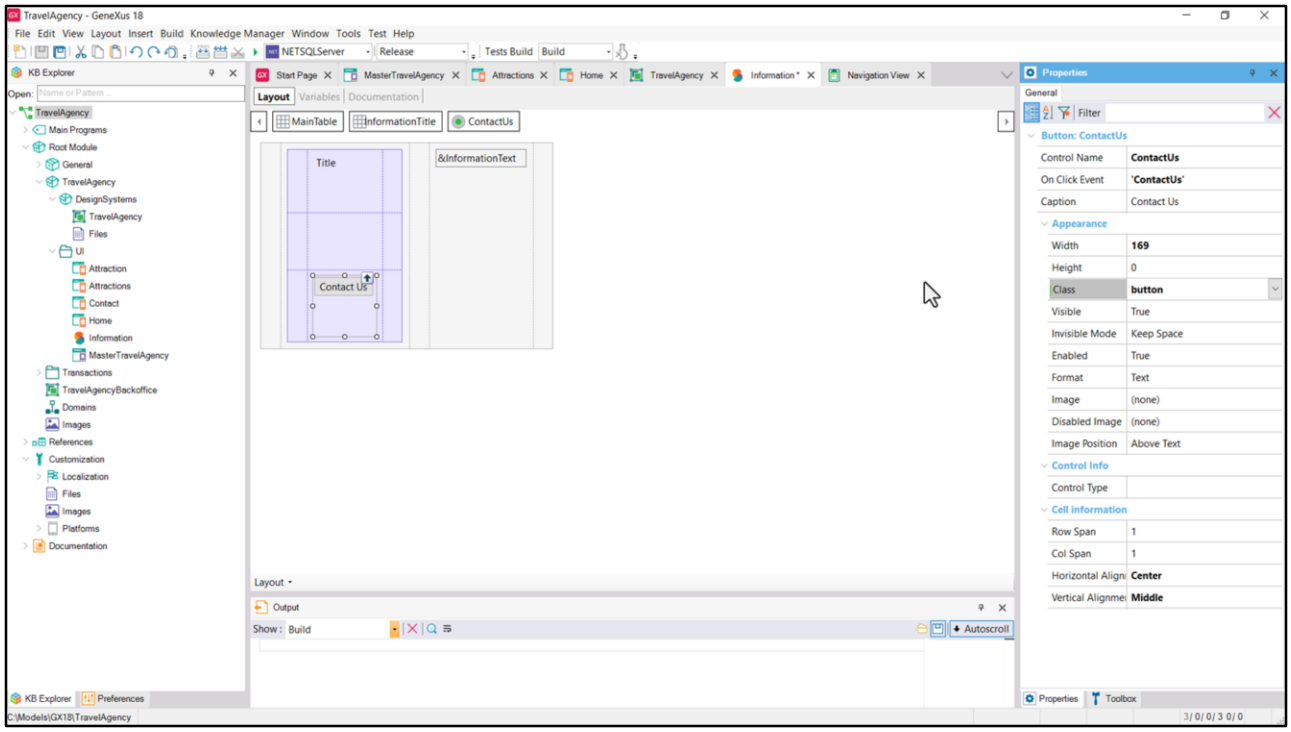
Esa clase no está definida en nuestro DSO, y es por ello que en ejecución vemos así el botón…

Podemos inspeccionarlo…

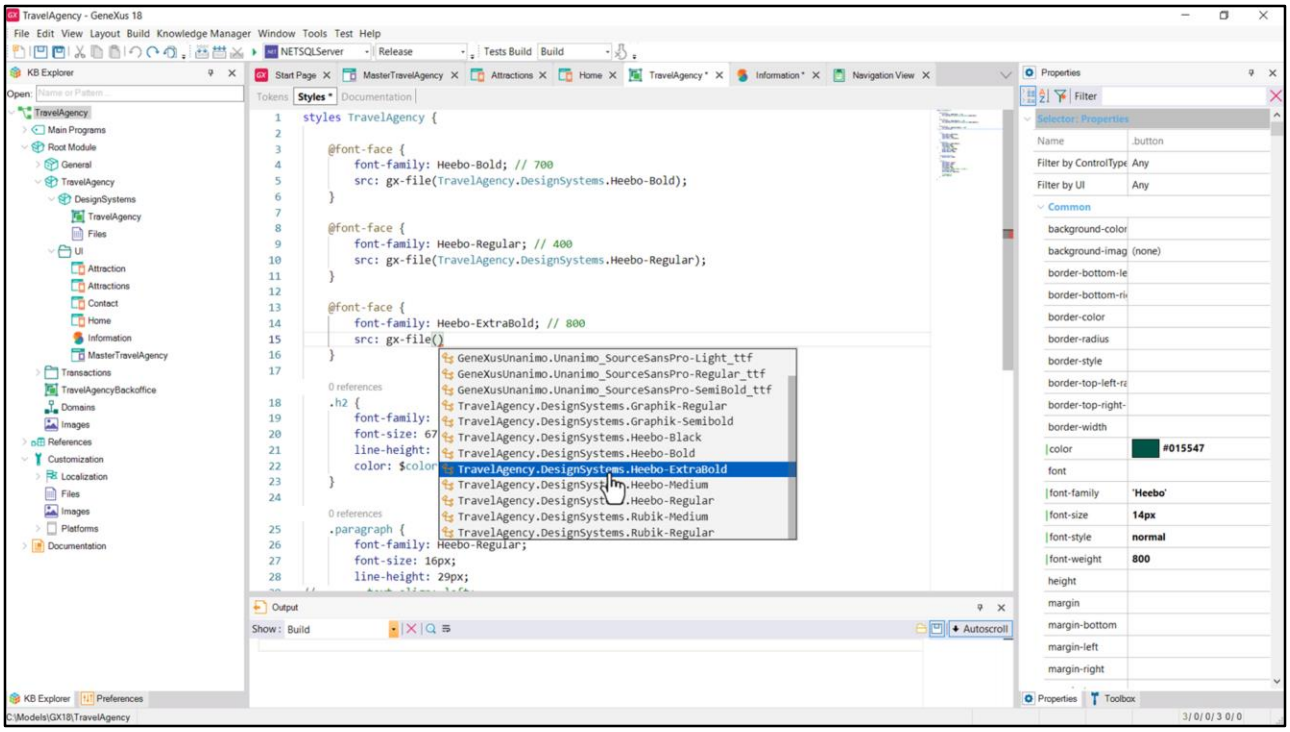


Ahora bien, en Figma no tenemos un elemento de tipo botón. Para construir un botón tenemos que hacerlo agrupando dentro de un contenedor dos capas: la capa del fondo, y una capa superpuesta que va a ser la que va a contener el texto, que sería el caption en el caso de nuestro botón GeneXus. De hecho, si vamos a observar las propiedades de este texto, que corresponde a la capa superior, vemos que Chechu le llamó a la tipografía, creó un estilo tipográfico de nombre Button, con estas características.

Lo que podemos hacer es copiar el código CSS como habíamos hecho antes, pegarlo en un Notepad, copiar las propiedades…

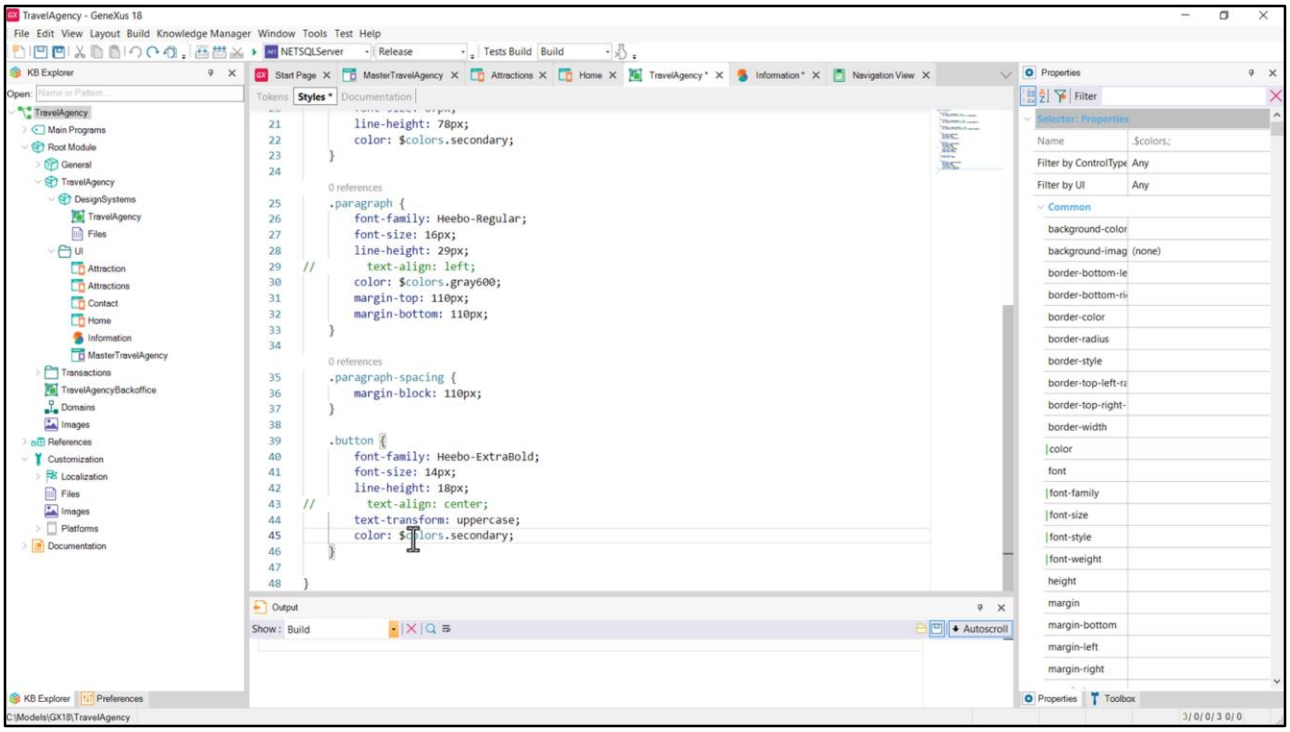


… e ir a GeneXus y asociárselas dentro de nuestro DSO a las de la clase Button. Pero recordemos que como estamos utilizando la nomenclatura BEM vamos a usar el nombre de clase en minúscula –como esto es case sensitive no va a ser lo mismo la clase en minúscula que la clase en mayúscula- bueno, la vamos a nombrar en minúscula…



y vamos a venir a nuestro DSO y vamos a escribir esta clase con minúscula.

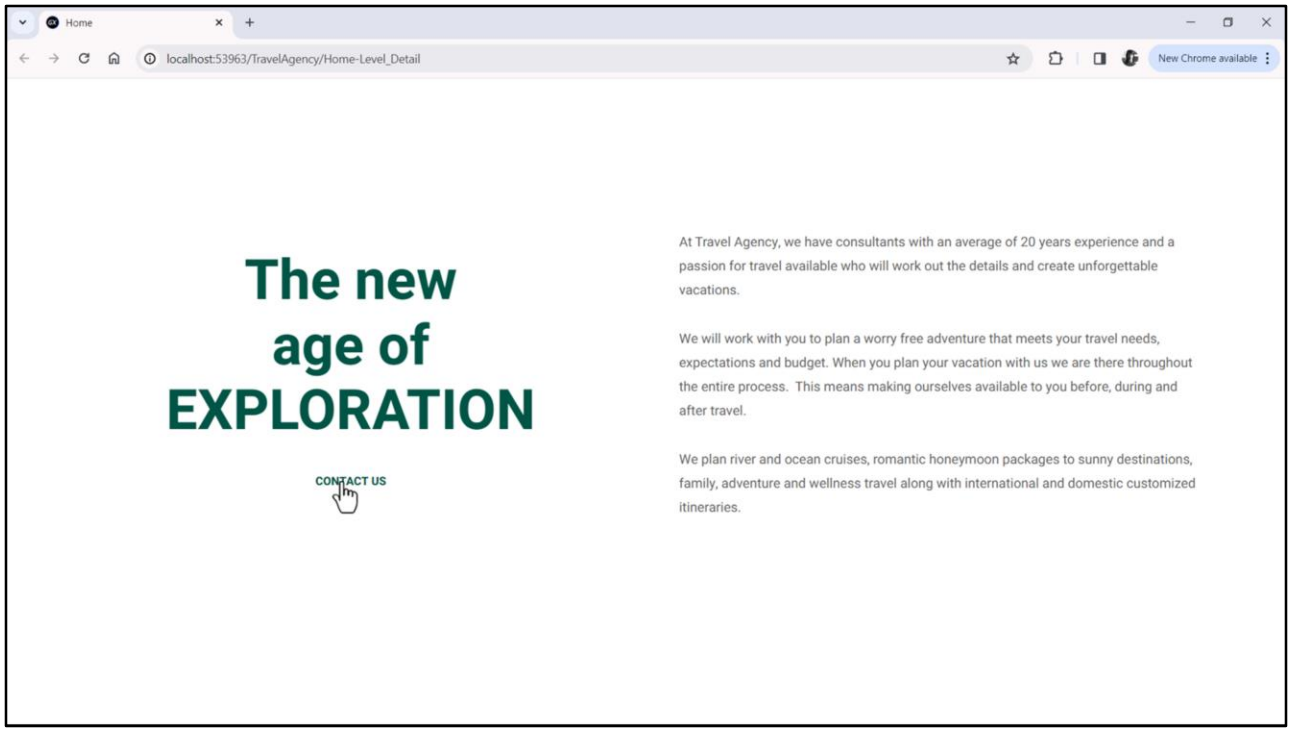
Copiamos aquí dentro las propiedades… y vamos a hacer lo mismo que hicimos antes para los otros controles, de los videos anteriores, es decir, vamos a introducir la familia de fuentes, esta Heebo que vemos que es de peso 800. Entonces venimos aquí y especificamos la regla font face… le voy a llamar ExtraBold… va a corresponder a este archivo que inserté antes en la KB.



Y ahora lo que hago, como sabemos, es esto. Bien.

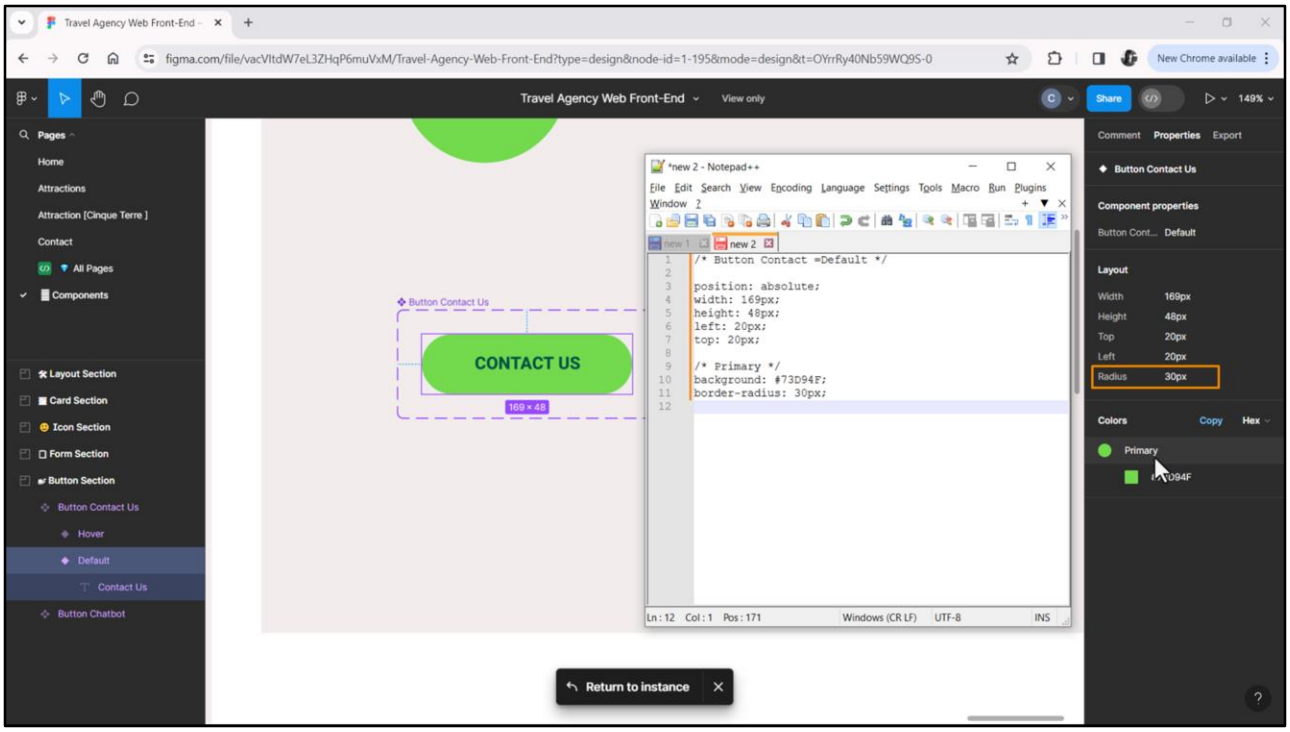
Text-align center no lo voy a necesitar, como vamos a ver. Y aquí tenemos la propiedad CSS text-transform que es para hacer que, independientemente de cómo esté escrito el texto, siempre quede en mayúsculas, todo en mayúsculas. Y luego, por otro lado, al color le vamos a asociar, no el valor absoluto, sino nuestro token de color, secondary… teniendo este cuidado… que acá nos quedó esto mal. Bien.

