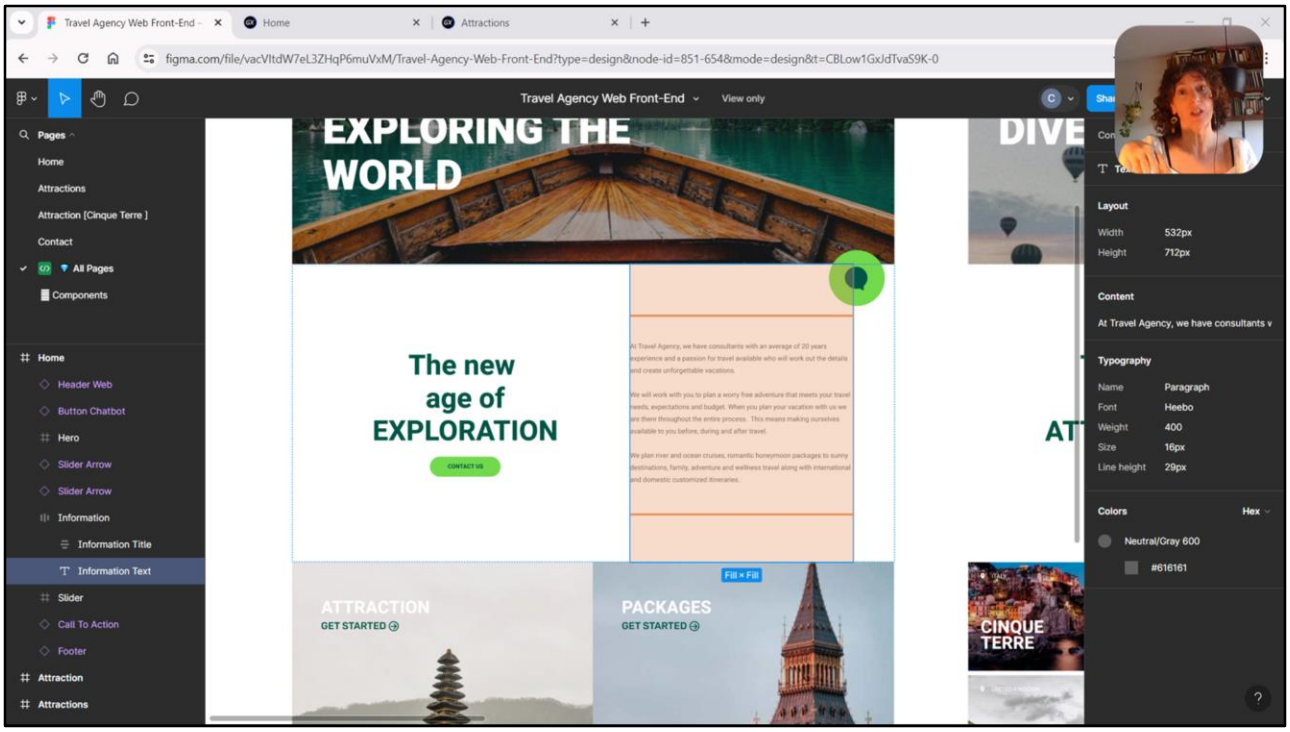
GeneXus by Globant

First Layout in GeneXus. Style (cont)

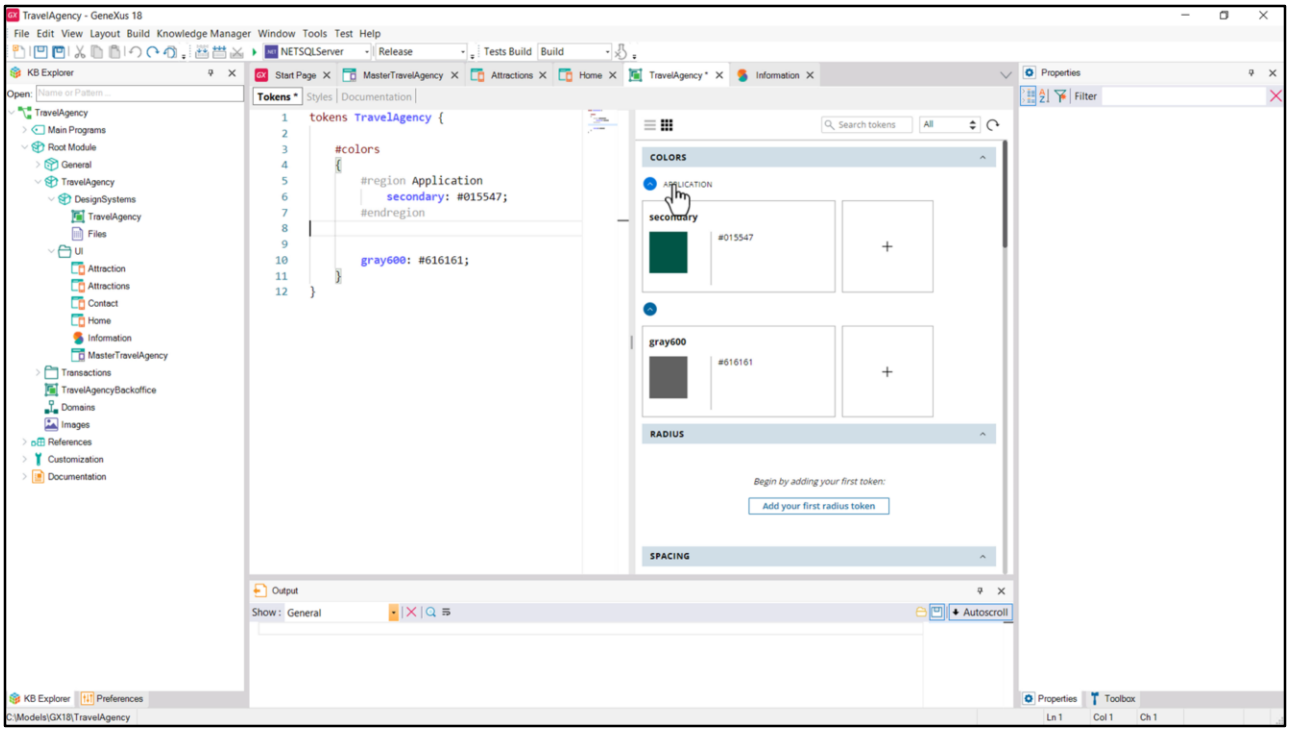
Spacing, logical properties, composite classes.

Cecilia Fernández



En este video daremos estilo al parágrafo de nuestro stencil y veremos algunas particularidades que no aparecían en el caso del video anterior, en el del textblock. Por ejemplo, veremos cómo setearle un margen superior e inferior al elemento, veremos las diferencias entre propiedades físicas, es decir, aquellas que tienen que ver con las coordenadas arriba, abajo, izquierda, derecha, versus las propiedades lógicas, es decir, aquellas que se establecen respecto al inicio y fin del elemento, entre otras cosas.

Dejaremos para el video siguiente el análisis del estilo del botón.



En principio, para dar estilo al párrafo vamos a empezar por hacer lo mismo que hicimos para el texto de la izquierda.

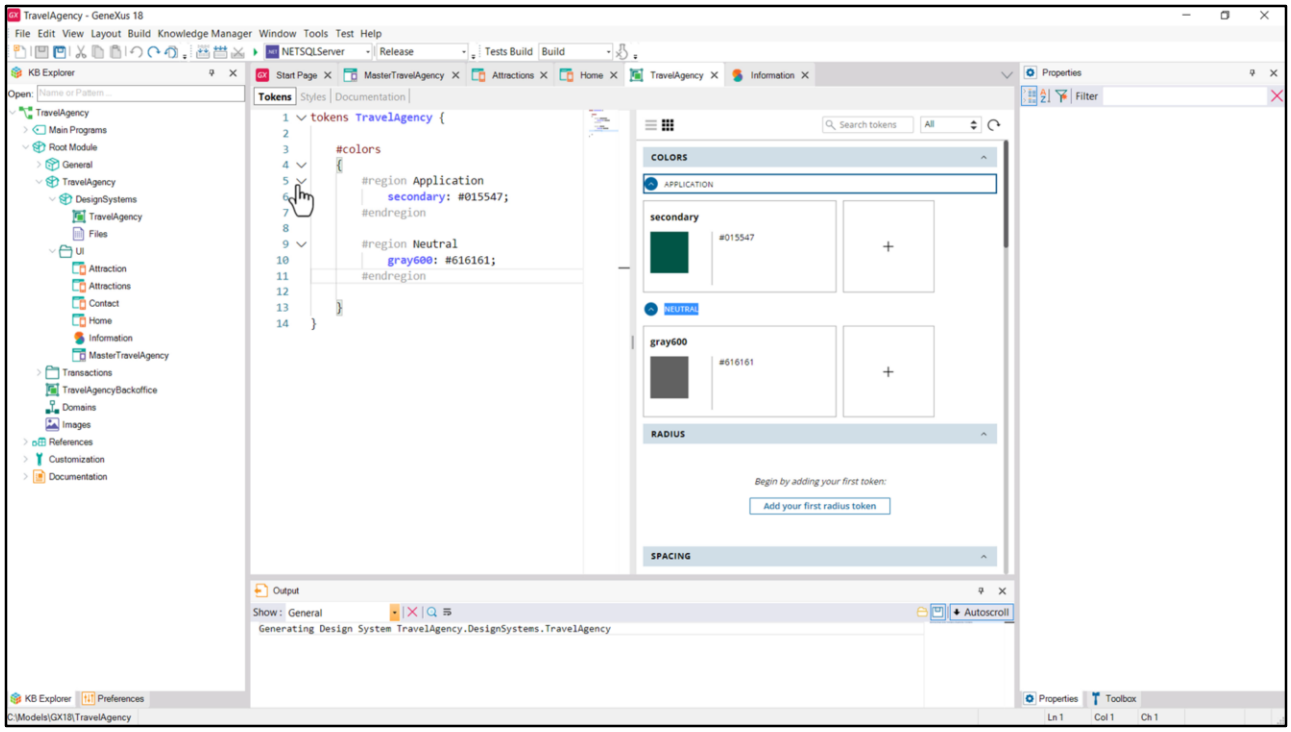
Entre sus propiedades, vemos en las tipográficas el estilo Paragraph. Y el color es este tipo de gris. Voy a copiar… y si pego… vemos que aparece esa propiedad… lo que me voy a copiar es el valor…

