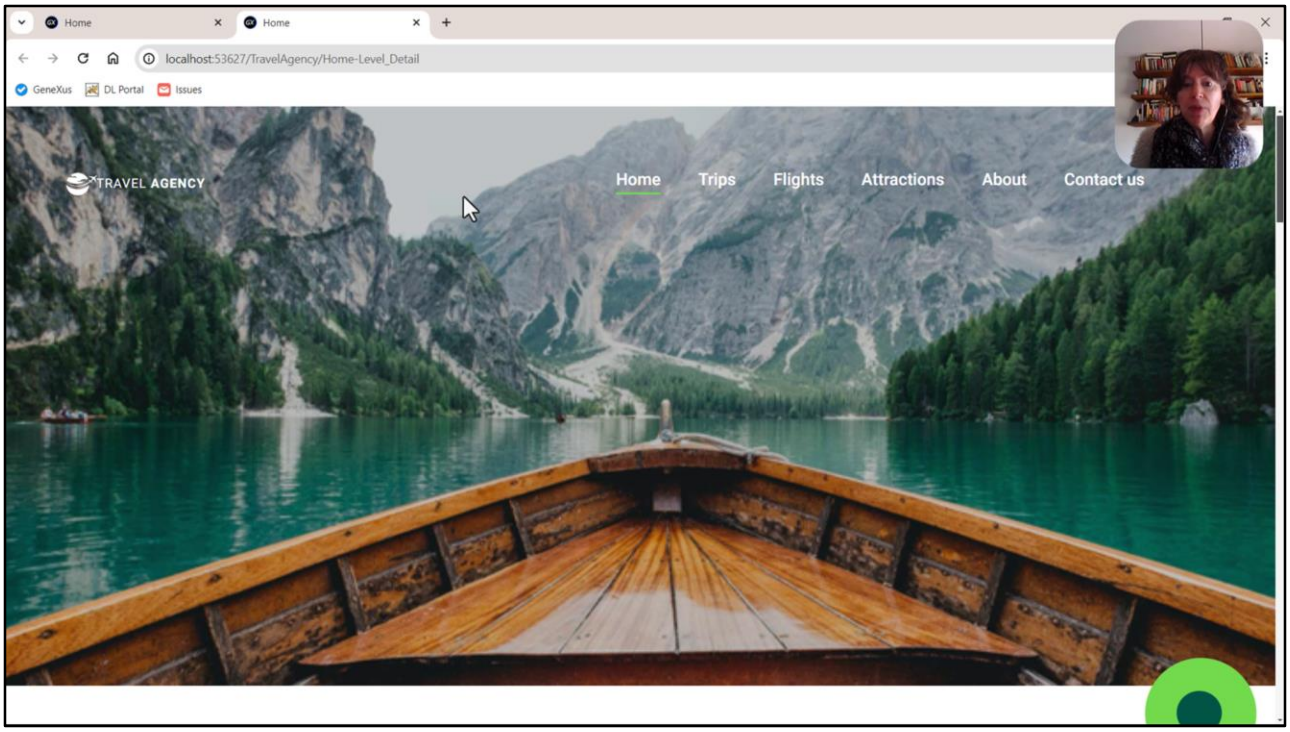
GeneXus by Globant

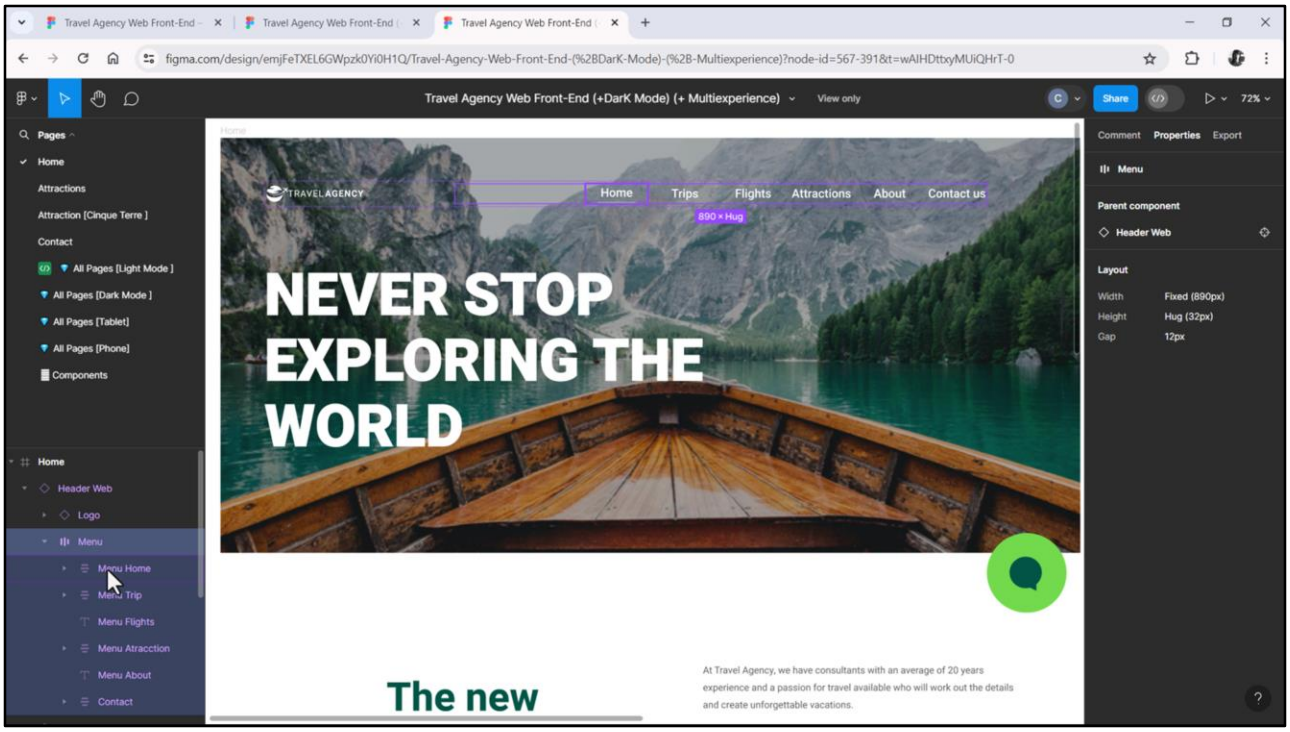
Header in GeneXus: logo, menu and title

Cecilia Fernández



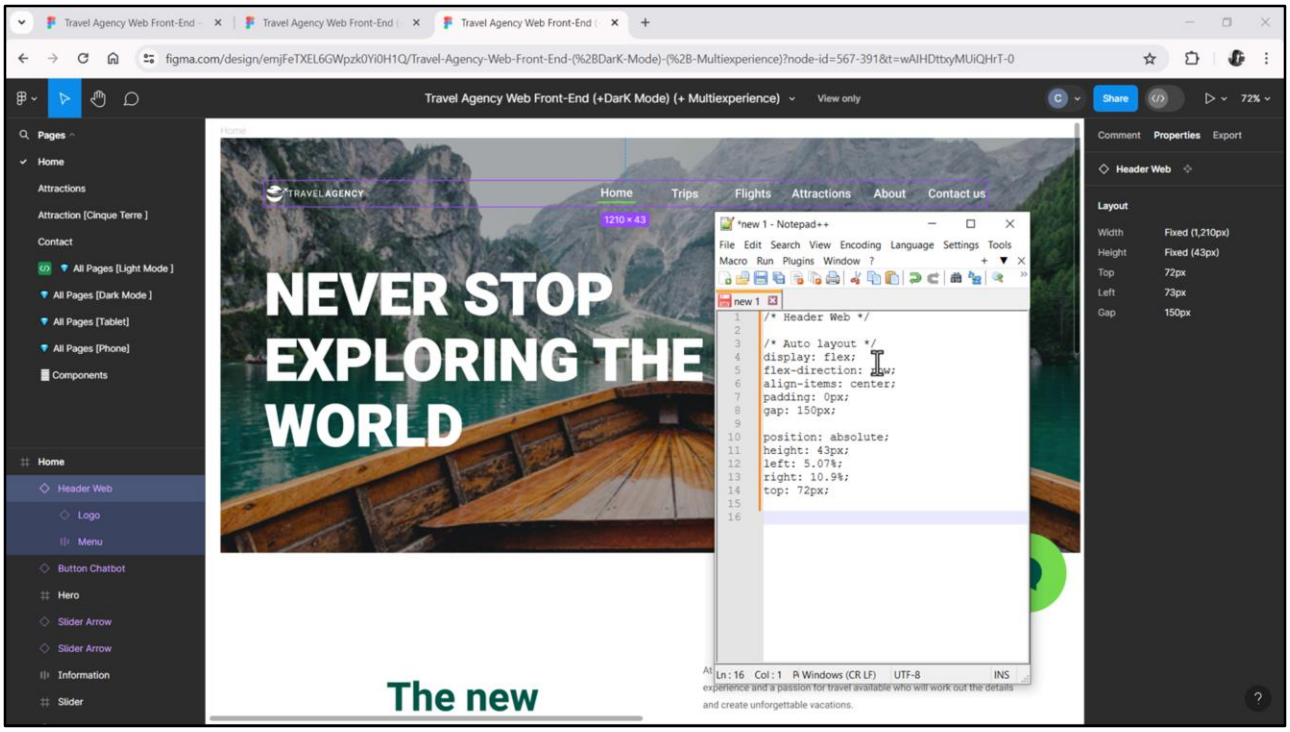
Bueno, en el video anterior habíamos obtenido esto, es decir, habíamos logrado implementar la imagen del Header. Lo que vamos a hacer ahora, en este video, es implementar el logo y el menú. Y vamos a dejar para otro video la implementación del cambio de pantalla en las navegaciones, es decir, que cuando yo por ejemplo presione sobre esta opción Attractions, no solamente me cargue la página de atracciones acá, que va a ser en el Contentplaceholder, sino también que cambie todo lo que necesita, es decir, la imagen Hero y también el texto que se superpone.

Si en este video nos da el tiempo, vamos a implementar también el texto. Si no, lo dejamos para el siguiente.



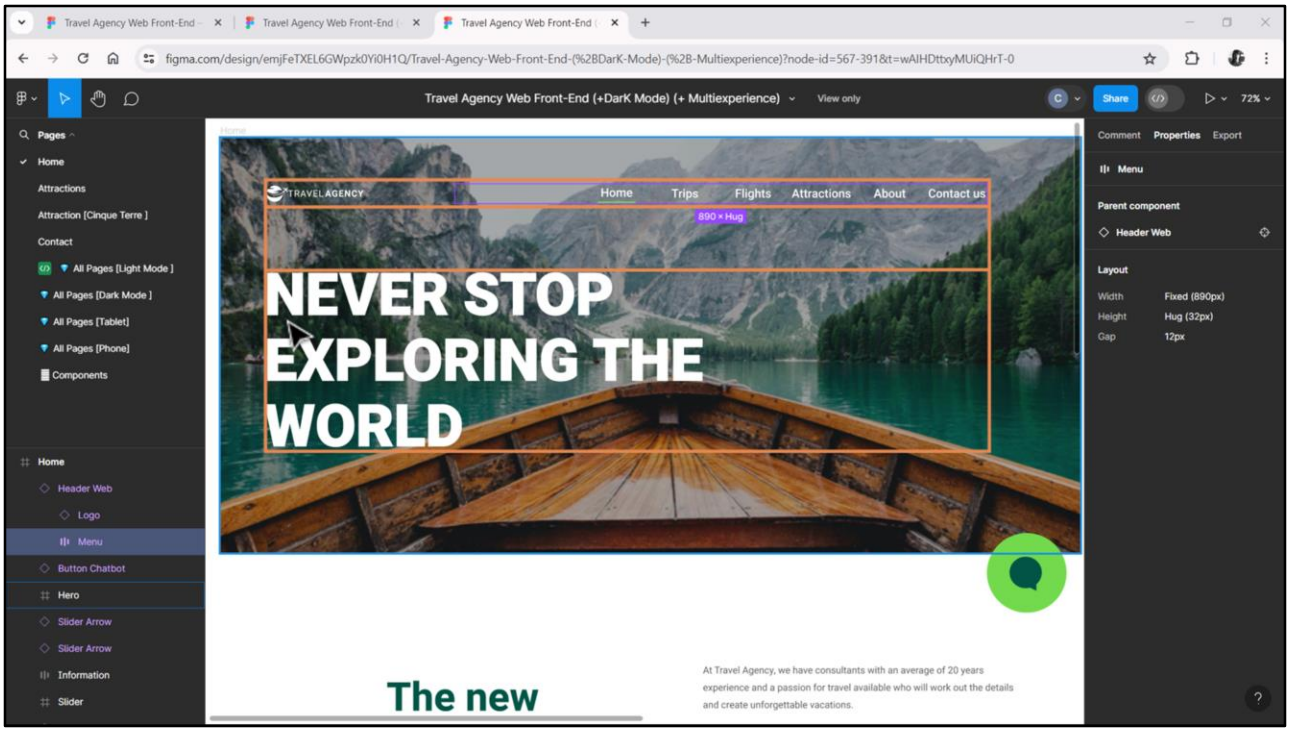
En Figma vemos que están implementados como un componente que contiene:

- Otro componente con el logo: que a su vez se compone de un ícono y dos textos.
- Y un contenedor con autolayout, es decir, Flex, para el menú, es decir, una sucesión de ítems.



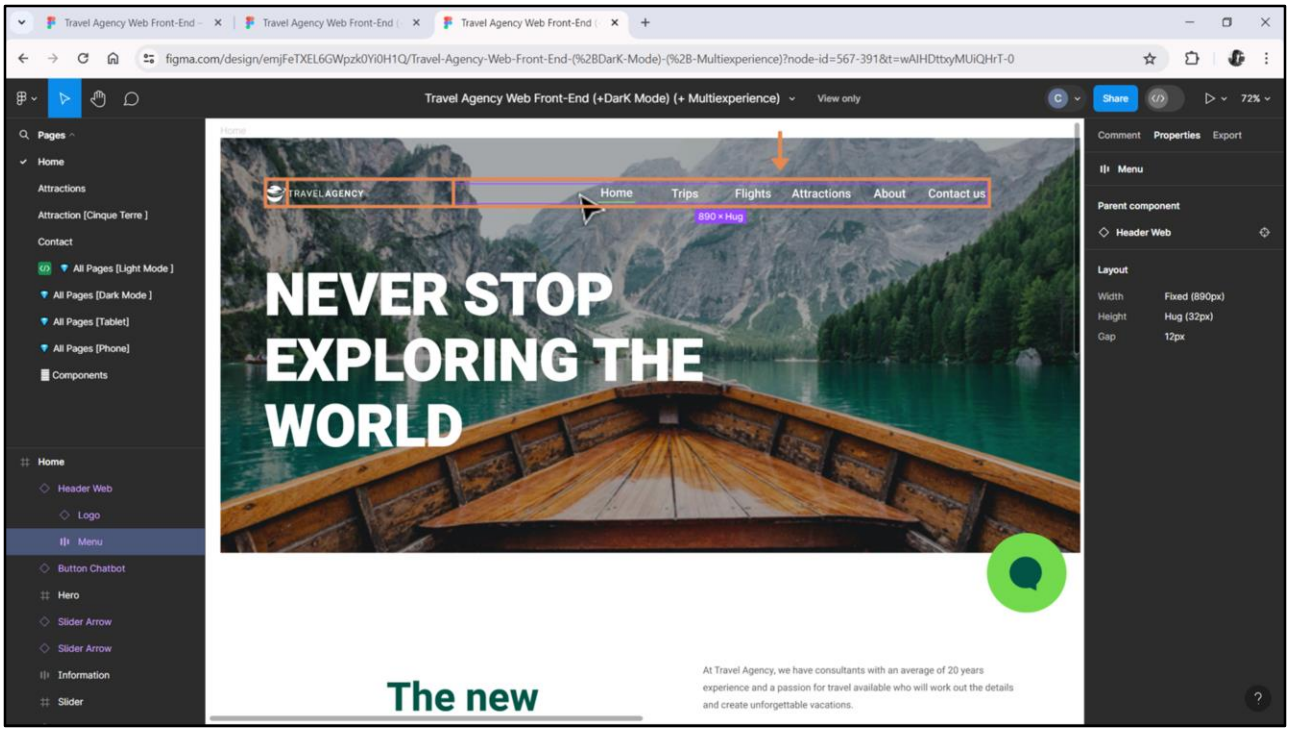
También podemos ver que Chechu agrupó ambos componentes en este “Header Web” que también en un contenedor con autolayout (lo vemos claramente en este gap que aparece aquí). Y si extraemos las propiedades CSS lo terminamos de verificar.

Y aquí vemos, por ejemplo, gráficamente, ese gap entre los dos elementos: el menú y el logo.

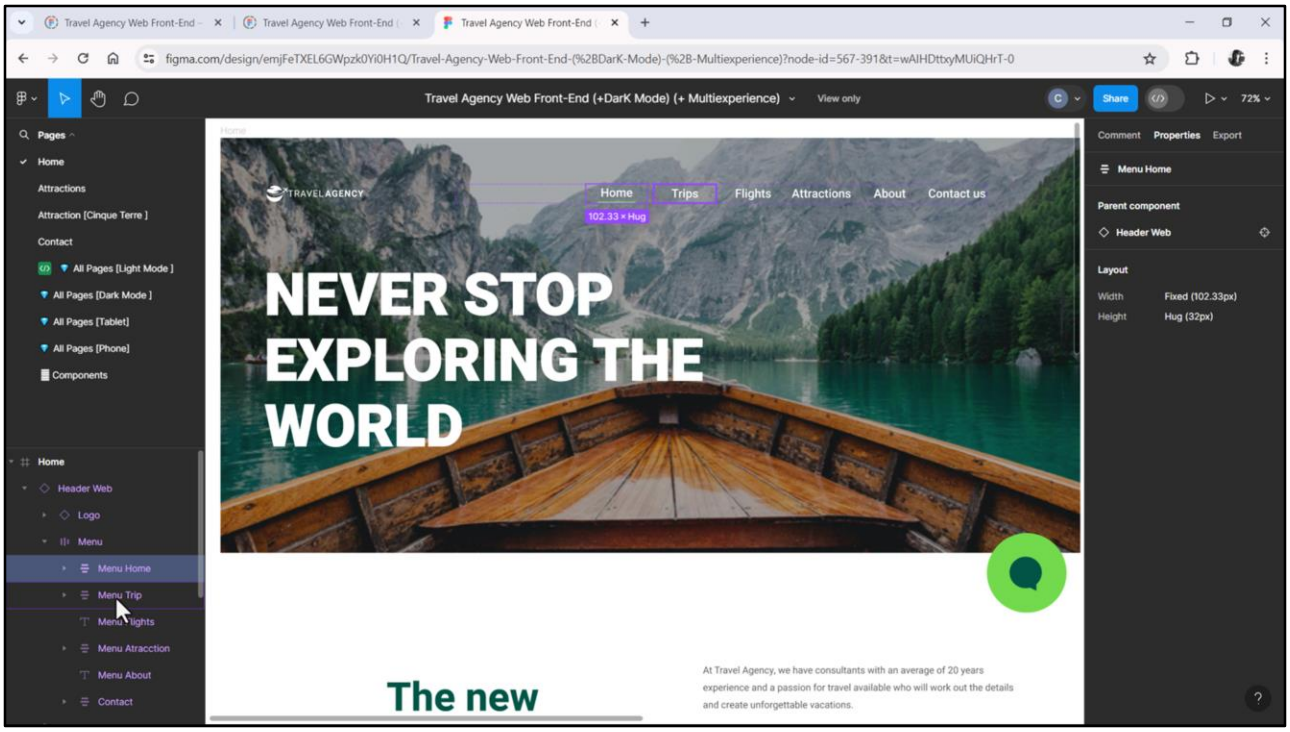


Pero nosotros no vamos a implementarlo como un flex, sino como una tabla, porque, como podemos ver, el logo está alineado por la izquierda con este título. Por lo que será más fácil utilizar una tabla para alinearlos.

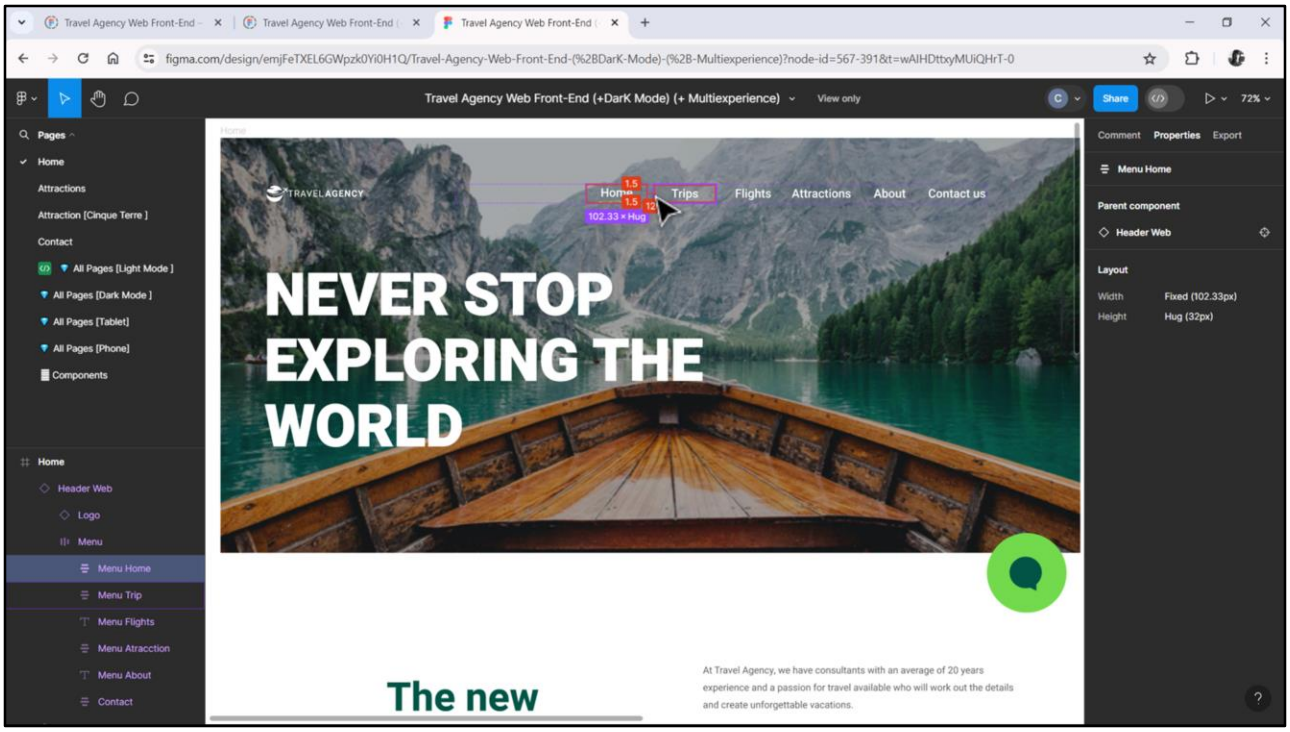
Tabla que tendrá 3 filas, la del medio para espaciado. Igual por el momento hagámosla de una sola fila, pensando sólo en logo y menú. Después le agregaremos las otras dos.



Siempre podemos imaginar muchas formas de implementar un layout. Voy a elegir colocar el ícono en una columna, las palabras “Travel Agency” del logo en otra columna, y una tercera columna para el menú propiamente dicho, que elegiré implementarlo igual que como Chechu lo hizo, con un flex.

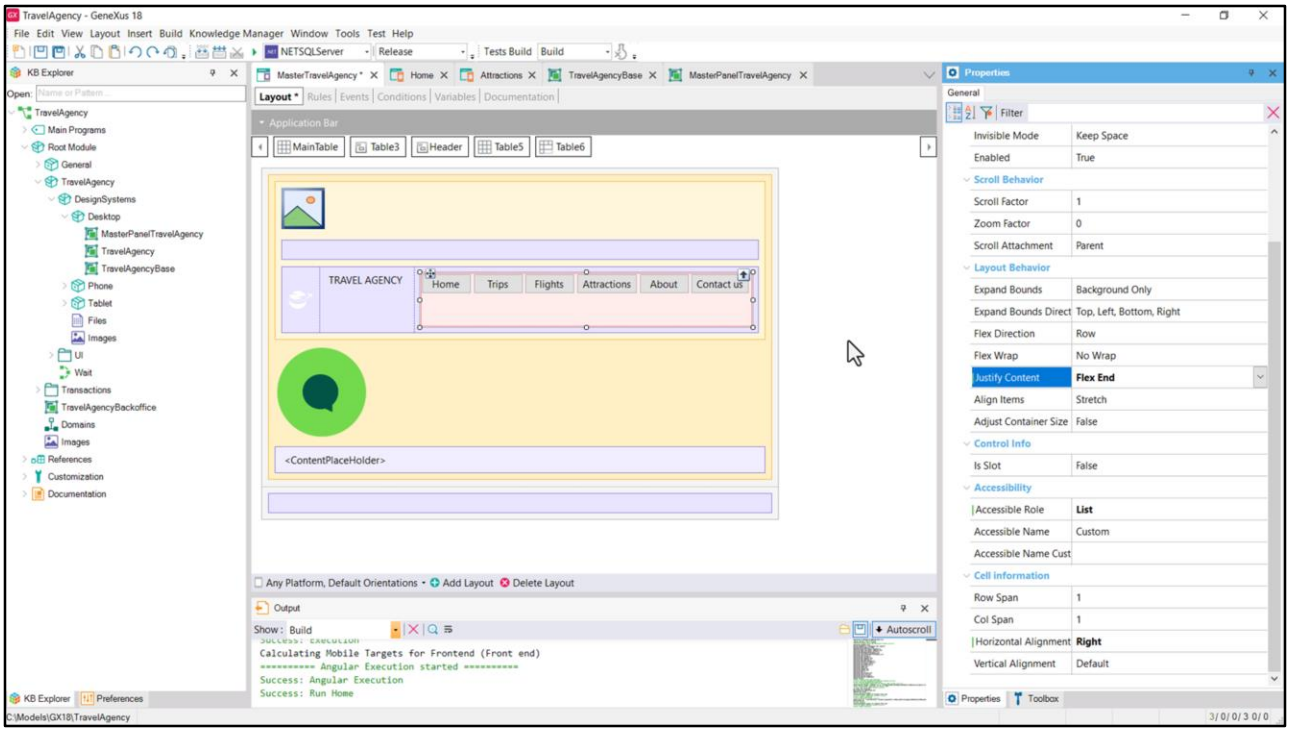


Será una sucesión en dirección horizontal de botones: uno para cada acción. Serán botones y no textblocks por todo lo que dijimos cuando hablamos de accesibilidad. Prefiero un flex en este caso antes que una tabla, porque, justamente, me da más flexibilidad: cada botón ocupará el espacio que necesite, de acuerdo a la cantidad de letras de su texto (nos importa sobre todo si vamos a querer la aplicación para varios idiomas). Y los puedo además equidistribuir a través de la propiedad **gap** aplicada a la clase del flex.



Acá Chechu no se dio cuenta (y lo digo con convicción porque acabo de preguntarle) de que lo que hizo fue fijar los anchos de los espacios de cada texto y eso fue lo que equidistribuyó, a 12 píxeles uno de otro, pero de este modo, fíjense que el espacio entre el texto Home y el Trips le quedó mucho más grande que entre Attractions y About, por ejemplo. Y esto es un error.

Por otro lado si me interesara que en caso de que no dé el ancho de pantalla para contener todos los textos, en lugar de una barra de scroll horizontal se haga wrap y algunos de los textos, de las opciones, digamos, del menú, aparezcan en una segunda línea, entonces eso no lo consigo con una tabla y sí con el flex.