Vamos a probar esto así como está.



Bien, vemos el cambio, ¿verdad? Que pasó todo a mayúsculas, que tiene el color que queríamos, el tamaño, etcétera.

Ahora nos está faltando lo que tiene que ver con el background.

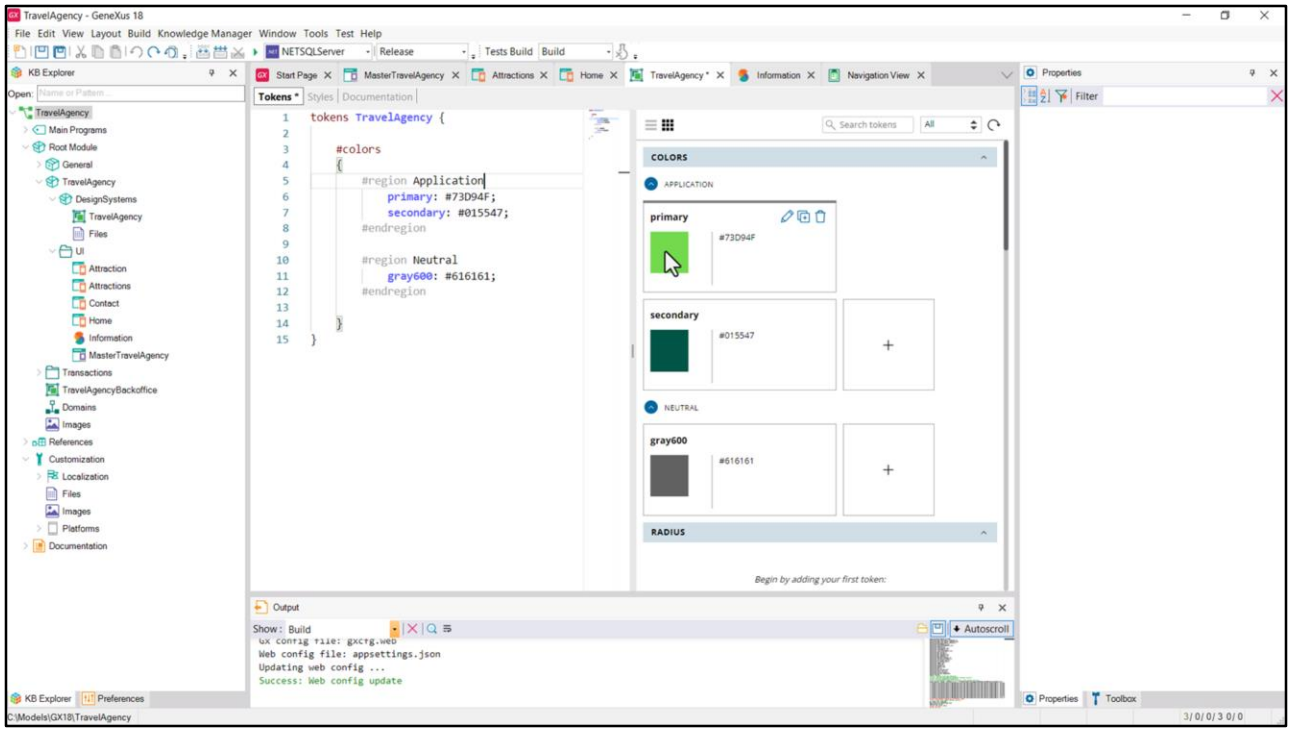


Entonces venimos a Figma y nos posicionamos sobre ese background.

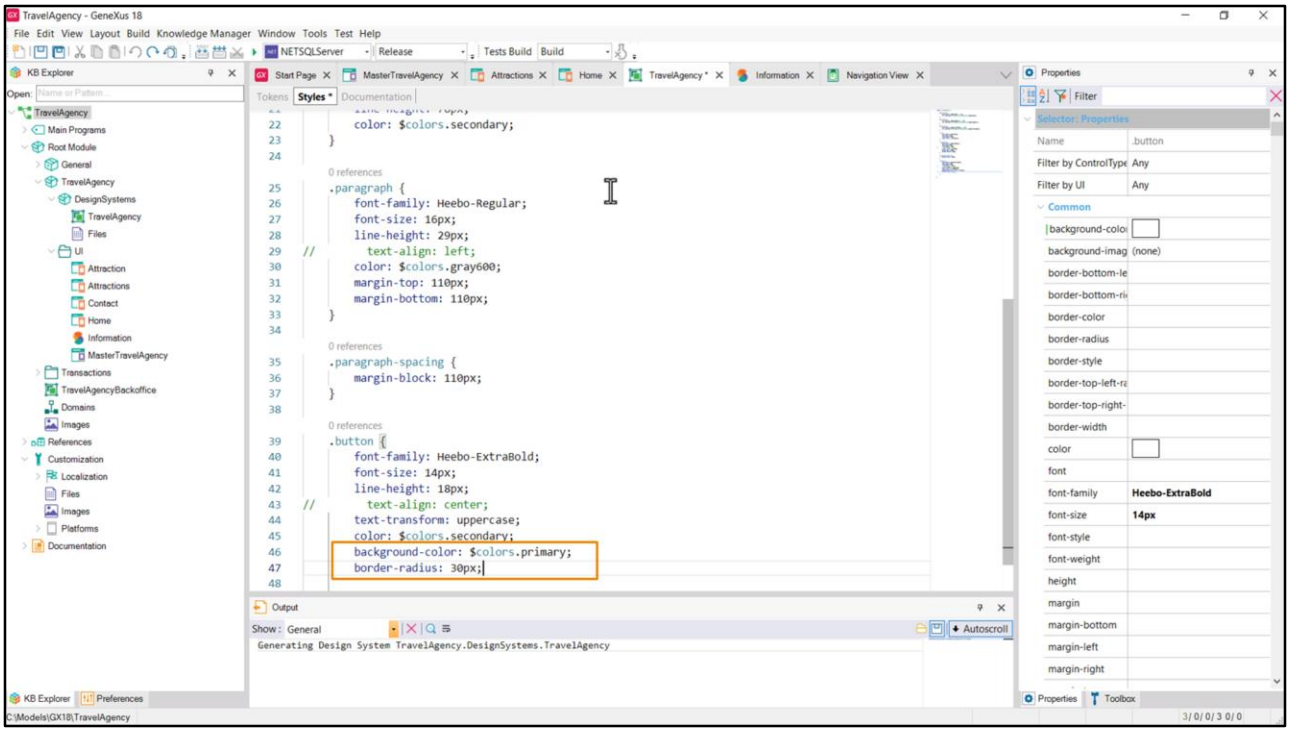
Y aquí vemos las propiedades. En particular atendamos a esta Radius, que es la que va a dar justamente estos bordes redondeados.

Vamos a, otra vez, elegir las propiedades CSS, pegarlas aquí… y vemos entonces la propiedad de color, background, que es esta de aquí, que está mostrando este color, verde, al que Chechu le dio un nombre al estilo de color, le llamó Primary. Que es lo primero que vamos a hacer, convertir esto en un token primary, con este valor. Y luego, además, tenemos que agregar esta propiedad CSS a nuestra clase button.

Entonces… vamos a copiar primero esto…

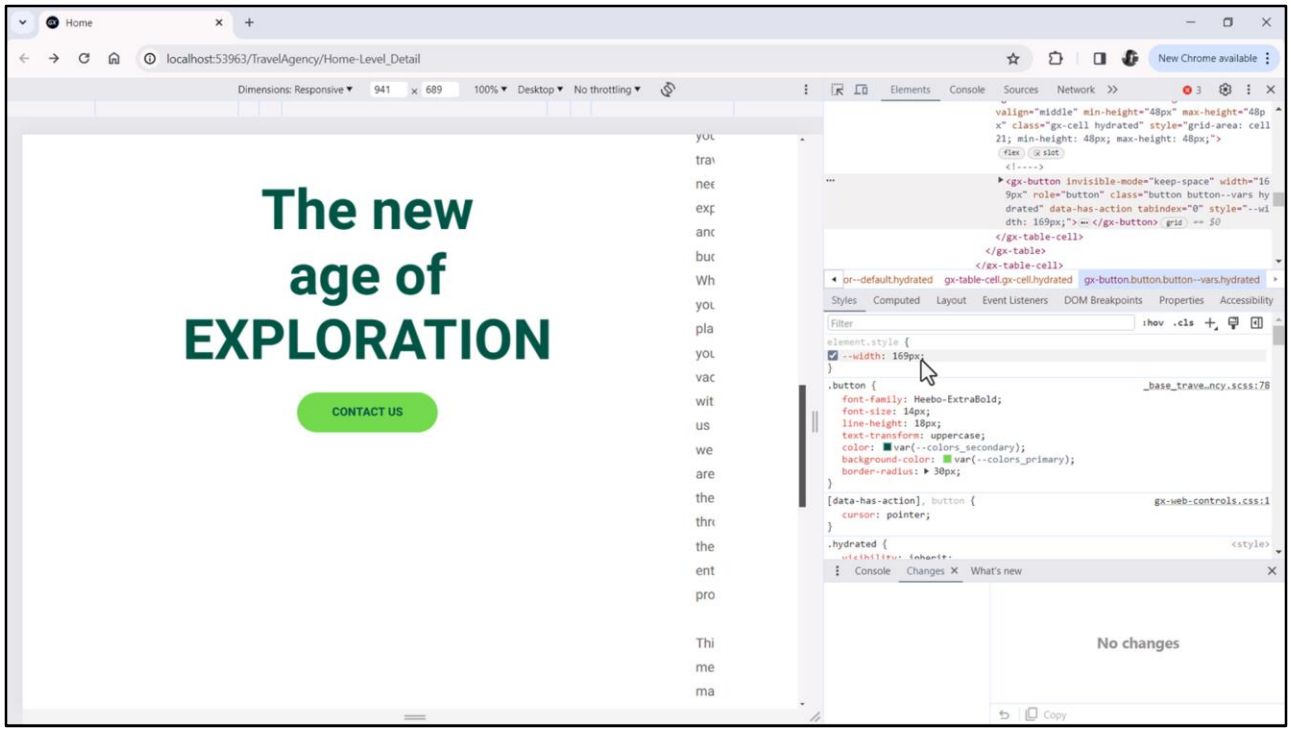


…venimos a nuestro DSO, a al solapa de tokens y agregamos este token, el primary.



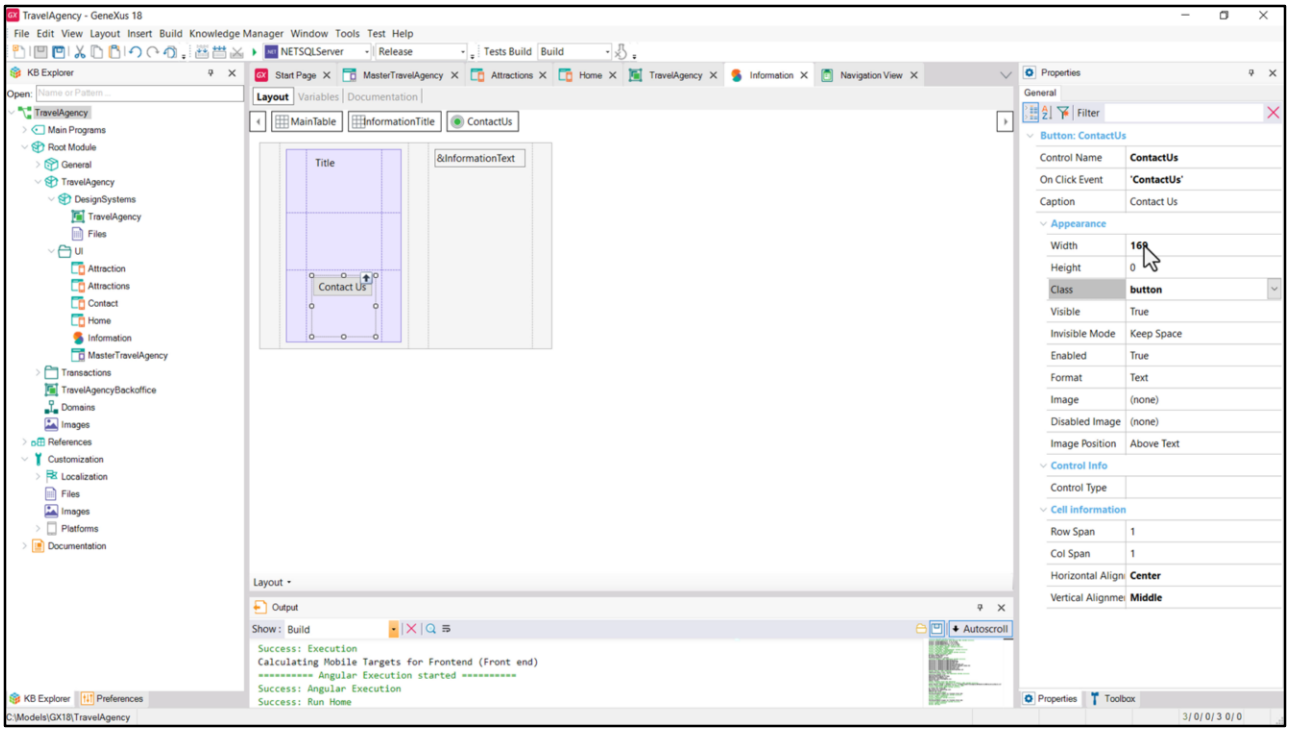
Y luego vamos a la clase, agregamos la propiedad background-color… y por otro lado vamos a copiar esta otra.