…porque voy a empezar por crear ese token, gray600, en la pestaña de tokens de mi DSO. Y allí lo copio. Vemos que está apareciendo en el editor de la derecha… podría modificarlo desde aquí, si quisiera, en lugar de hacerlo desde aquí.

Y voy a empezar a dar un orden dentro de la información, definiendo aquí dentro lo que se conoce como regiones, que son bloques en el DSO que solo sirven a los efectos de organizar internamente la información. No tienen ninguna otra función.

Entonces podemos definir una región… con numeral… ahí le damos un nombre a la región, le vamos a llamar Application, por ejemplo, a la región que va a contener los tokens de color de la aplicación. Aquí está el Secondary, pero luego tendremos también el Primary, y demás. Cuidado que acá me quedó un signo de más… ahí está.

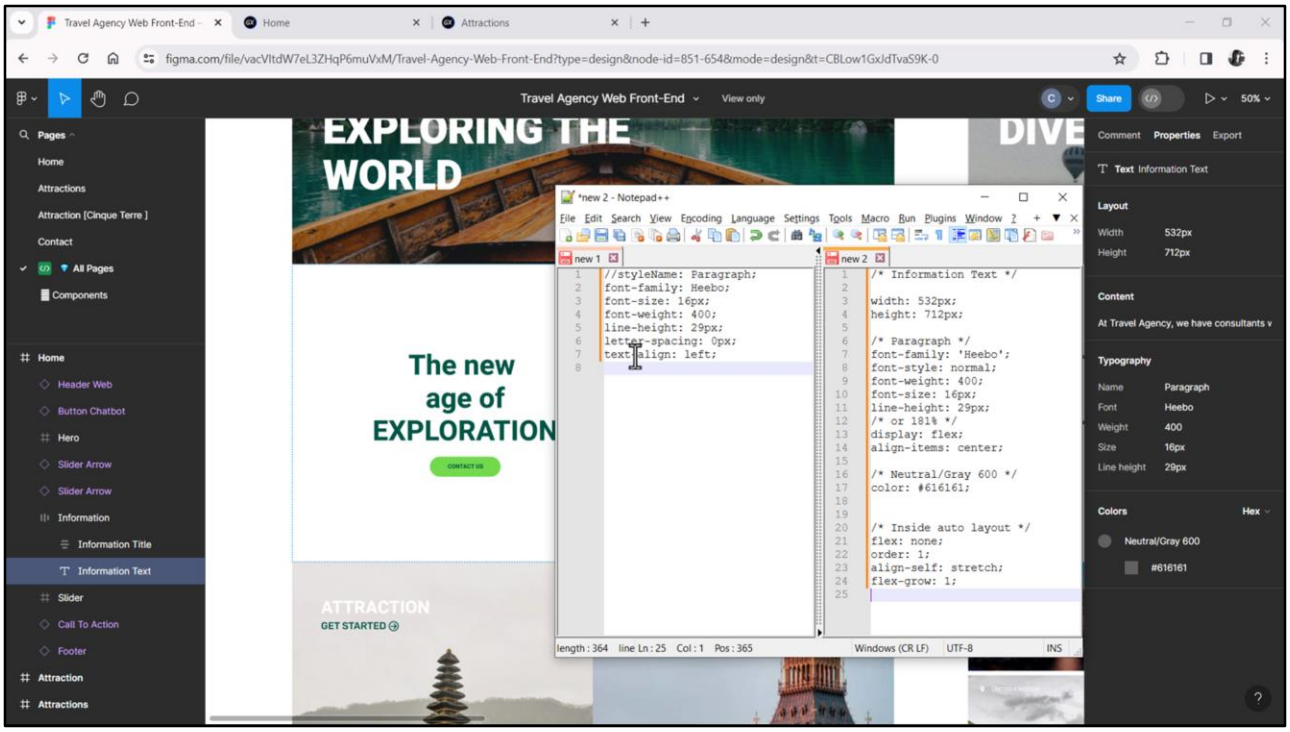
Vemos que al haber agregado esa región ya me está apareciendo, en el editor de colores de la derecha, en el gráfico, me está apareciendo separado lo que es esta región de esta otra que todavía no le pusimos ningún nombre porque es la que quedó por fuera.



Podríamos dejarla así, sin colocarlo dentro de ninguna región, o puedo definir también una región, a la que le voy a llamar Neutral para los colores neutrales. Y allí adentro vamos a colocar este gris, porque evidentemente luego habrá otros grises.

Bien, y ahí vemos que podemos colapsar… cuando esto crezca va a tener más sentido. Pero ya lo dejamos de esta manera para ir organizando entonces la solapa de Tokens de esta forma. Vamos a grabar.

Acá también… vemos que aparecen estas posibilidades de colapsar y expandir la región. Y lo mismo va a valer para la pestaña de estilos. Luego lo veremos.



Ahora vayamos en busca de las propiedades CSS del parágrafo… si las copiamos de acá directamente… parecerían ser exactamente las propiedades CSS del estilo de Chechu.

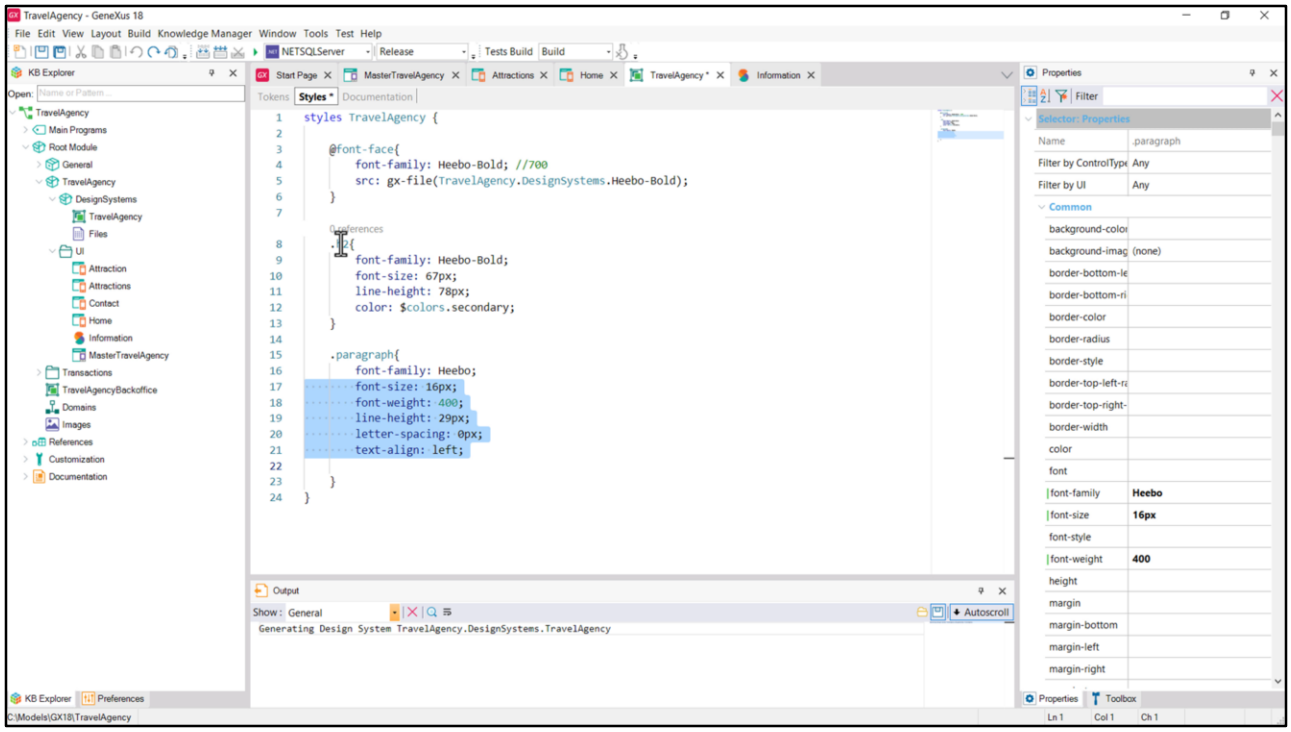
Veamos la diferencia con pedir explícitamente las propiedades como código CSS…

Vemos que hay algunas que están apareciendo acá, como letter-spacing o text-align y que no están apareciendo entre las propiedades CSS listadas.