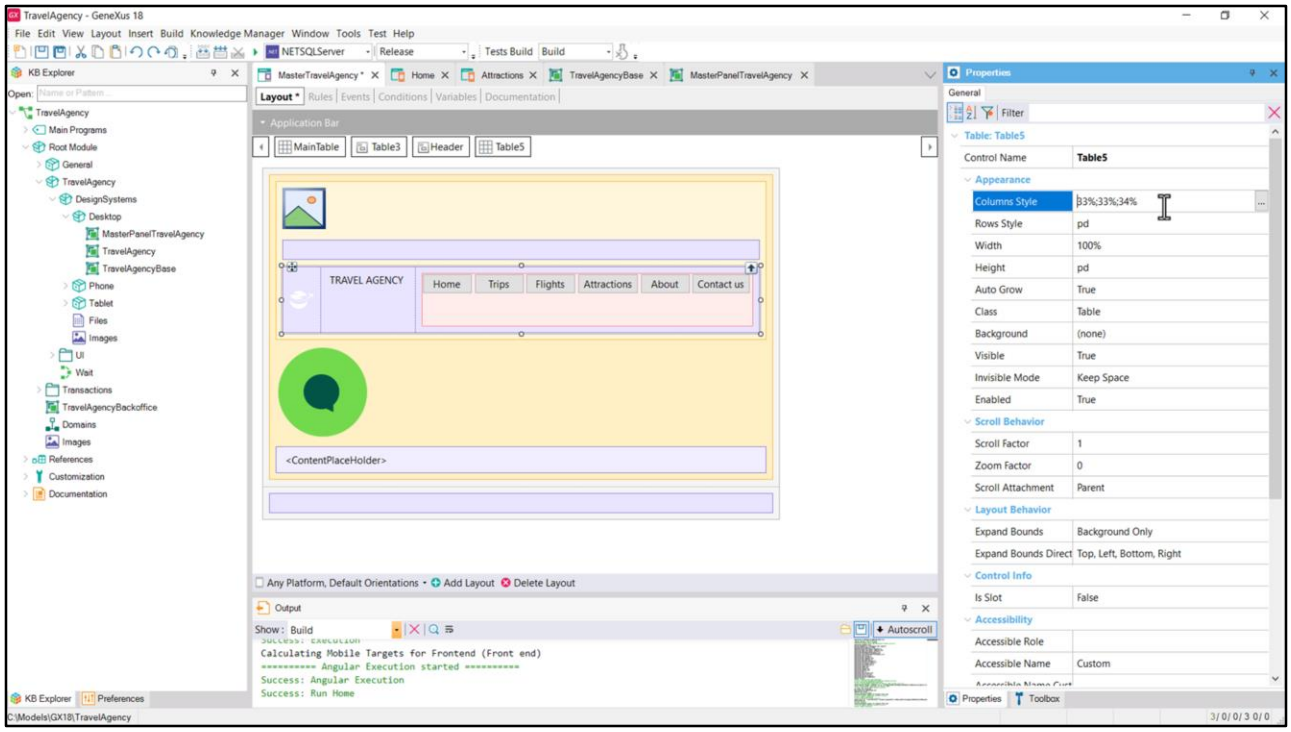


Empecemos entonces por insertar una tabla con 3 columnas.

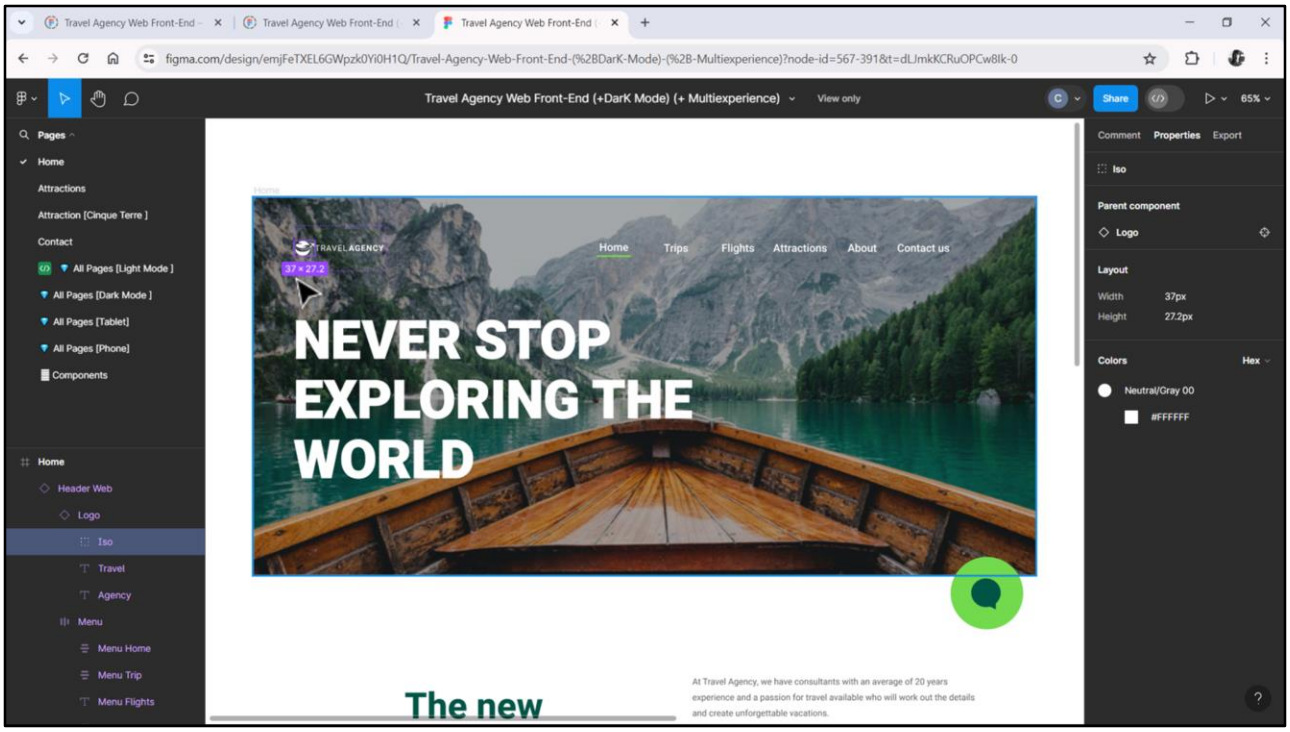
En la primera colocaremos el ícono de Travel Agency. En la segunda colocaremos un textblock, de caption (por ahora) TRAVEL AGENCY. Y en la tercera colocaremos un contenedor Flex.

En ese flex ya insertemos los 6 botones: uno para cada acción. A este contenedor flex vamos a colocarle el Accessible Role: List, para indicar justamente que será una lista de ítems. Podemos ya aprovechar y darle alineación horizontal por la derecha.

La dirección del Flex vemos que es la correcta, es la de fila. Para Wrap por ahora dejemos la opción default que es que no haga wrap. La justificación del contenido cambiémosla por Flex End, para que todos los botones se justifiquen respecto al final del flex y no al inicio, así se ubican dejando el espacio libre de adelante y no de atrás.
La alineación de los ítems respecto al otro eje, al y, la vamos a querer centrada, pero por ahora dejaré la default, lo veremos luego.

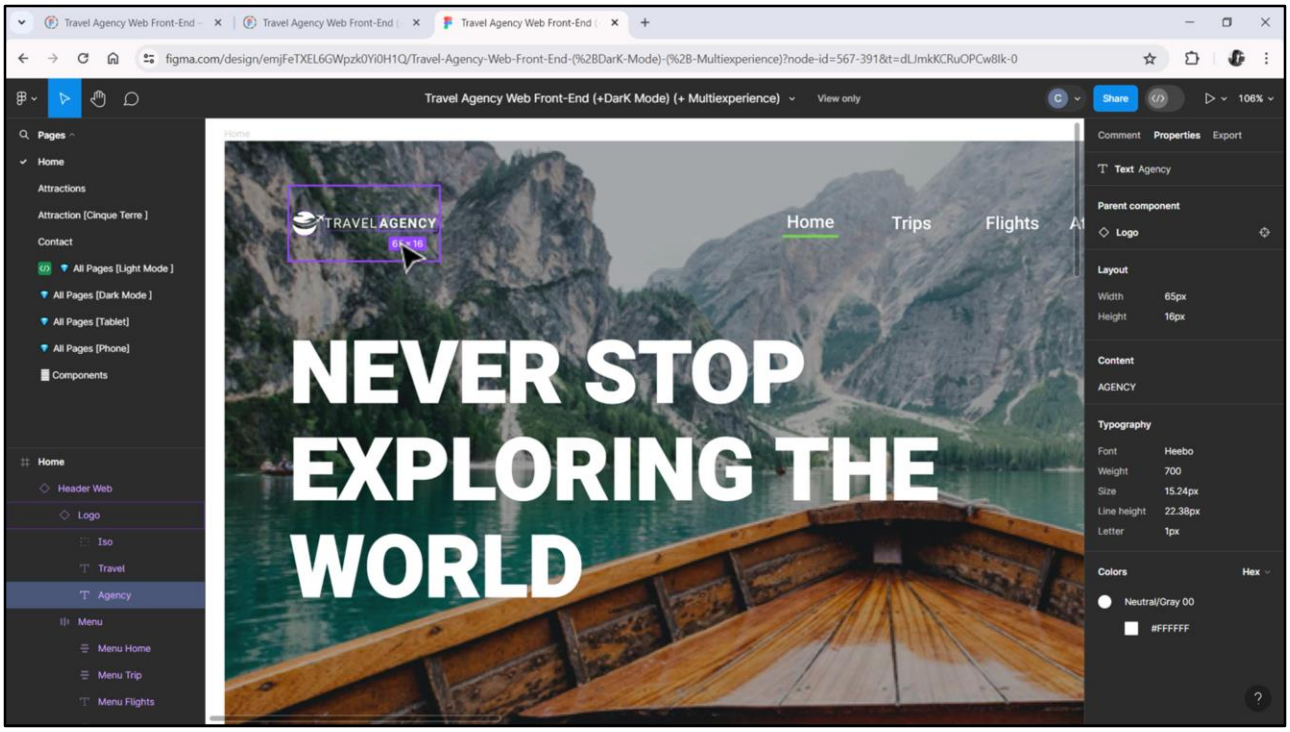


Ahora establezcamos las dimensiones de las tabla: las columns style, rows style, Height.

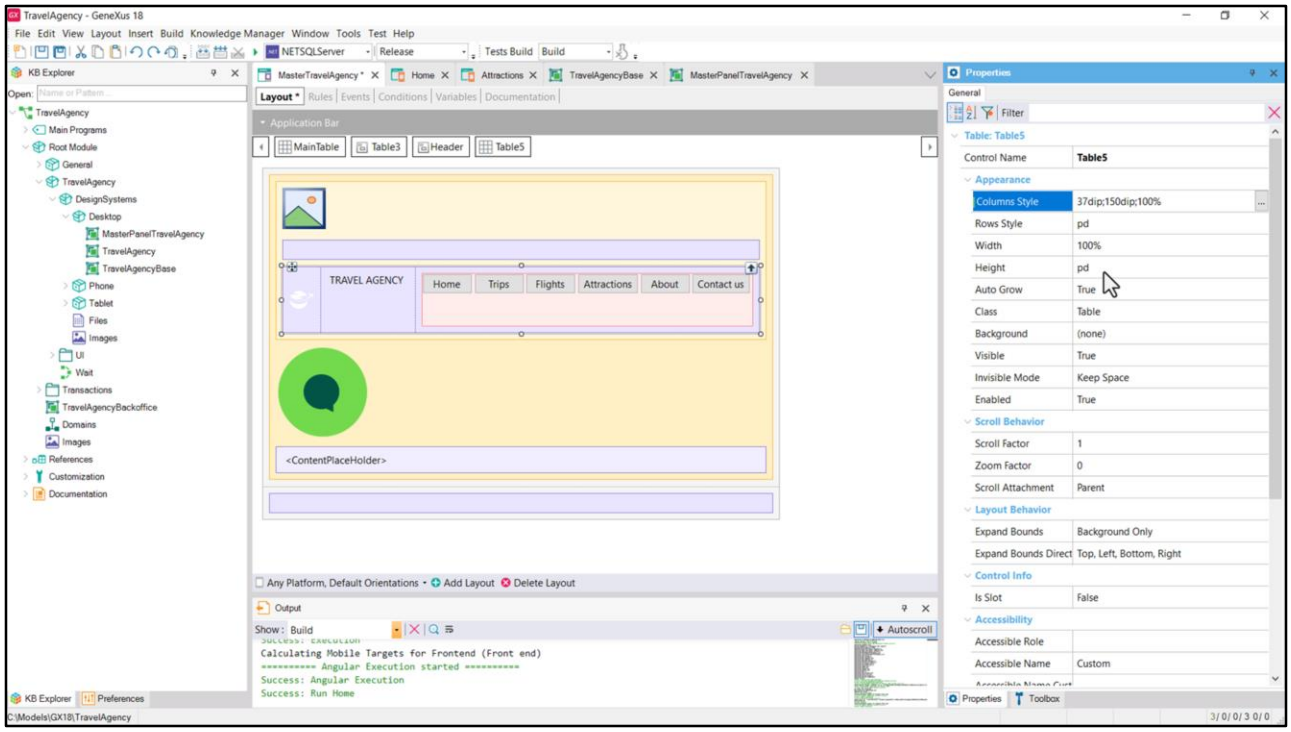


Podríamos colocar de alto de la tabla estos 43 dips.

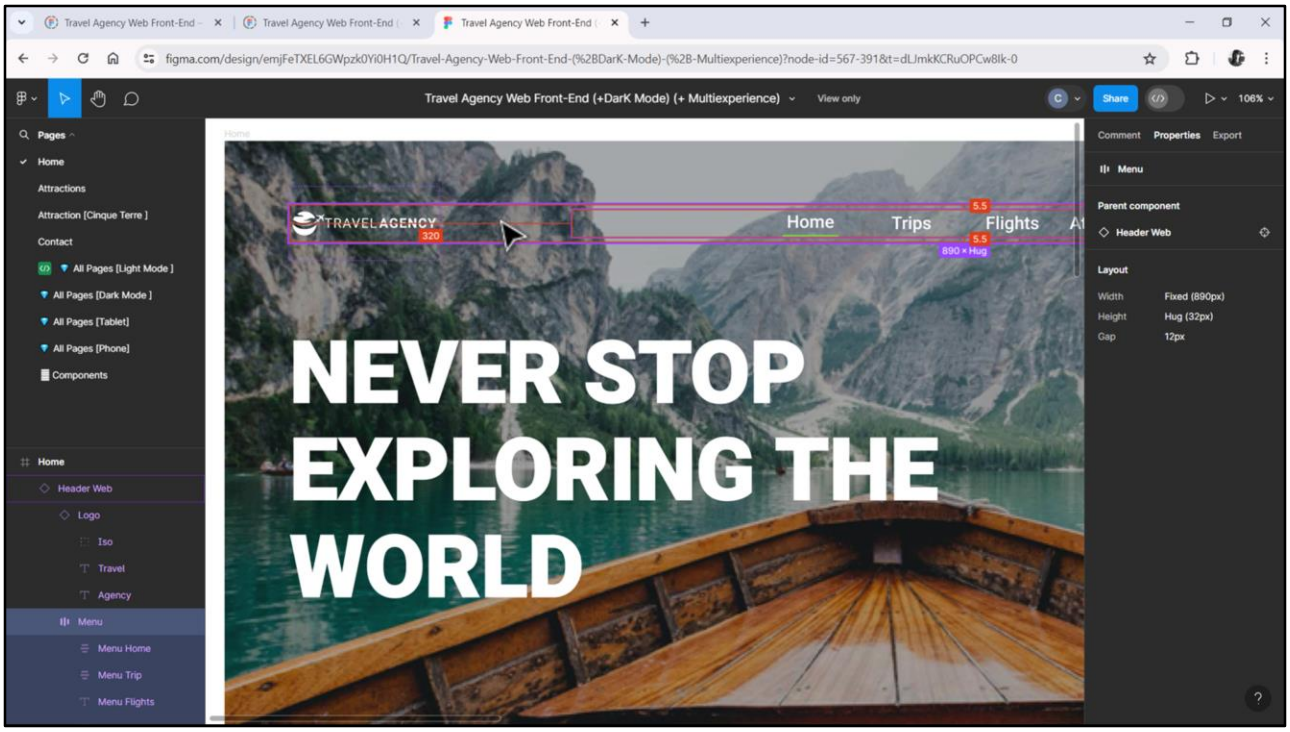
Y luego… el ícono tendrá 37 dips de ancho, así que le daremos ese ancho a la columna 1.



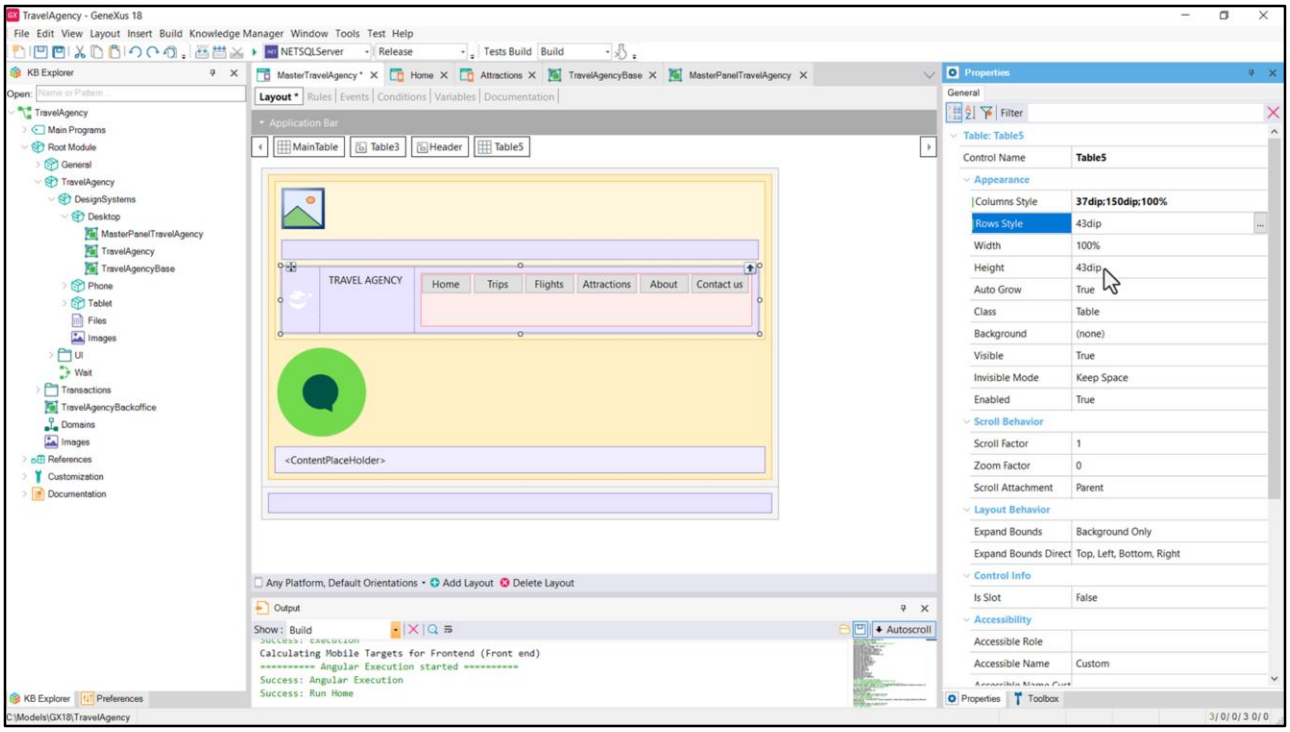
Luego, pegado, sin espacios, viene el texto Travel Agency, que suma 62 + 65, 127. Así que podríamos dar a la columna 2 un ancho de 150 dips, y a la 3 el 100% del ancho restante…



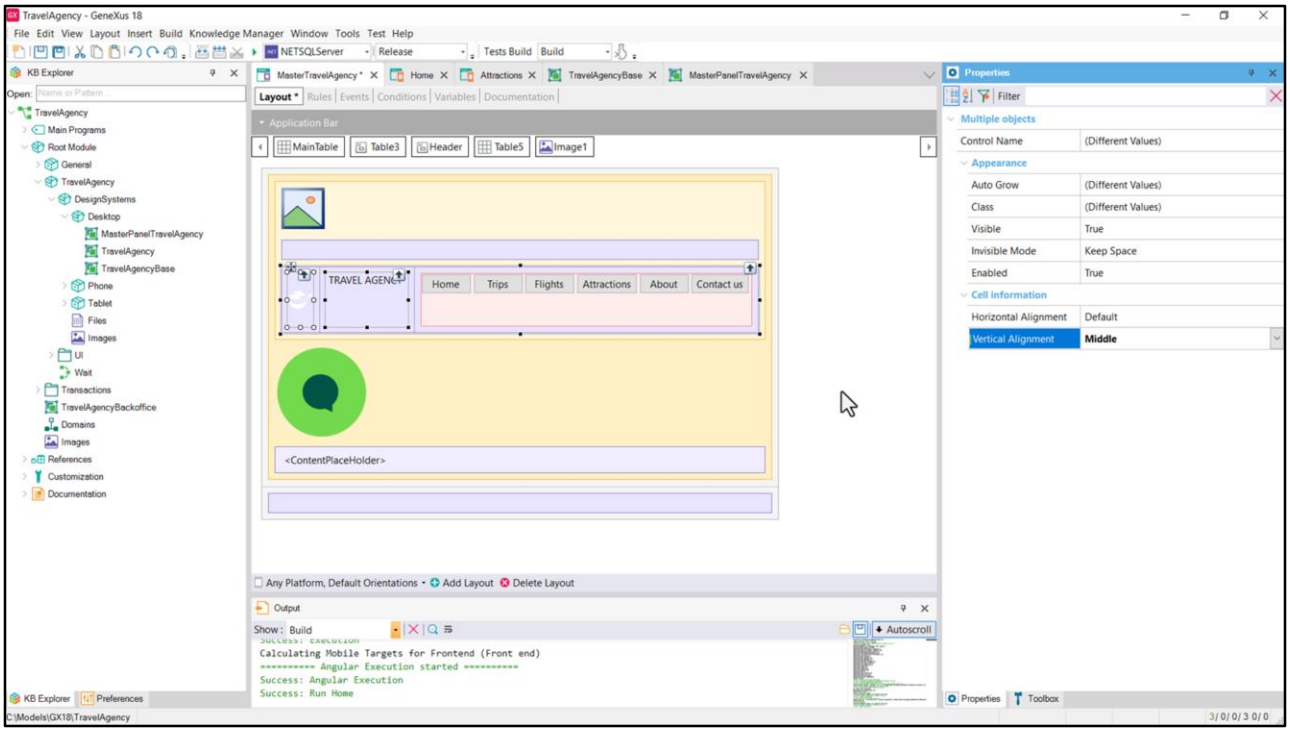
Así que… 37 dips, 150 dips, 100%... ¿Y el alto de la fila o de la tabla?



Sería este, de 43 dips. Y vemos que todos los controles aparecen centrados verticalmente respecto a sus bordes.

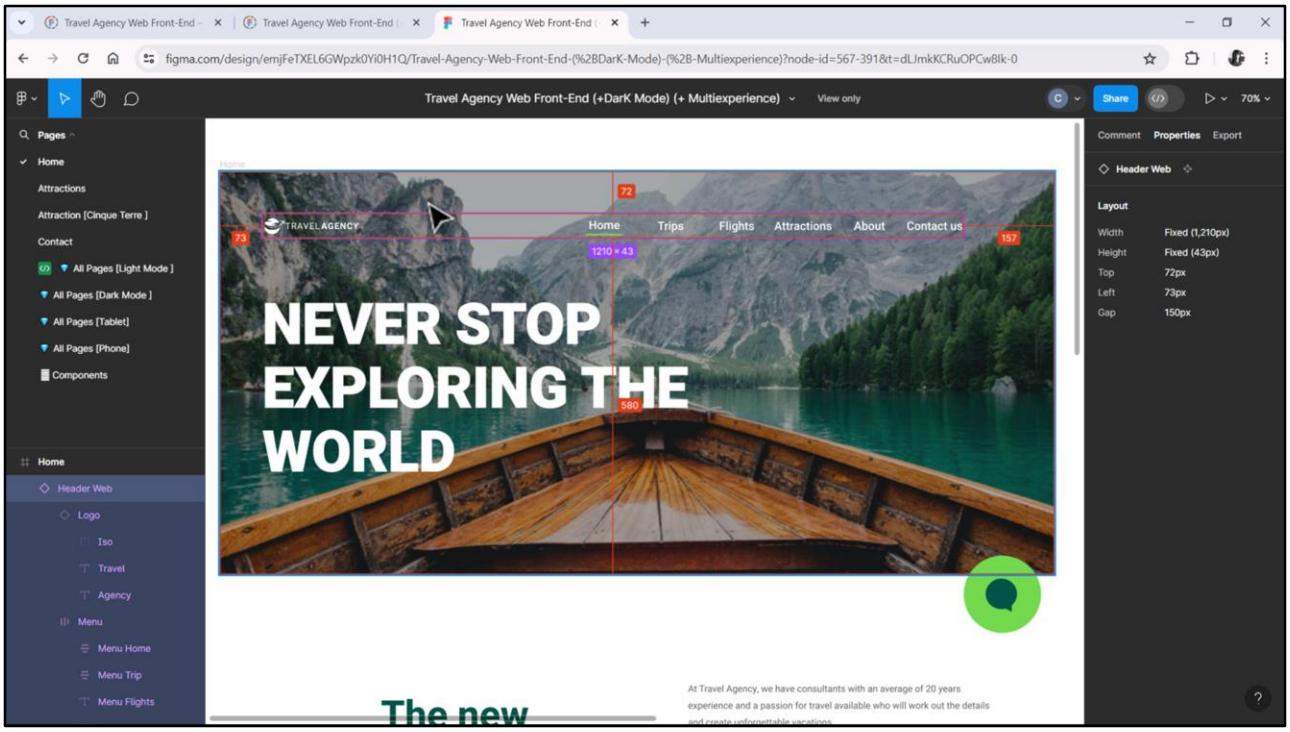


Bien, entonces coloco los 43 dips aquí. Automáticamente el Height me va a quedar de ese mismo tamaño…

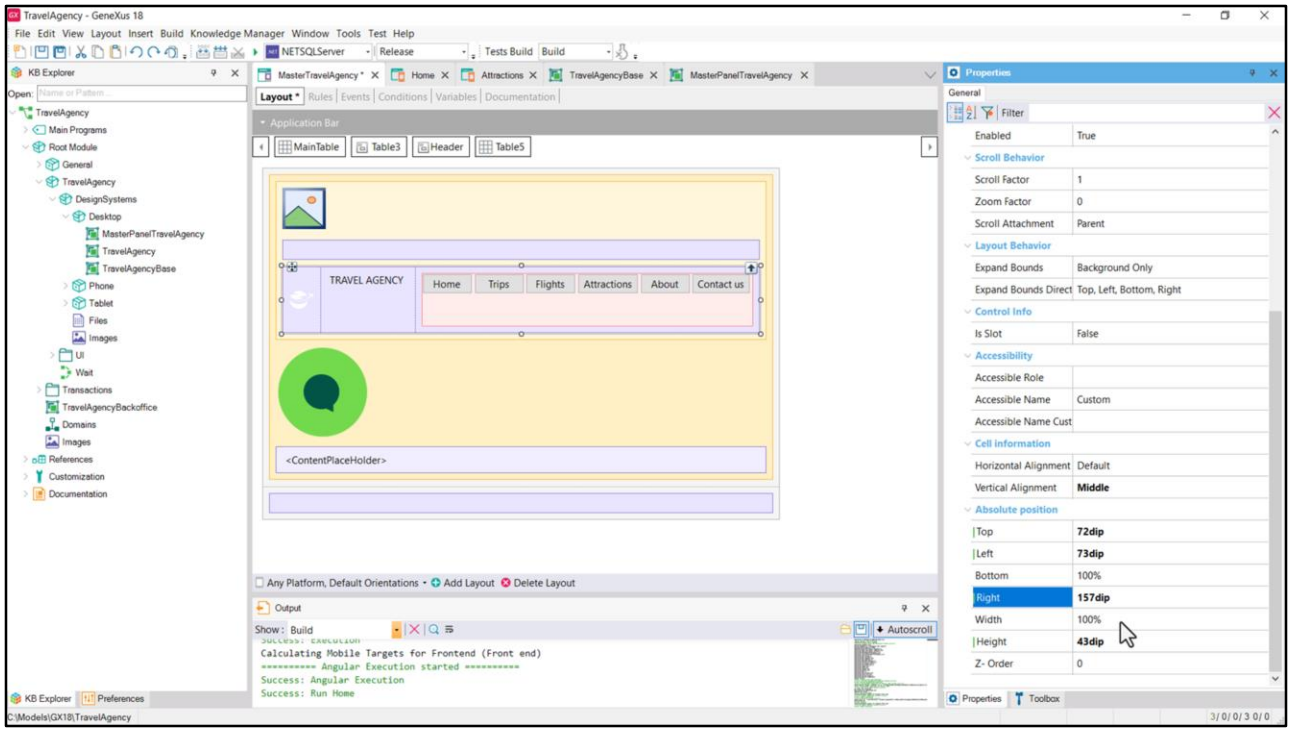


Y luego selecciono los tres elementos… aquí el flex… y coloco la propiedad Vertical Aligment para los tres en Middle.