Probemos esto así ahora…

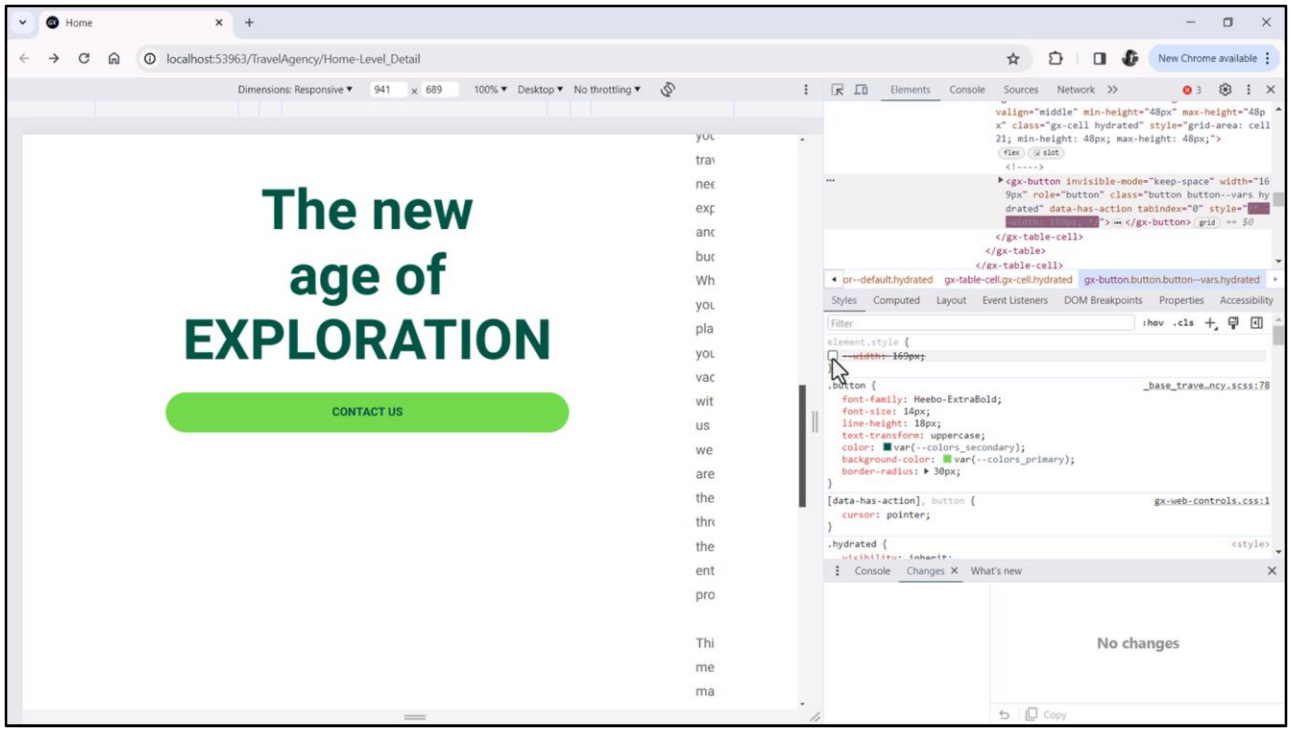


Bueno, parecería que ya con esto estaría.

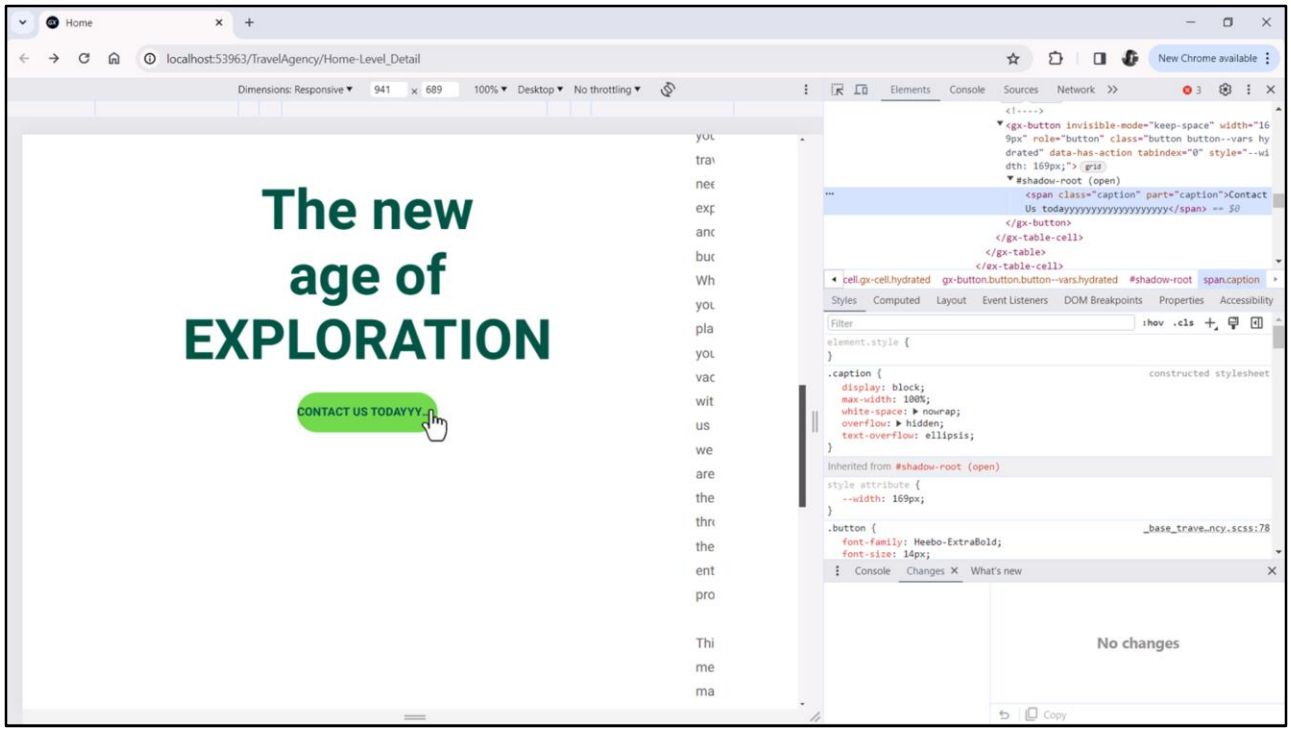
Sin embargo, pensemos qué pasaría si tradujéramos la aplicación a un idioma en el cual este texto fuera más largo. ¿Qué pasaría con el ancho del botón? Si lo inspeccionamos vemos que está asumiendo como ancho este valor, 169 píxeles, que ¿de dónde sale?



Del valor que le habíamos dado a la propiedad Width a nivel del botón dentro del stencil. Habíamos prefijado el ancho del botón a este valor. Y habíamos dicho que si dejábamos el valor default que era un 0, lo que iba a pasar era que el botón se iba a expandir para ocupar el ancho de la celda en la que se encontraba.

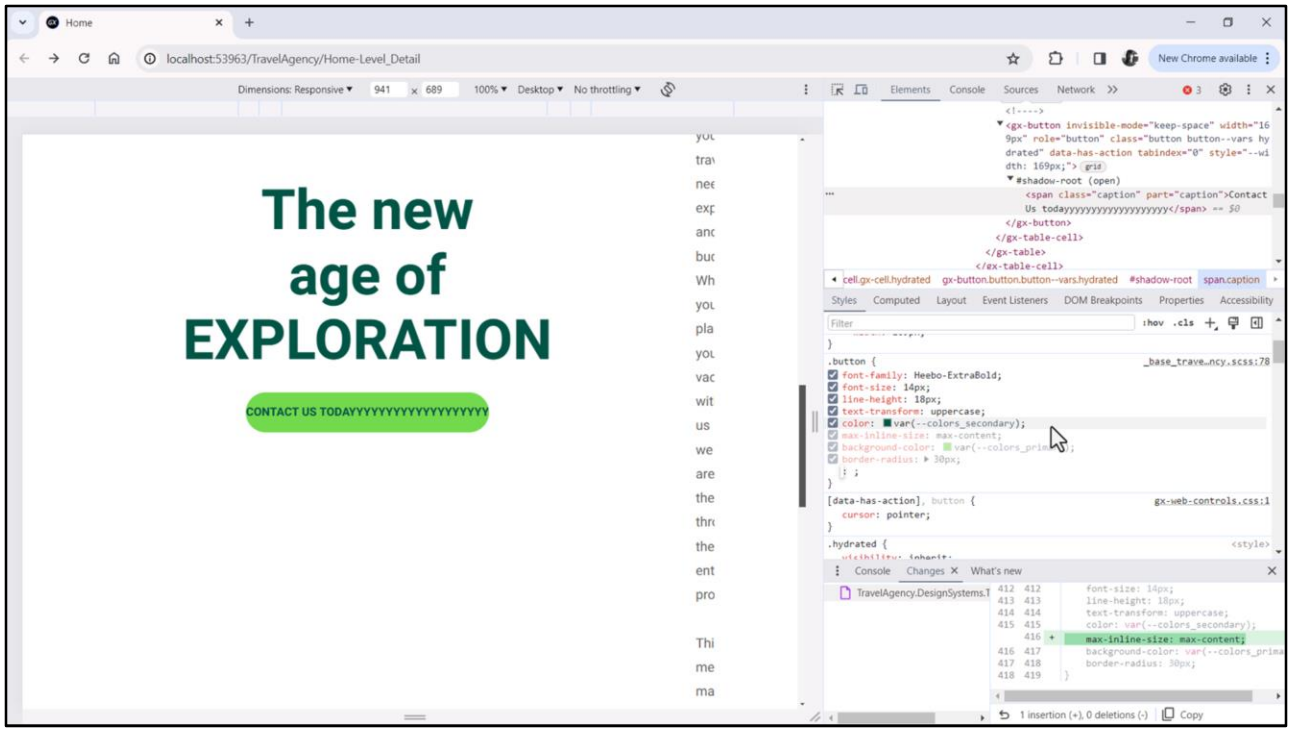


Y lo podemos ver muy claramente aquí mismo si hago esto.



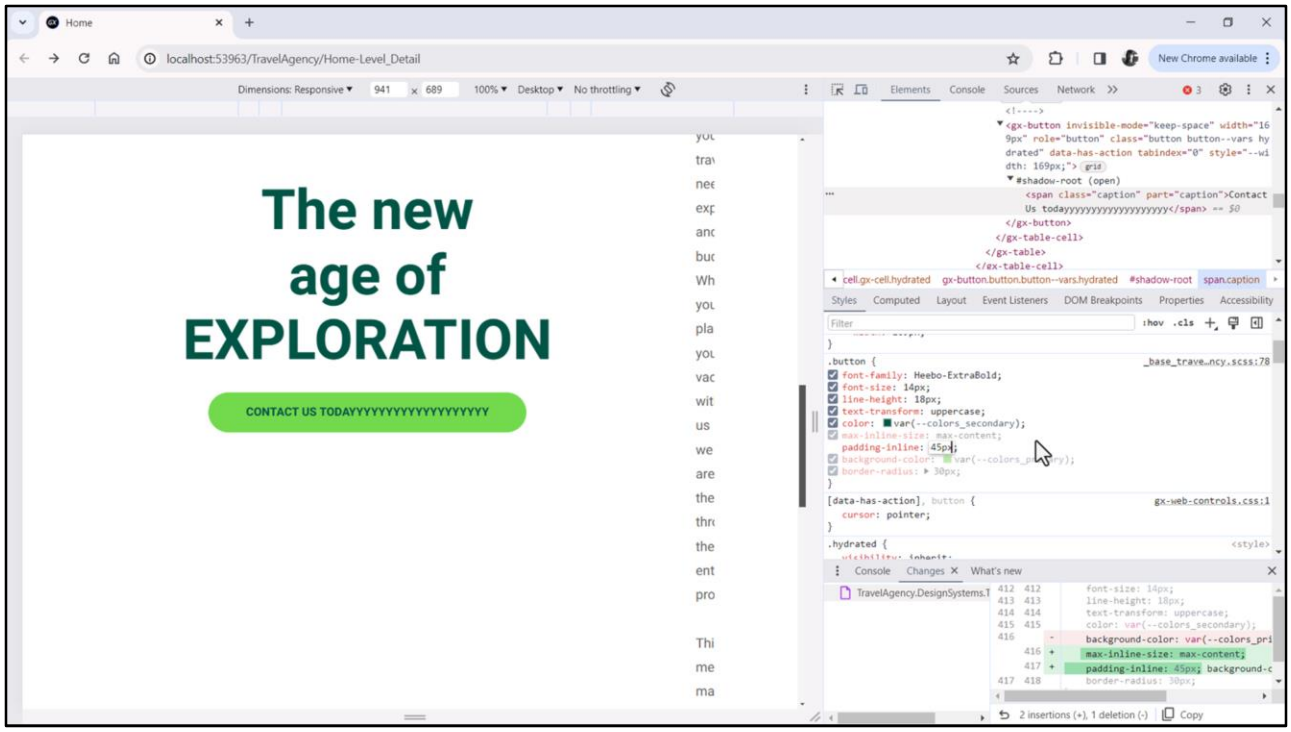
Bien, pero, preguntaba, qué pasa entonces si agrego a este Caption, si lo hago más ancho. Por ejemplo…

Bueno, lo que estamos viendo es que el texto se está expandiendo hasta el límite del ancho del botón. Y allí lo que está haciendo es colocando una elipsis, es decir, truncando, cortando el texto… y este no es el comportamiento que vamos a desear. Lo que vamos a querer es que el borde del botón se expanda también, dejando un aire a los costados. Es decir, no nos va a interesar el ancho fijo de 169 dips (eso está muy bien para el texto “Contact us” exactamente, con esas dimensiones, pero no si traducimos la aplicación).



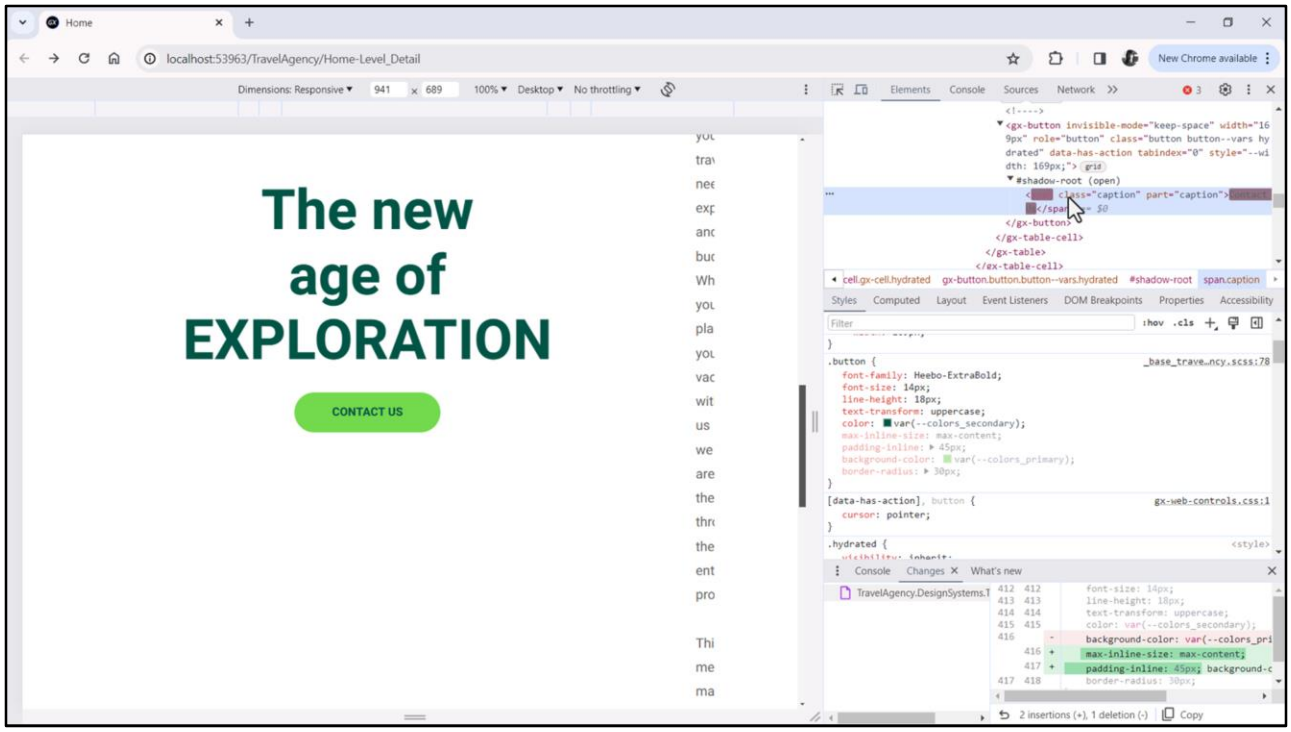
Por tanto… vamos a probar lo siguiente. Vengo a las propiedades de la clase y voy a agregar una propiedad max-inline-size, y voy a elegir como valor de esa propiedad, max-content. Es decir que el tamaño máximo inline, es decir, en la dirección horizontal, corresponda al máximo del contenido.