Presumiblemente se deba a que son default: si no se escriben van a asumir estos valores.

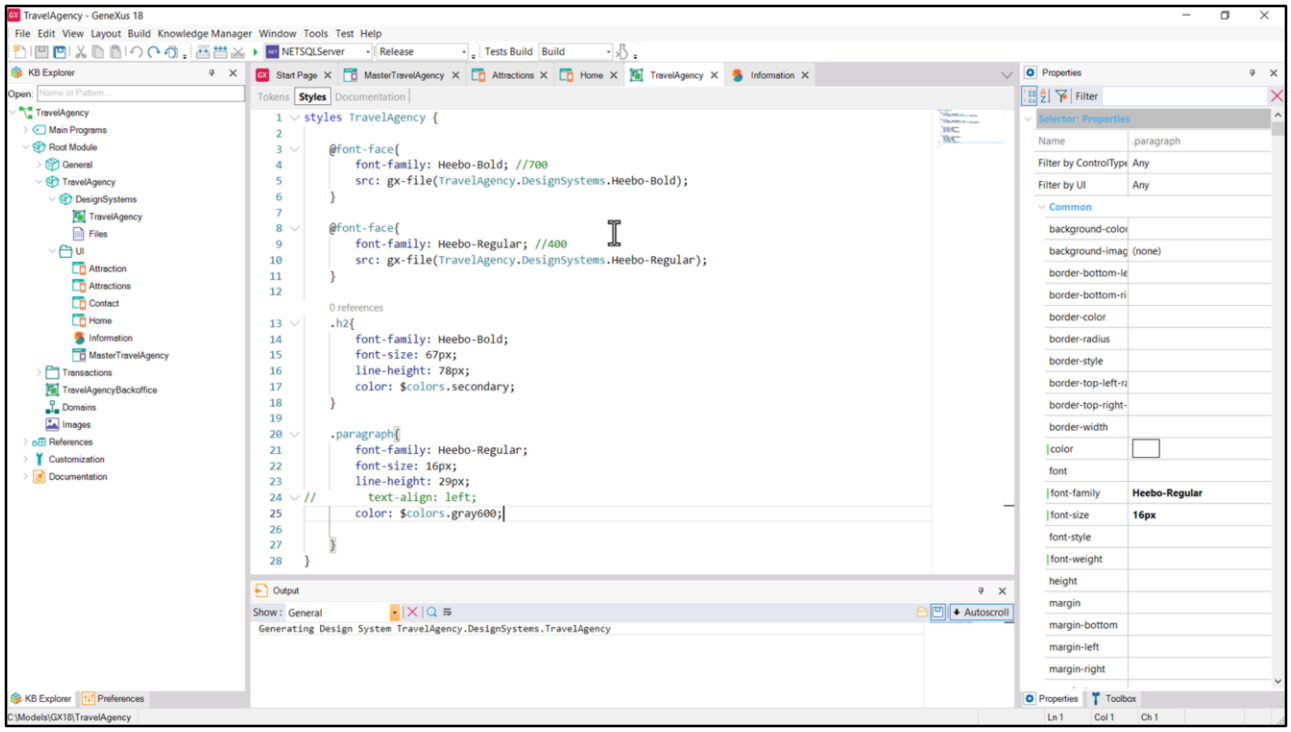
Por otro lado, acá está la font-style normal que aquí no, pero que no está evidentemente porque está contemplada dentro de la familia de fuentes y el peso.

Copiemos estas de aquí, entonces, y vayamos a nuestro DSO…



Y en la solapa de estilos seleccionemos la clase que utilizaremos, a la que llamaremos igual que el estilo de Chechu: “paragraph”. Copiemos las propiedades.

El uso de minúsculas es para respetar la convención de estilos de CSS BEM, que entre otras cosas nos dice que todas las clases deberían ir en minúsculas. Hay que tener en cuenta que todo lo que escribimos en un DSO (al igual que lo que sucede con html en conjunción con CSS) es case sensitive, por ende no será lo mismo una clase paragraph que empieza con minúscula y una Paragraph que empieza con mayúscula. Hay que tener cuidado con eso.



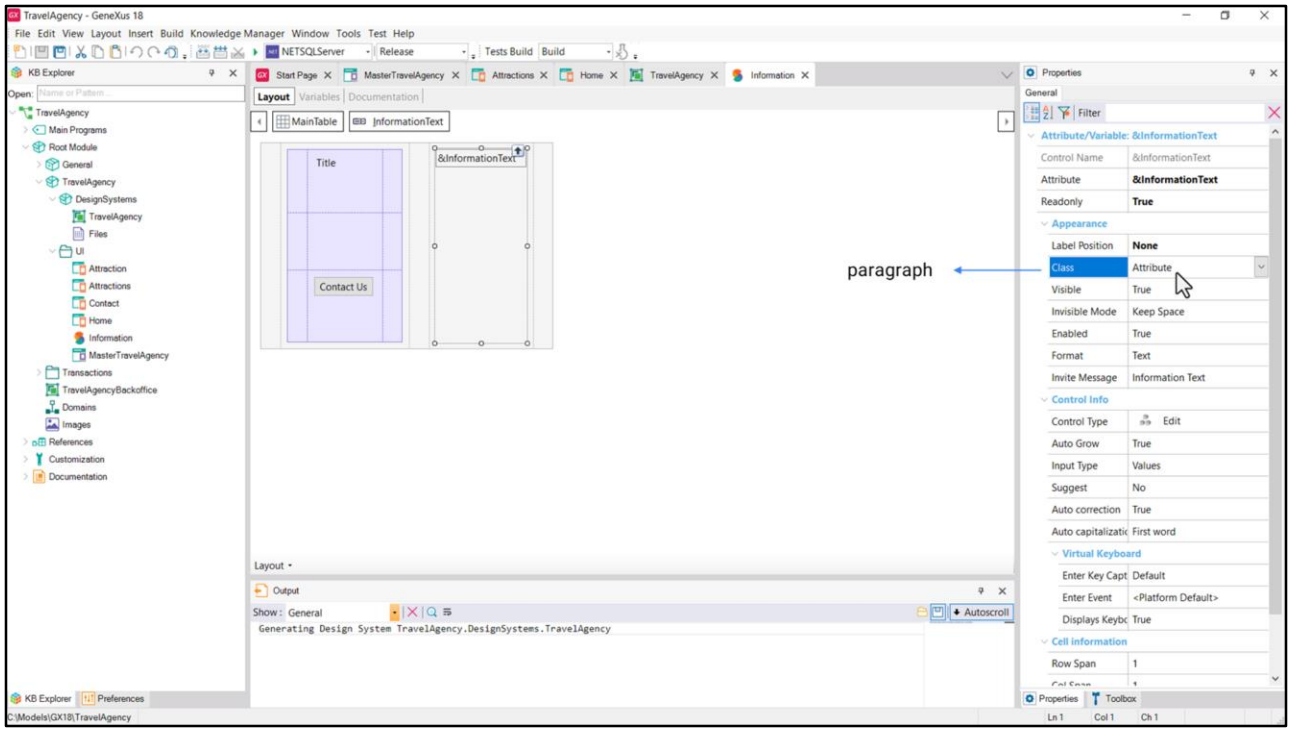
Tenemos que integrar la fuente en nuestro caso, así como hicimos para el caso de la clase h2, a nuestro DSO. Por tanto utilizaremos la misma regla font-face.
Font-family… le llamaremos Heebo… ¿y cuál de las Heebos es la de peso 400? Era la regular, así que le llamaremos de esta manera… vamos a escribir como comentario los 400 puntos de peso.

Y otra vez, ¿dónde se encuentra? En el archivo que habíamos integrado en el video anterior a la KB. Entonces, a partir de que hice eso, puedo llamarle aquí de este modo.

Esta propiedad la voy a eliminar, porque ya está contemplada en esta otra… font-size 16, line-height…
Y estas dos en principio no las necesitamos, decíamos, porque el espacio entre las letras, si no se especifica nada, ya es de 0 píxeles. Así que la voy a eliminar.
Y por ahora esta la voy a dejar comentada.

Sólo nos resta agregar la propiedad de color, el color que asumirá el texto que tenga a ésta como su clase, que será el token de color gray600.