Bien, y ahora, ¿dónde queda ubicada esta tabla dentro del canvas que la contiene? Tenemos que proporcionar su posición absoluta.

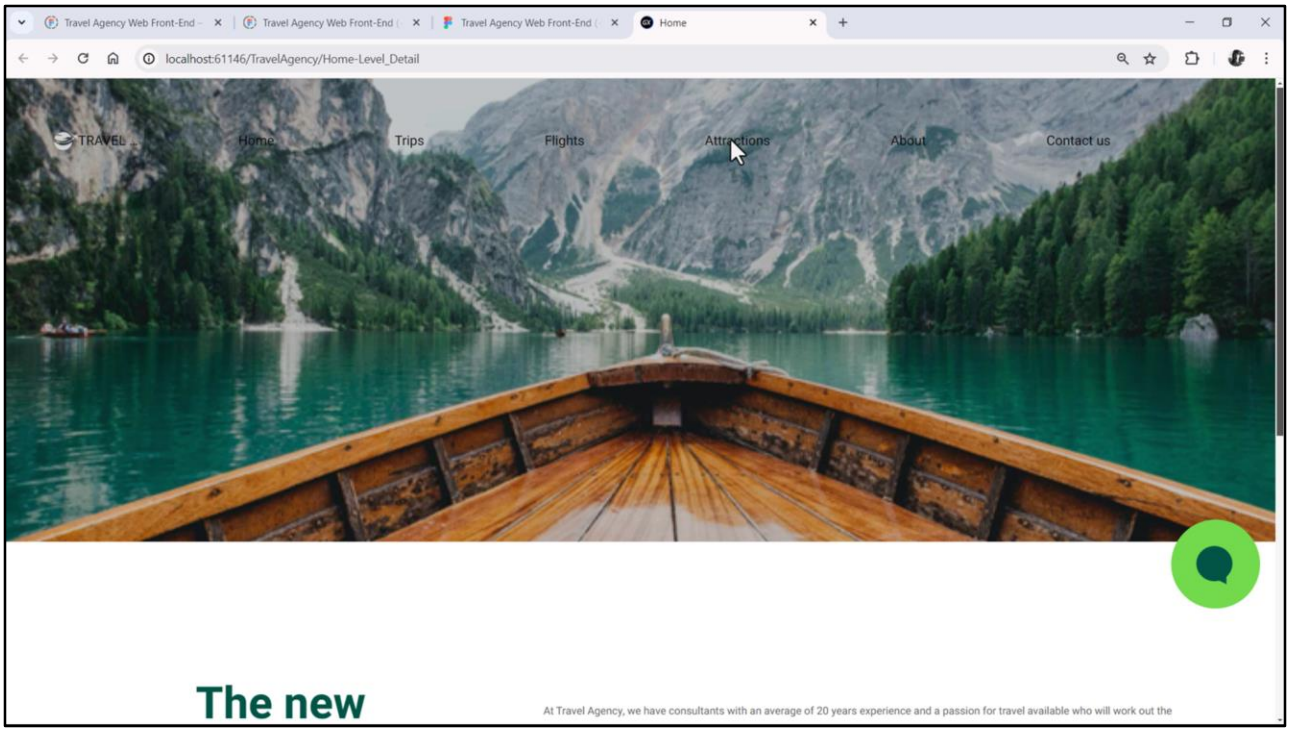


Y es: de arriba 72 dips, de la izquierda 73 y de la derecha 157. Con eso ya nos alcanza. De bottom será el 100% restante del alto del canvas.

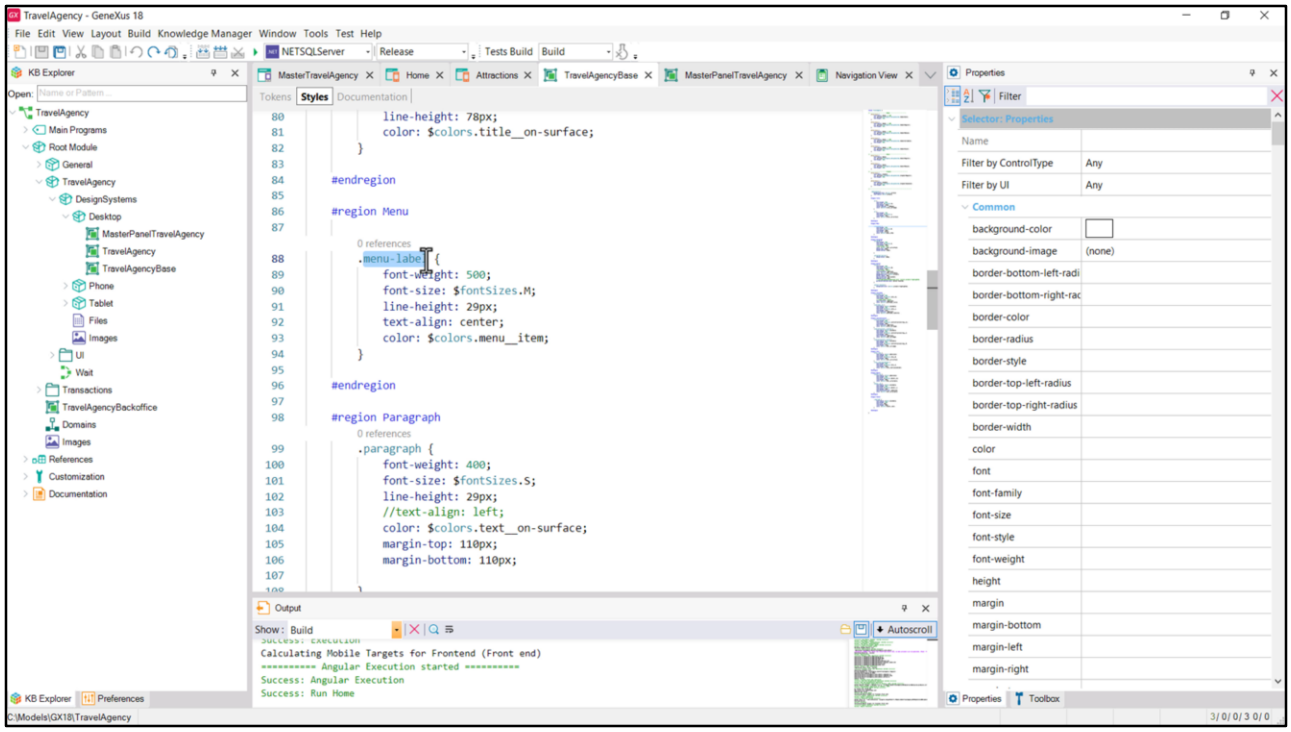


Así que… top 72 dips, el alto de la tabla era de 43 dips, y así me queda bottom del 100% restante.
Left 73 y Right 157.

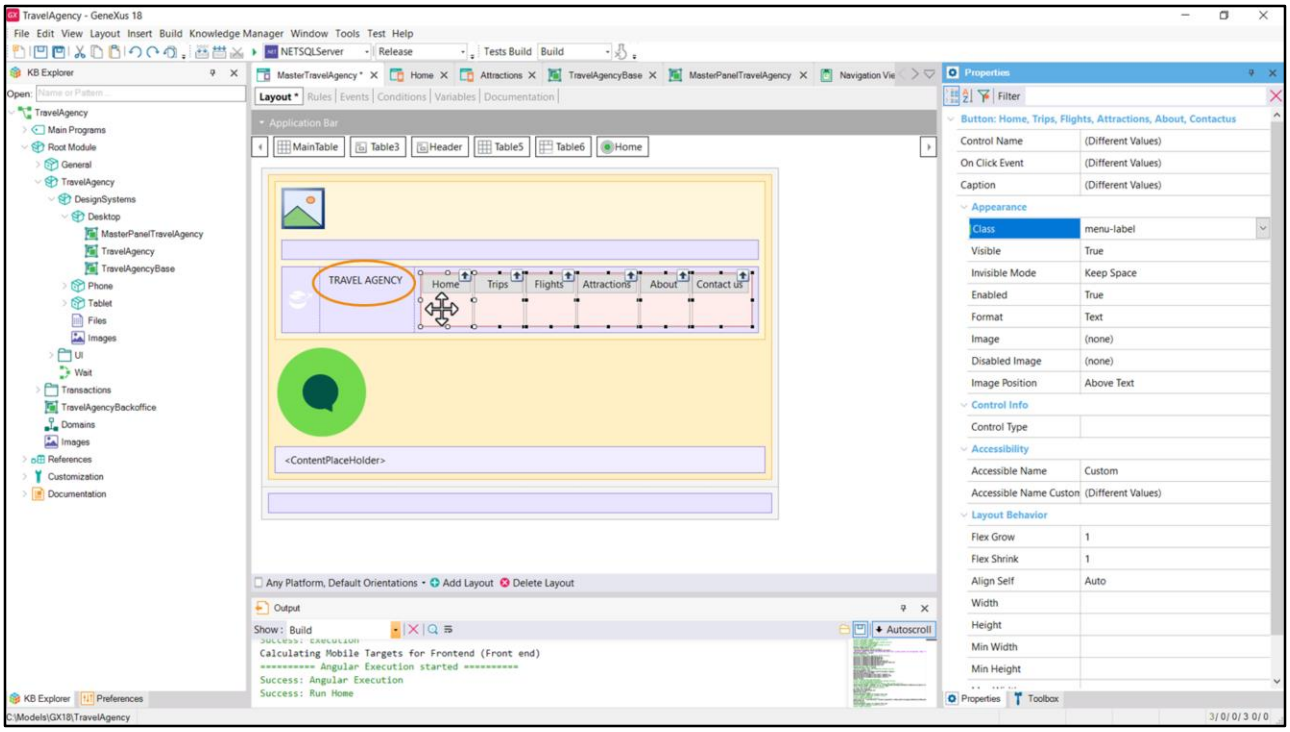
Vamos a probar lo que llevamos hecho.



Bueno, tenemos unas cuántas cositas para trabajar, ¿no? Para empezar, los estilos tipográficos de los textos…

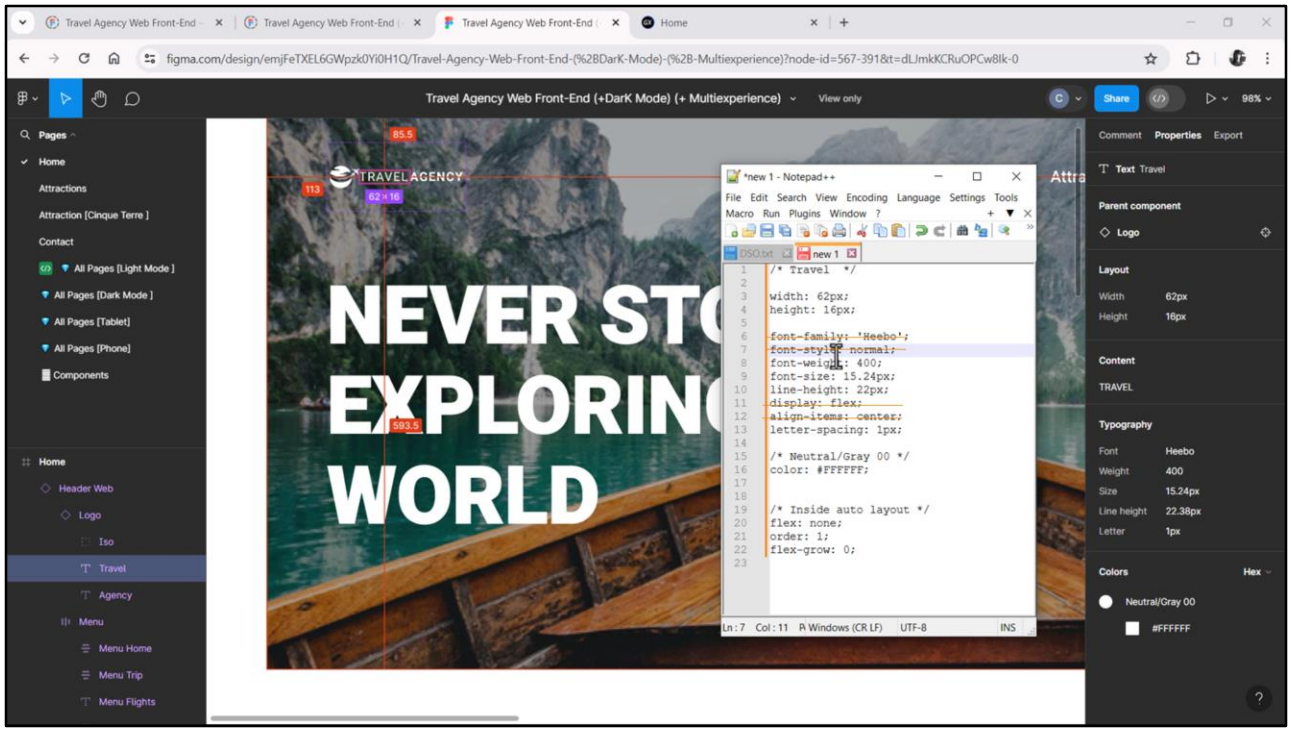


En el DSO base teníamos el estilo de las etiquetas del menú…



Así que empecemos por asignarle esta clase a todos los botones del menú. Los seleccionamos todos, y les cambiamos la clase por esta… vemos que efectivamente quedó cambiada en todos.

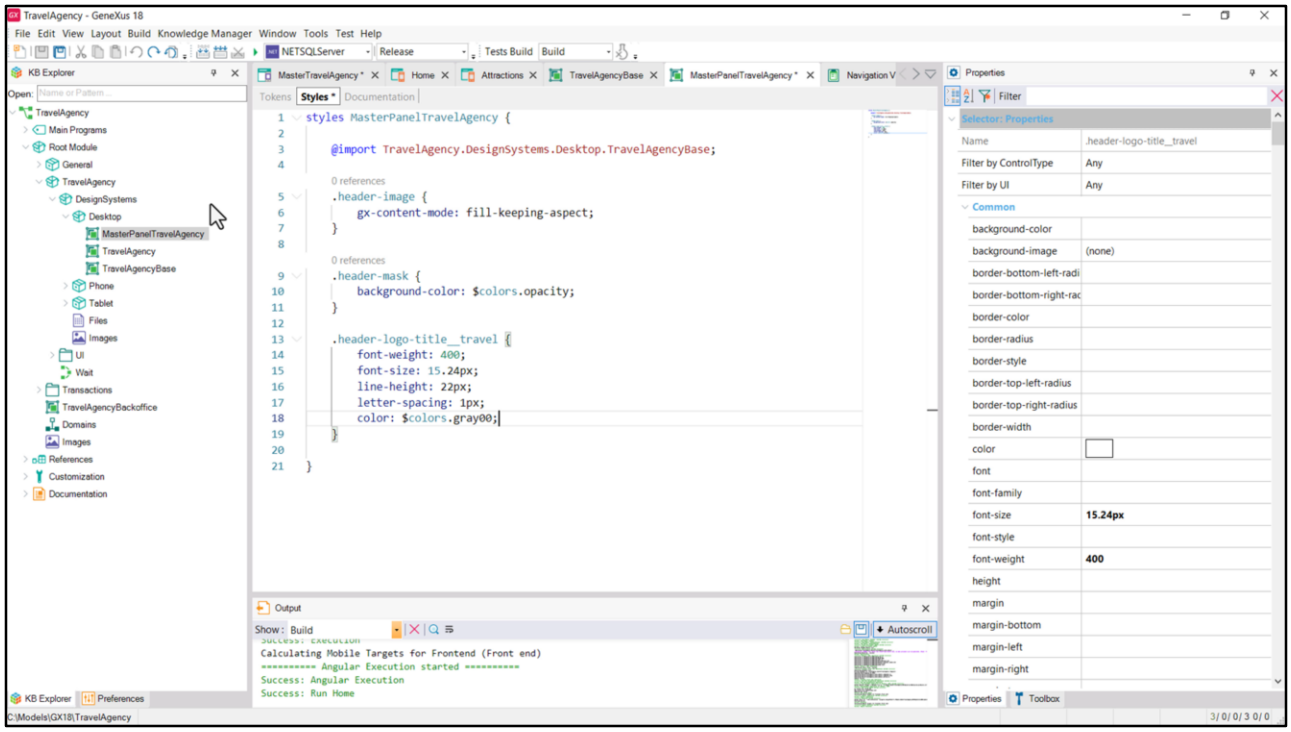
No habíamos definido una clase para el estilo tipográfico de estos textos…



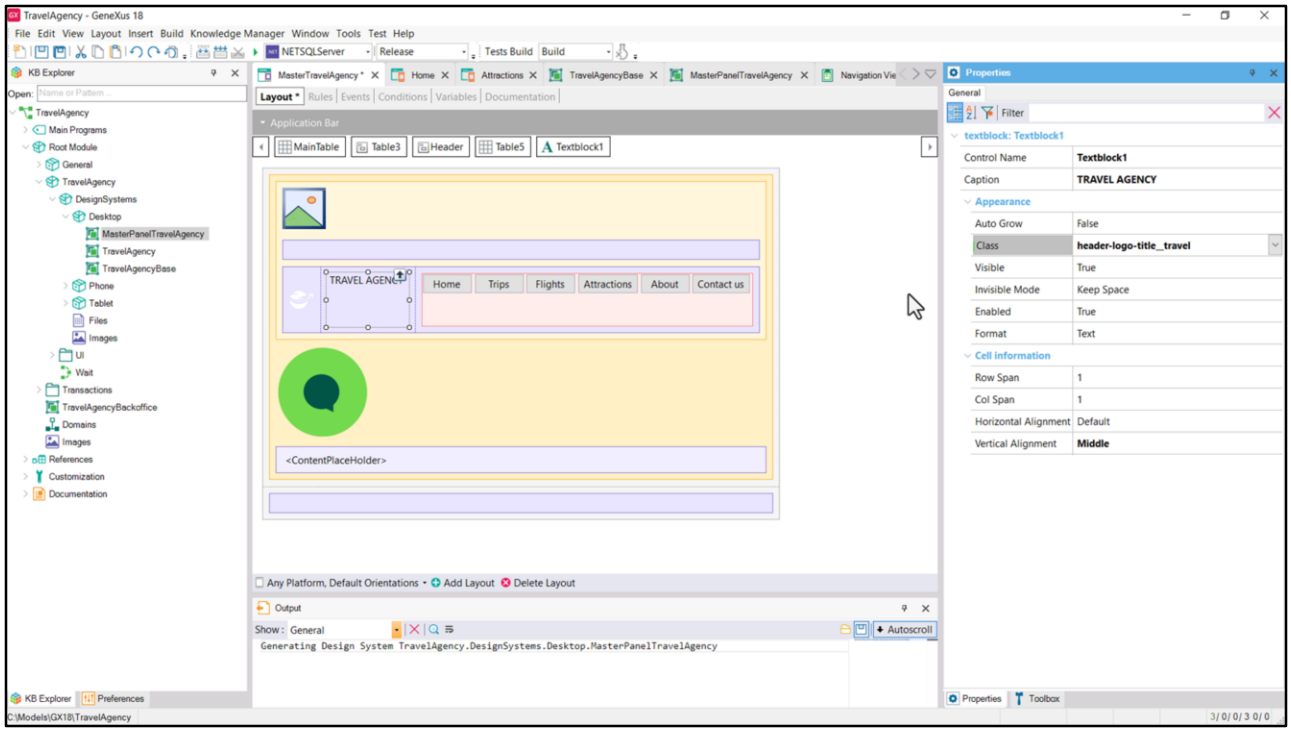
Vemos así que Chechu no había creado estilo… y que la única diferencia entre un texto y el otro es el peso de la fuente. Uno es de 400, la otra es de 700.

Bueno, vamos a copiar el CSS porque tendremos que definir una clase para estos textos. Quito estas propiedades que serán las de la fuente default, así que no las voy a necesitar… quito estas dos que son del elemento en el flex de Chechu, y la que voy a necesitar agregar además de estas 4 será el color de nuestro token gray00.

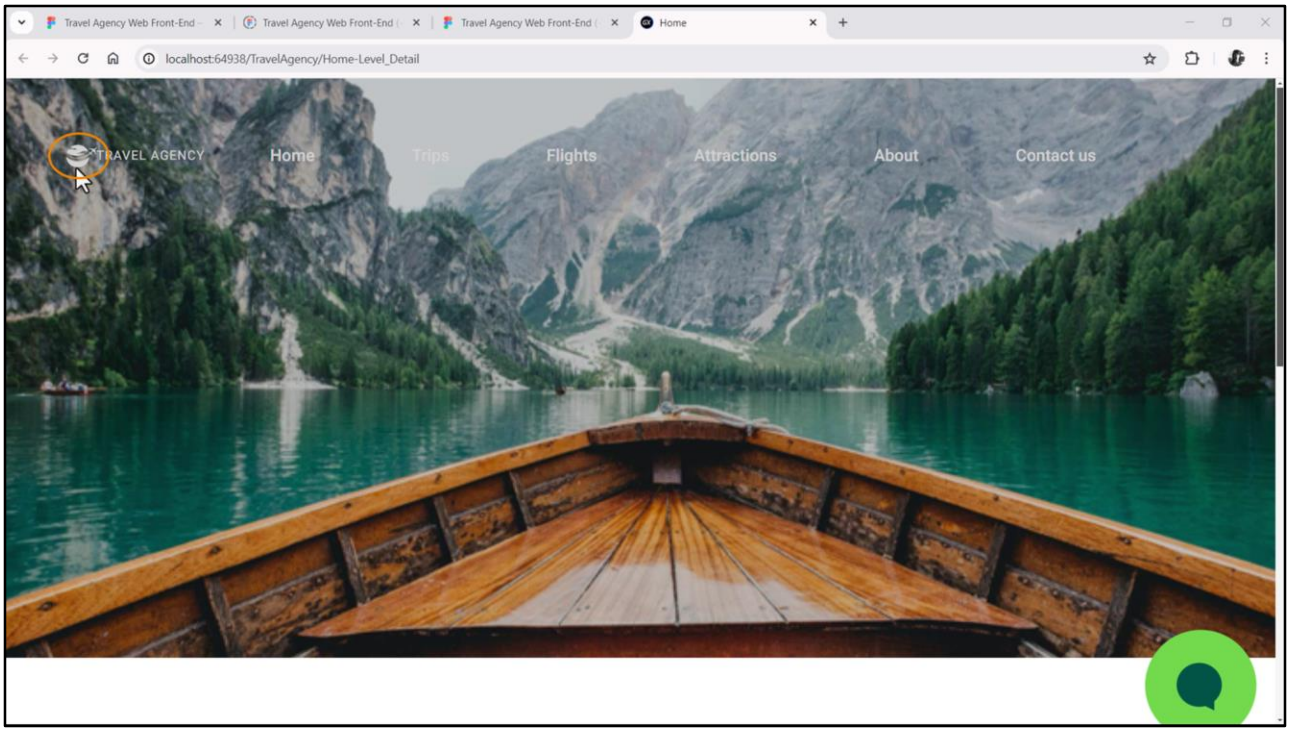
Copio estas 4…



Y en el DSO del Master Panel agrego una clase a la que llamaré…así. Ahora después veremos por qué este “travel” en el nombre…
Pego las propiedades… Agrego la de color…



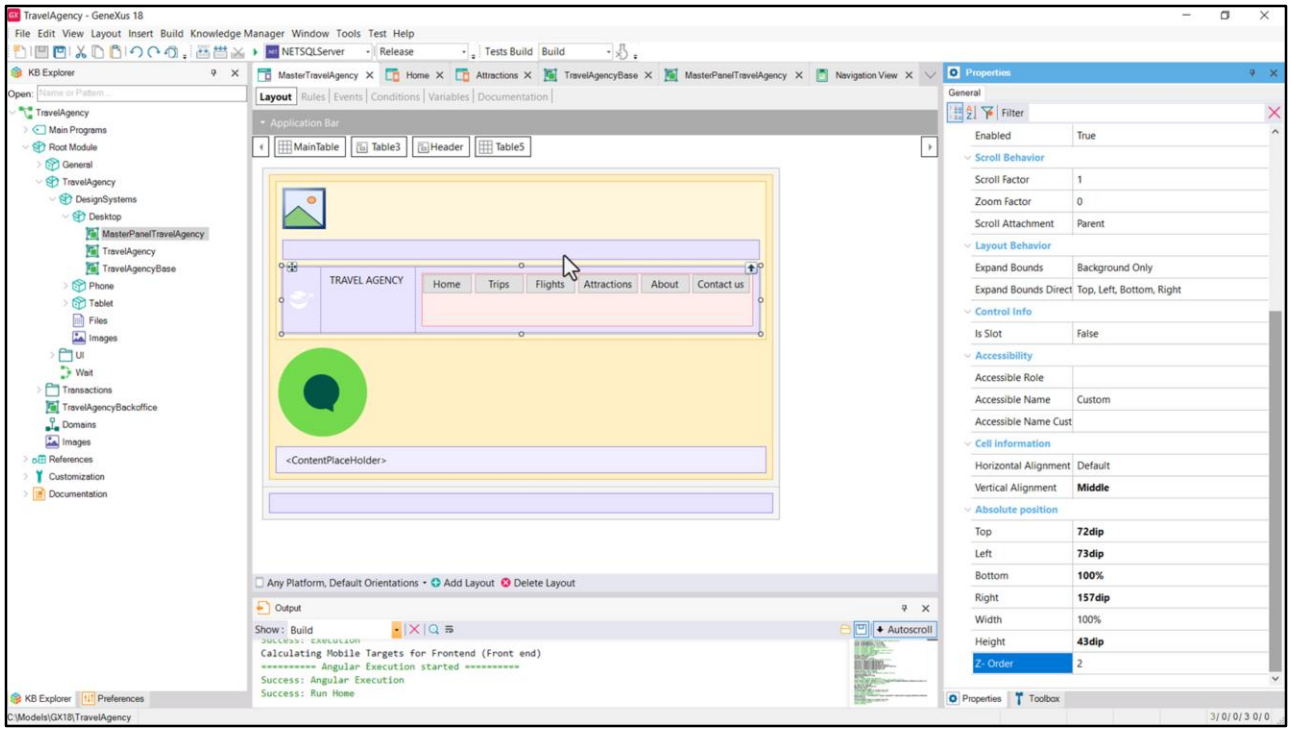
…Y le asocio la clase al textblock.



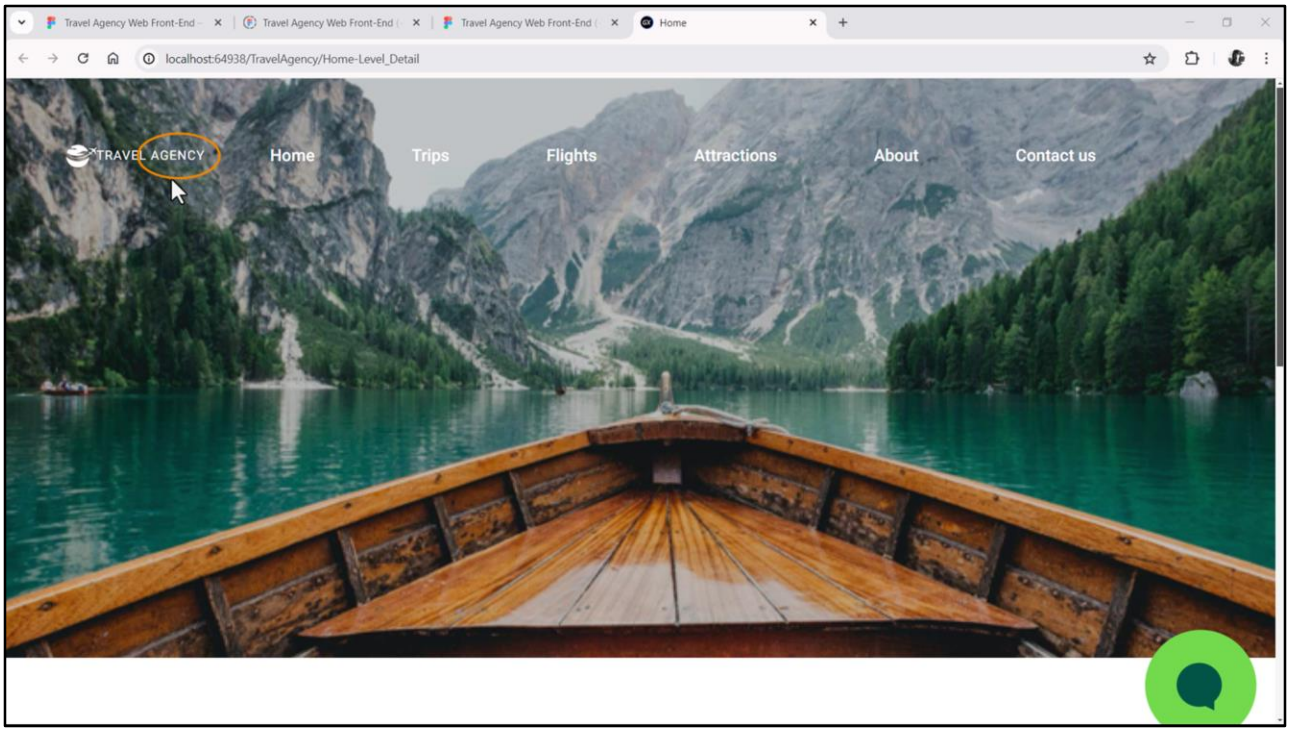
Probemos…

Si este ícono no les apareciera, prueben hacer un “Rebuild all” e intenten de vuelta. O vuelvan a insertar el ícono en la KB. Cuando hicimos el movimiento de módulo de los íconos en el video de Assets, no sé si se acuerdan, pueden haber quedado internamente mal ubicados allí. Así como les digo se arregla.