Y acá lo estamos viendo: cómo se expandió el botón para ocupar entonces todo el contenido.



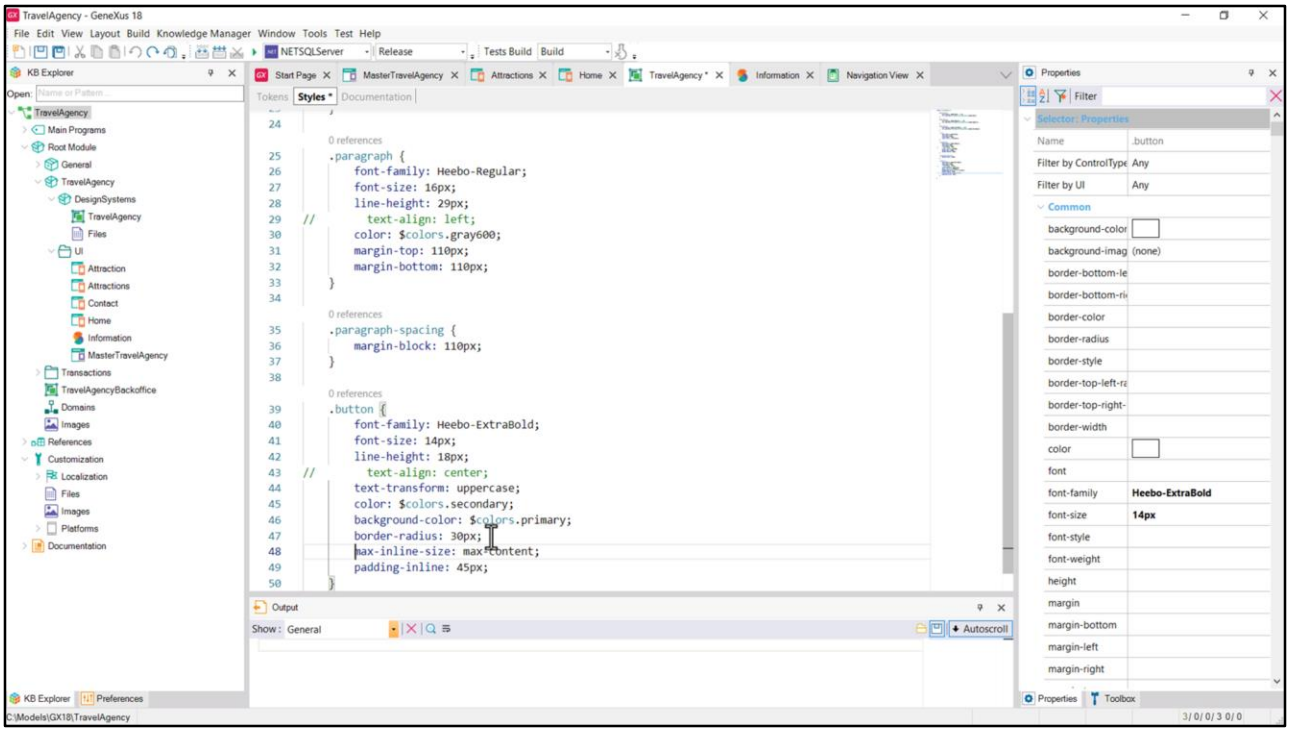
¿Qué nos estaría faltando? Dejar un padding, que en el contenido también esté considerado un padding… padding de un lado y padding del otro. Entonces, por ejemplo, probemos, padding-inline (para utilizar, otra vez, como sugeríamos antes las propiedades lógicas en lugar de las físicas), y por ejemplo coloquemos un 45 píxeles. Y ahí lo vemos.

Está entonces teniendo 45 píxeles respecto al inicio, 45 píxeles respecto al fin y el ancho del botón está siendo el que necesita para que todo ese contenido esté envuelto.

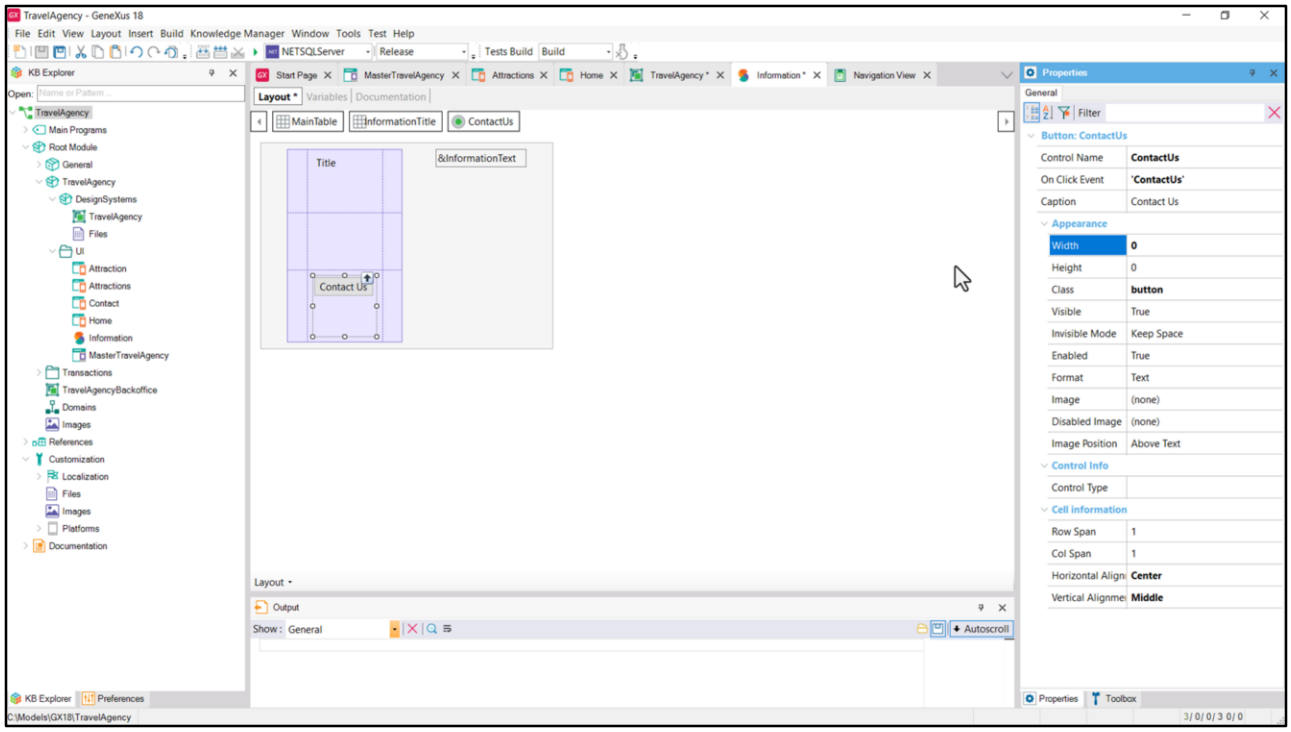


De hecho, observemos que si ahora volvemos a dejar el caption como queríamos, Contact Us, es inapreciable la diferencia con lo que teníamos antes.

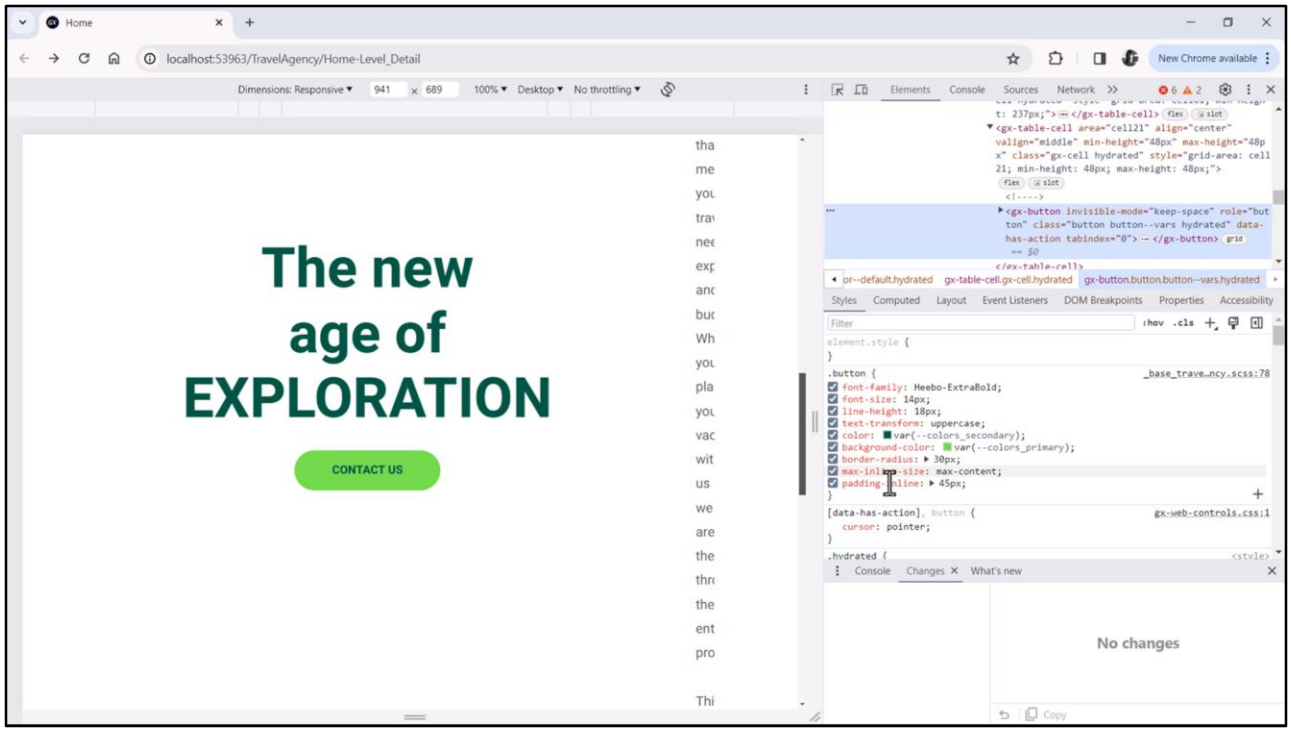
Sin embargo esta forma va a permitir que si traducimos la aplicación y cambiamos el ancho del texto el botón se resizee para que siempre se vea bien.



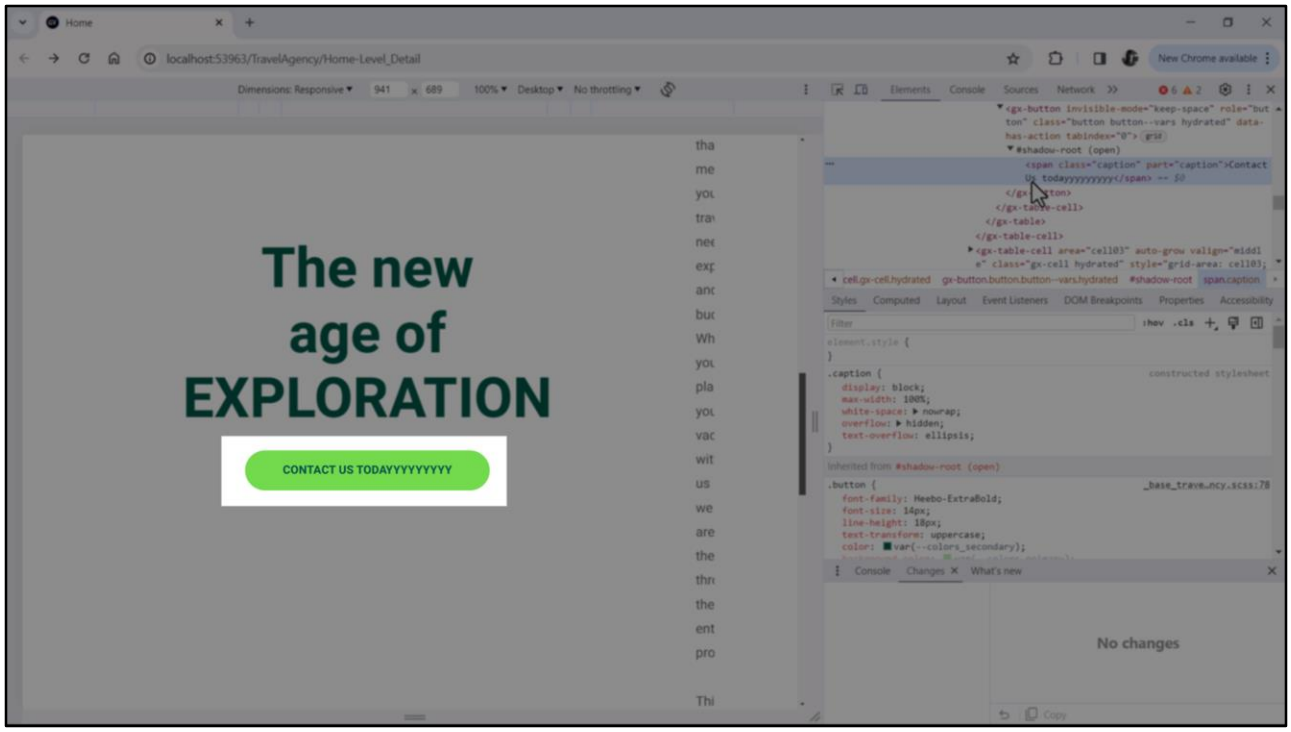
Bueno, entonces llevemos estas dos propiedades a nuestro DSO.



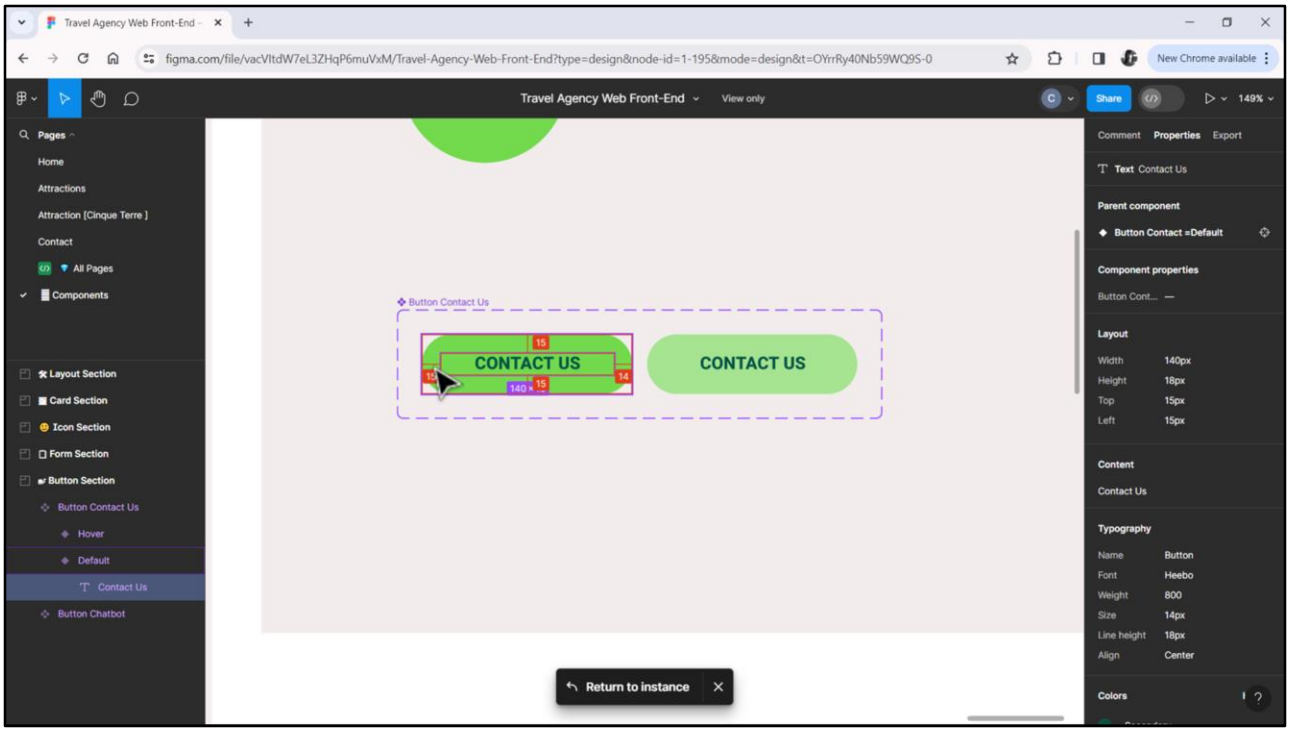
Y quitemos la propiedad Width, dejemos el default a nivel del control botón en el stencil.