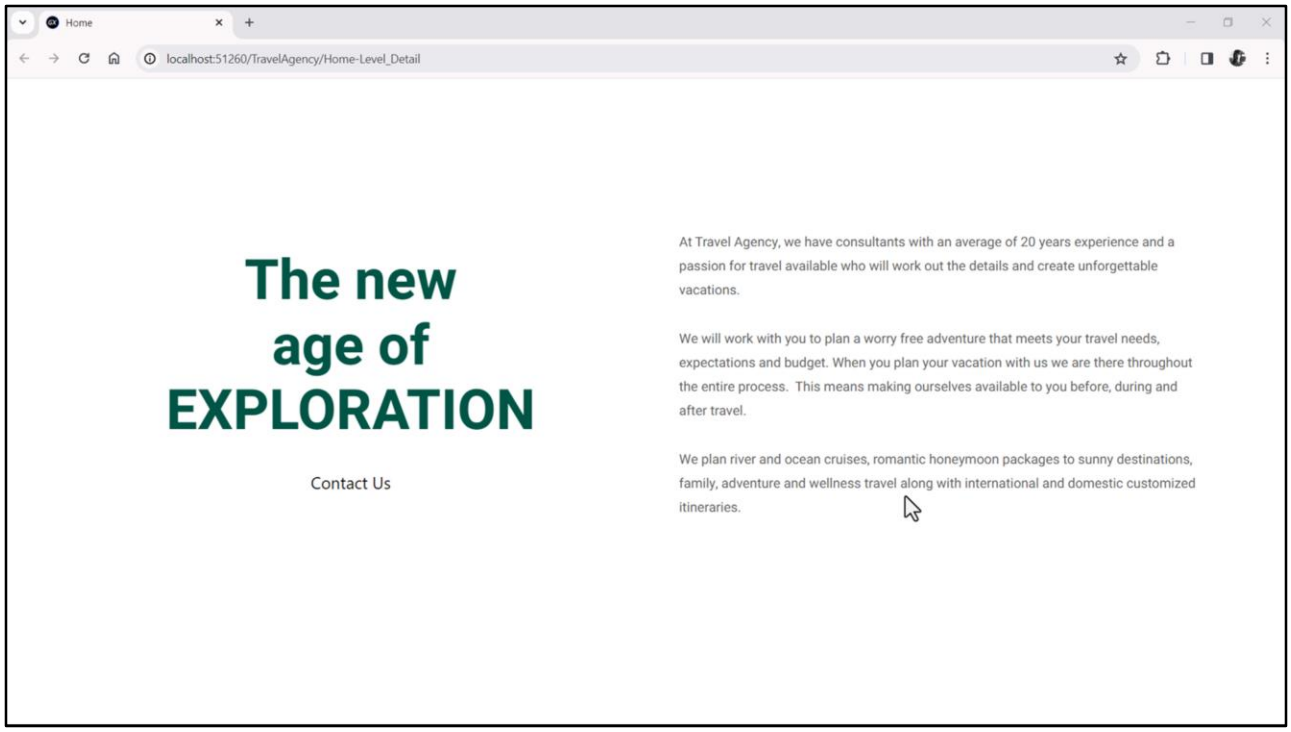
Bien, grabamos.



Y lo que nos resta es asociarle al control variable, en este caso, la clase que acabamos de crear. No será esta clase Attribute, que era la que tenía por defecto, que, como no está seleccionada en nuestro DSO es como si estuviera pero vacía, es decir, sin asociarle valor a las propiedades, por lo que asumirá los valores default en el navegador. Decíamos, entonces, cambiarle esta clase por la paragraph.

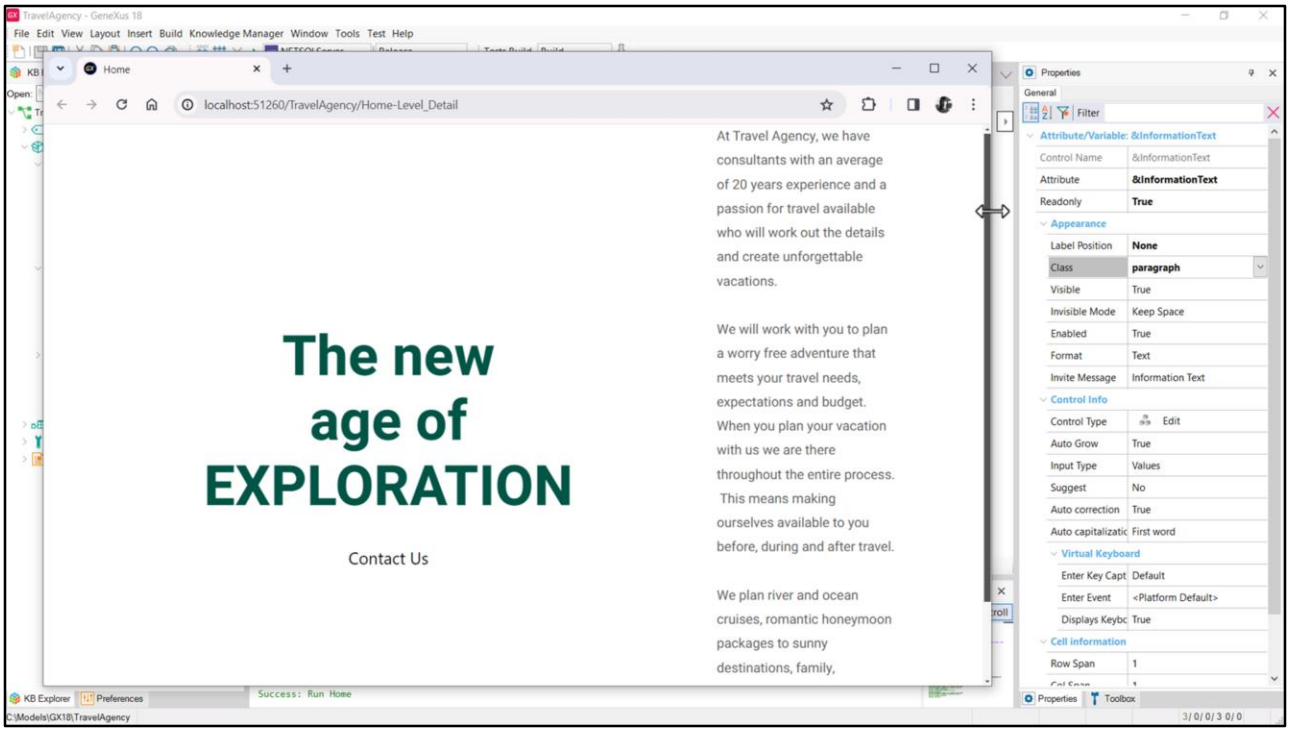
Hecho esto, grabamos… como vimos antes…queda cambiado automáticamente para la variable instanciada en el panel Home, y lo mismo será para la variable instanciada en el de Attractions.

Bien, entonces vamos a ejecutar para probar…

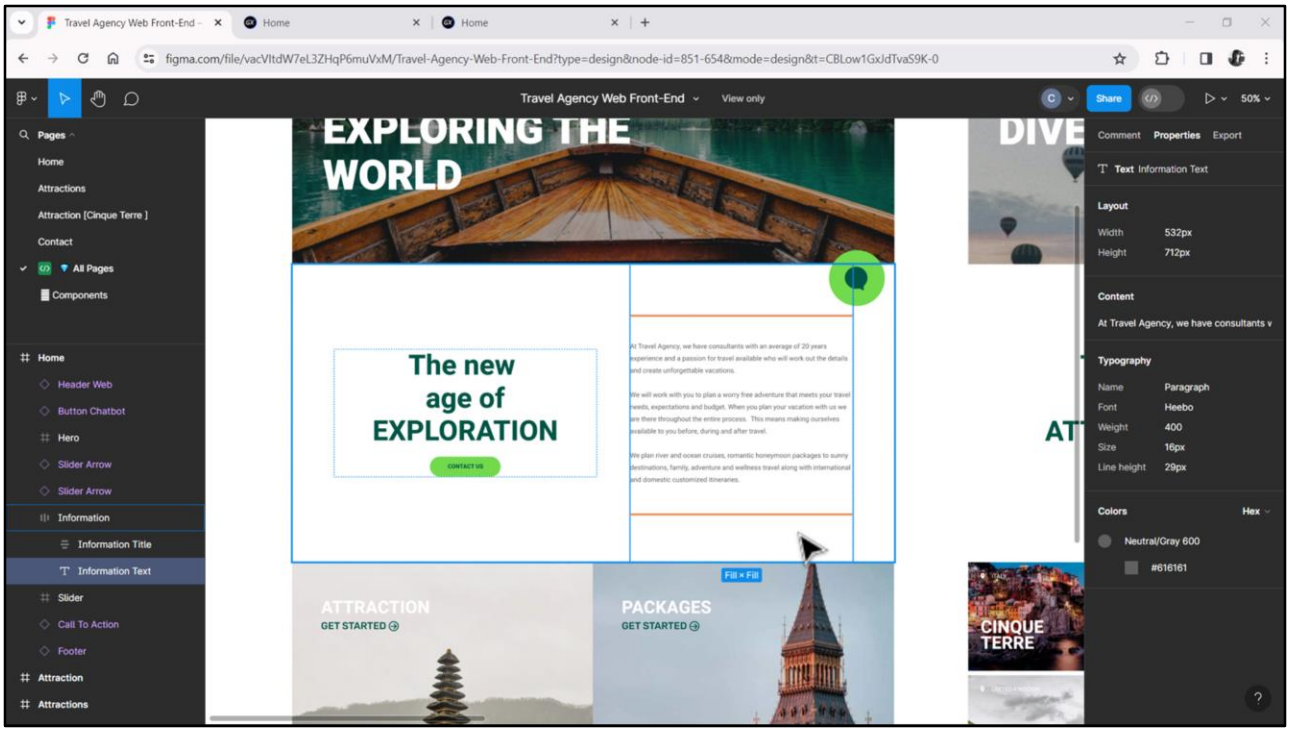


… perfecto.