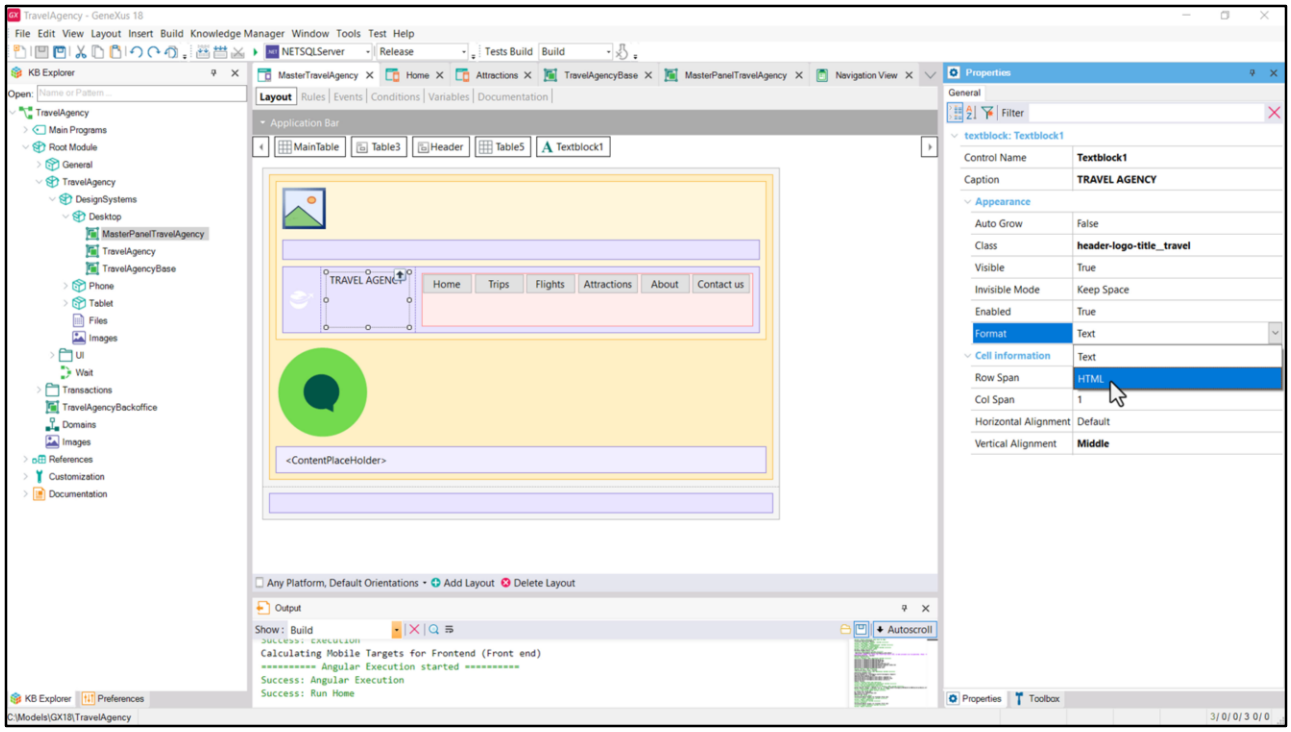
Está pasando algo raro… Se están viendo grises estos textos y no blancos… Es que están quedando por debajo de la máscara.



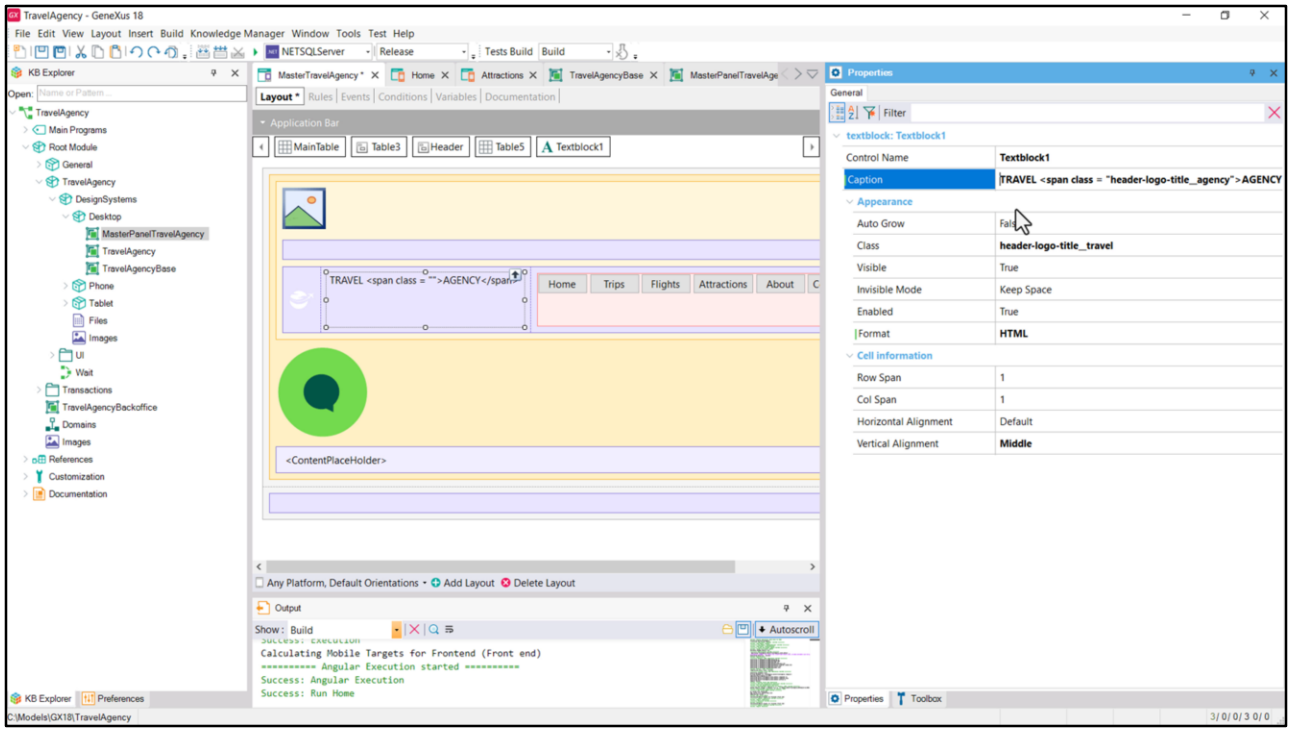
Le había dejado el Z-order default a esta tabla, el 0, y la máscara en cambio tenía el 1. Voy a asignarle el 2, para que esté por encima. Probemos ahora…



Bien. Tenemos ahora que trabajar en el menú que no está bien, pero también nos está faltando cambiarle a este texto el peso de la fuente, para que se vea de 700 y no de 400. Empecemos por ahí, así después nos dedicamos en lo que resta, y más tranquilos, al menú propiamente dicho.

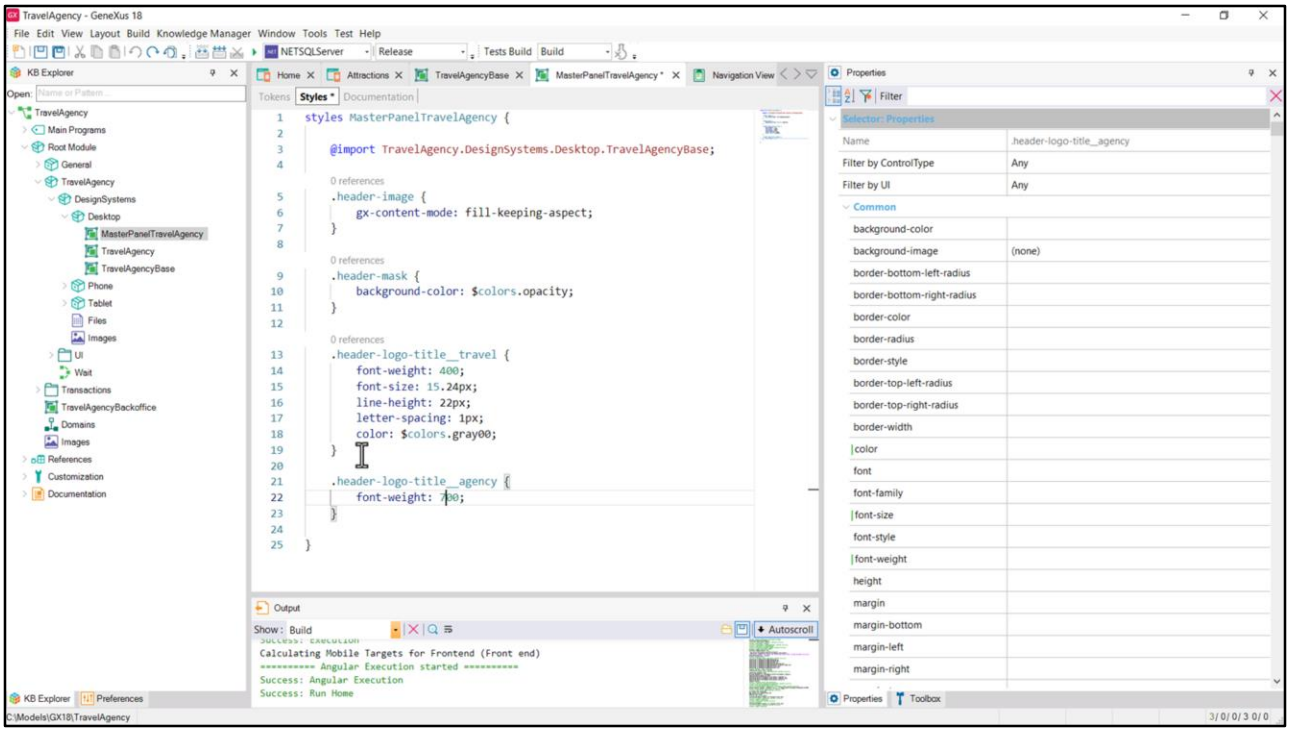


Como estamos en la aplicación Web, podemos cambiar el formato del texto por HTML. Esto para que el Caption pueda ser interpretado como html. Esto nos permitirá asignarle otra clase al texto AGENCY a través del tag span de HTML.

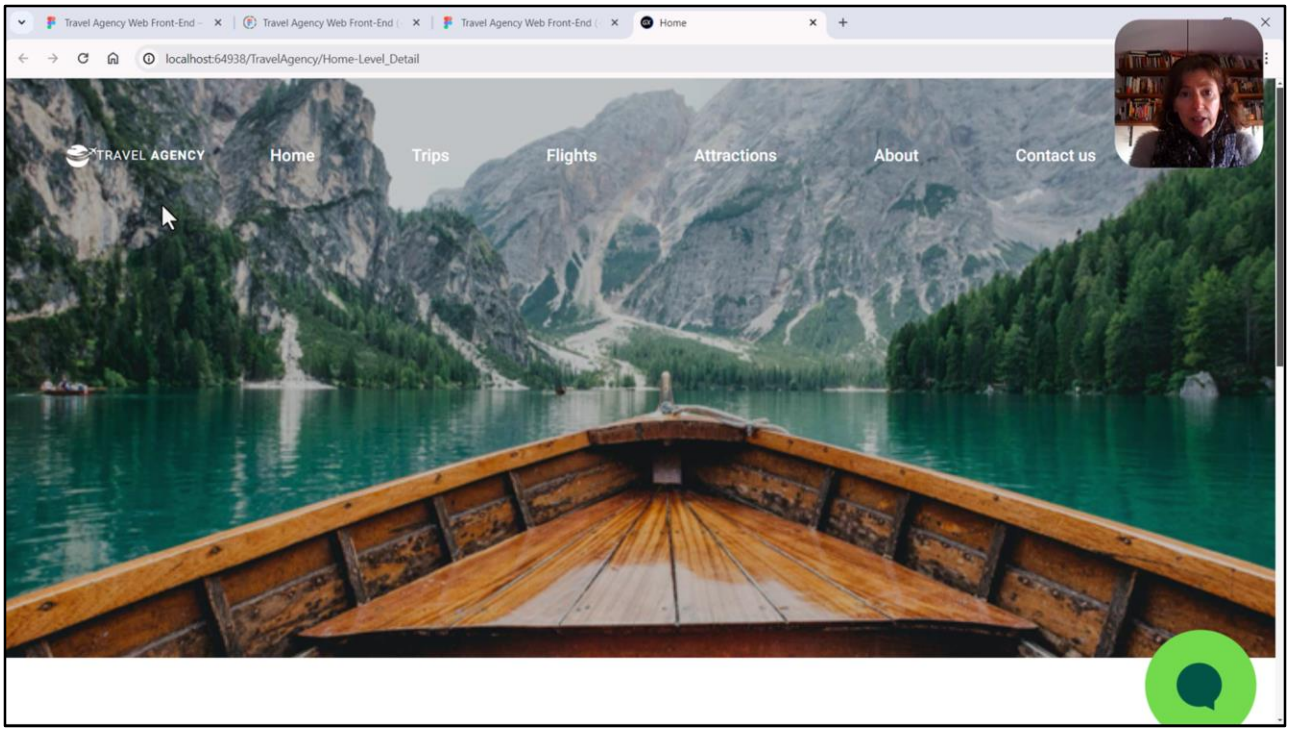


Tenemos la clase del textblock esta, “header-logo-title__travel”, y lo que haremos será asignarle a AGENCY la clase que crearemos, “header-logo-title__agency”, para de esta manera poder, a la palabra AGENCY, cambiarle el peso a la fuente. Es lo único que queremos hacer. Todo lo demás queremos que tome las propiedades de “header-logo-title-travel”.

Entonces copiamos este texto…



…Venimos al DSO, y vamos a especificar esa clase. A la que lo único que le haremos será cambiarle el peso de la fuente: de 400 a 700.



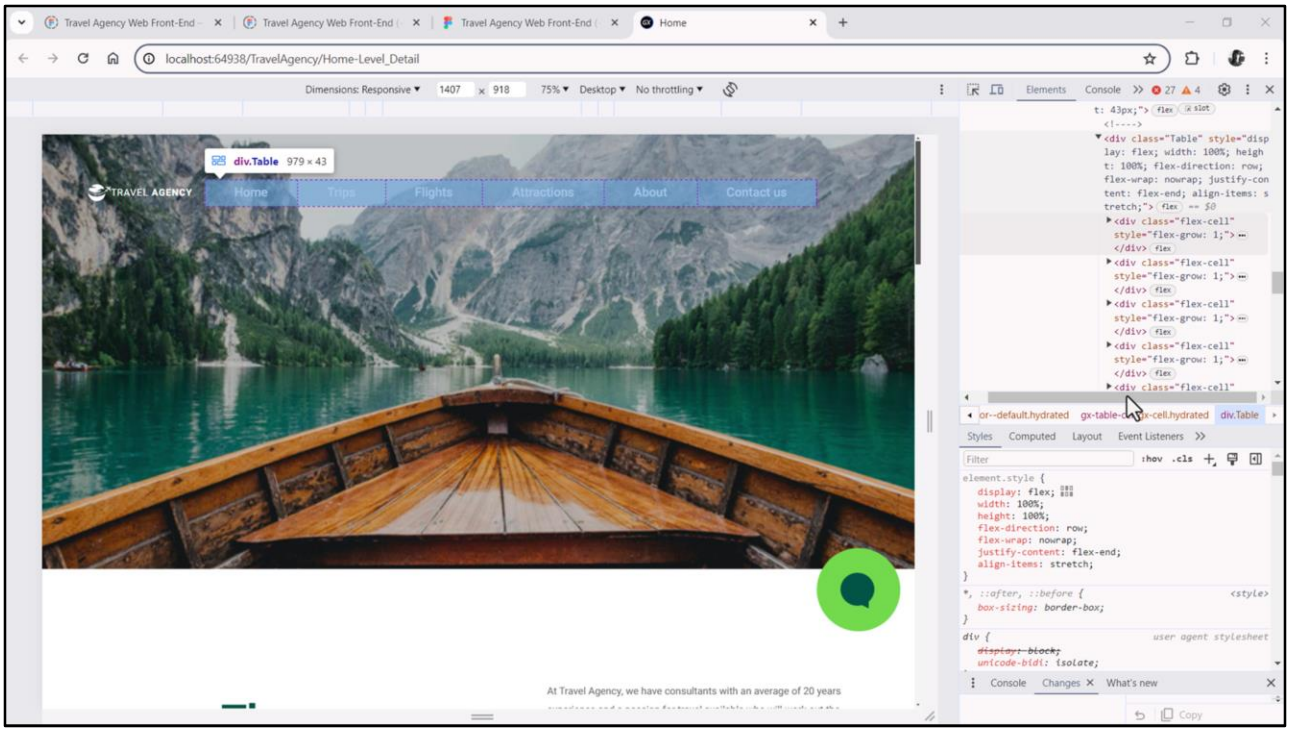
Y acá lo podemos ver.

Esto que acabamos de ver no significa que Angular permita colocar cualquier html en los textblocks y en los campos editables con formato html. De hecho va a eliminar cualquier definición de un atributo style que se coloque allí. Pero lo que sí permite es lo que hicimos, usar clases del DSO dentro de ese html.

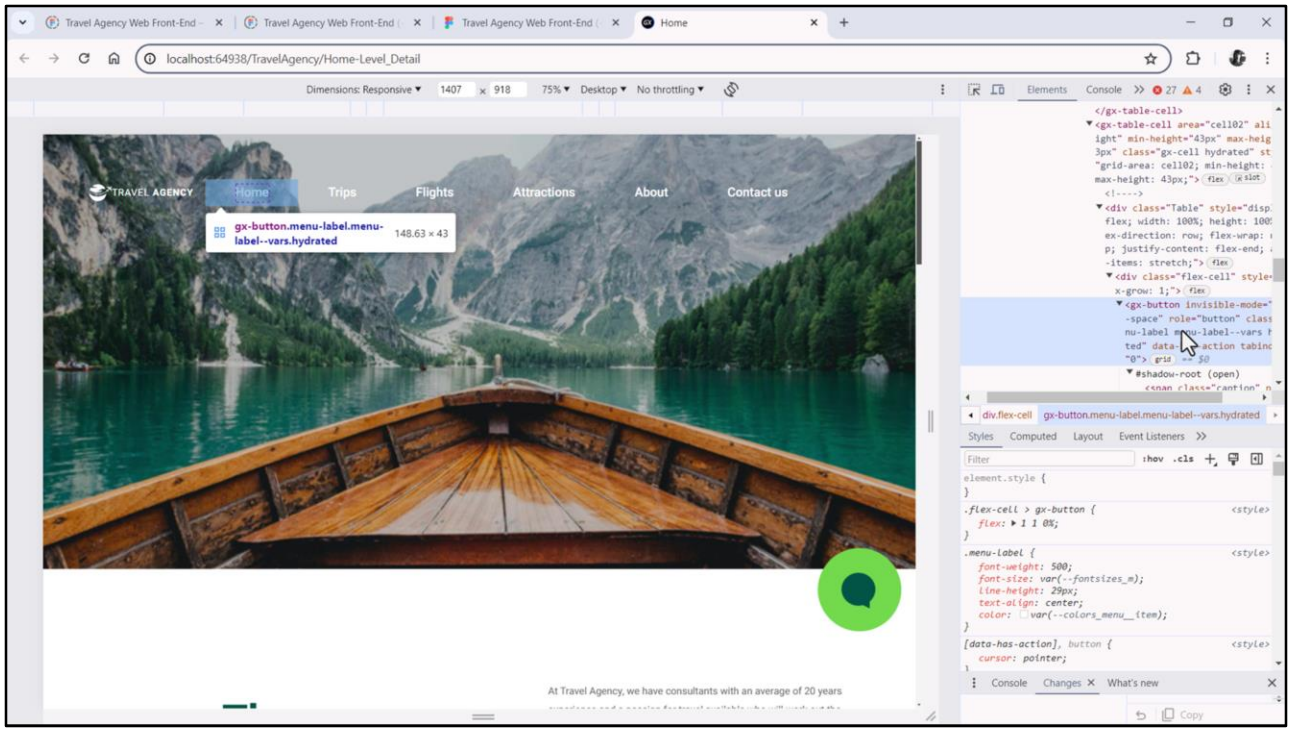
Todo lo contrario de lo que sucede con Android o Apple: ellos no permiten usar clases del DSO dentro del html y no eliminan el atributo style.

Así que en un principio, para que la solución funcione para ambos casos, es decir para Angular y también para Android y Apple, lo que nos haría falta sería agregar entonces un atributo style donde coloquemos allí las propiedades de la clase que, en nuestro caso, la tenemos en el DSO. Así, de esa manera, si ejecuta Angular va a eliminar todo lo que tenga que ver con el atributo style, lo va a eliminar, no lo va a tomar en cuenta… pero sí va a tomar en cuenta la clase. Y en caso contrario, si ejecutamos para Apple o para Android, va a eliminar, no va a tomar en cuenta la clase y sí va a tomar en cuenta el atributo style.

Como para tamaño desktop solamente tendremos la aplicación Angular en este caso no tenemos que hacer nada.



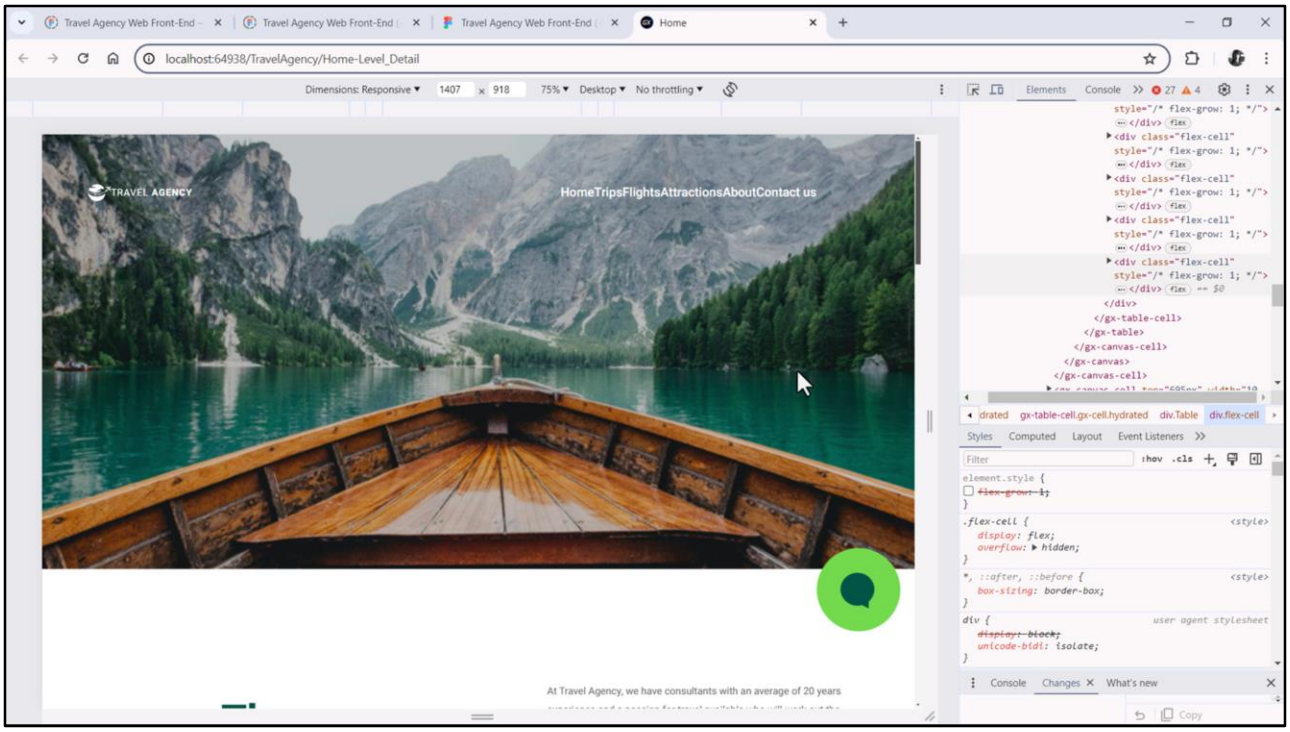
Ahora sí, ¡al menú! Si lo inspeccionamos tal como lo tenemos hasta ahora, vemos aquí la tabla que corresponde al Flex, y vemos que tiene una celda para cada botón.



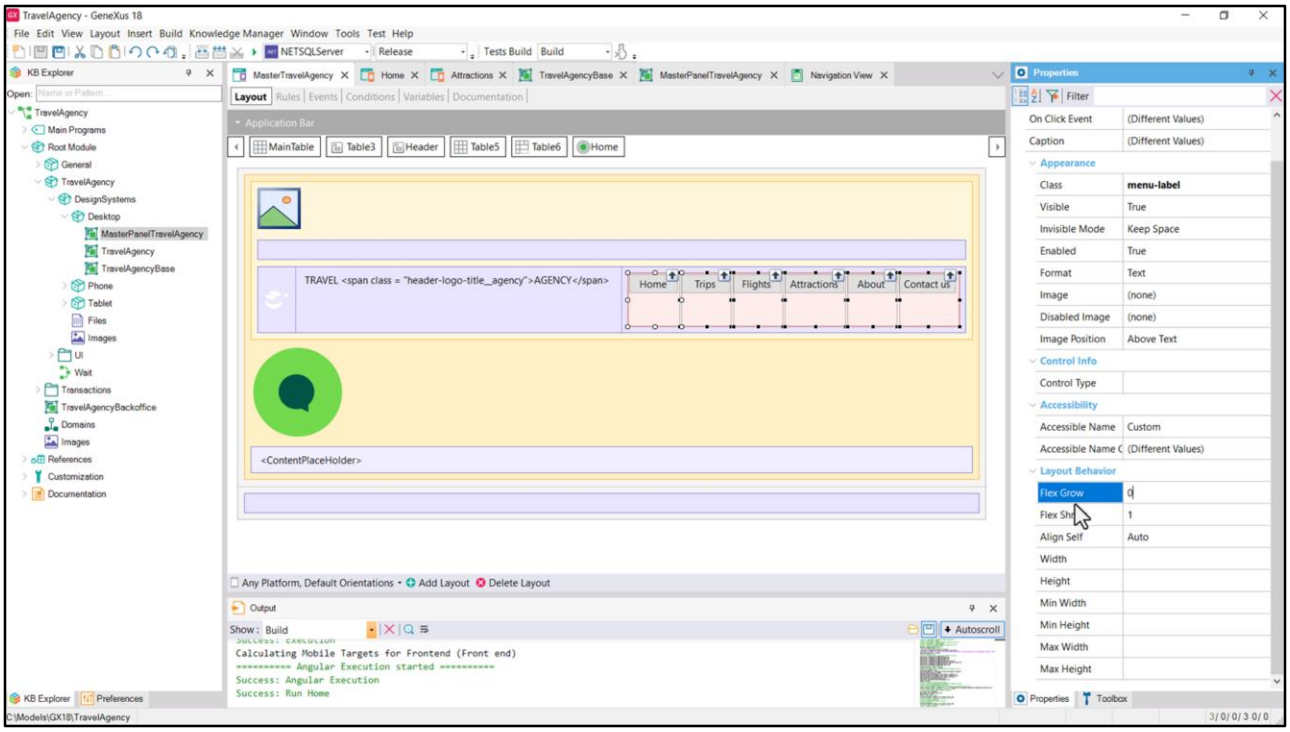
Así, aquí vemos la celda del primer botón, el botón y su caption.