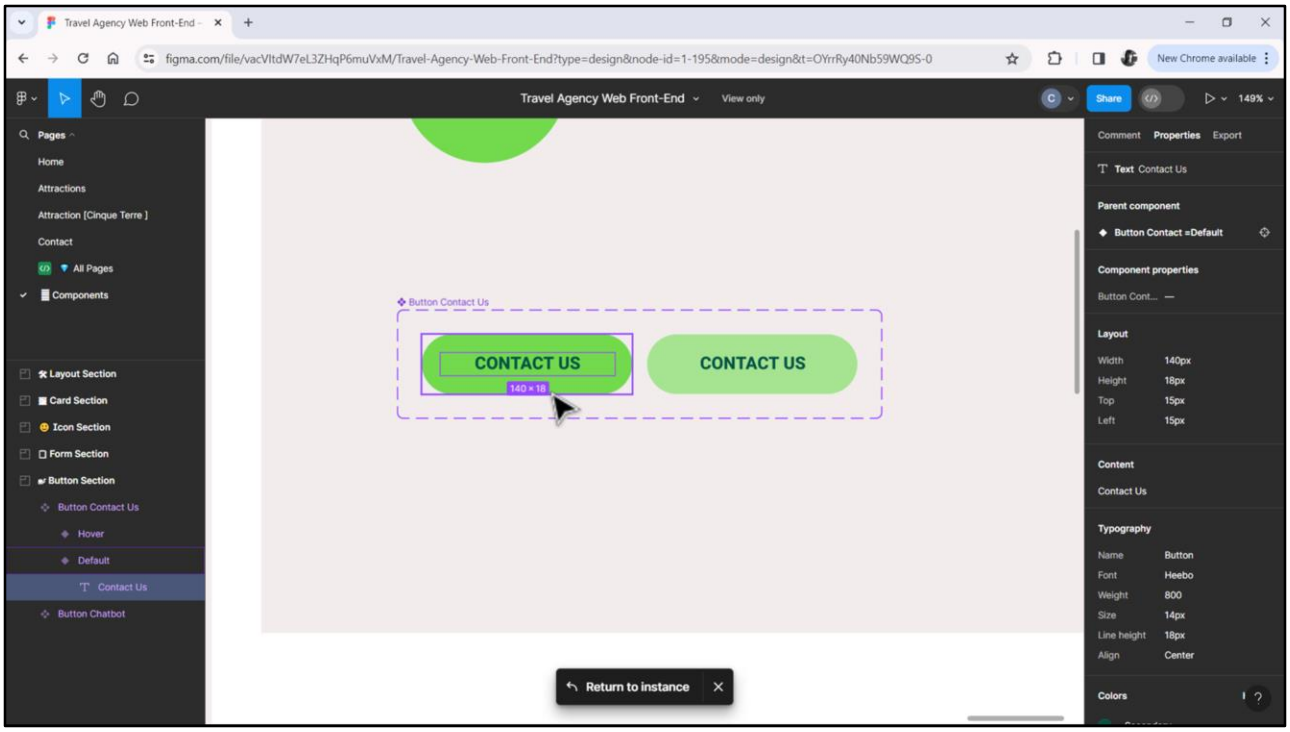
Y ahora probemos. Bueno, como decíamos es inapreciable… pero ahora vemos incorporadas las propiedades en la clase, no vemos la width, y si modificamos el caption…



…vemos cómo automáticamente se resizea, ahora sí, como deseábamos.

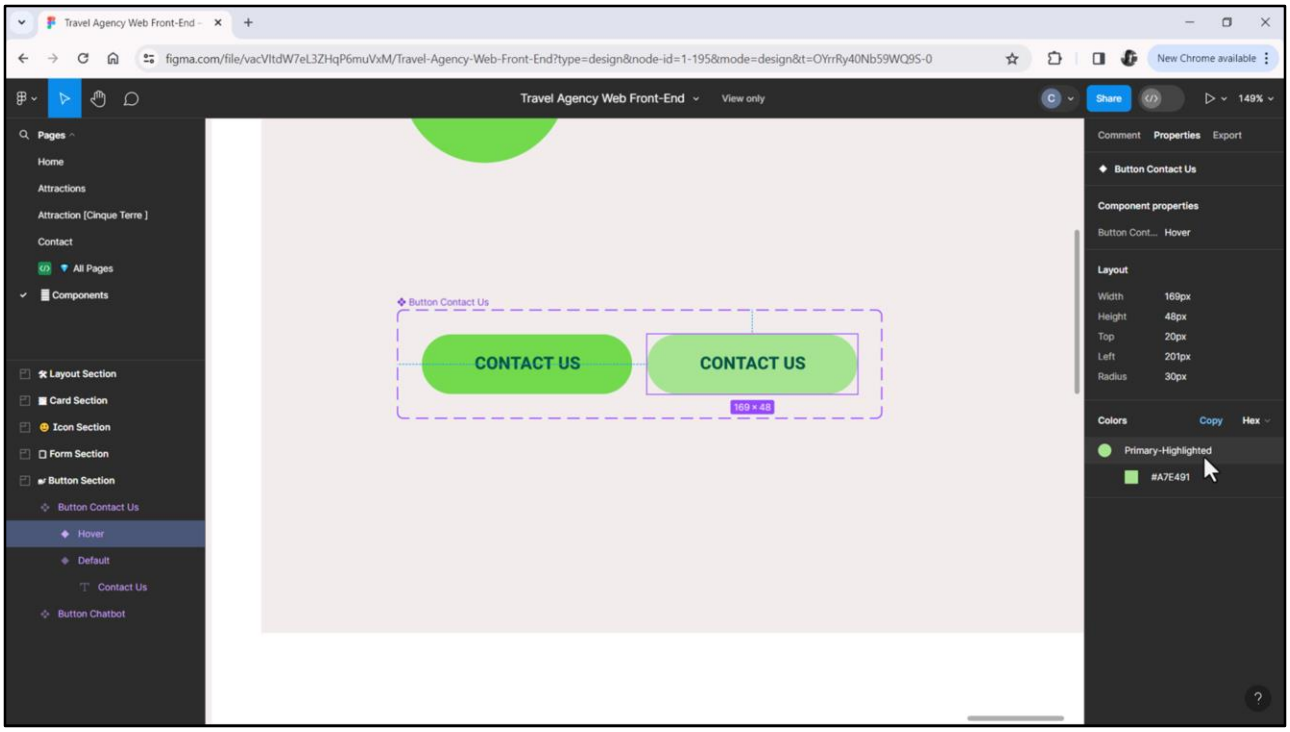


¿Podríamos habernos dado cuenta de algo de esto en el archivo en Figma? Bueno, así como quedó expresado no. De hecho, si observamos, Chechu definió la caja de texto como ocupando 140 píxeles de ancho, es decir, un ancho fijo, dejando hacia los costados 15 píxeles de un lado y 14 píxeles del otro. También podemos ver 15 píxeles de arriba y 15 píxeles de abajo, que nosotros no hemos contemplado. ¿Por qué? Porque al botón le dimos el alto de 48 píxeles que corresponden a la suma de estos 15 más estos 15 más…



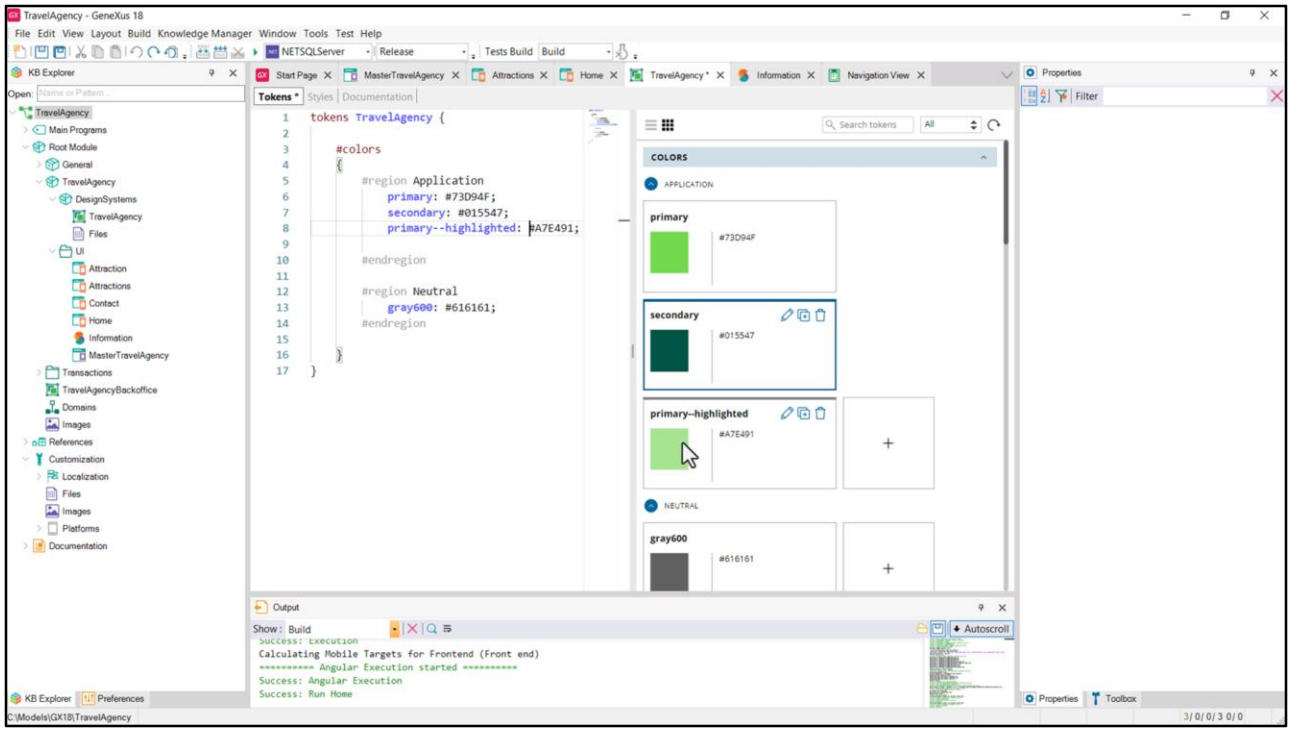
… estos 18 de acá del texto. Estos no los consideramos porque el botón no va a crecer hacia arriba o hacia abajo conforme a la traducción. La traducción lo que va a hacer es, en todo caso, expandir de derecha e izquierda, pero no en la otra dirección.

En definitiva, en nuestro caso, nuestra implementación hubiera correspondido a un diseño en Figma en el cual este ancho fuera hug, es decir, envolviera al contenido, y además tuviera un padding-left y un padding-right, si lo voy a decir en términos físicos, de esos 45 píxeles que proporcionamos.

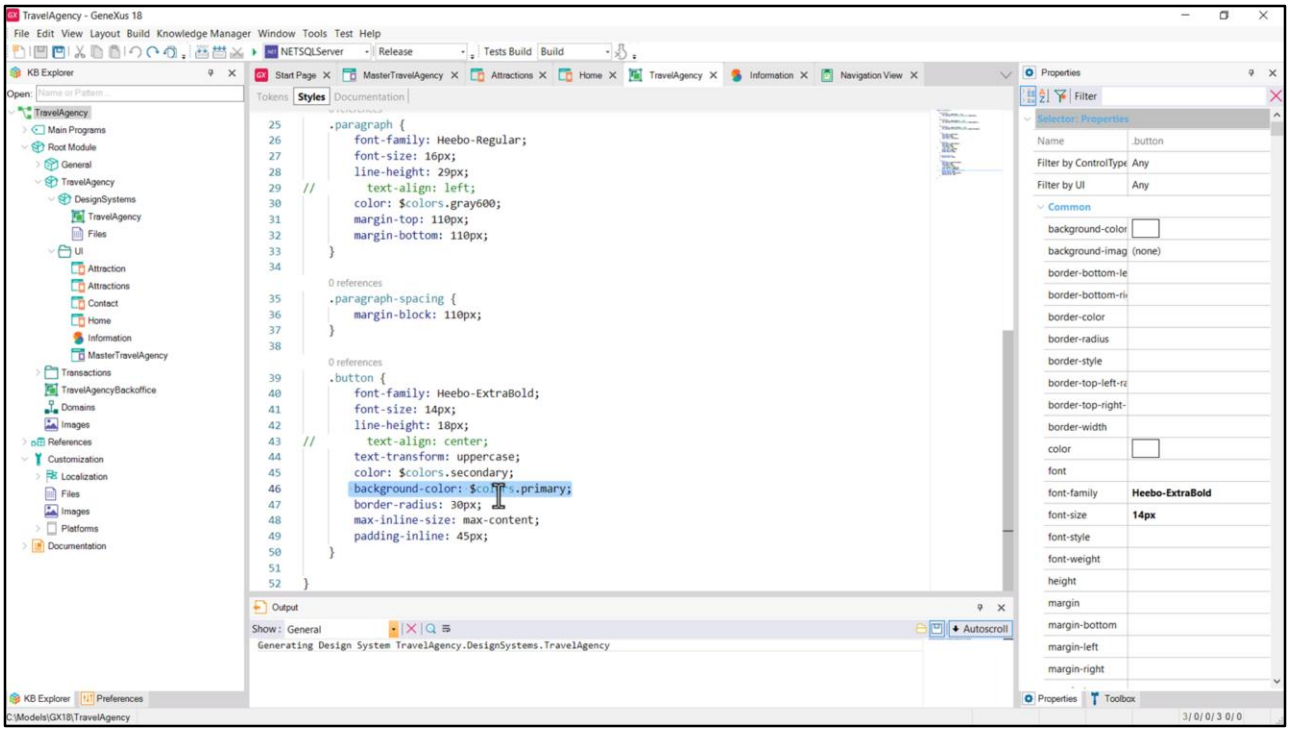


Bien, entonces ahora estamos en condiciones de estudiar cómo hacemos para que el botón cambie su aspecto cuando se hace hover sobre él, también podría ser cuando se lo cliquea, cuando se lo activa, para que asuma este estilo, que, por lo que vemos, en lo único en lo que difiere del estilo default es en el background-color. Que asume este color al que Chechu le dio este nombre, Primary-Highlighted.