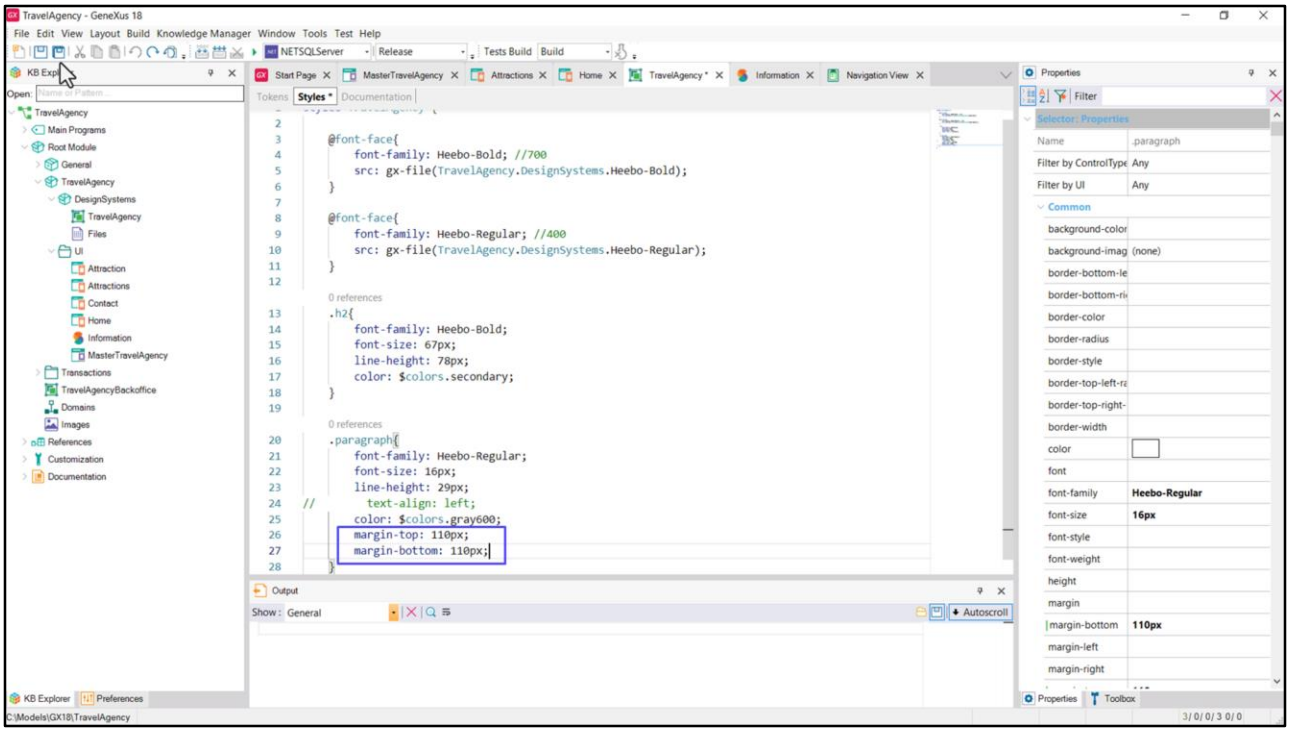
Es claro que la propiedad text-align left no era necesaria, porque la dejamos comentada y aún así se está alineando por la izquierda.



Es momento de reflotar algo que habíamos dejado pendiente en uno de los videos anteriores… Y era el no haber definido un espaciado vertical para este texto… ¿recuerdan que si disminuíamos el ancho de pantalla se pegaba a los bordes superior e inferior de la tabla?

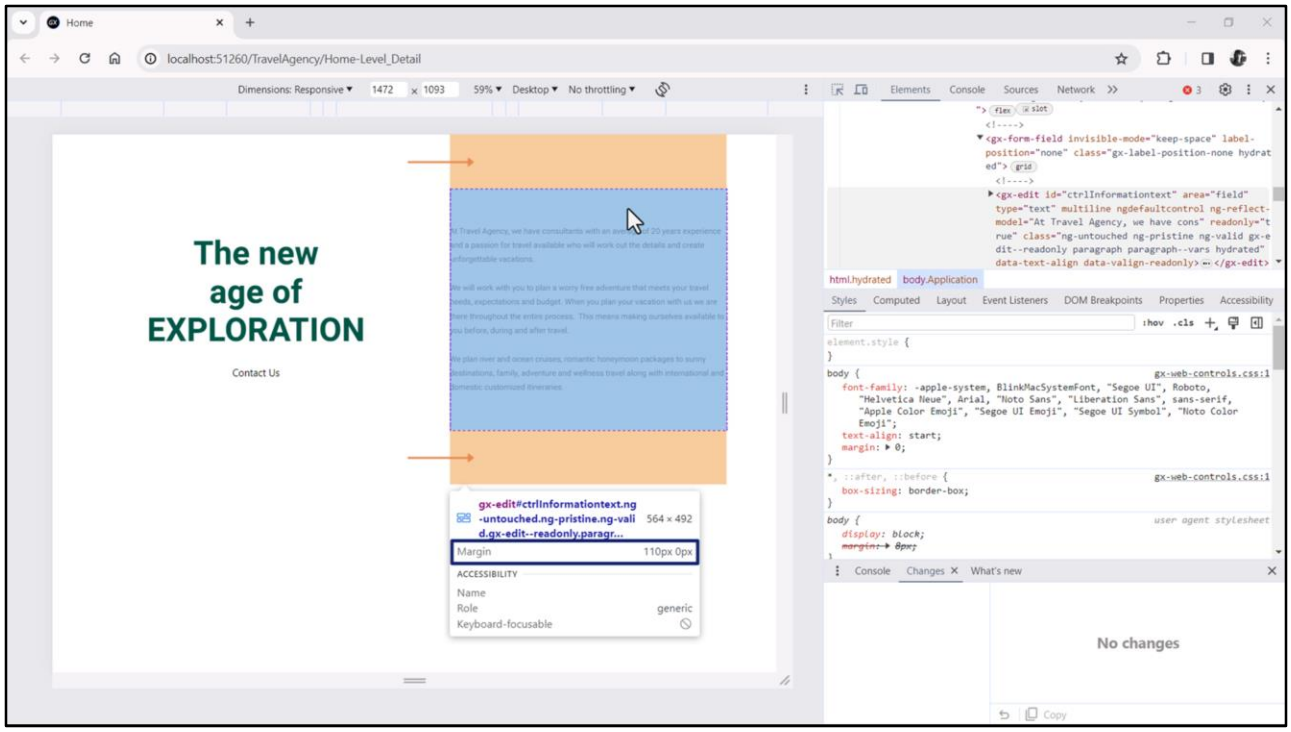


Es que tenía Fill para el alto. En aquel video habíamos visto que un buen espaciado de arriba y de abajo podía ser de 110 píxeles y que podíamos lograrlo especificándole un margen superior e inferior a este elemento, de ese valor. Como si Chechu hubiera podido agregar aquí estas propiedades… no puede, claramente; el estilo tipográfico es sólo tipográfico. Y tampoco tiene otra forma de hacerlo.