Y lo mismo para los demás botones.

Observemos que está permitiendo que cada botón crezca de modo de ocupar equitativamente todo el espacio del flex libre.



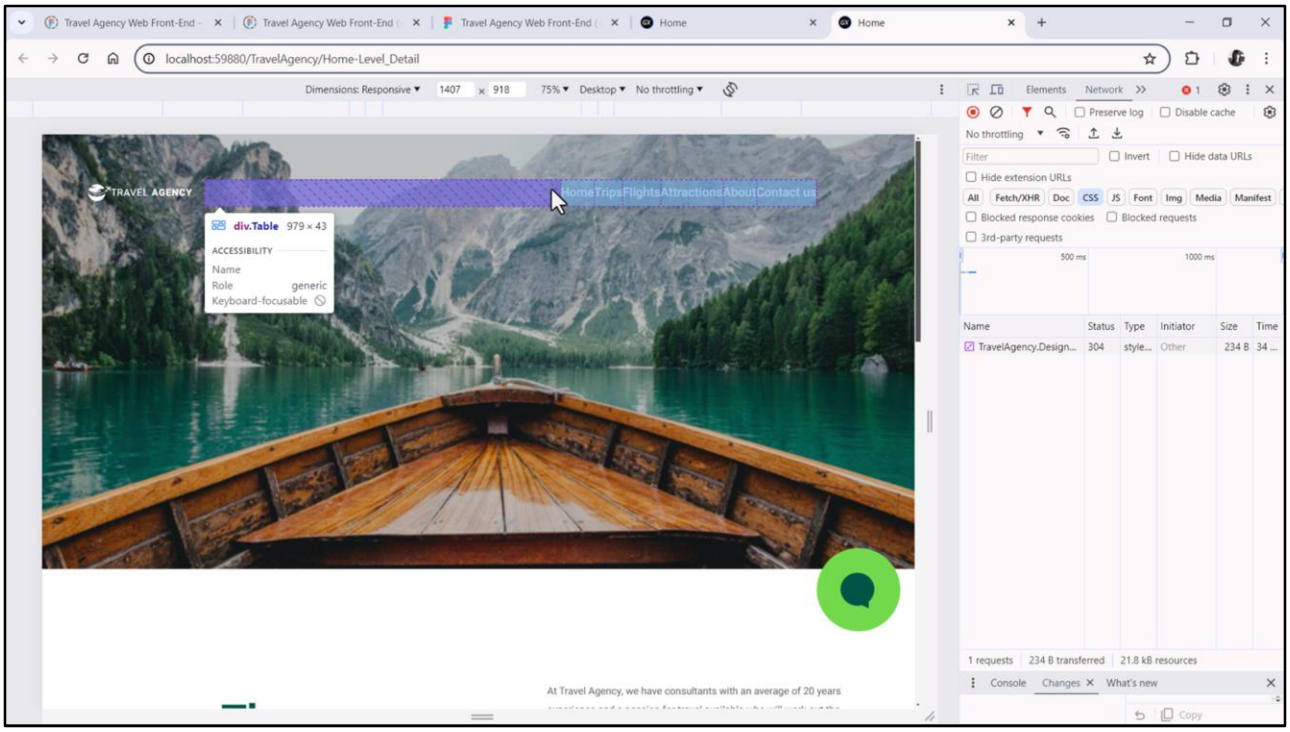
Esto se debe a la propiedad Flex Grow en 1, que tiene cada uno de los botones. Veamos qué pasa si las quitamos…



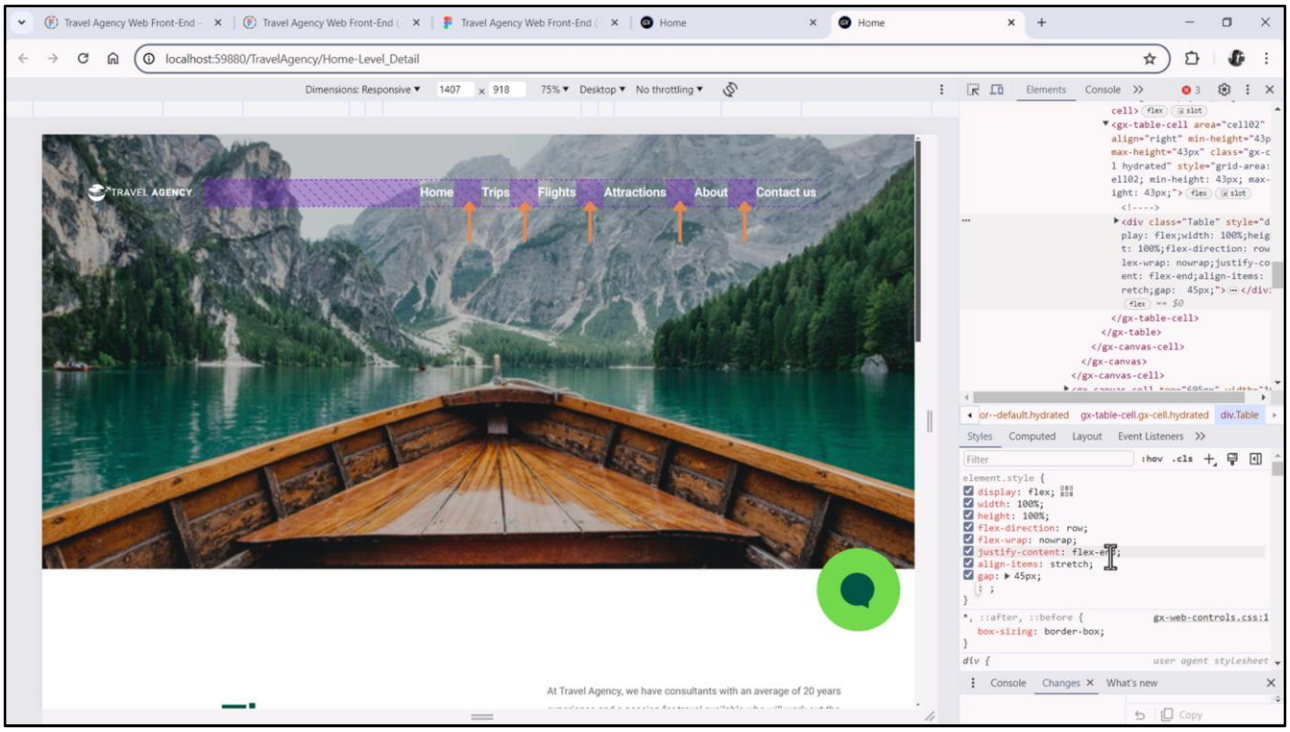
En GeneXus vemos que si bien el contenido está justificado de acuerdo al final del flex, cada botón tendrá por default la propiedad Flex Grow en 1. Veamos la descripción de la propiedad…

Dice que determina qué tanto este hijo crecerá si queda espacio libre para distribuir entre los demás ítems de la línea. Si el ítem tiene un valor positivo para esta propiedad, entre todos los que tengan valor positivo se distribuirá proporcionalmente el espacio libre.

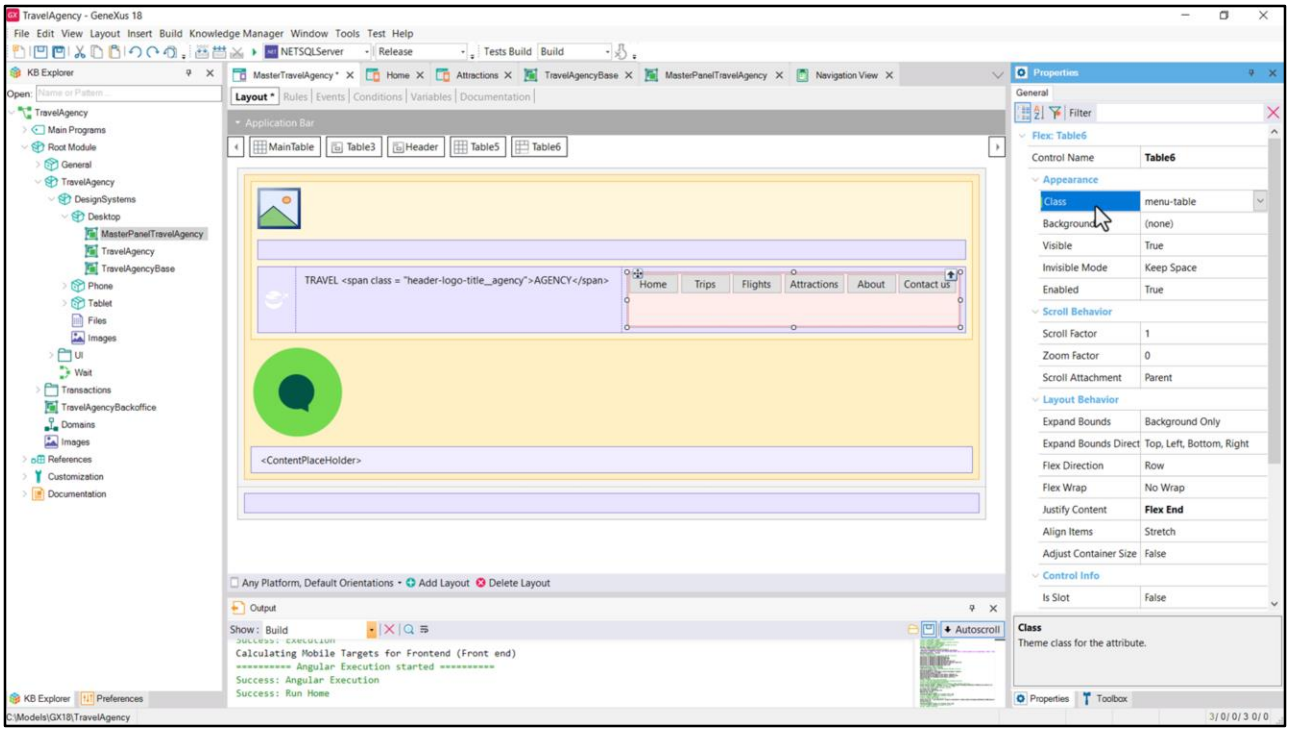
Todos tienen por defecto el mismo valor, 1. Entonces seleccionemos todos los botones y cambiemos su valor por 0, para que ocupen exactamente su espacio y no más. Ejecutemos…



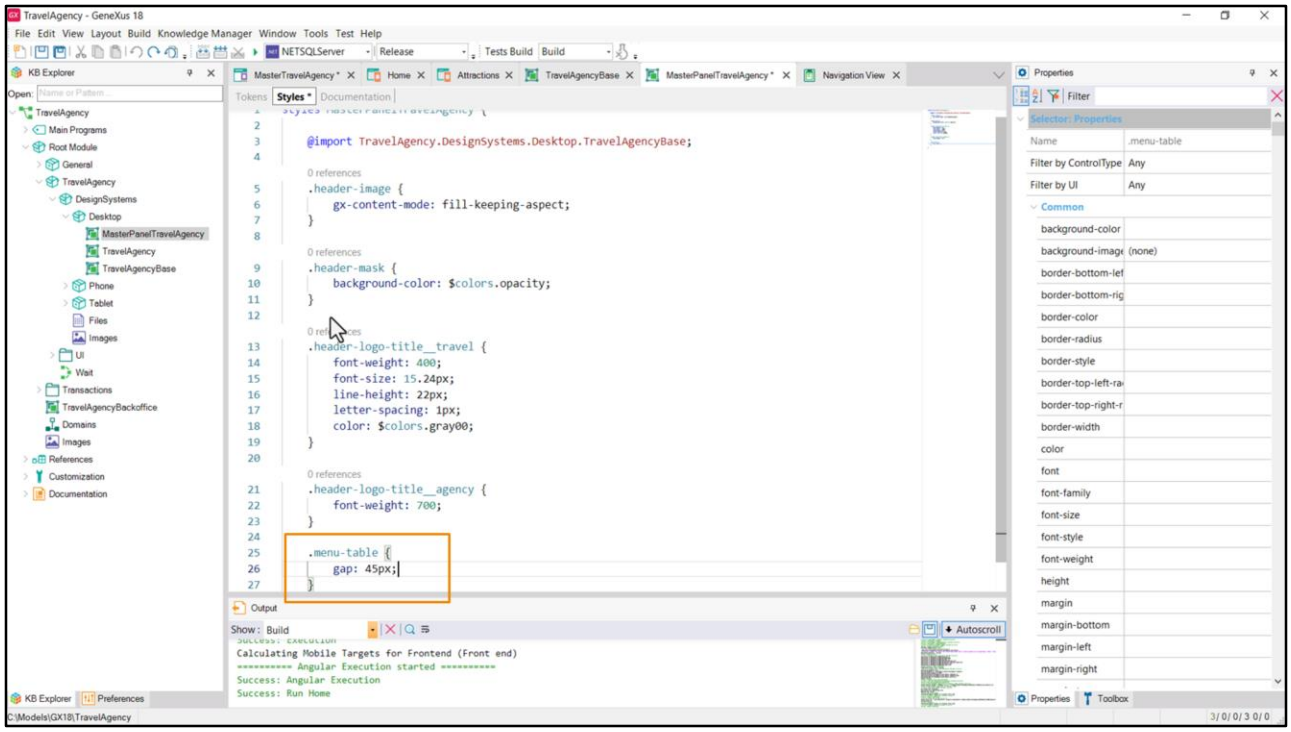
Si ahora inspeccionamos… vemos todos los botones contra el final del flex, y sin espacios entre ellos.



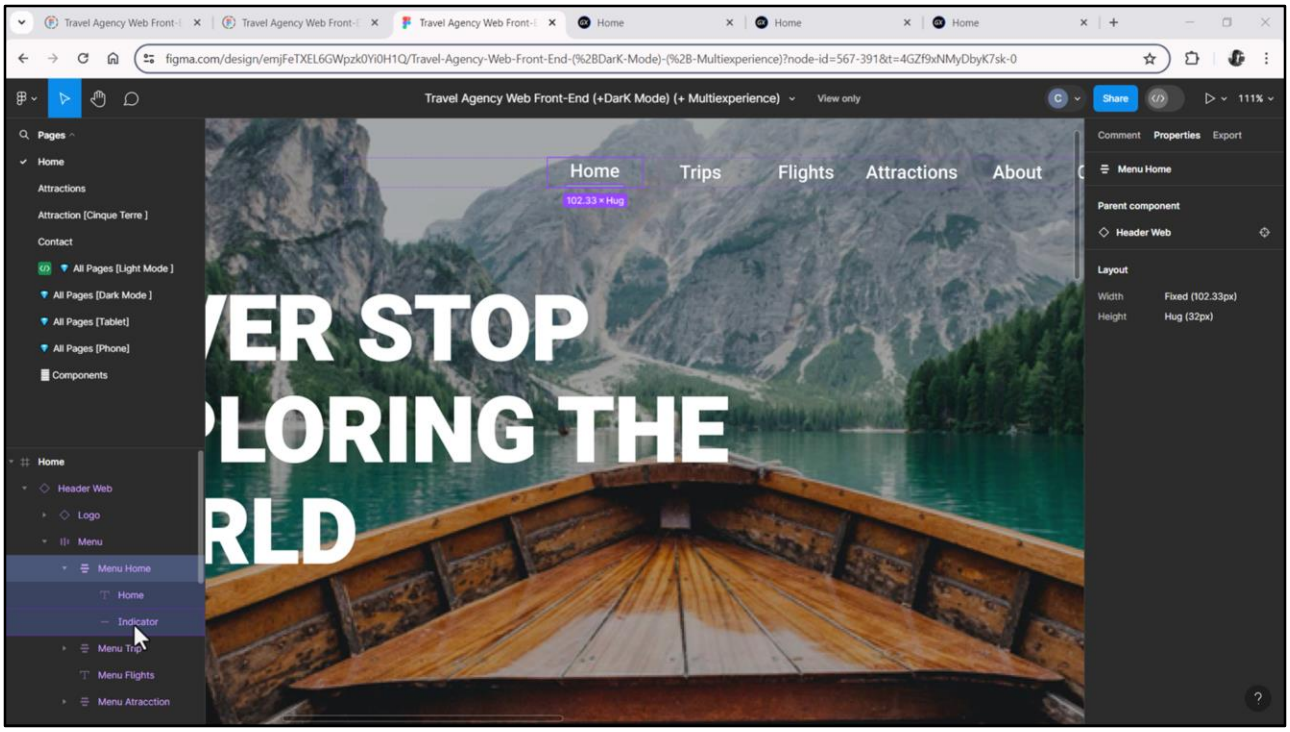
Para darles un espaciado uniforme de forma tal que la distancia entre el fin de un caption y el principio del otro sea la misma, probemos la propiedad gap, con 45 píxeles por ejemplo. Bien, luce bien. Allí vemos claramente cómo están justificados contra el final del flex, por eso queda todo este espacio libre adelante, y cómo entre ellos está funcionando este gap de 45.



Esta es una propiedad a nivel del flex… y no la tenemos como propiedad estática del control, así que tendremos que especificarla en una clase, a la que llamaré menu-table.

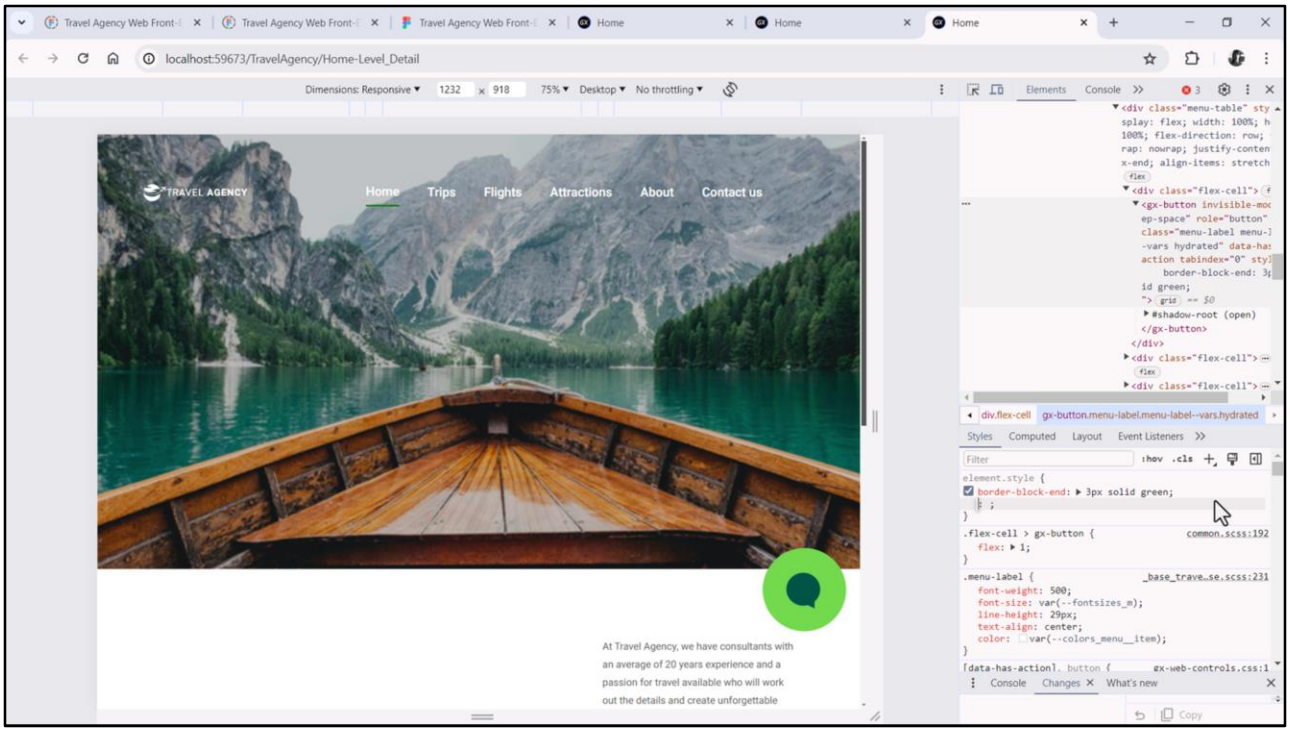


Y allí coloco la propiedad gap.



Ahora tendríamos que pensar cómo implementar el indicador de la opción seleccionada.

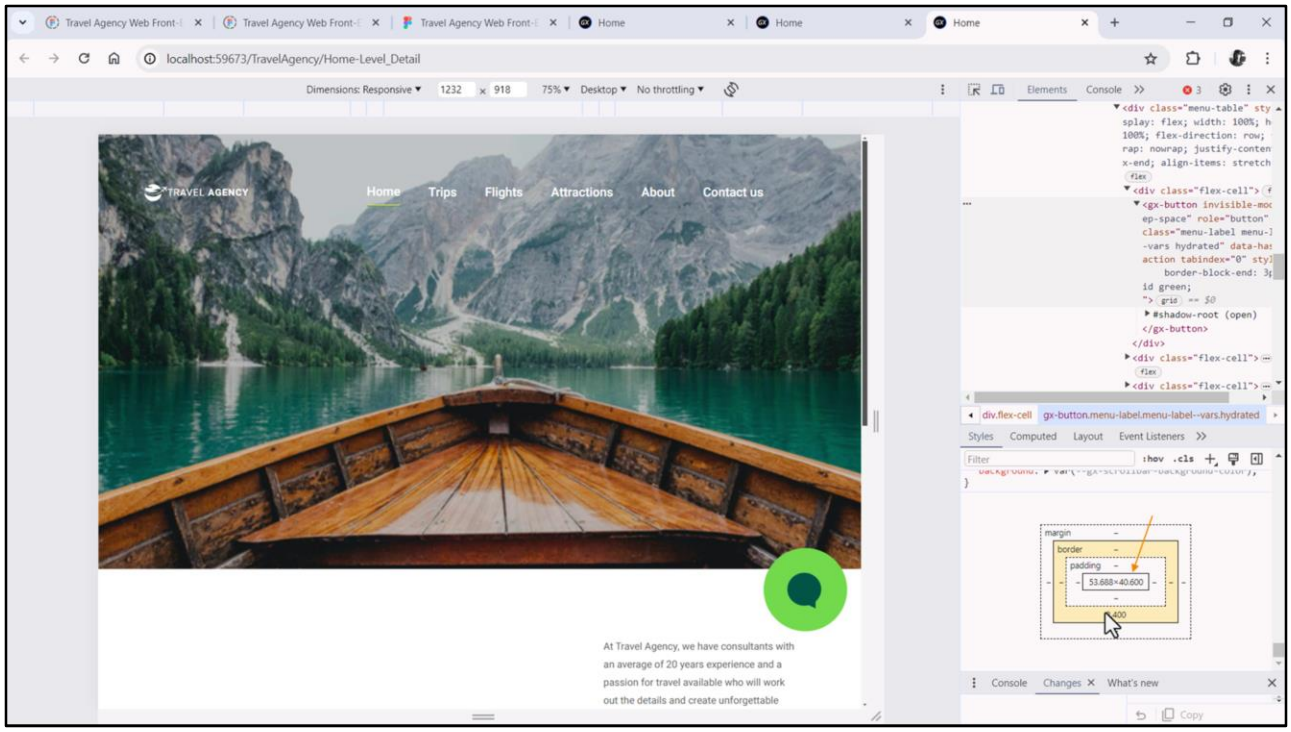
En Figma vemos cómo Chechu implementó cada opción como un flex vertical, con el texto encima y el indicador como una línea debajo.



Como se trata de una acción utilizamos botón en GeneXus, y no texto, y como todo control, tiene 4 bordes, uno a cada lado.

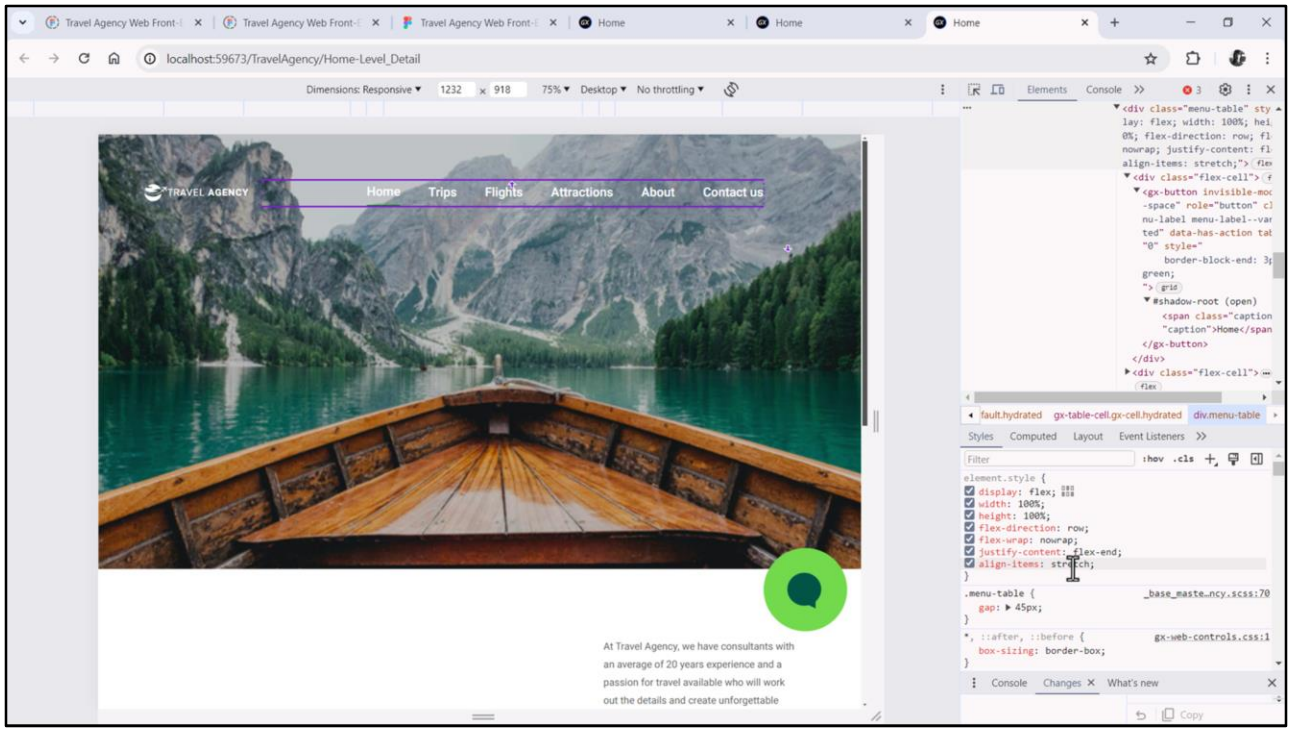
Nos alcanzará con que los bordes de arriba y de abajo tengan un valor positivo pero sean transparentes para todos los botones, y en cambio el de abajo tenga color sólo para el botón seleccionado.

Probemos con este elemento… la propiedad es la border, y utilizaré la lógica y no la física, así que será border-block, con lo que me estaré refiriendo a los dos bordes en dirección vertical: el de inicio y el de fin. Si quiero solo el de fin entonces coloco end. Y allí indico que quiero que sea de 3 píxeles, por ejemplo, con línea sólida, y el tercer parámetro sería el color, por ejemplo, green.