Entonces lo copio…

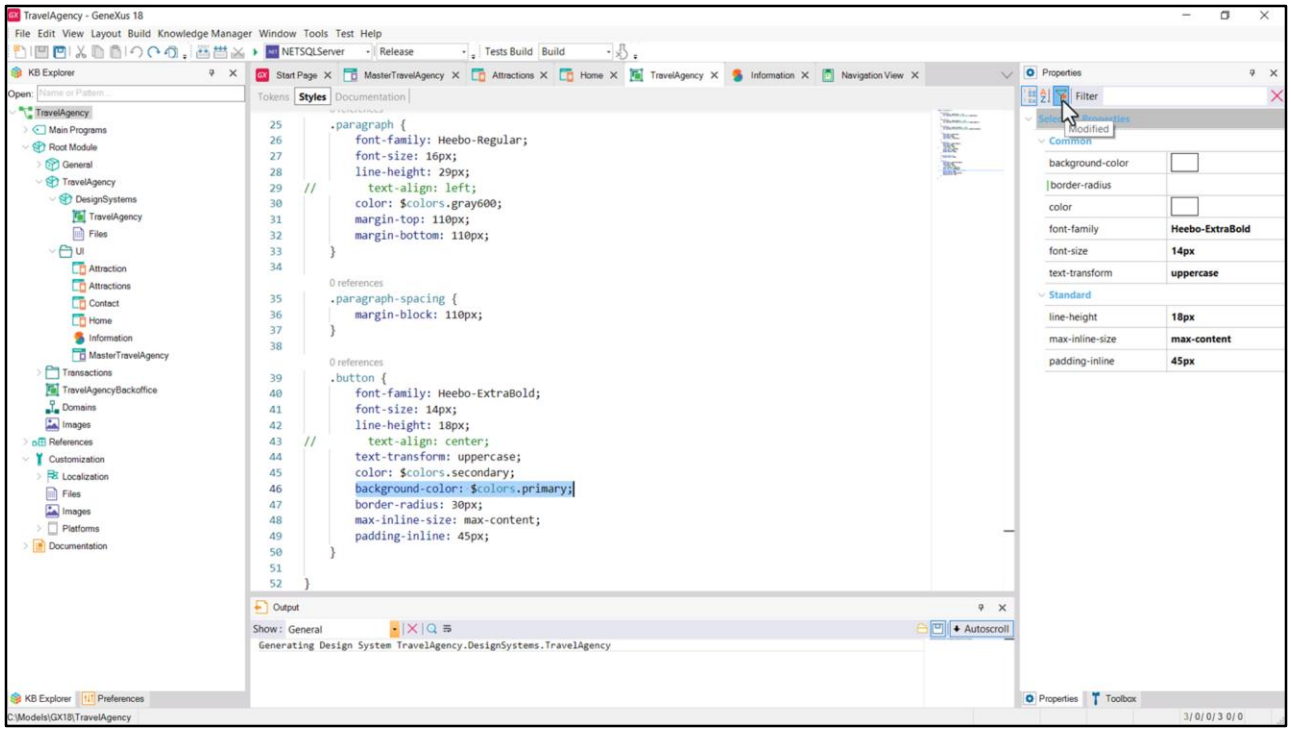


…y en la sección de tokens agrego…



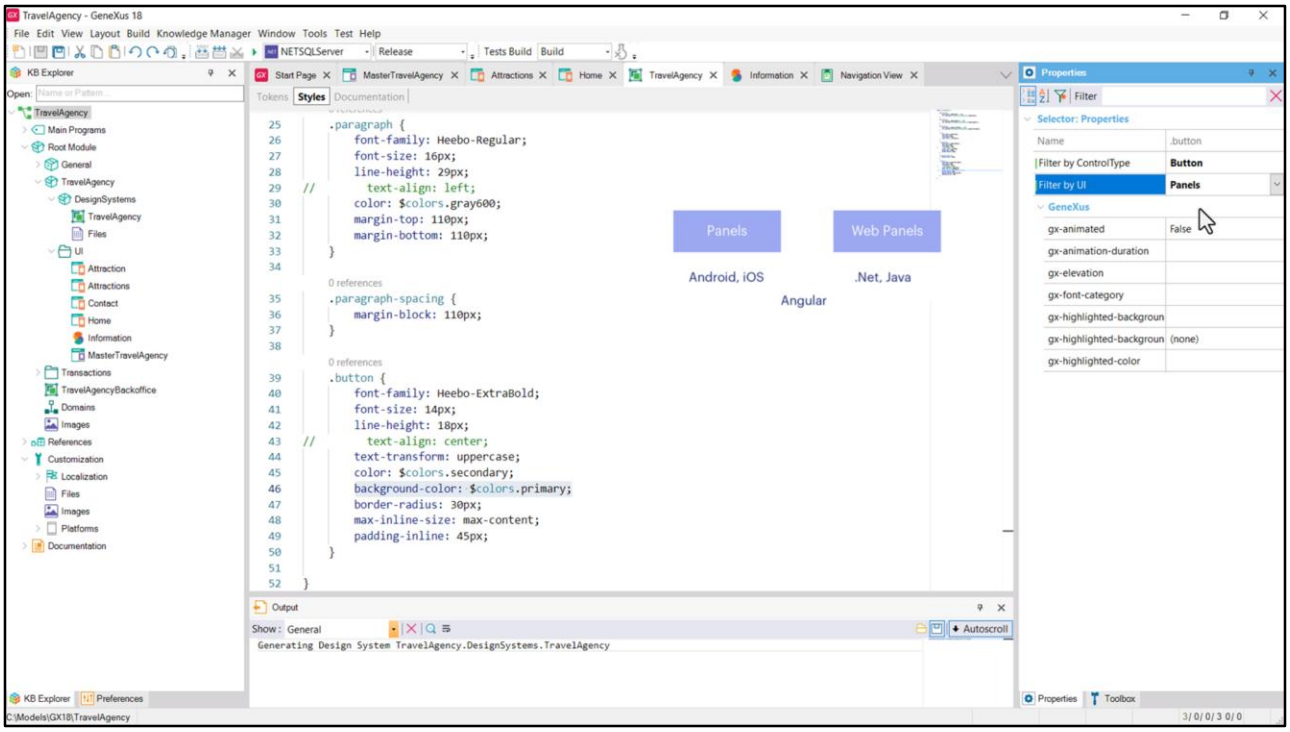
Y ahora lo que tengo que ver es cómo hago para que la clase asociada al botón modifique su propiedad background-color para asumir el token primary—highlighted cuando se activa el botón.

Notemos que en alguna medida estamos hablando de comportamiento, ¿no? Cuando el botón es activado necesitamos cambiar una de las propiedades de la clase que gobierna su estilo.



Bueno, pues, resulta que las clases no solamente soportan la utilización de propiedades CSS sino también propiedades específicas de GeneXus.

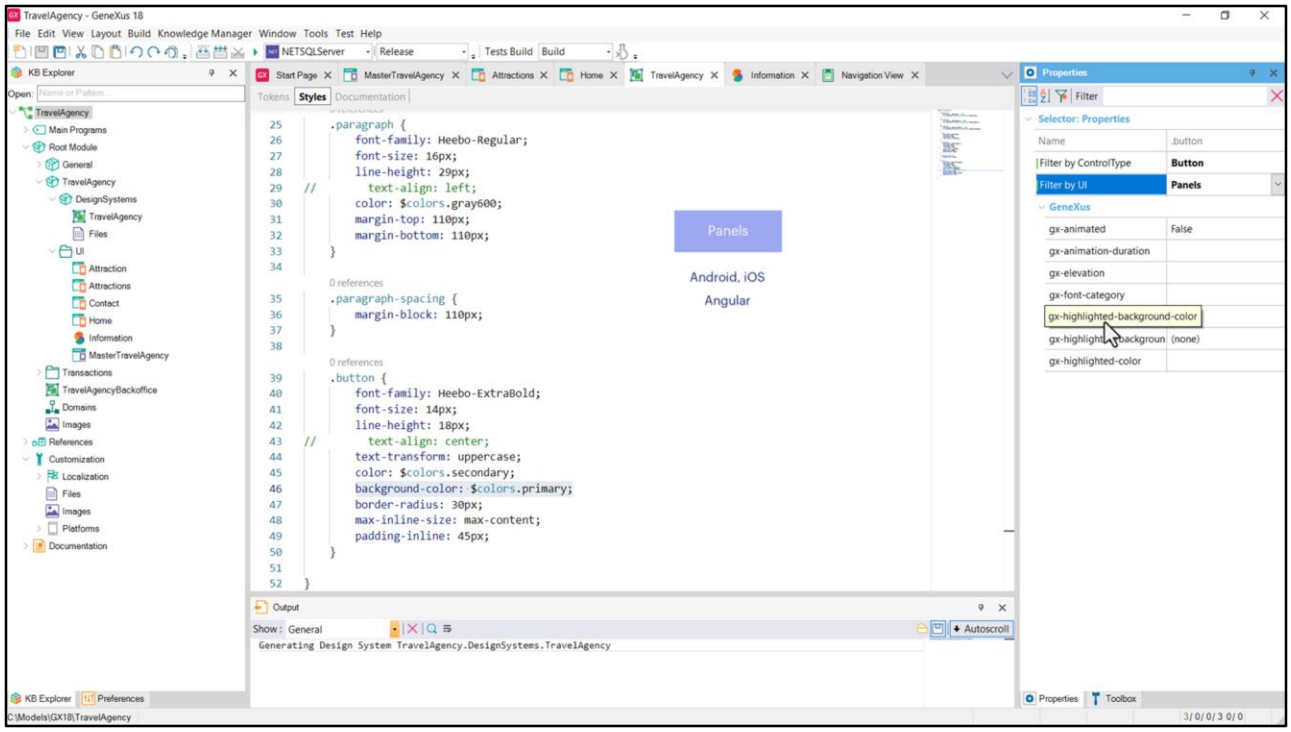
Así, por ejemplo, si venimos a la ventana de propiedades… bueno, acá podemos filtrar las propiedades que estamos utilizando a nivel de esta clase…



…pero además vemos que tenemos estos dos filtros para poder visualizar no todas las propiedades sino, por ejemplo, las que correspondan a controles de tipo botón. Al hacer esto me va a mostrar solamente las propiedades válidas para ese tipo de control.

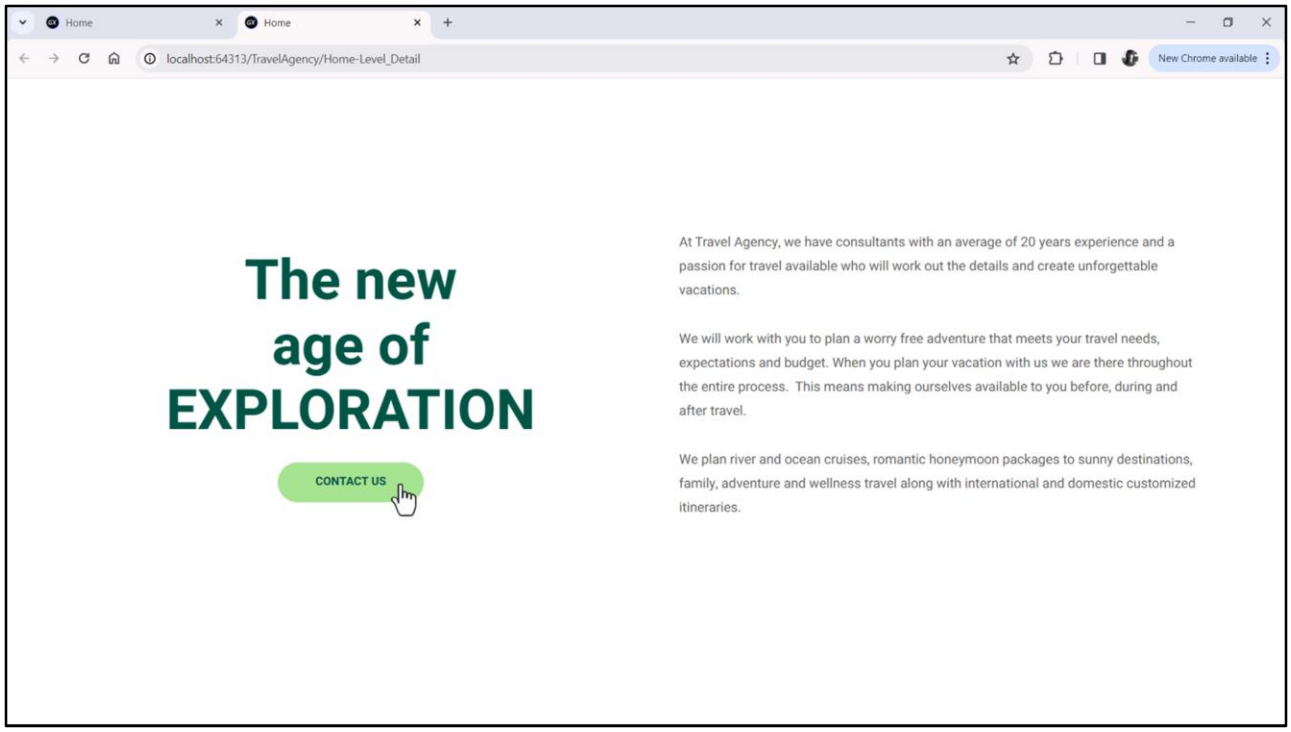
Y por otro lado, también puedo filtrar por el tipo de interfaz de usuario… es decir, o todas, que es lo que estoy filtrando ahora… o podría elegir el mundo Panels o el mundo Web Panels…

Si filtro por el mundo Panels que es en el que estamos ahora… (recordemos que el mundo Panels surgió cuando los dispositivos inteligentes, es decir, cuando las aplicaciones nativas para teléfonos y tablets; luego Angular se incorporó a este mundo... Pero Angular es un mundo intermedio, es un híbrido entre el mundo Panels y el mundo Web Panels y eso lo vamos a ver ahora en un ratito)…



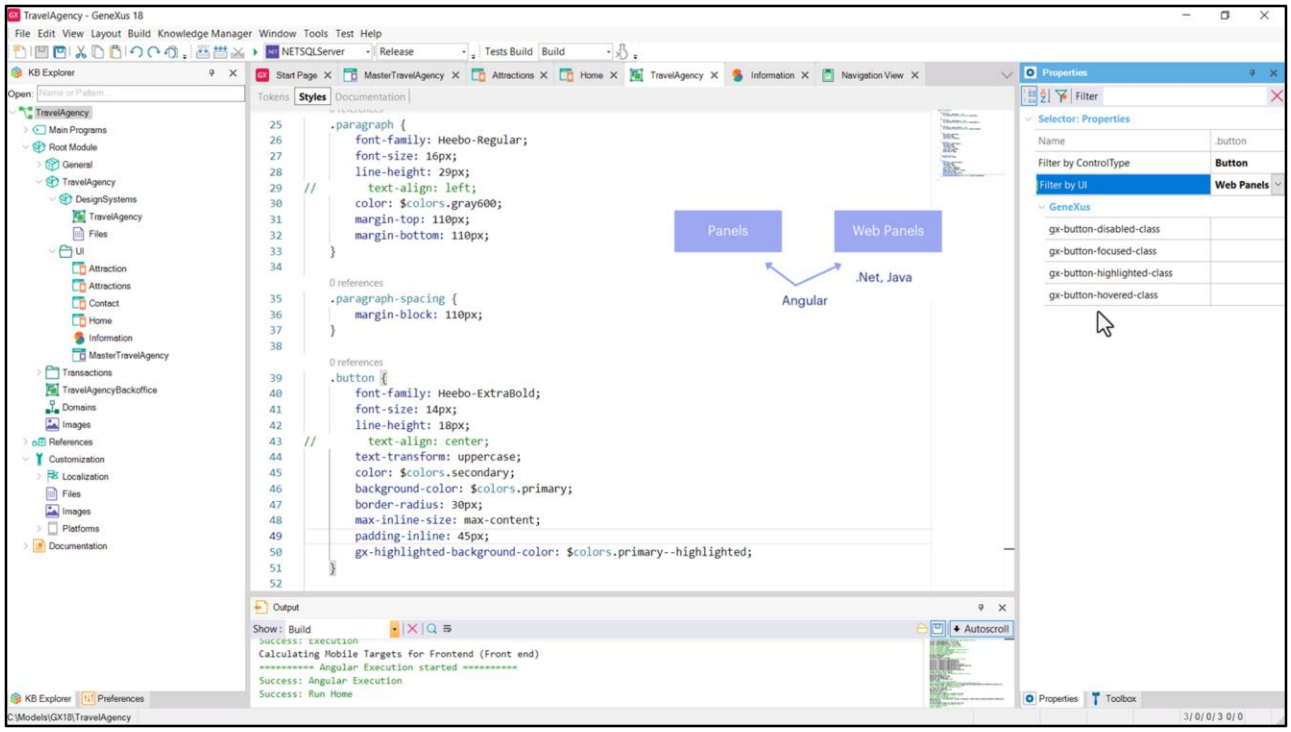
Pero bueno, ahora estoy filtrando por interfaz de usuario Panels. Y acá vemos que me están apareciendo estas propiedades GeneXus, (gx, guion y un nombre de propiedad)… y bueno, estas son las que podría eventualmente utilizar dentro de la clase, de una clase que le voy a asociar a un control de tipo botón en un panel… y bueno, son clases, que tienen cada una su comportamiento.