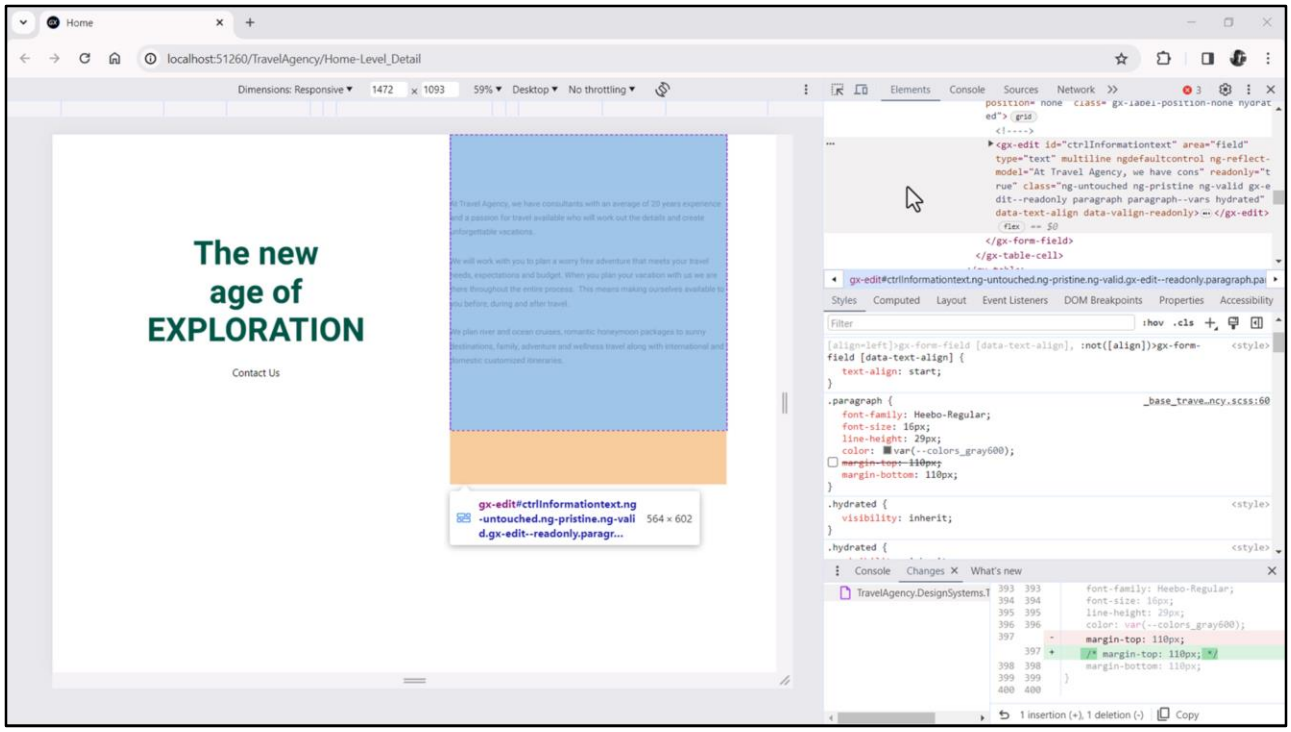
Pero nosotros en nuestra clase no tenemos esa limitación. Así que podríamos escribir esto…

Probemos…

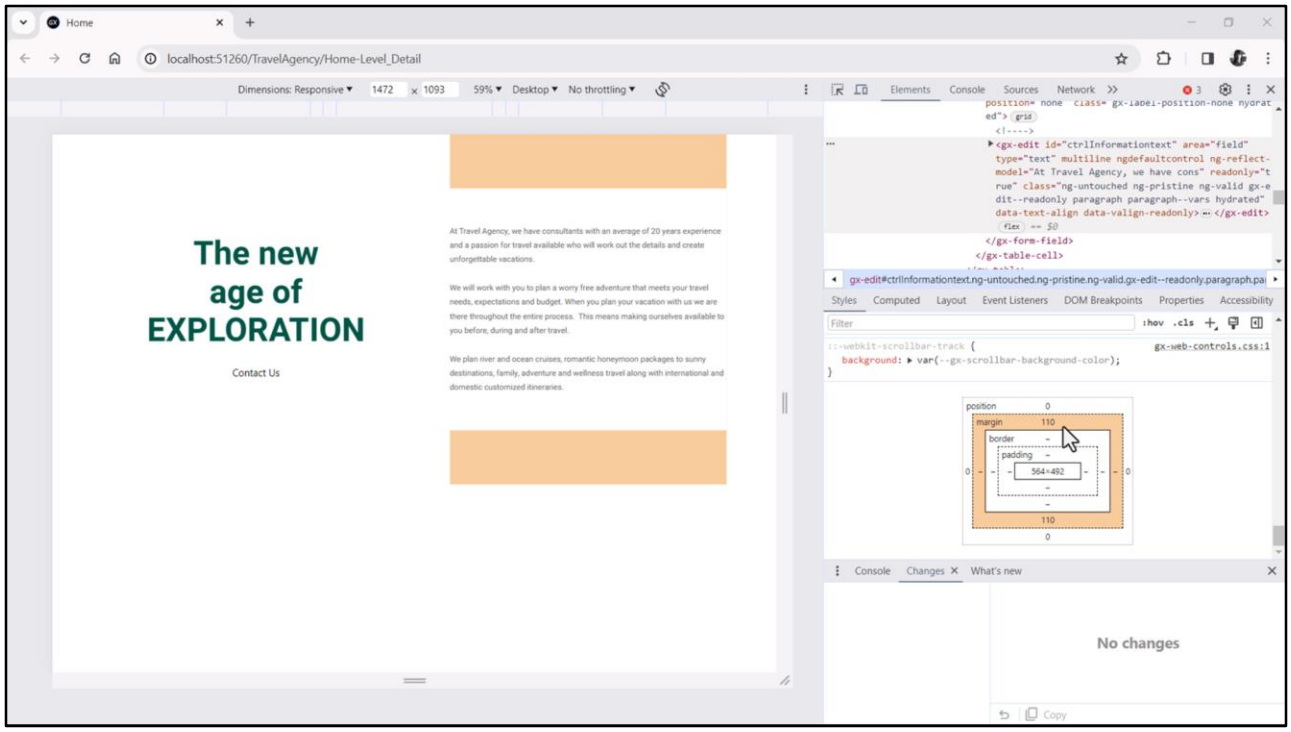


Vemos ese margen respetarse, arriba y abajo.

Vamos a inspeccionar el html resultante. Acá vemos claramente el elemento y el margen superior e inferior.



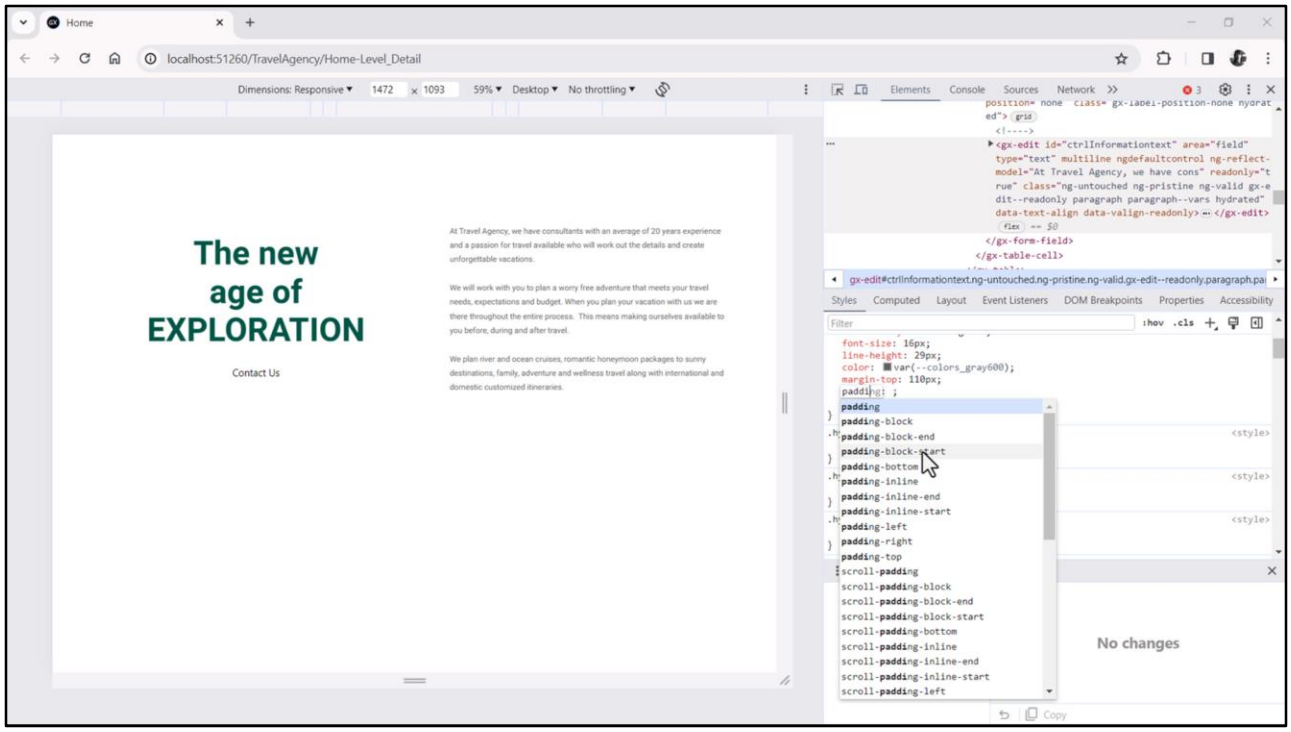
Si buscamos acá la clase que está aplicada, que es la paragraph, vemos las dos propiedades, y por ejemplo, podríamos apagar margin-top y ver cómo al hacerlo efectivamente desapareció ese margen superior.