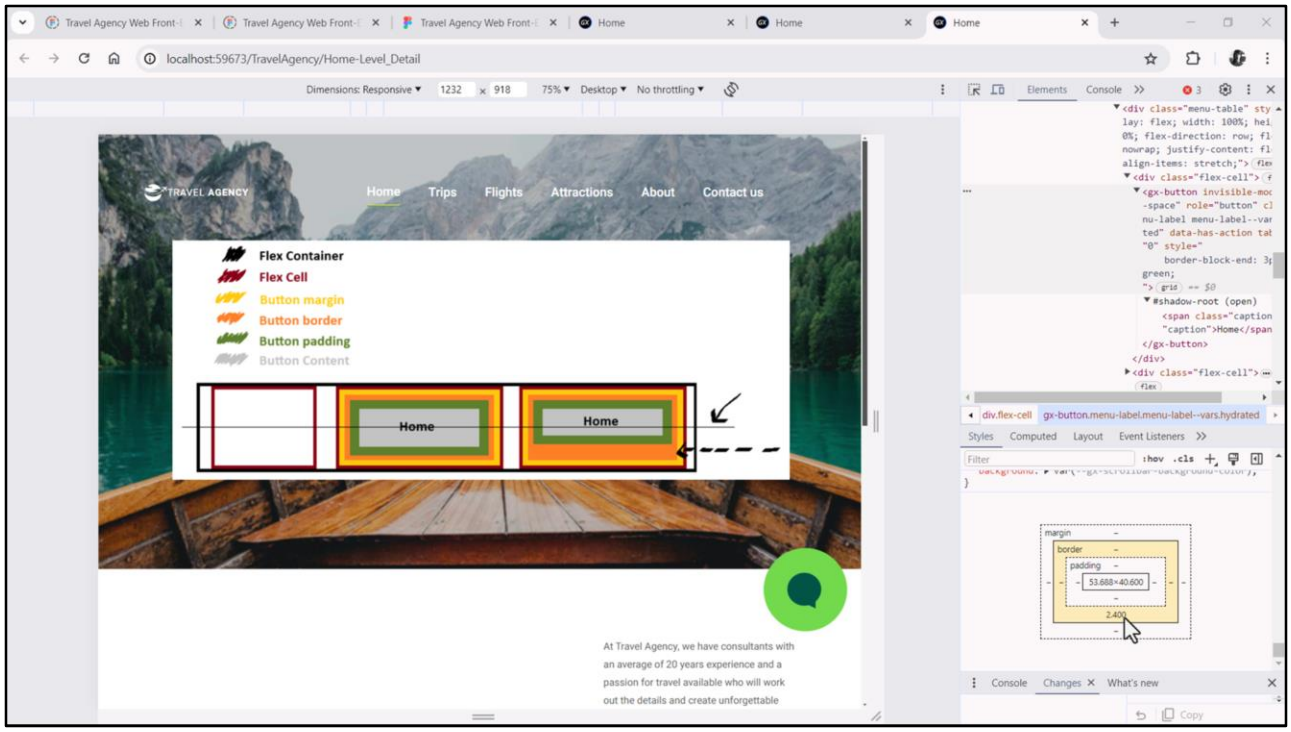


Este 2.4 en lugar de 3 en un error de precisión de punto flotante. Se suele dar si se hace zoom en la pantalla en múltiplo diferente de 100%, que es el caso. No le prestemos atención ahora.

Sí observemos que al darle 2.4 al borde, el alto del botón será de este valor, para que juntos sumen los 43 píxeles del alto del flex.

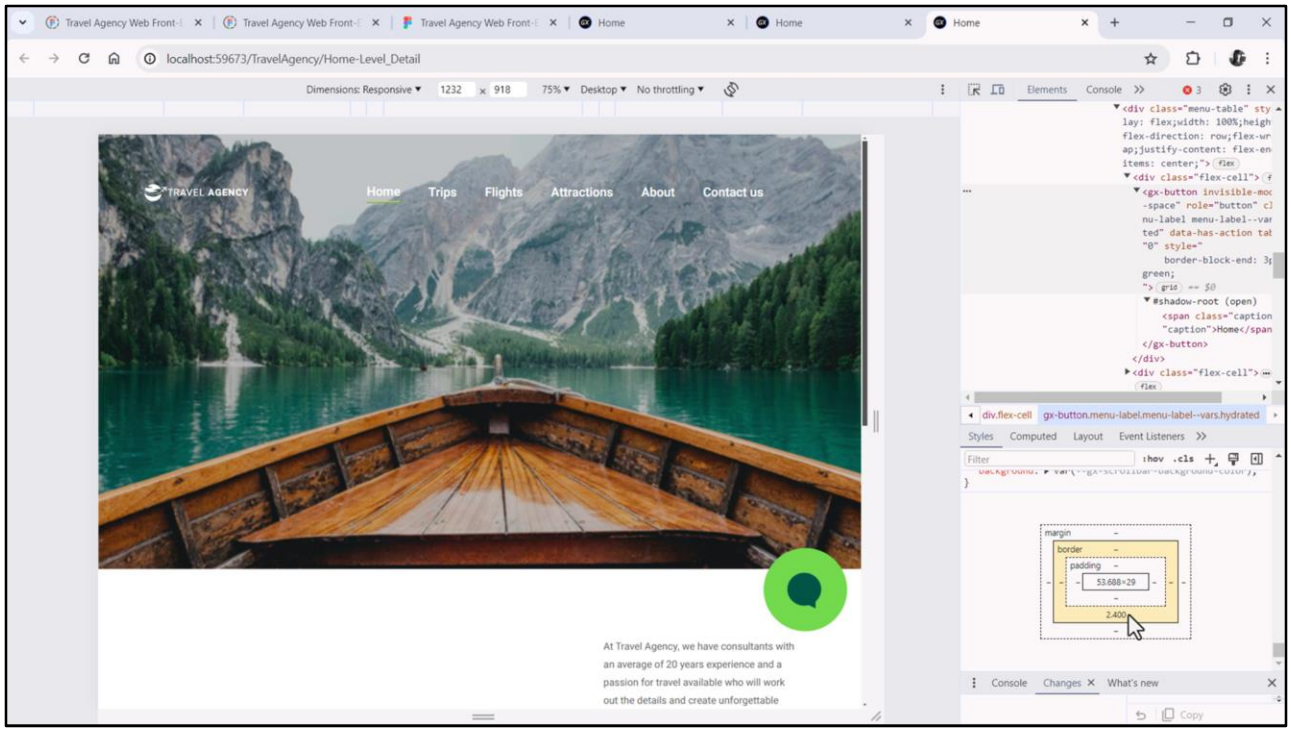


¿Cuál es el problema con esto? Vemos que el flex tiene la propiedad align-items con el valor default que es Stretch. Esto hace que cada ítem se estire para ocupar verticalmente todo el alto de su celda, que ocupará a su vez todo el alto del flex. En este caso 43 píxeles.

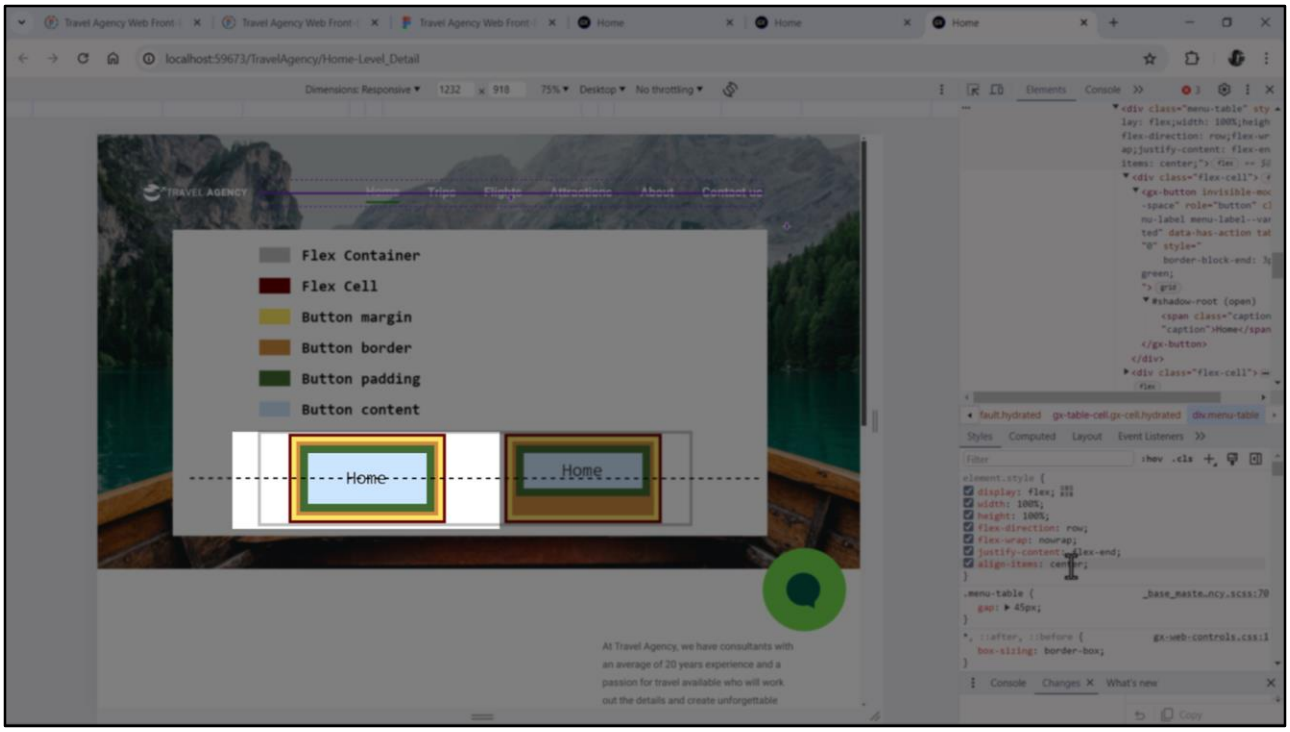


Pero cada caja del botón incluye no sólo el contenido del botón, sino padding, border, y margin. Por lo que, si dejamos borde inferior de 3 píxeles, lo que veremos es que el texto del botón no nos quedará centrado por la línea media, y por tanto no nos quedará centrado respecto al logo de Travel Agency.

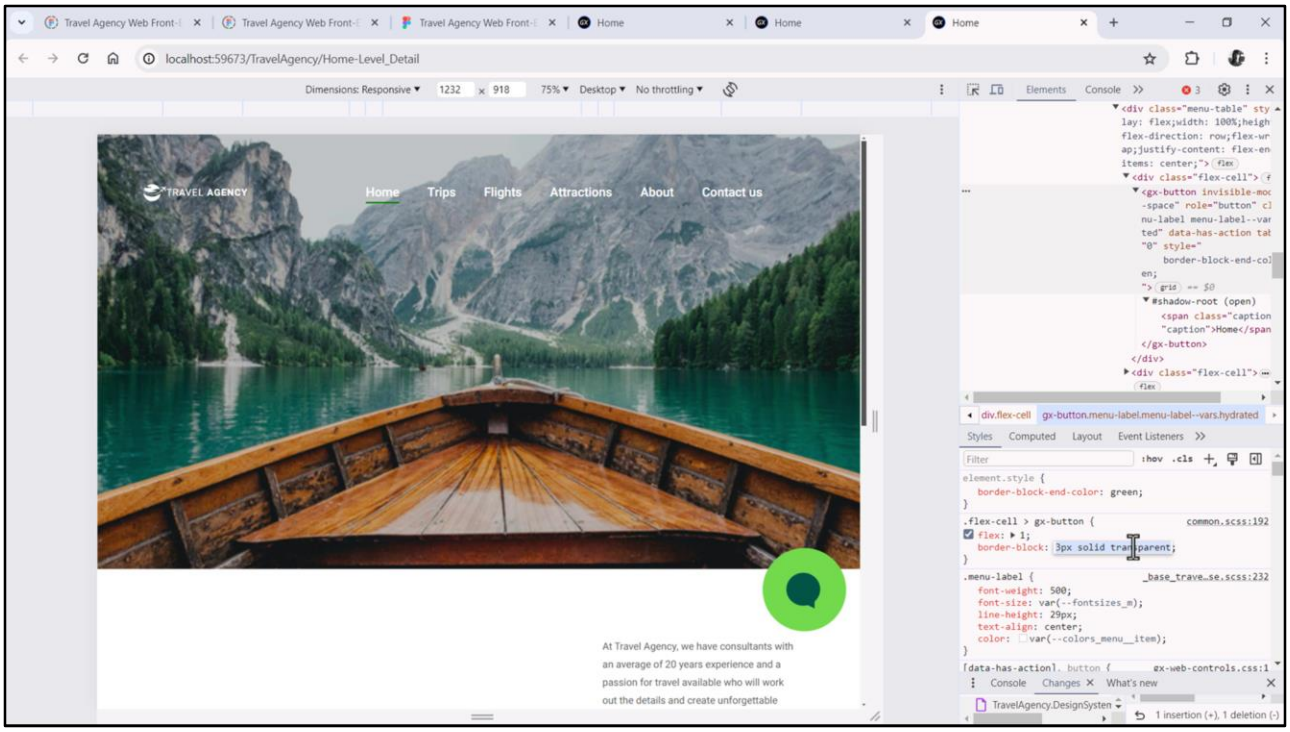
En este dibujo medio desprolijo estoy representando con el botón de la derecha esta situación en la que estoy dejando un borde inferior más grande que el de arriba (que en nuestro caso es 0: el de debajo de 3 píxeles, el de arriba de 0). Y al hacer esto, no me va a quedar centrado el contenido del botón. Ese es el problema.



Cambiar la alineación de los ítems del flex por center no nos soluciona el asunto. Hacer esto lo único que consigue es que el alto del elemento no sea el alto del flex, sino que corresponda al alto del texto, del botón, más padding, border y margin… Así que será de 32 píxeles en lugar de 43. Pero en lo que hace a la alineación por la línea media, estamos en las mismas.



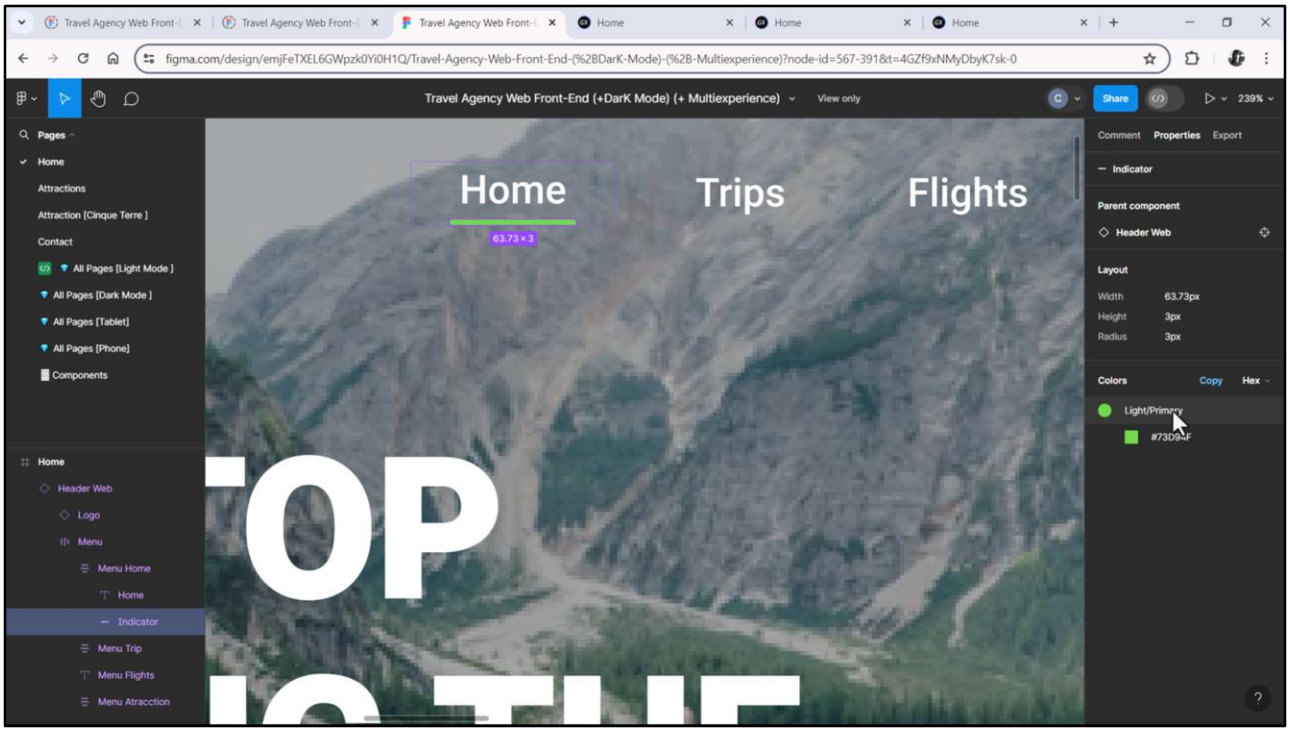
Una solución será darles el mismo espesor a ambos bordes, como en el ejemplo de la izquierda. Allí sí nos quedará centrado verticalmente el texto. Y al de arriba asociarle como color transparente.



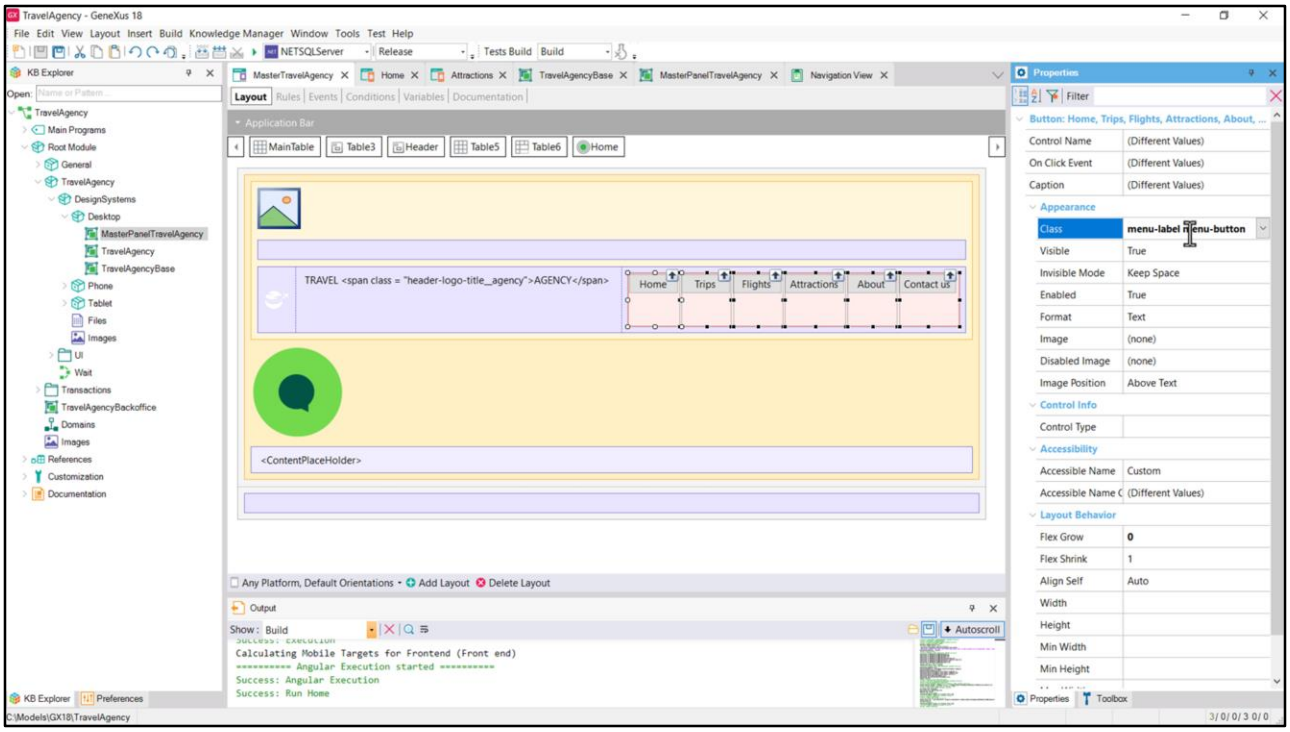
Para probarlo aquí posicionémonos sobre el elemento que corresponde al botón Home. Esta propiedad está comandando el estilo de este elemento particular. En este otro selector tenemos todo lo que aplica a todos los botones de las celdas del flex, así que probemos allí colocar border-block (es decir, valdrá para arriba y abajo), de 3 píxeles, solid, y color tansparente.

Y luego para el elemento que esté seleccionado, no necesitamos repetir esto, sino sólo darle color green. Así que elegimos esta propiedad… y el valor green.

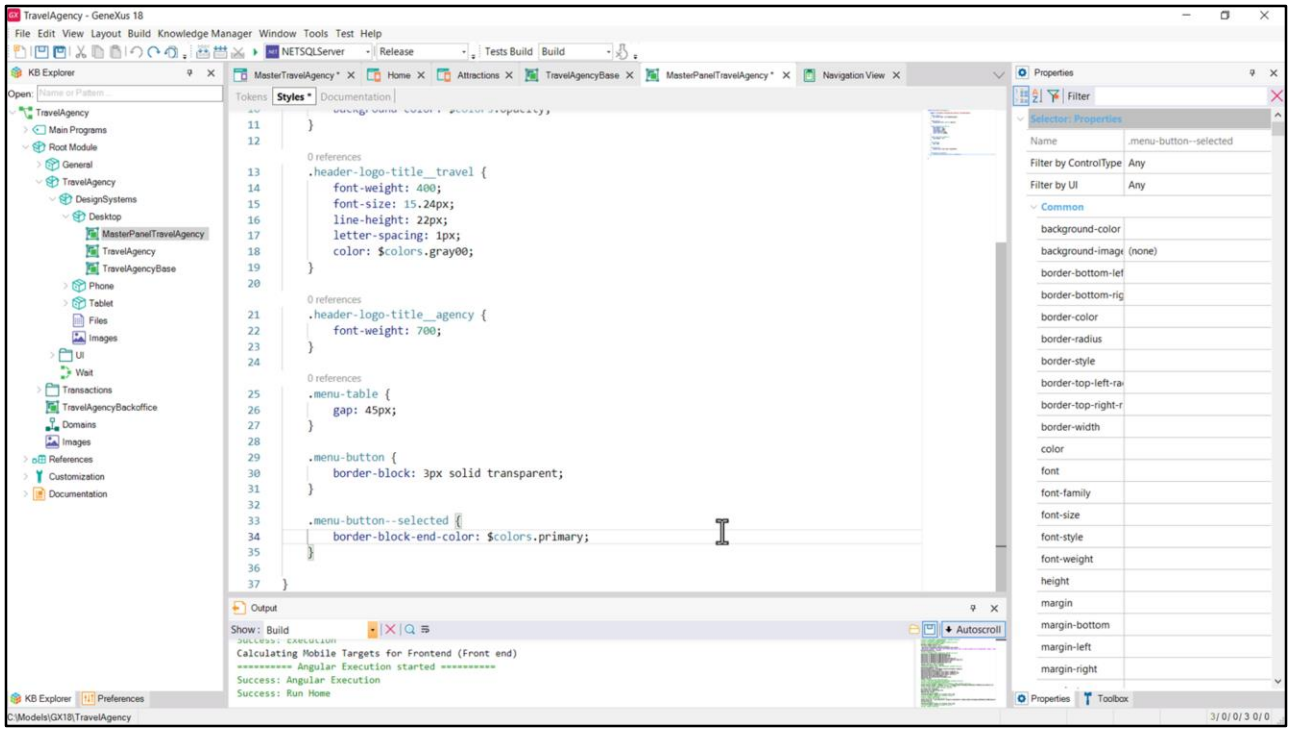
Con esto, tendremos entonces aplicada esta propiedad para todos los botones, y en particular para cada uno, se le cambia el color al borde inferior de transparente a este verde; para cada uno que corresponda, en el momento en que corresponda.



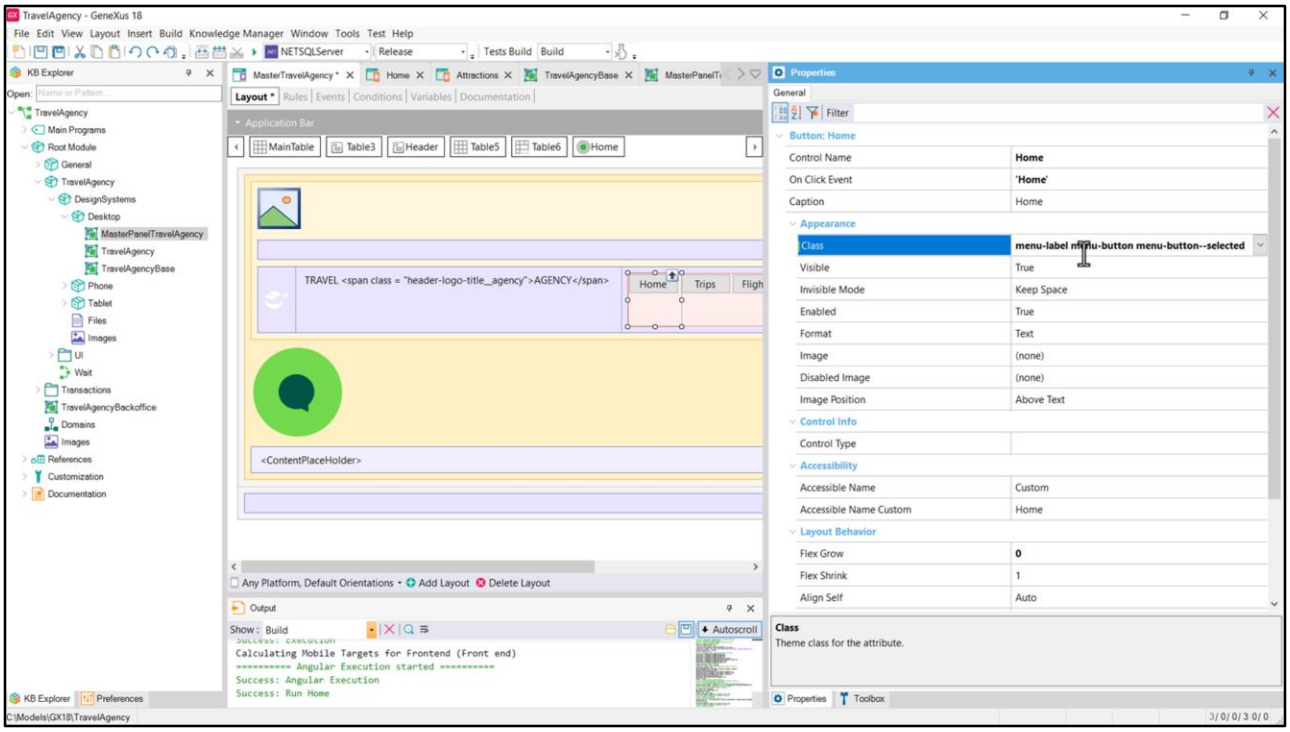
En verdad si vamos a Figma… el color será el primary, y está bien esto de los 3 píxeles. A este radio no le prestaremos atención en esta oportunidad.



Entonces a la clase que tienen todos los botones, que comanda el estilo tipográfico de las etiquetas, agreguemos otra clase, a la que llamaré menu-botón…

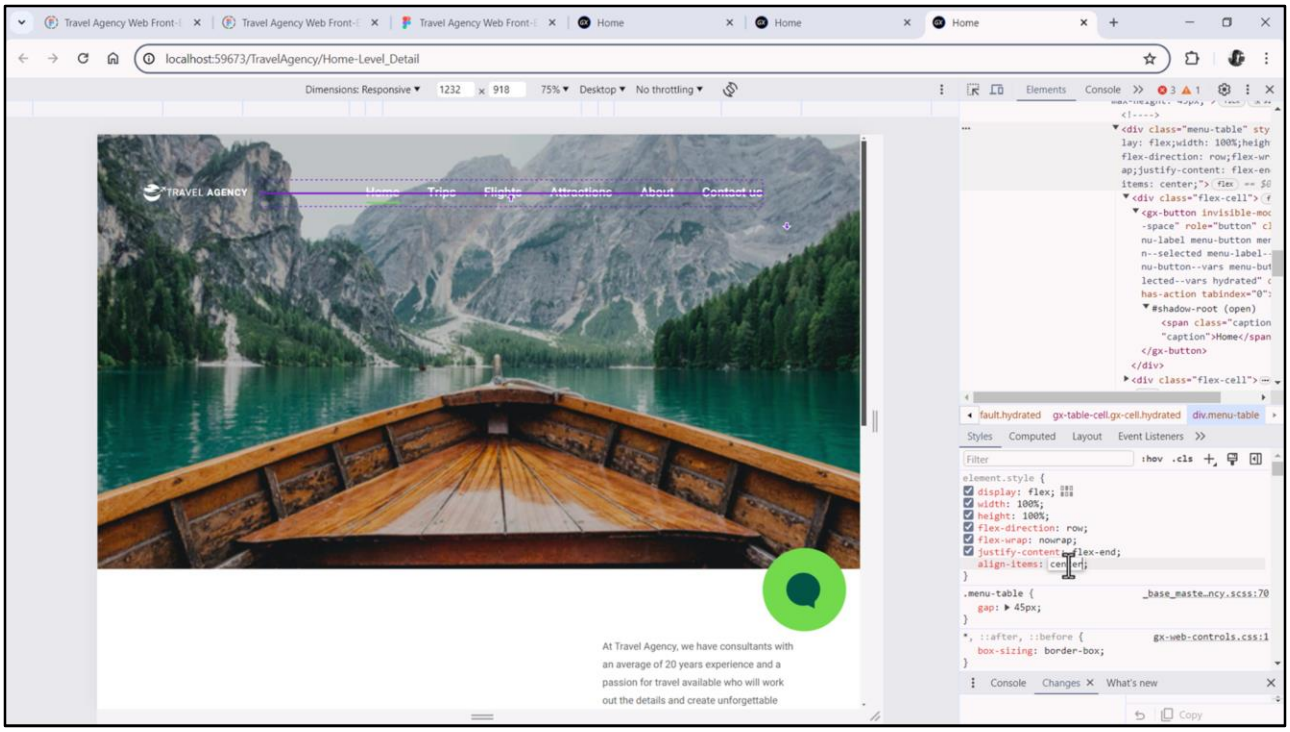


….A la que le asignaré el borde transparente de abajo y arriba… Y luego crearé otra clase para cambiarle el color por el primary al borde de abajo.