Vemos entre las clases esta: gx-highlighted-backgroud-color, que es casualmente una propiedad que nos sirve justamente para esto que estamos buscando… le voy a colocar como color el del token de color primary--highlighted. Vamos a probar…



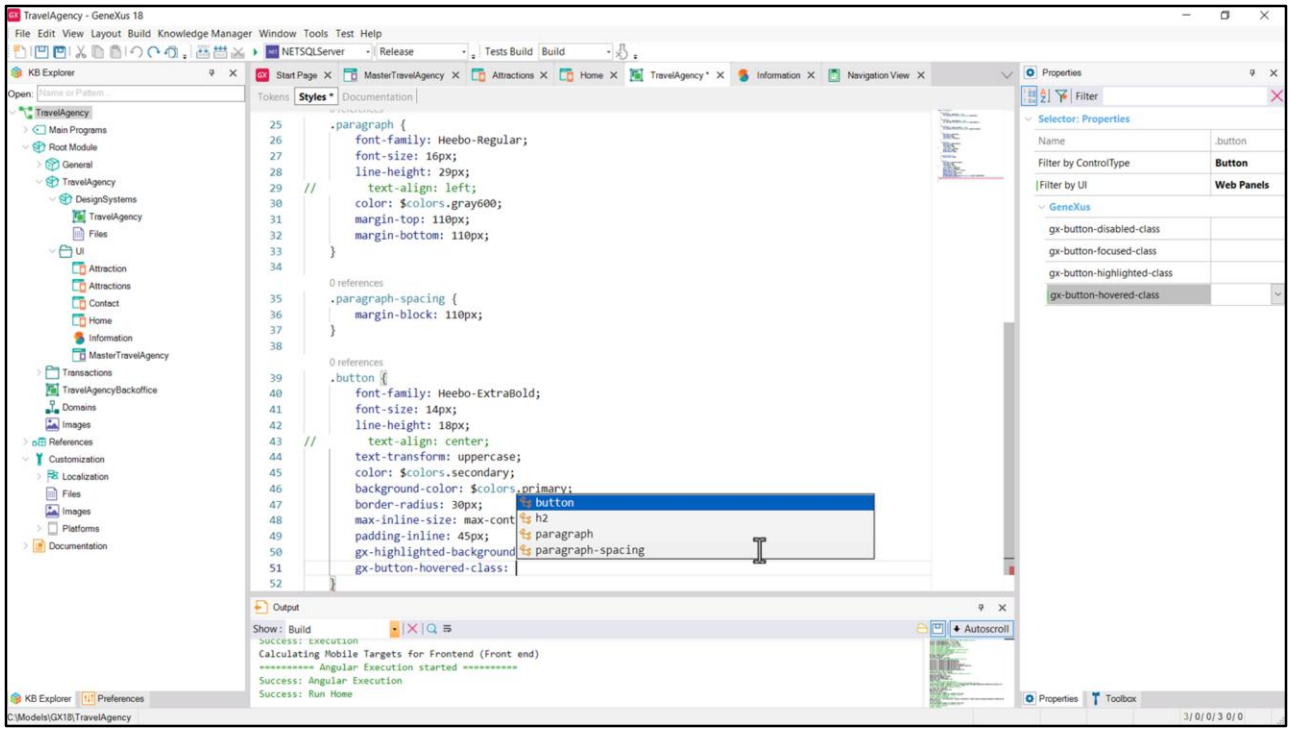
Y acá vemos que cuando presionamos aparece efectivamente el background color en otro color. Estoy presionando con el mouse, no lo solté porque cuando lo suelte va a llamar al panel Contact que no tiene nada programado por el momento, por eso vemos esto… por eso vemos esto.

¿Y si quisiéramos que esto sucediera al hacer hover y no solamente al activar el botón?

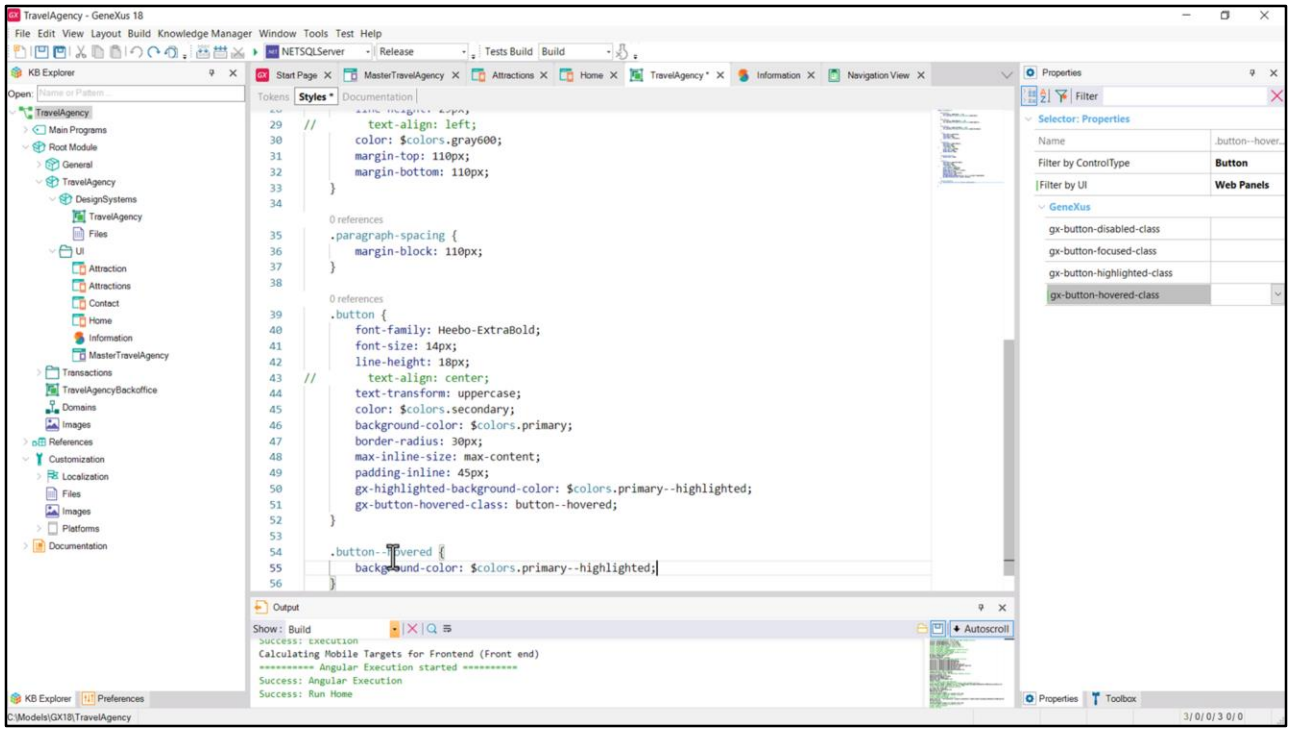


Si venimos a buscar una propiedad gx-hover-background-color, no la encontramos. Es decir las propiedades aplicables a un botón dentro de un panel son estas y no incluyen esa. Es que, insisto, estas propiedades surgieron con el mundo panels para aplicaciones nativas: Android, iOS.

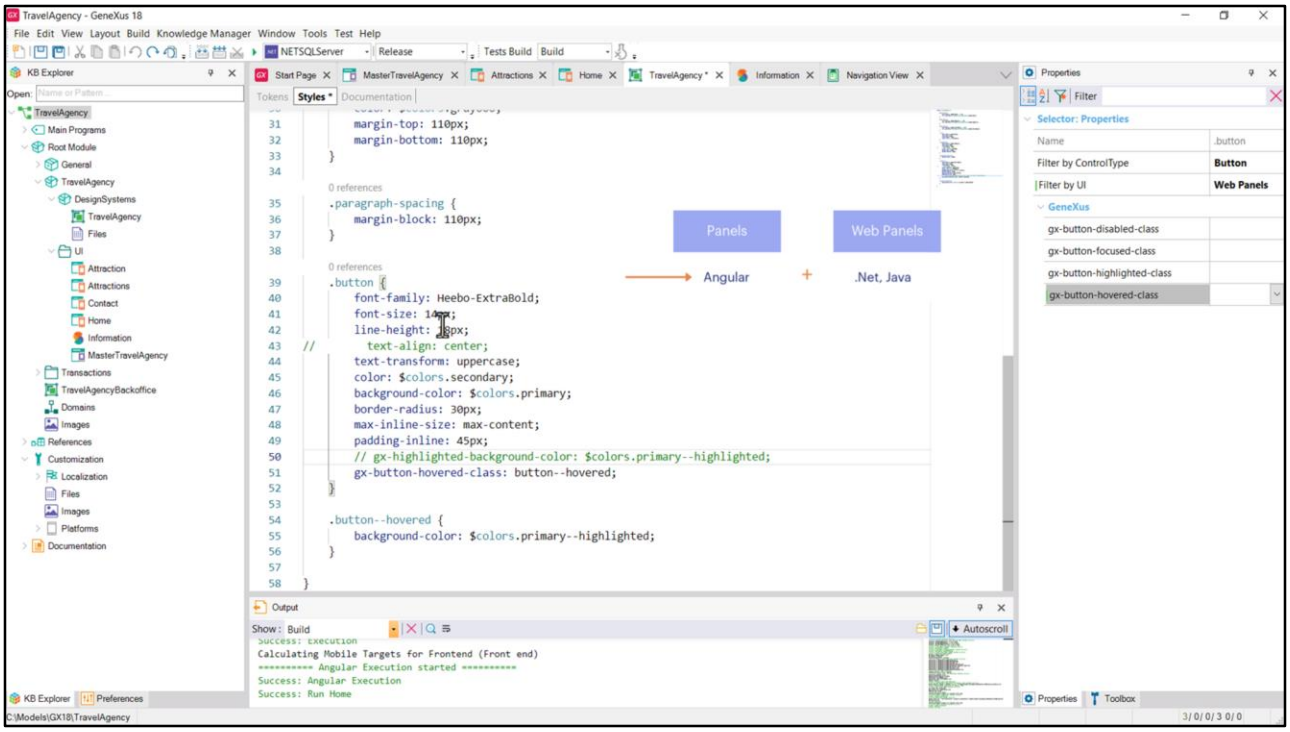
Sin embargo, observemos qué pasa cuando cambio el filtro por el de Web Panels. Vemos que aparecen estas otras propiedades asociadas a los botones, que sí incluyen esta hovered. Estas propiedades, lo que están indicando –vean que terminan con class- lo que le estamos indicando si elegimos esta propiedad es que…



… vamos a hacerlo… le estamos indicando cuál es la clase que va a comandar el estilo del botón que tiene esta clase asociada cuando se haga hover sobre él. Observemos que se nos están ofreciendo las 4 clases que este DSO tiene especificadas hasta el momento.



Vamos a crear una nueva clase a la que vamos a llamar button--hovered, y en ella… la vamos a especificar es donde vamos a especificar que el background-color sea el primary--hightlighed.

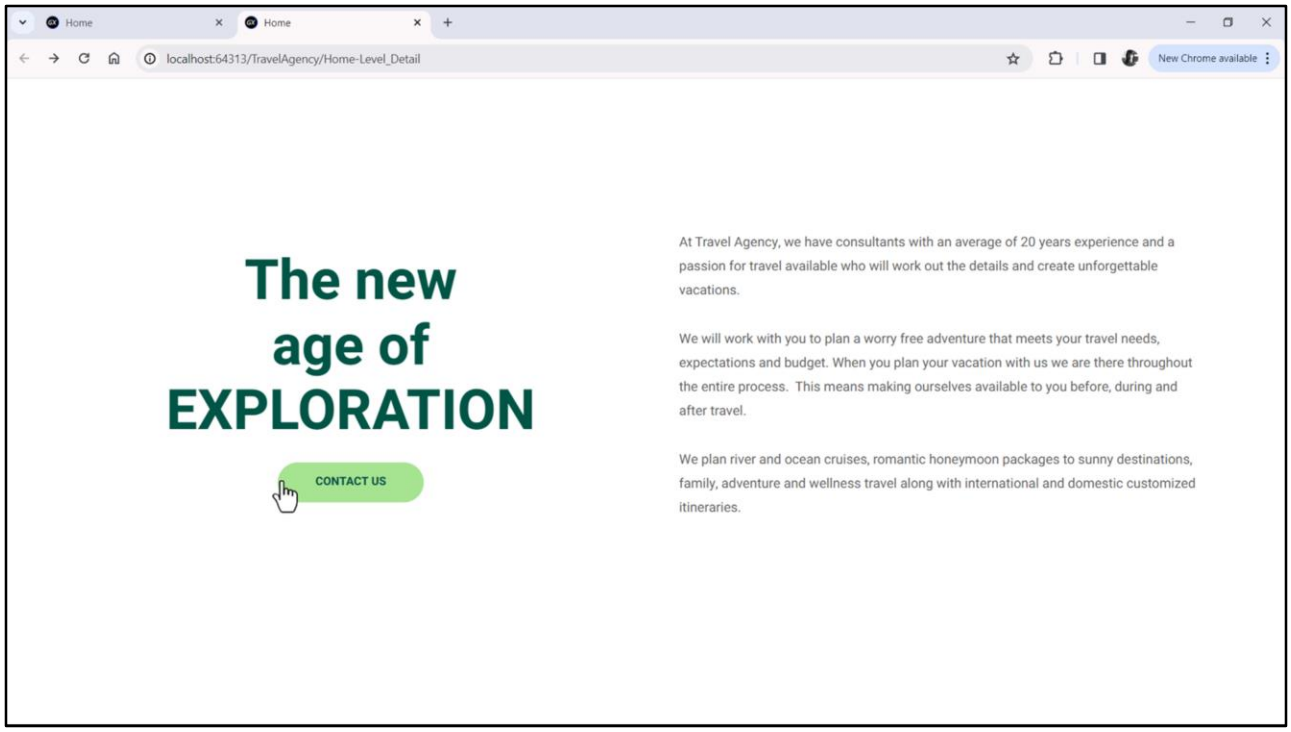


Voy a dejar primero comentada esta para que se aprecie bien la diferencia. Estoy utilizando una propiedad GeneXus que podríamos a primera vista pensar que no va a tener efecto sobre nuestro panel porque estamos en el mundo de los Panels y no de los Web Panels. Sin embargo como se trata de propiedades que tienen que ver con el mundo web sí van a ser tomadas en cuenta.