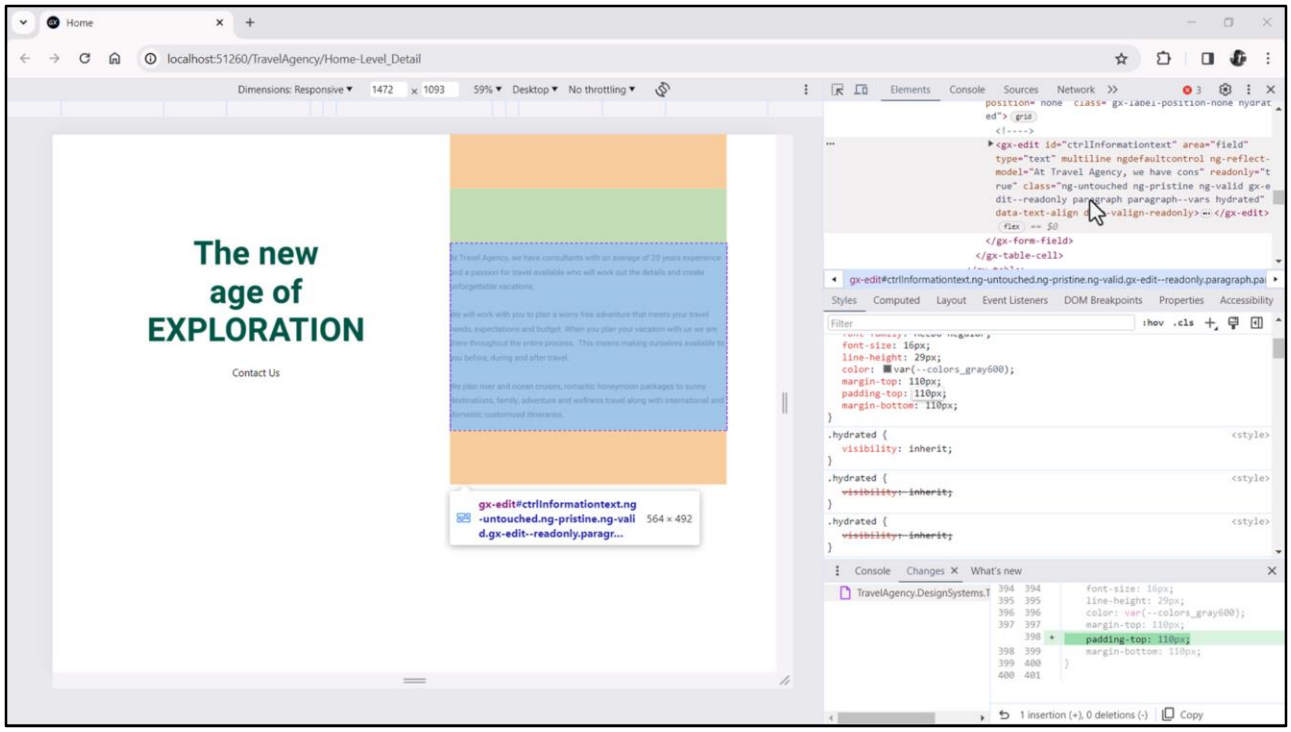
Podemos visualizar abajo del todo justamente esos 110 píxeles de margen de arriba y de abajo. Vemos que no se la ha definido borde, ni padding, y acá tenemos el elemento interno.

¿Por qué elegimos dar ese espaciado a través de margen superior e inferior y no lo hicimos a través de padding? Visualmente veríamos lo mismo, vamos a probar cambiar el margin de 110 por un padding de 110, para el margin y padding superiores. Vamos a dejar el margin inferior tal como está, para ver la diferencia.

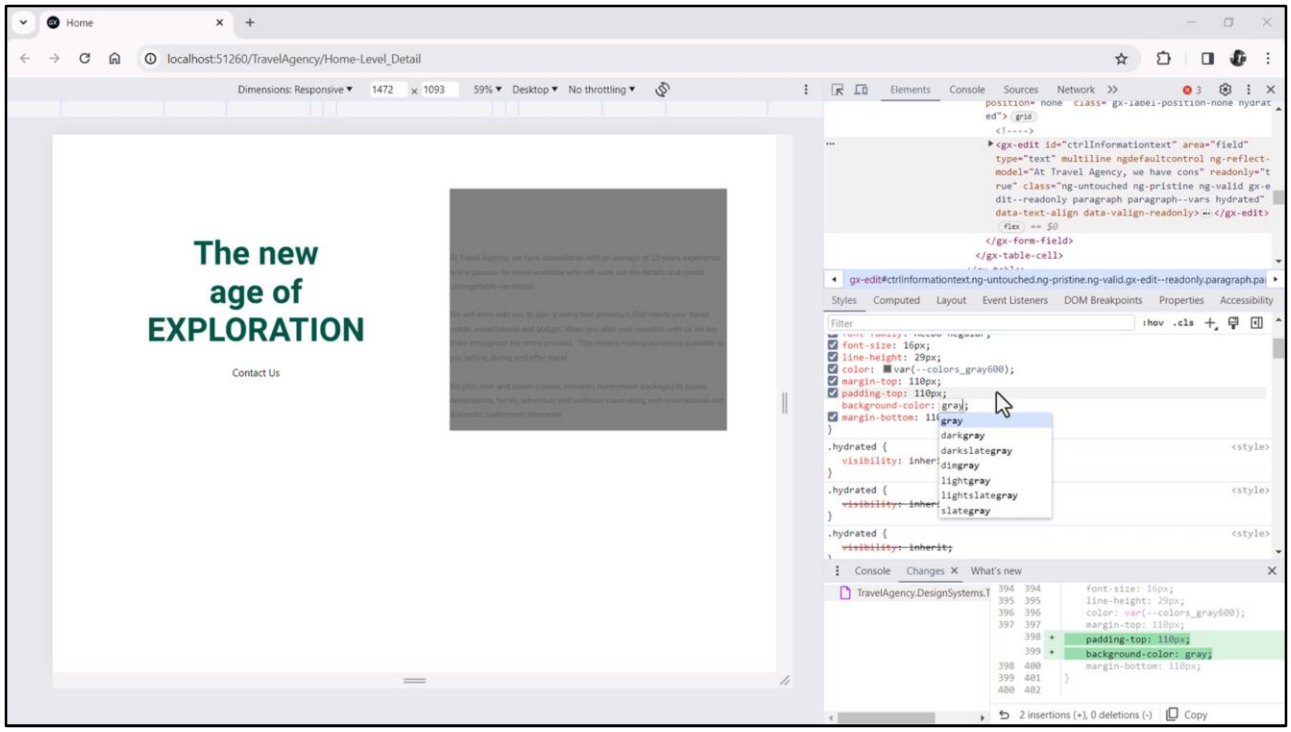
Pero acá ya lo voy a adelantar: el elemento es lo que hay adentro del borde, de este límite hacia adentro. El margen estará por fuera del elemento, y vamos a ver qué implicancias tiene eso.



Entonces antes, voy a volver a la clase, y vamos a colocar la propiedad padding… padding-top vamos a elegir… los voy a dejar mencionados porque en un ratito vamos a entrar en esto, ¿pero vemos estas padding block block block y padding inline? ¿Sí?