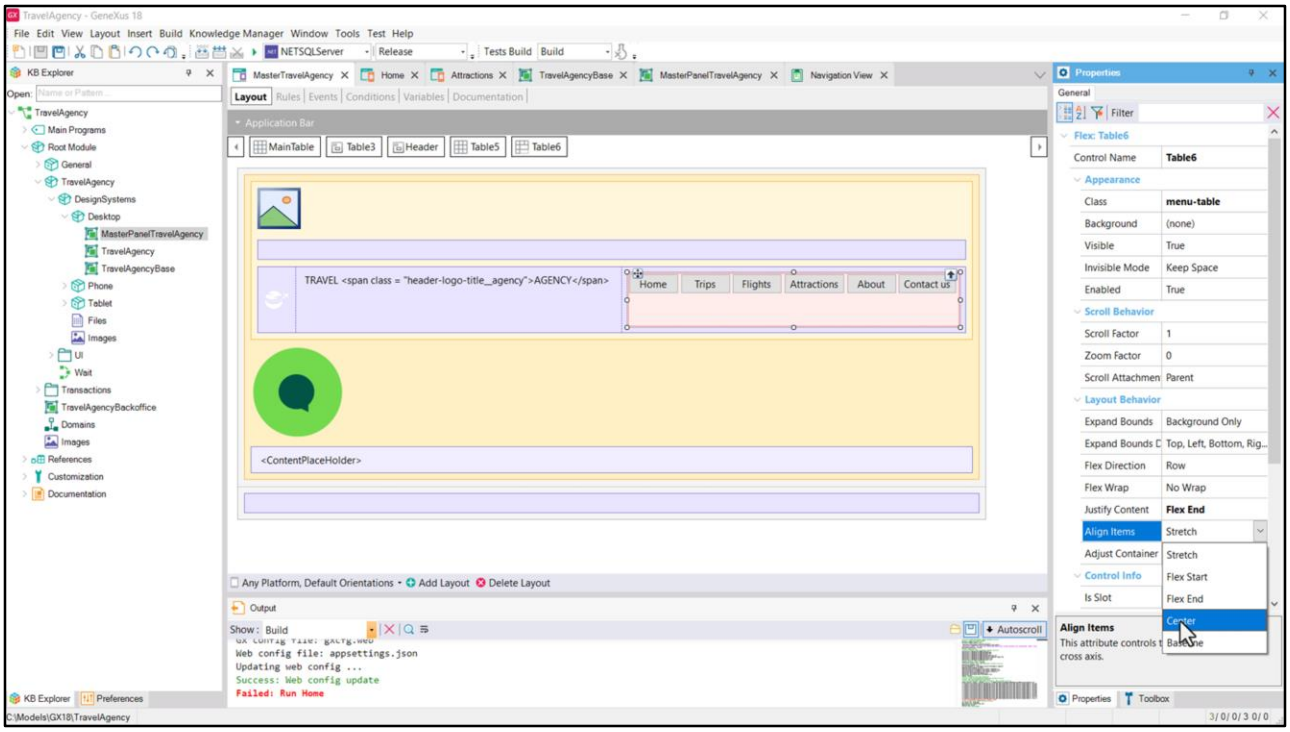


Por ahora se la voy a asignar al botón Home para que veamos el estilo.

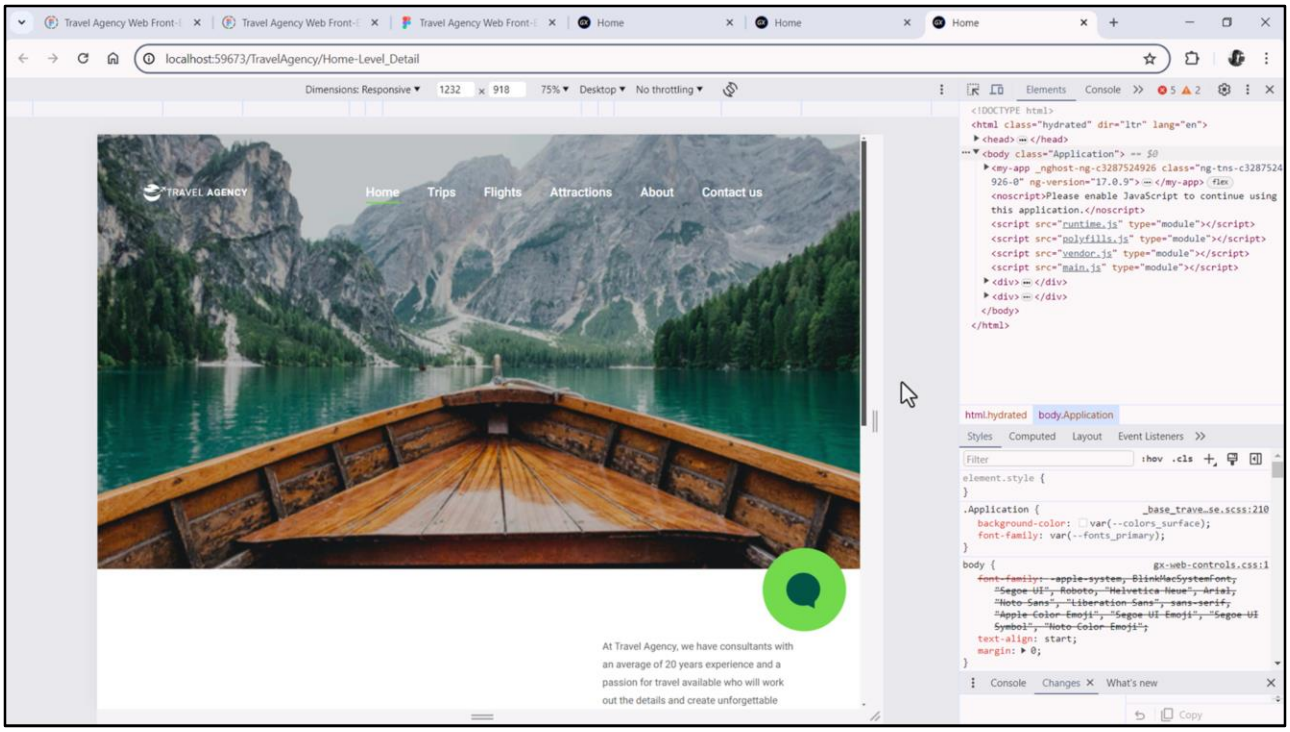
Cuando estudiemos las navegaciones allí veremos cómo cambiarlo dinámicamente de acuerdo a quien se esté cargando en el contentplaceholder.



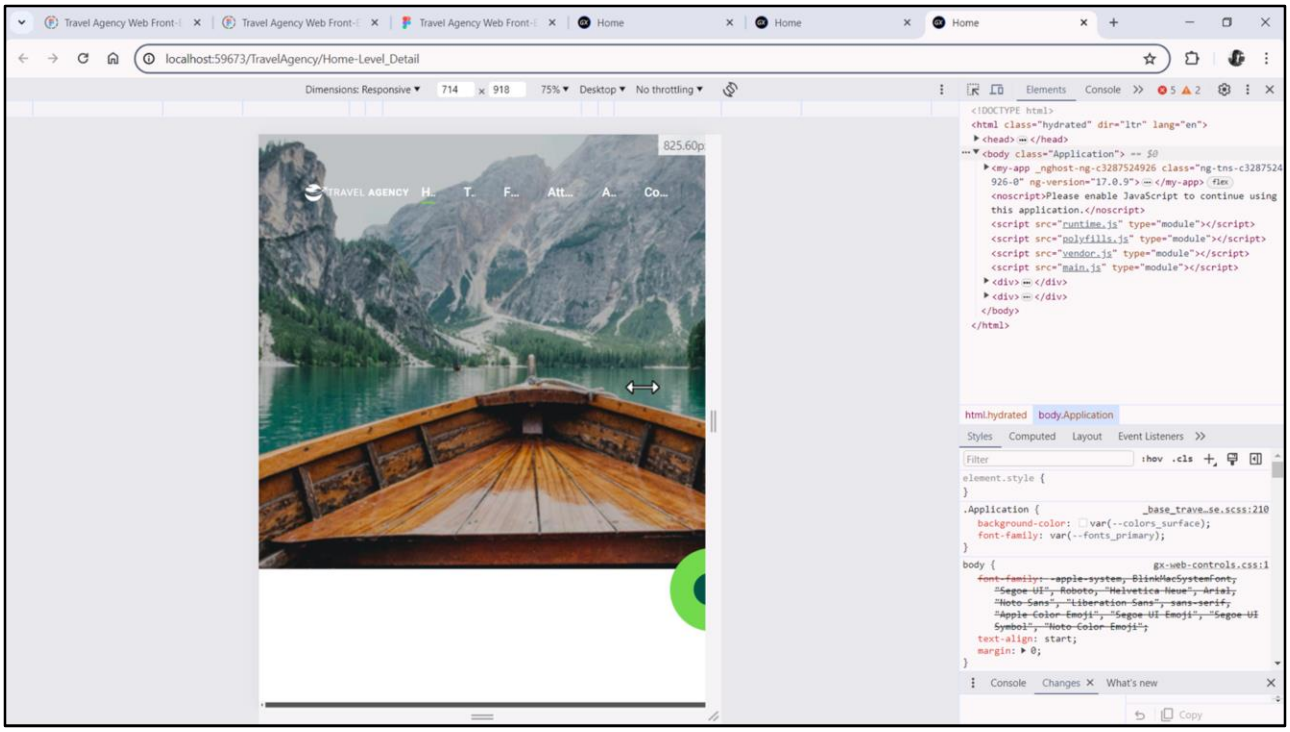
Tal vez pueda parecernos que hay mucha distancia entre el texto y la línea del borde. Es que, recordemos, el botón está ocupando todo el alto de la celda, que está ocupando todo el alto del flex, debido a que le habíamos dejado el valor default a la propiedad align-items, que era stretch. Si lo cambiamos a center conseguimos que el alto sea el mínimo necesario para contener a su contenido más padding, borde y margen. Y además que se centre por la línea media.



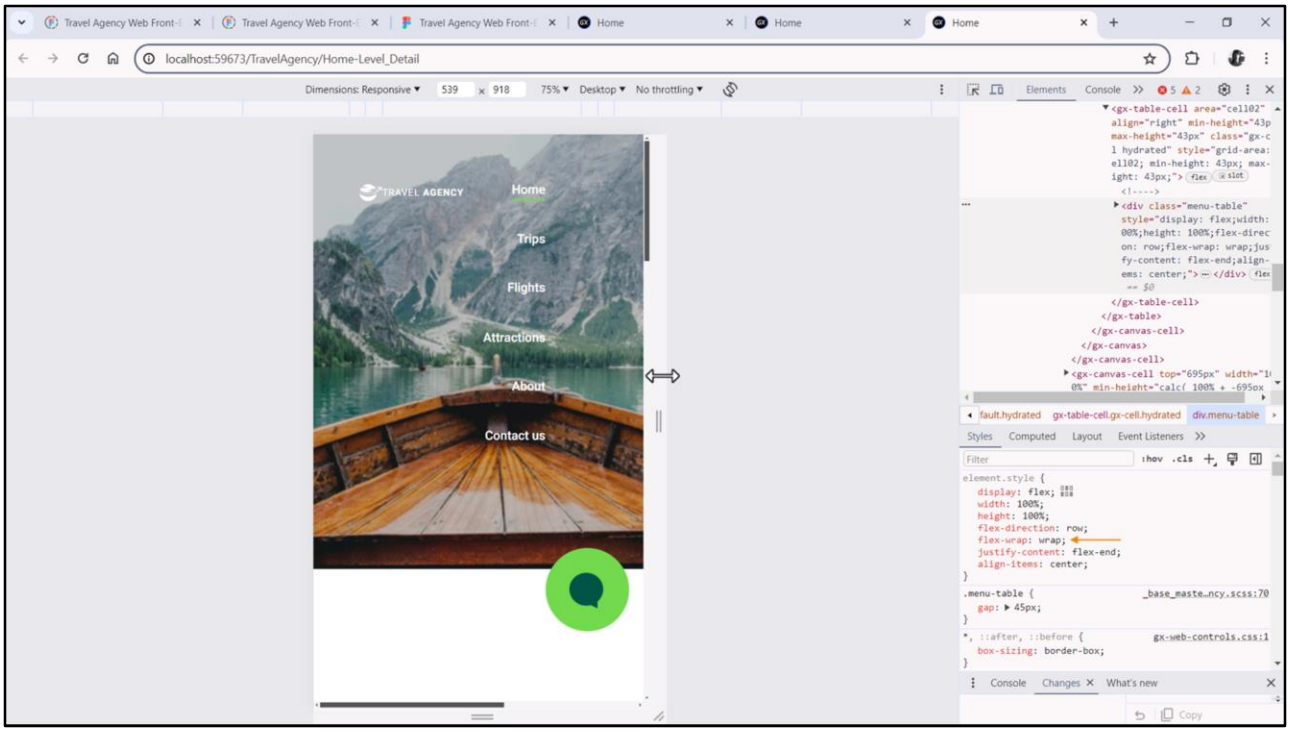
Esto lo conseguimos con la propiedad estática a nivel del flex… (hay un bug en el editor abstracto que hace que no se vean los botones, pero estar están. Es solamente un problema visual. En el editor de GeneXus web ya está corregido)



De hecho si ejecutamos… vemos todo tal cual lo esperamos.

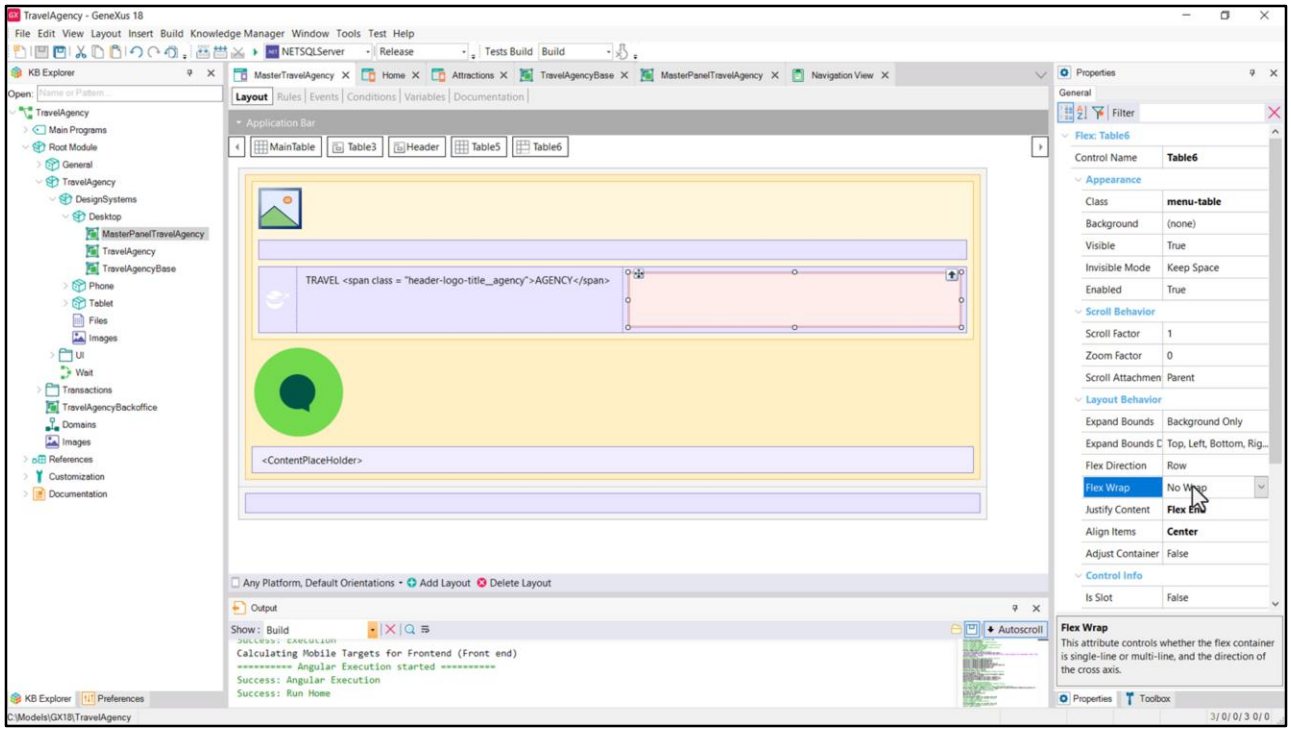


Una última cosa respecto al menú: vean qué sucede si disminuimos el ancho de pantalla. Los botones empiezan a acortarse para entrar en el espacio que tienen, dejando el gap que indicamos, claro.

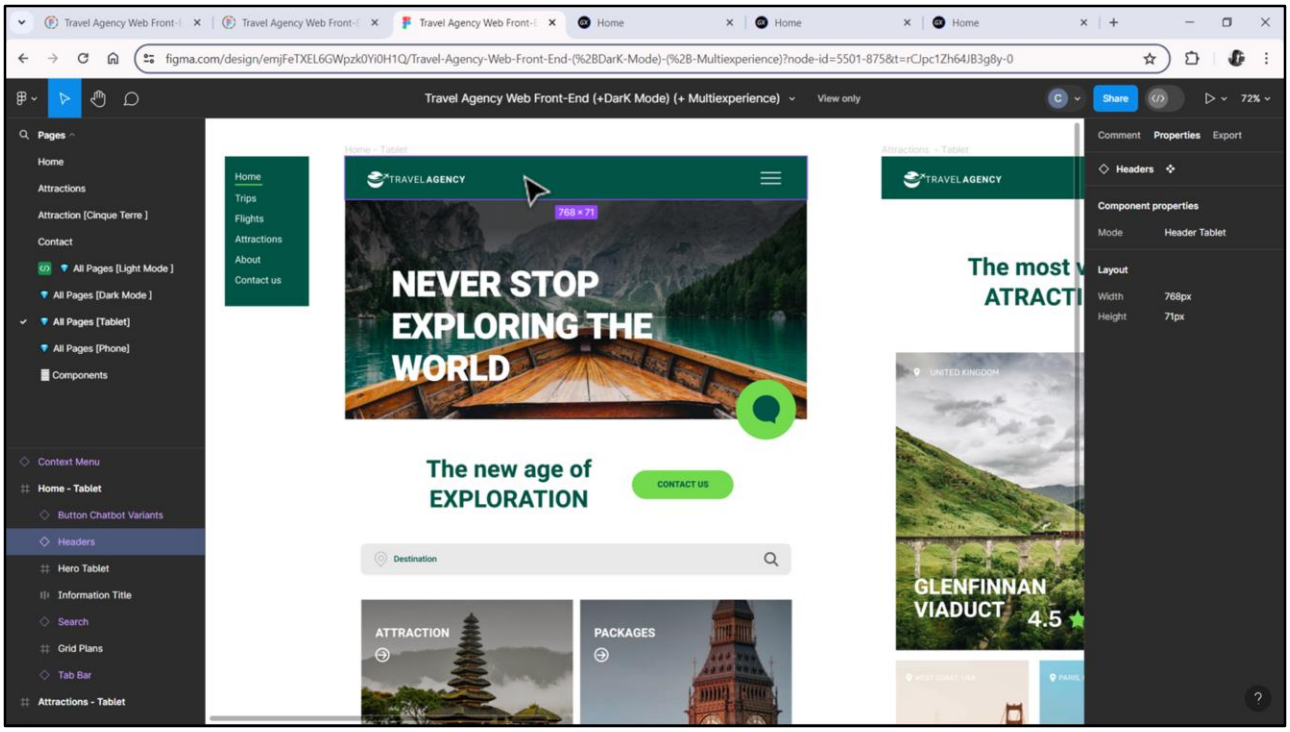


Una de las características que ofrece el contenedor flex, a diferencia de la tabla, es que permite el wrap, de modo tal que los ítems que dejen de entrar en la fila se coloquen en otra.

No nos va a interesar para nuestro caso, pero quería mostrarlo.

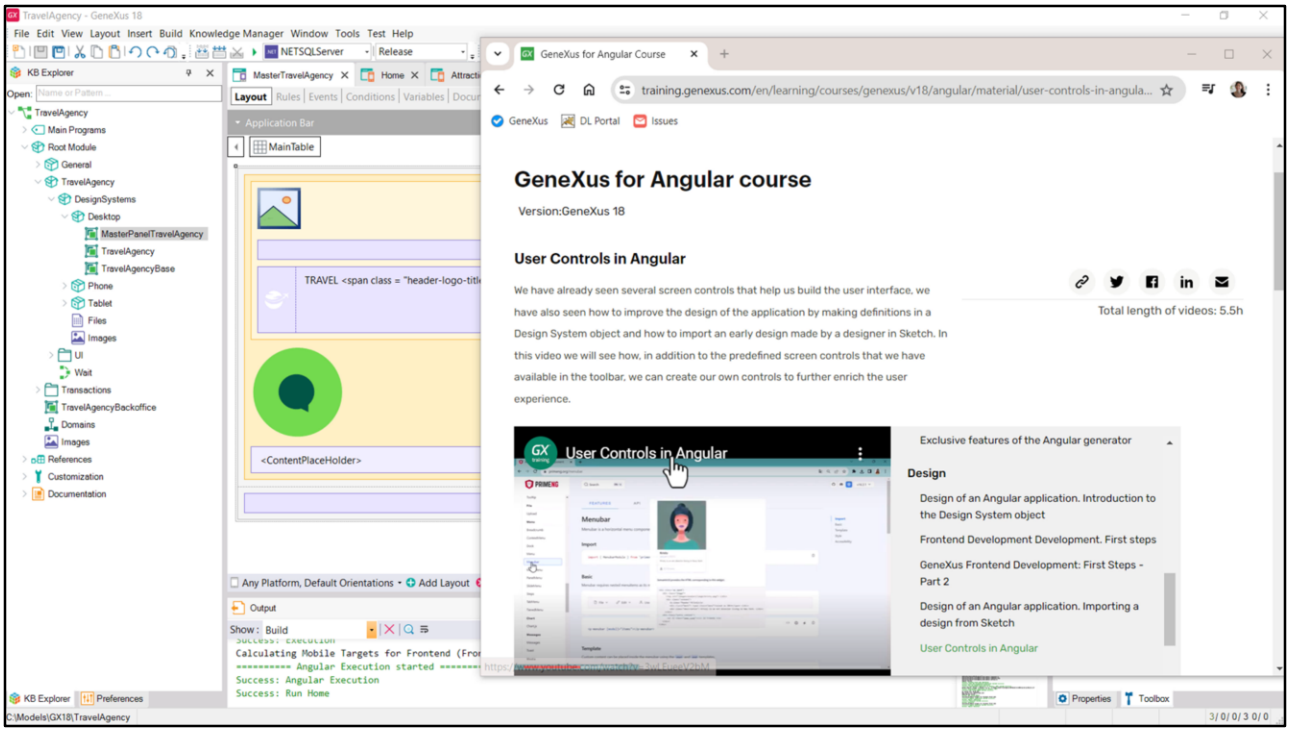


Y vean dónde está la propiedad.



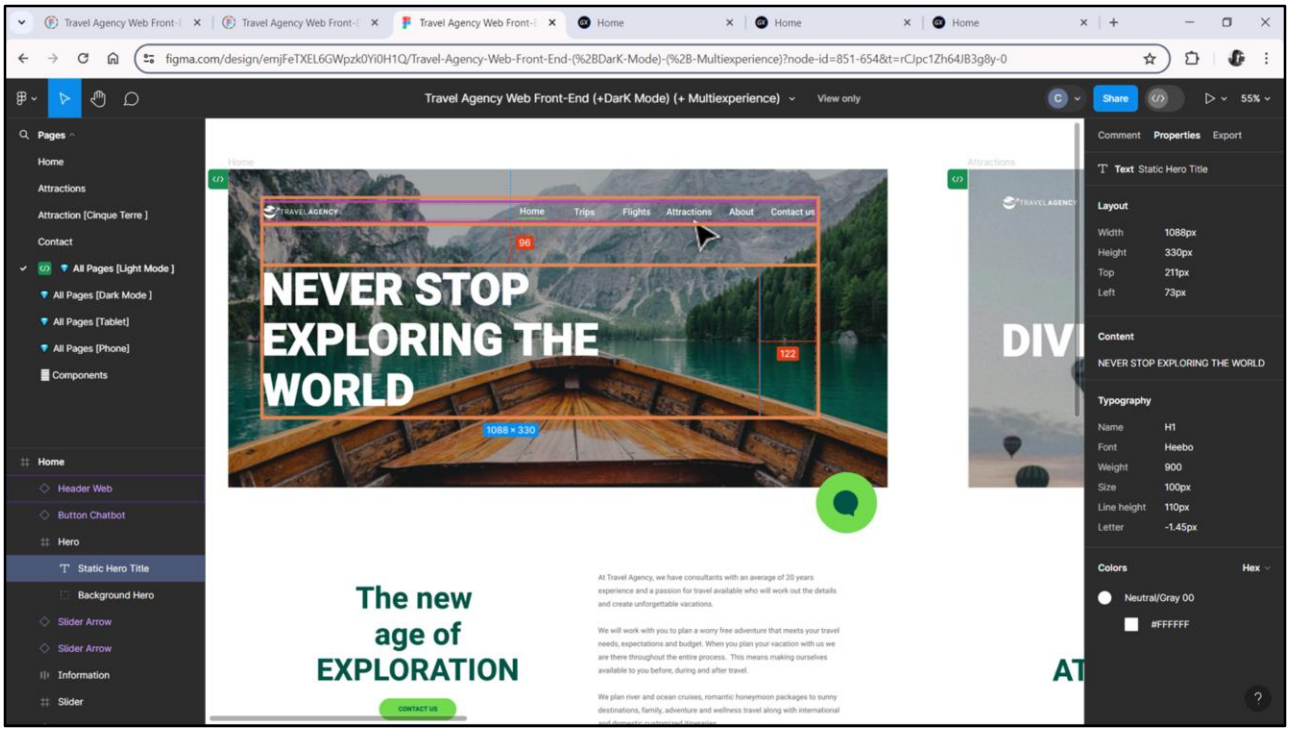
Recordemos que Chechu armó otros diseños para tamaño de pantalla el de Tablet… donde el menú será el clásico hamburguesa, y para tamaño Phone también.

Volveremos sobre estos breakpoints cuando analicemos la multiexperiencia, en el próximo módulo.



Ya aprovecho a comentar que si necesitamos controles más sofisticados que incluyen diseño y comportamiento, podemos incluirlos en GeneXus definiendo **User controls**, es decir, objetos que se llaman de esta manera, user controls, en los que podemos establecer el html e intevenirlo mínimamente para que pueda utilizarse en cualquier objeto GeneXus con infertaz. Es decir, lo incorporamos a la KB y ya queda disponible para ser utilizado.