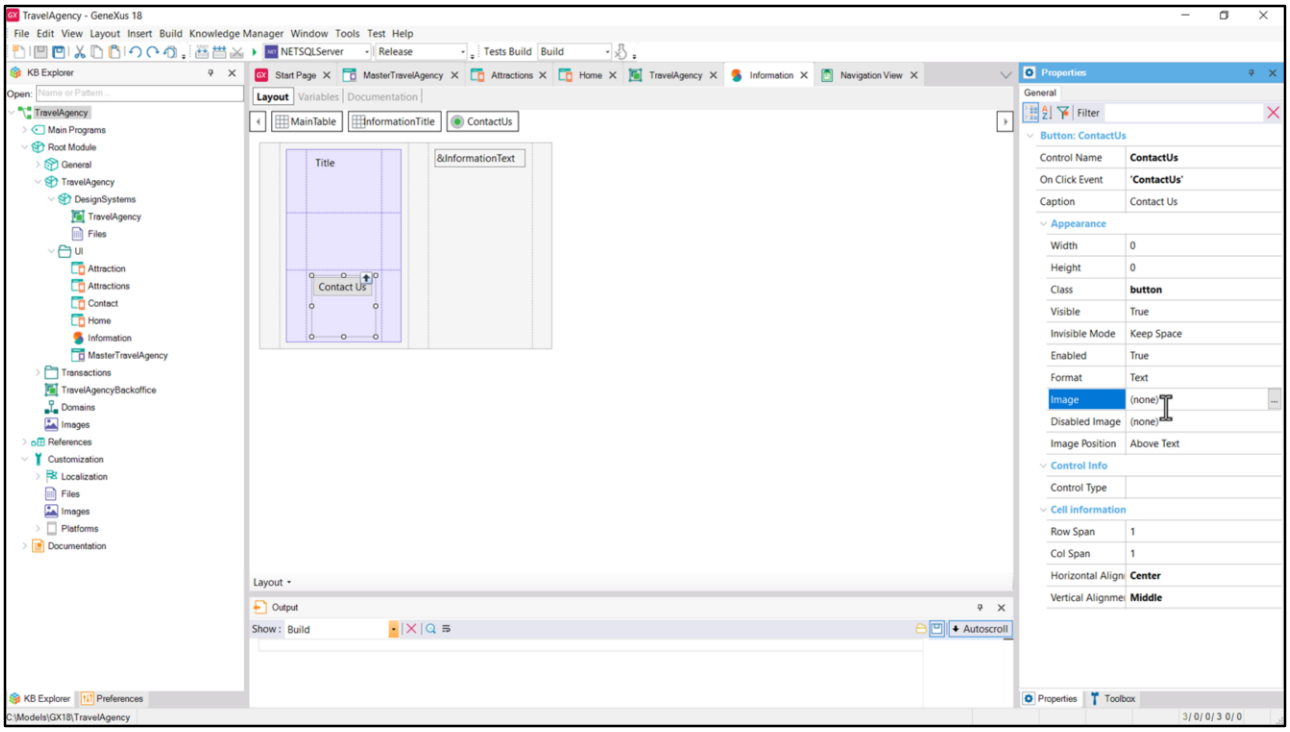
Es decir, voy a poder utilizar para un panel ejecutado para Angular, no para este panel ejecutado para Android o para iOS, para Angular voy a poder utilizar estas propiedades también.

Entonces lo que estoy diciendo aquí, voy a resumir es: cuando al botón que tiene esta clase asociada se le haga hover, entonces, a estas propiedades que están aquí correspondientes a la clase del botón se le van a agregar o sobrescribir las que estén dentro de esta otra clase, que la tenemos aquí seleccionada. Y lo que estamos indicando dentro de la clase es que sobrescriba esta propiedad, la background-color. Podríamos agregar nuevas propiedades que no sobrescriban a estas, sino que simplemente agreguen nuevos comportamientos de estilo y bueno, ahí se agregarían. En este caso como se llama igual que una que ya está definida aquí, bueno, se va a sobrescribir.

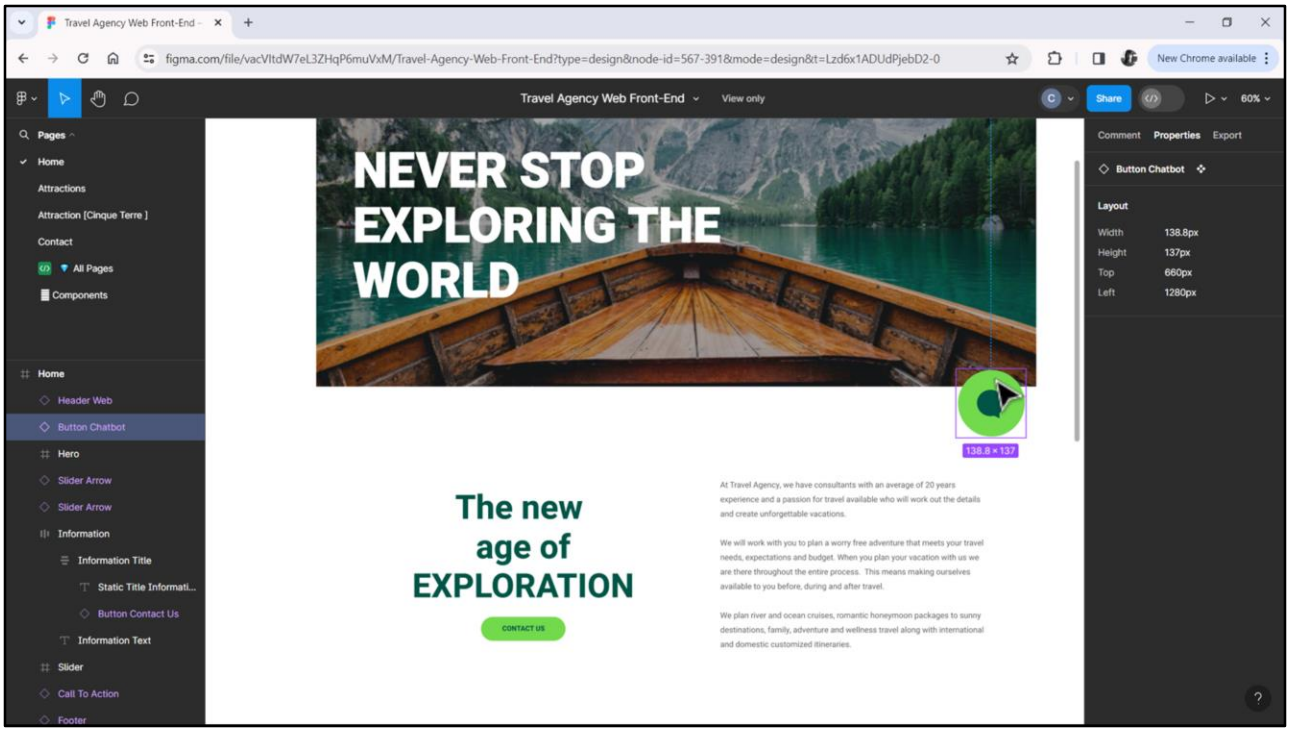
Vamos a probar.



Bien, ahora veamos qué pasa cuando paso el mouse sobre el botón.

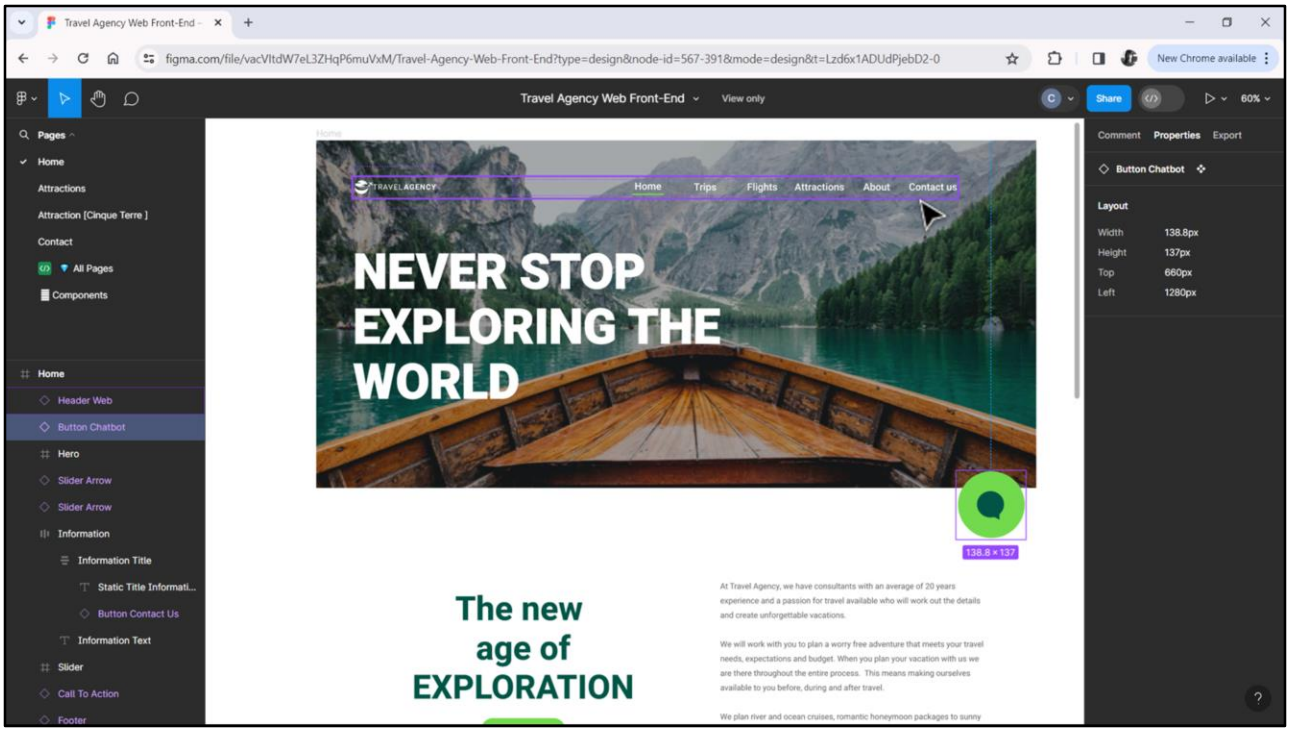


Bueno para terminar comentemos que un botón puede tener un texto, una imagen, o ambos.



Un caso en el que vamos a tener un botón con imagen y sin texto va a ser el del chatbot, que lo vamos a implementar cuando implementemos el Master Panel.

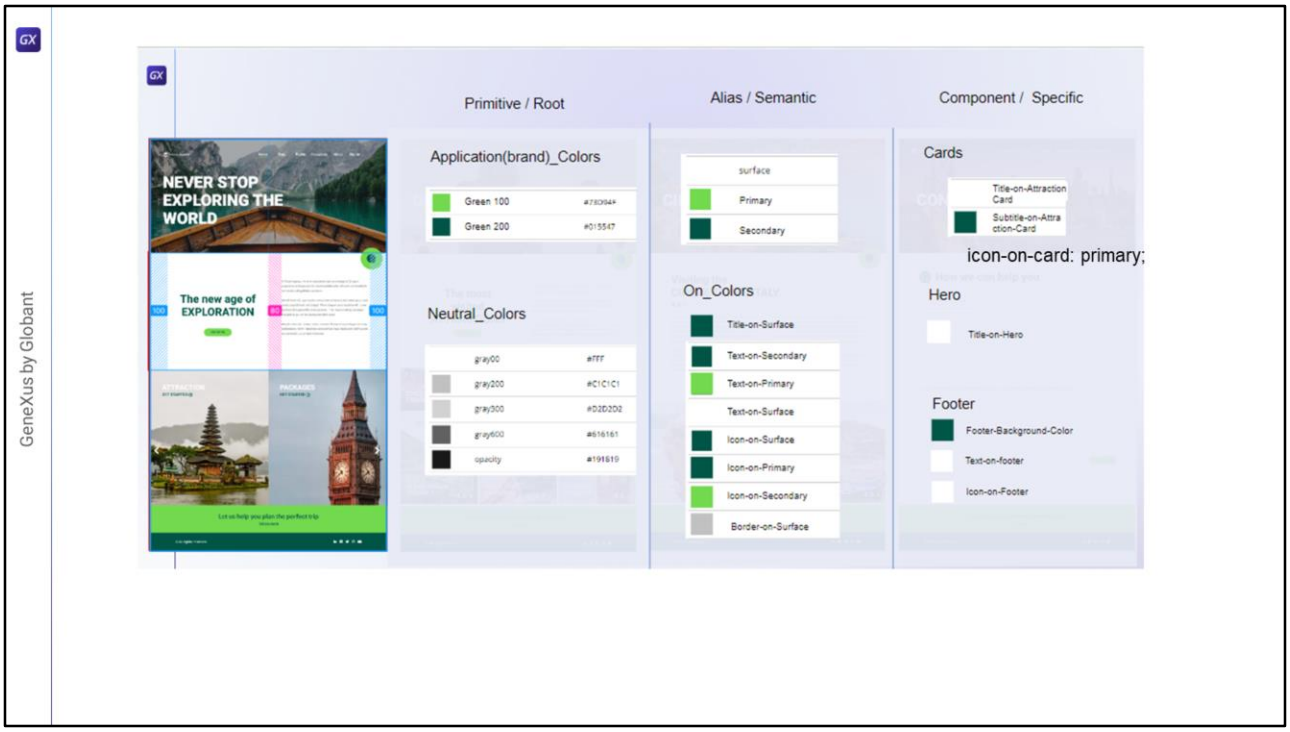
Pero acá vemos esto que parece una imagen, en realidad va a ser un botón con una imagen. ¿Por qué va a ser un botón, ya lo dejo adelantado? Porque todo elemento cliqueable, es decir que va a responder a una acción, debe ser implementado como un botón.



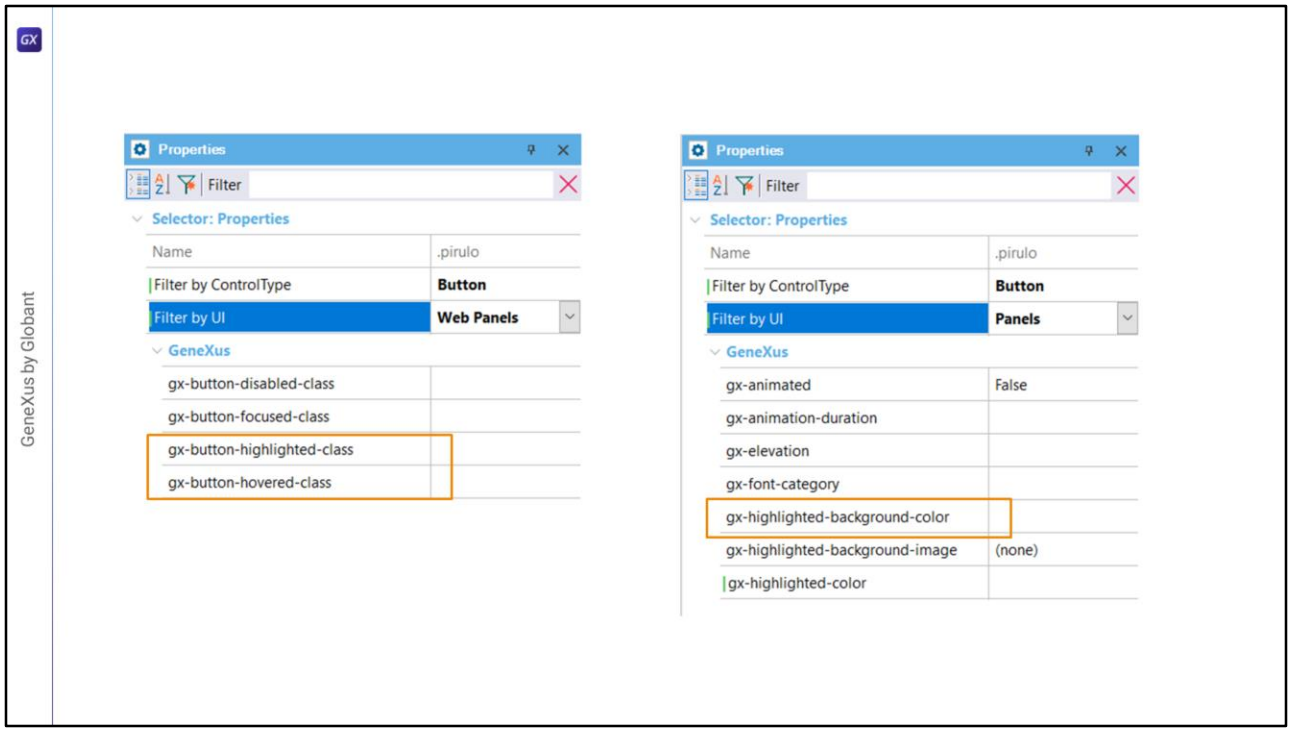
Así, por ejemplo, los elementos del menú, que se tratarán de botones que tendrán texto y no tendrán background, es decir, el background será transparente. ¿Y esto por qué? Por cuestiones de accesibilidad como veremos oportunamente.

Bueno, los espero ahora en el siguiente video.


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



Ahora sí, veamos cómo hacer que el botón luzca como queremos.



Ahora sí, veamos cómo hacer que el botón luzca como queremos.