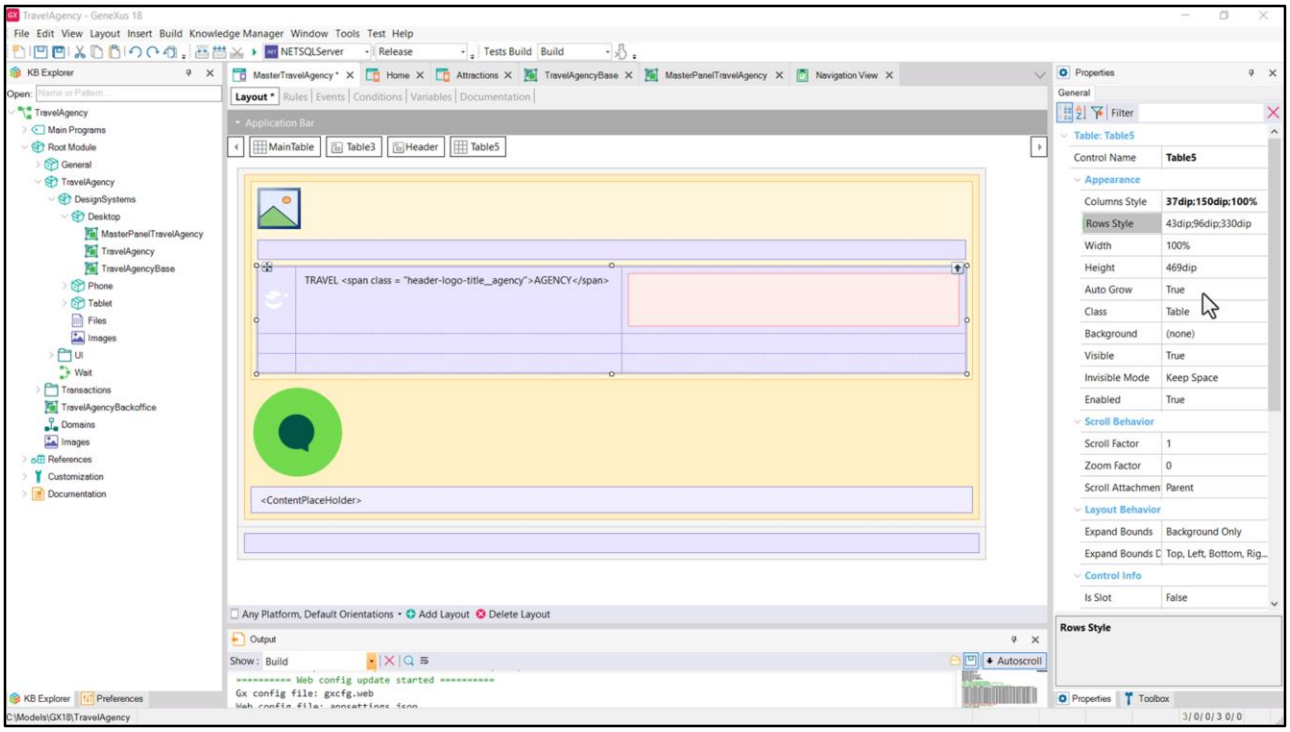
Esto valdrá para la plataforma Web (tanto Angular como Net y Java).



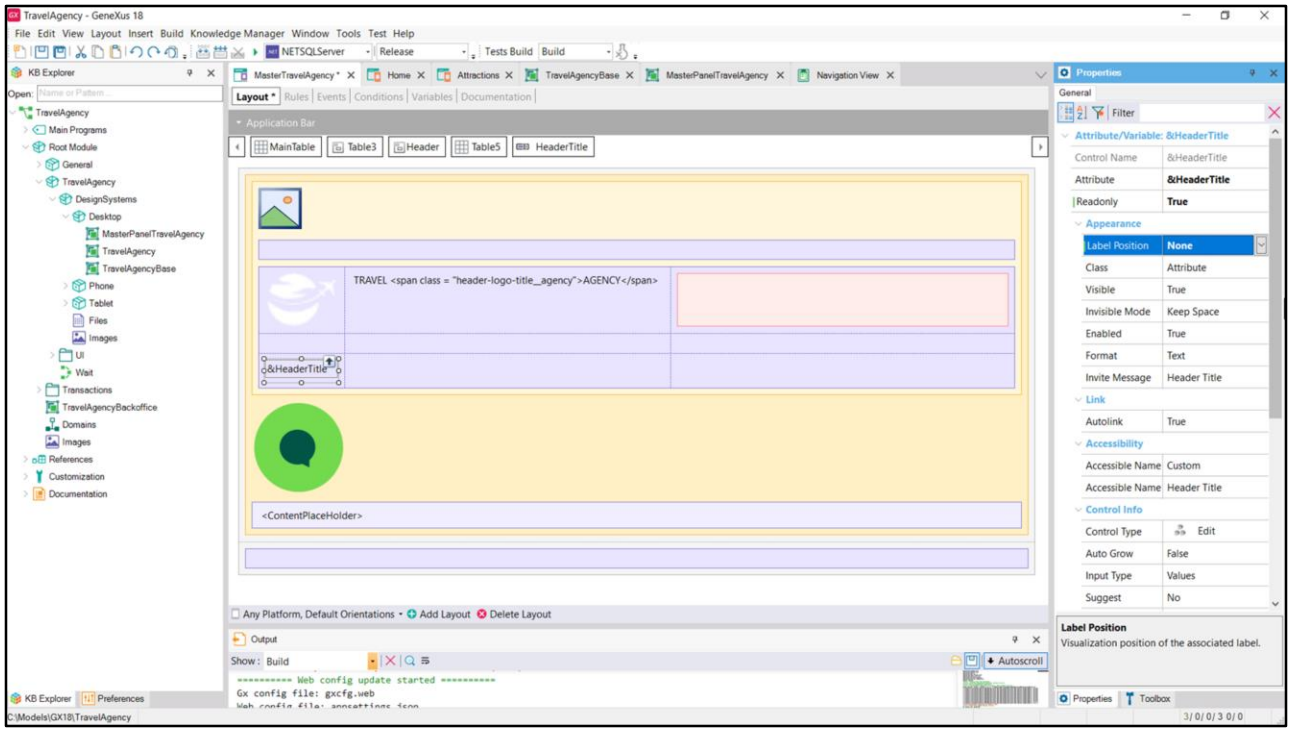
Agreguemos ahora rápidamente lo que nos está faltando al Header para terminar de implementarlo: el texto que sobresale sobre la imagen. Recordemos que su estilo tipográfico era el H1. Y copiemos su contenido al portapapeles.

Se alinea a la izquierda junto con el ícono de travel agency, y es por eso que habíamos pensado en utilizar la tabla. Ya extraigamos las medidas: el alto de la fila será de 330 píxeles, y estará de la del menú a esta otra distancia, a 96 píxeles…

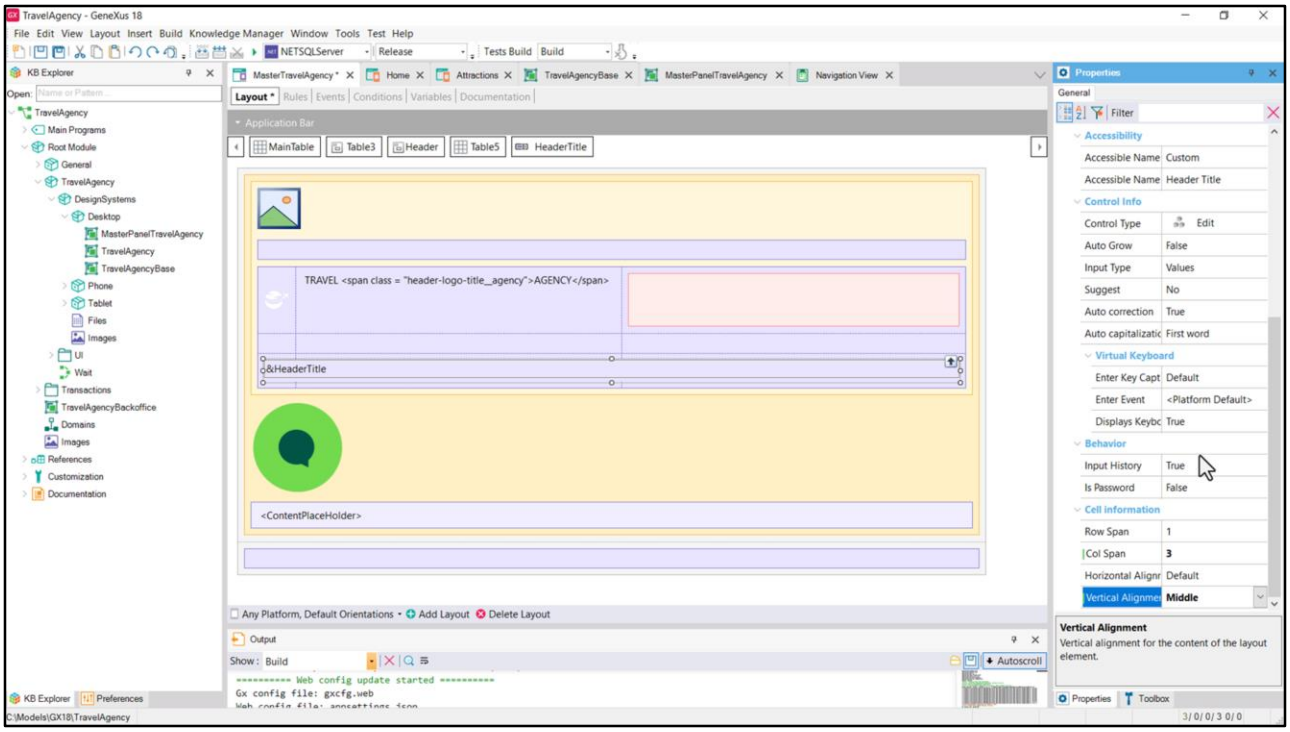
Claramente elegiremos implementar el texto no como un textblock sino como una variable, dado que, justamente, variará su contenido entre las pantallas.



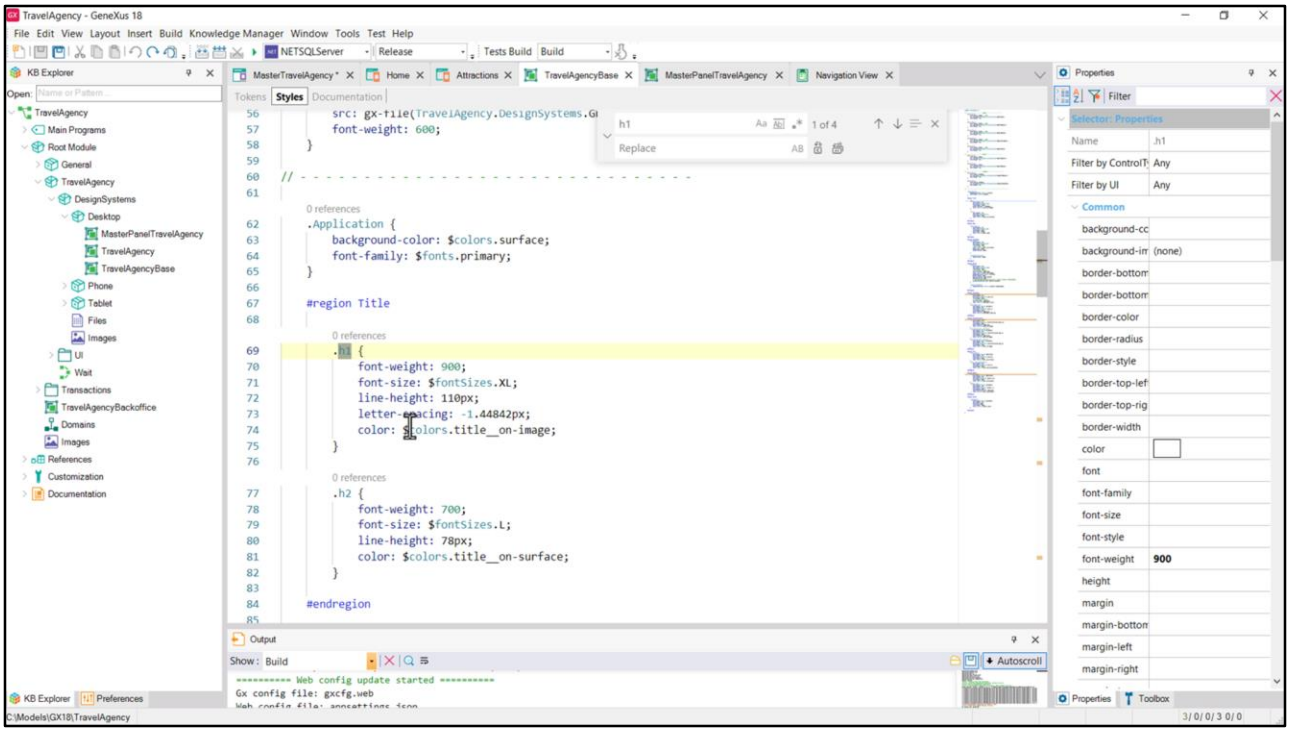
Insertamos entonces dos filas a la tabla. La segunda será de 96 píxeles y la tercera de 330.



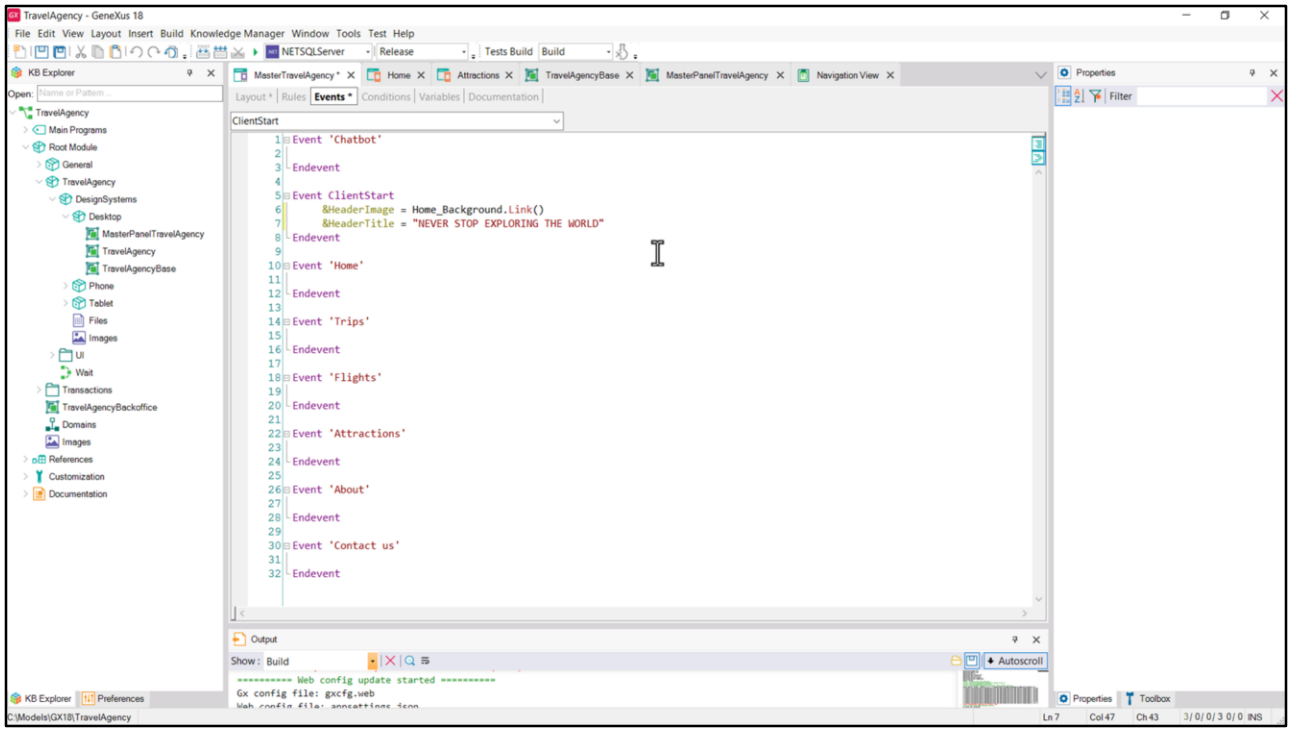
En la tercera, insertamos un control variable… de nombre HeaderTitle…. Y tipo de datos varchar.
Será readonly y sin mostrar la etiqueta.



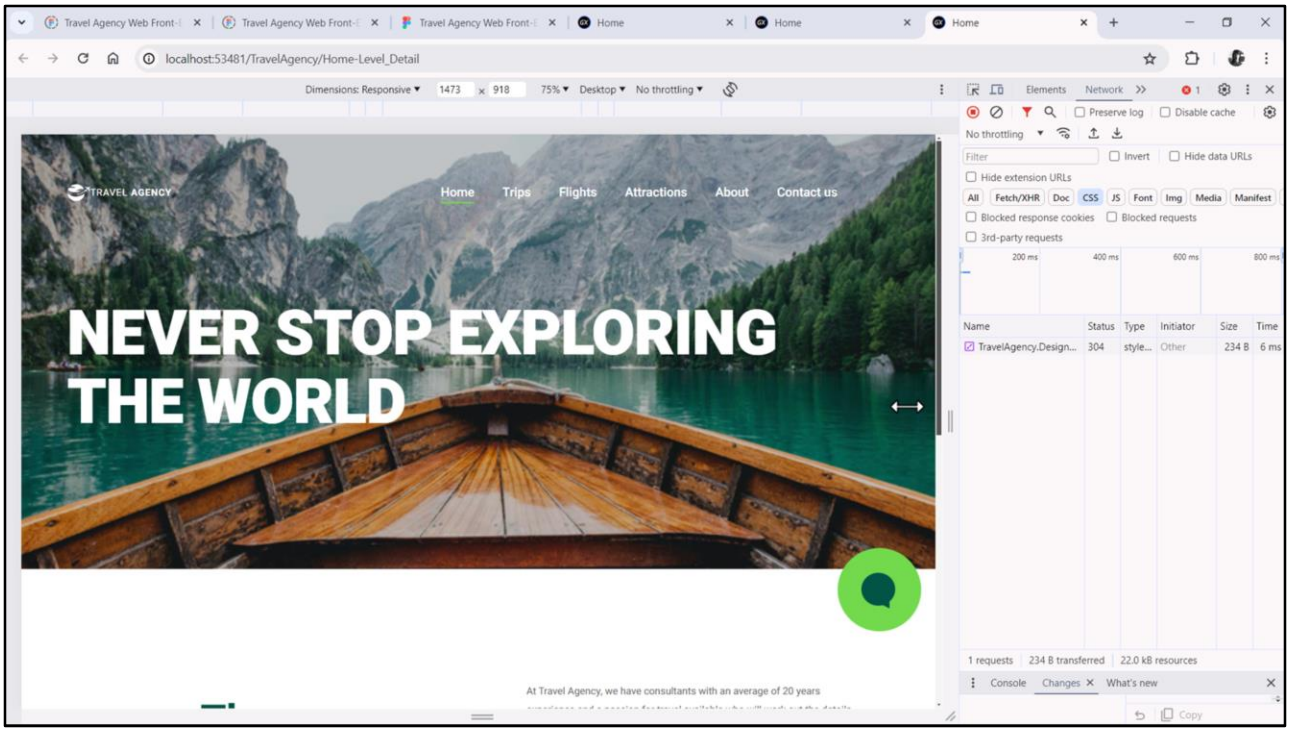
Pero además querremos que se expanda para ocupar las tres columnas. Y querremos que su contenido esté alineado verticalmente por el medio.



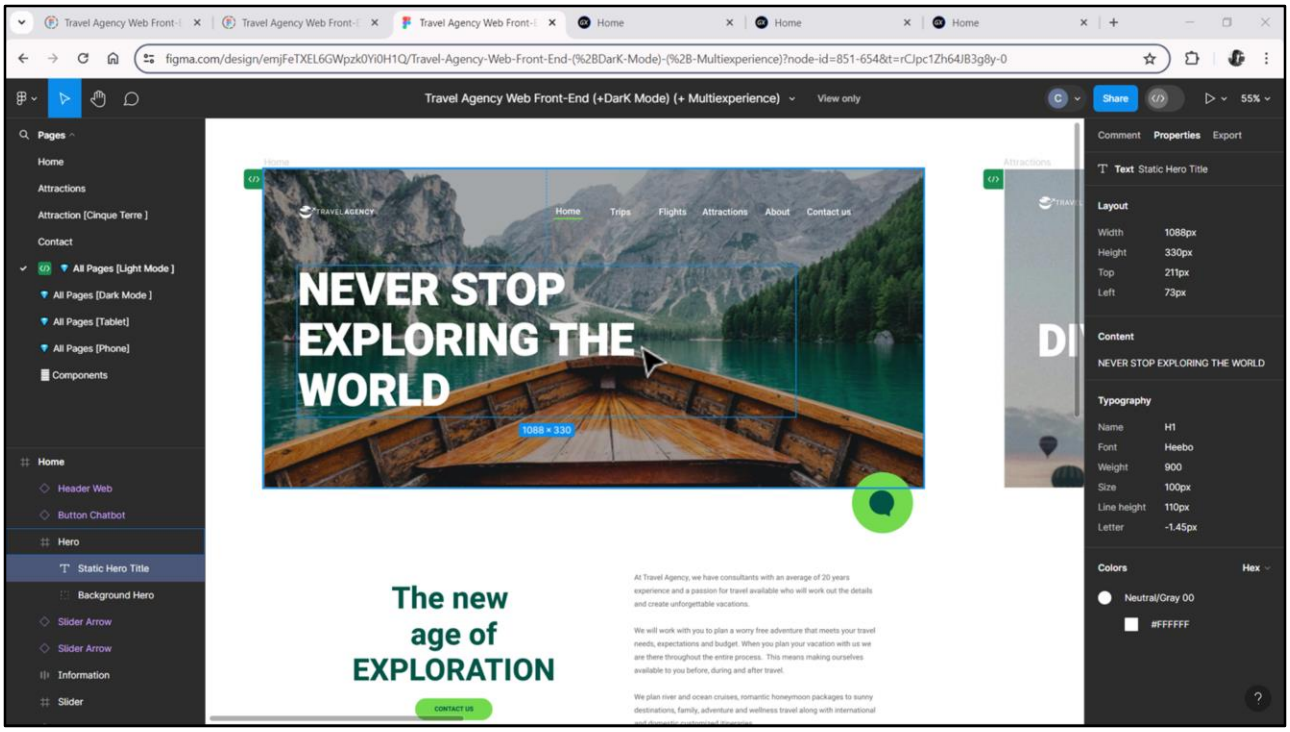
Por otro lado su clase será la que habíamos llamado h1 cuando introdujimos en el módulo de preparación todas las clases para la tipografía.



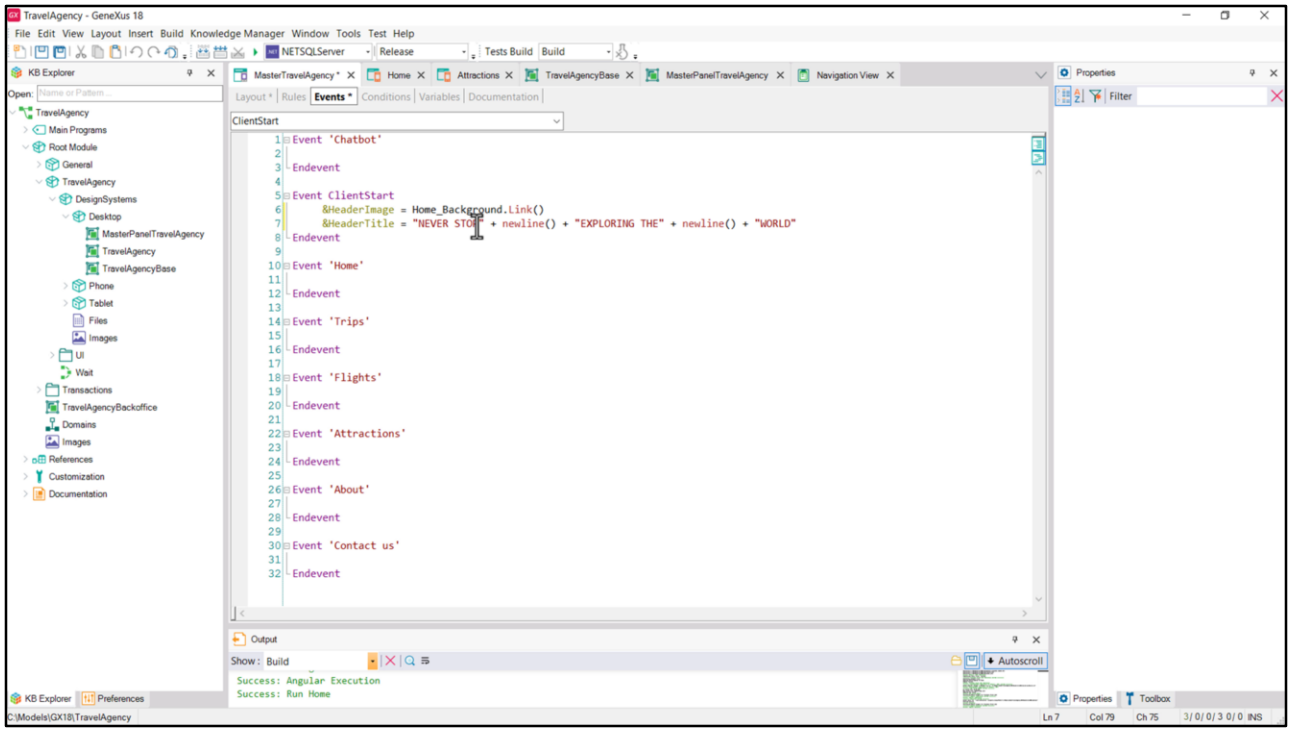
En el evento ClientStart asignémosle valor. Por ahora lo dejaremos así, estático.



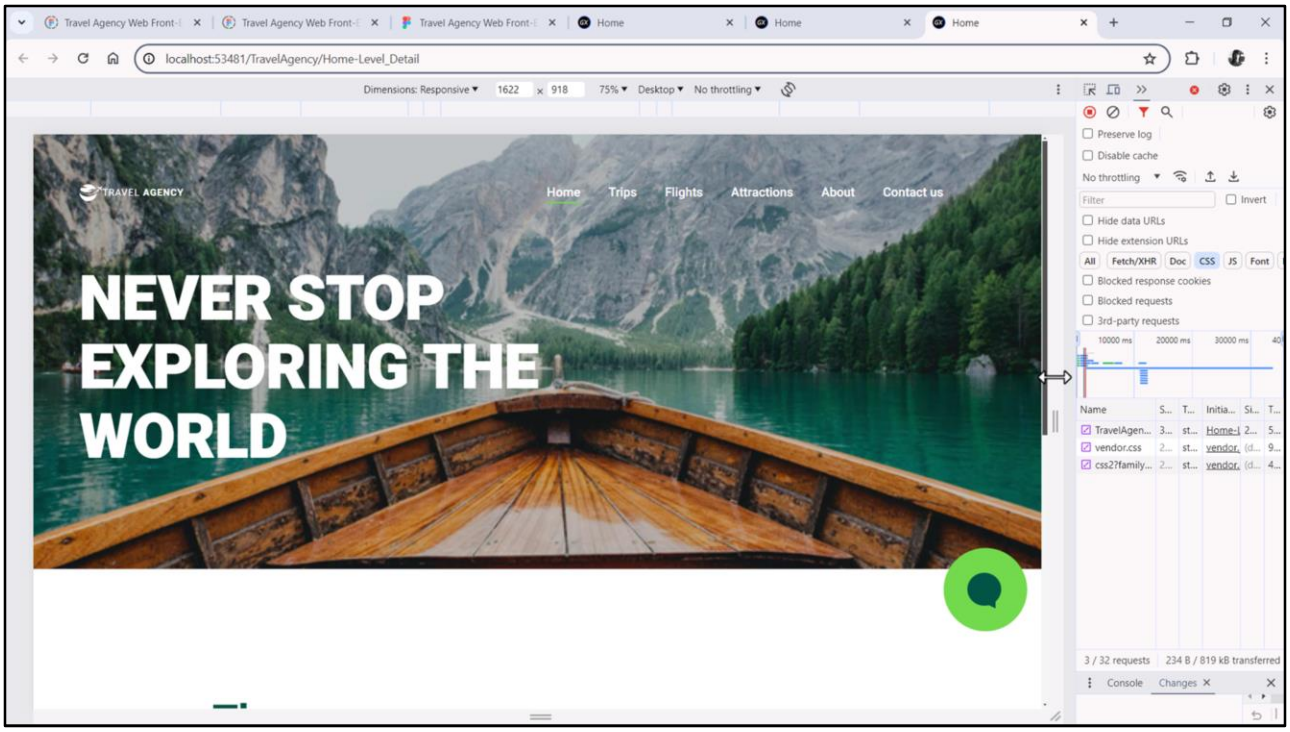
Probemos. Si queremos que siempre quede en exactamente 3 líneas…



…como acá, y exactamente estas…



… alcanzará con agregar newlines().



Bien, ahora estamos listos para implementar las navegaciones, logrando cambiar el Header de acuerdo a quien se esté cargando en el contentplaceholder en cada oportunidad.

Bueno, continuamos en el próximo video.


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