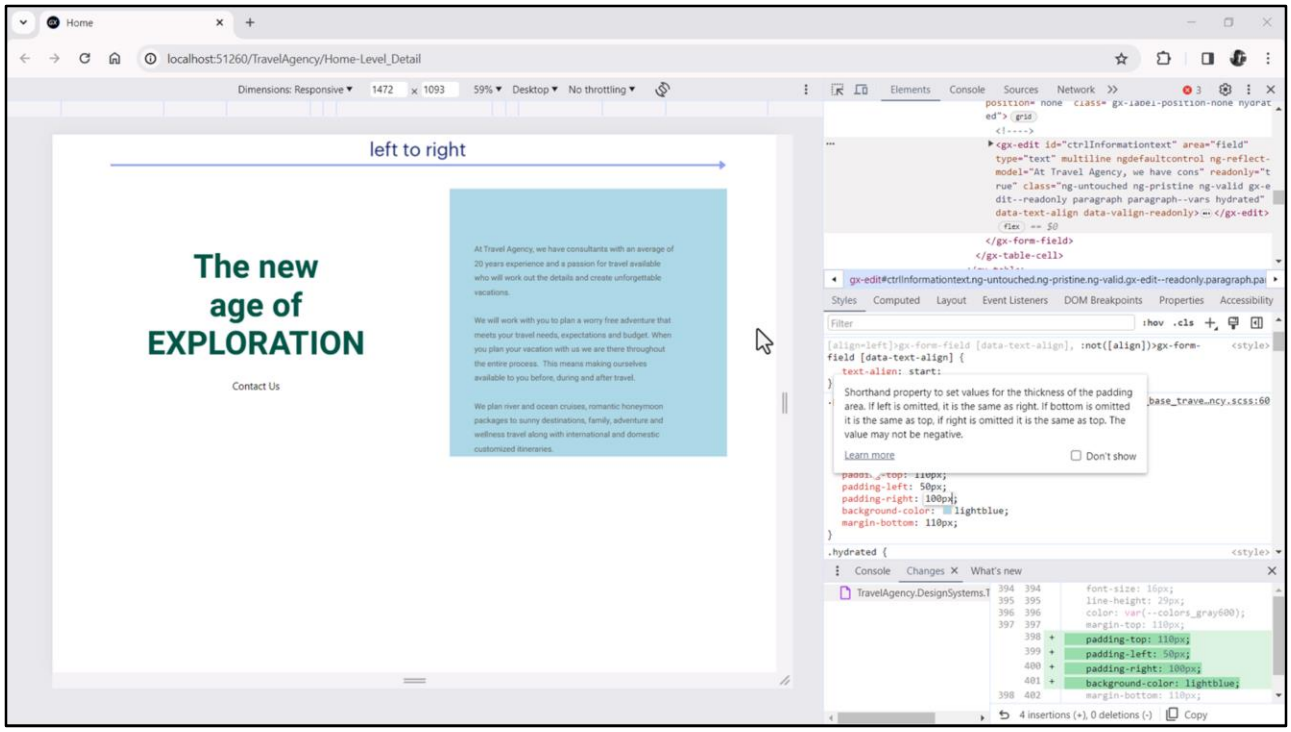


Bien, padding-top 110 píxeles. Y eso lo vemos claramente allí indicado: es esa área en verde.



Vamos a ver qué pasa –voy a dejar las dos: el margen superior y el padding superior- y vamos a ver qué pasa si le doy un color de fondo a este elemento. Background color… y por ejemplo vamos a colocar un gris.

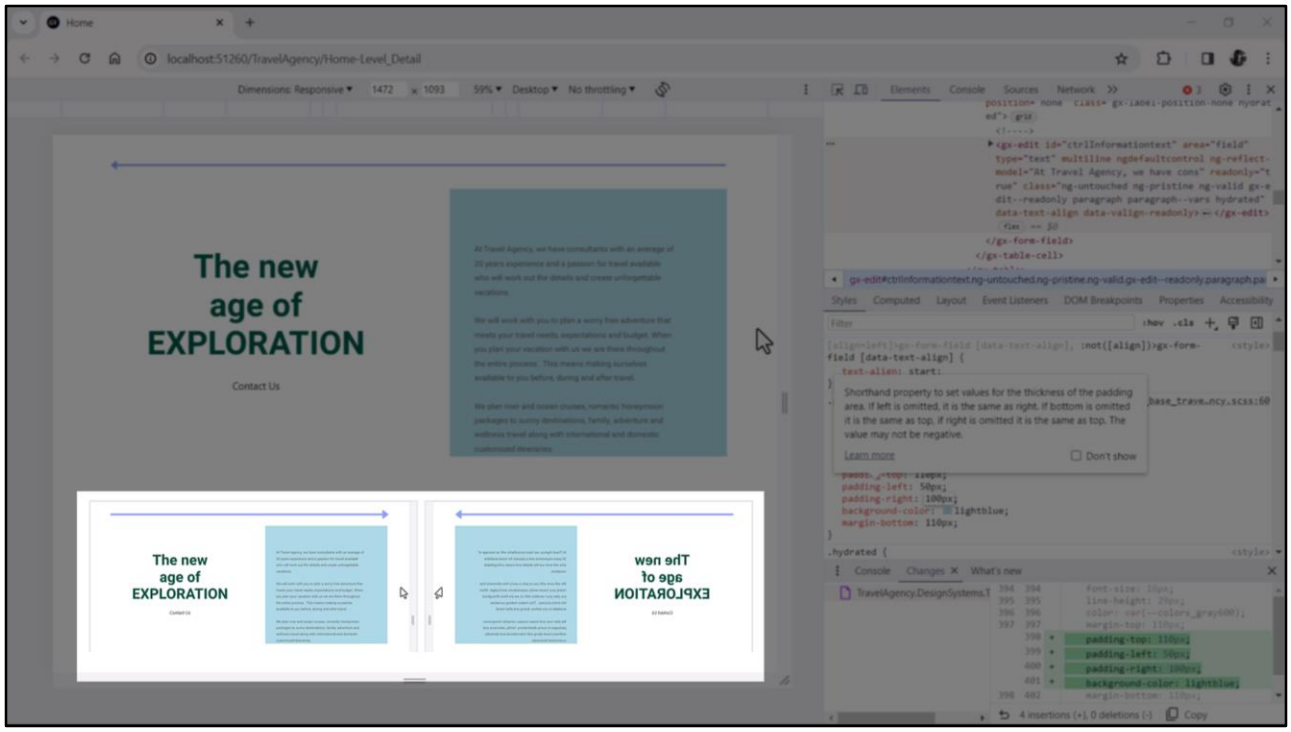
¿Qué está pasando? Vemos que está quedando en gris tanto el área interna del control, como el área del padding, esa área verde. Mientras que al área del margen no se le está aplicando ese color de fondo. Lo mismo, si implementáramos un evento, cliqueable a nivel de este contenido, esta área también sería cliqueable, mientras que el área de margen no lo sería.



Bueno, ahora sí, vamos a entrar en este asunto de la diferencia entre las propiedades físicas (padding top, right, bottom, left) versus las propiedades lógicas, y ahora vamos a ver de qué se trata eso.

Antes, vamos a cambiar este gray por uno más claro. Light… bueno, lightblue voy a dejar. Y lo que voy a hacer para que se note bien la diferencia de las cuestiones es escribir un padding… left… de 50 píxeles… y un padding right de 100, el doble. Es decir, del borde izquierdo tenemos un padding que es la mitad del padding de la derecha.

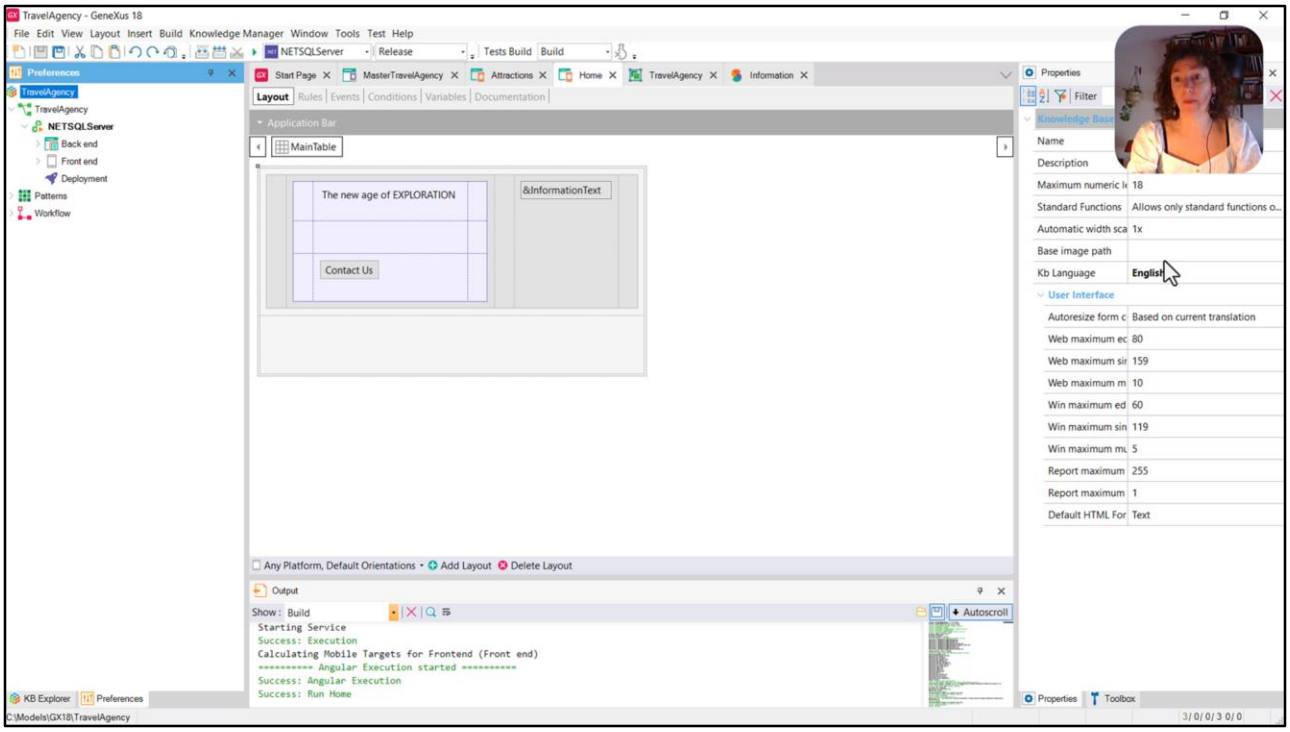
Ahora bien, la pregunta que me surge… estamos ejecutando esta aplicación con una dirección del texto que va de izquierda a derecha…



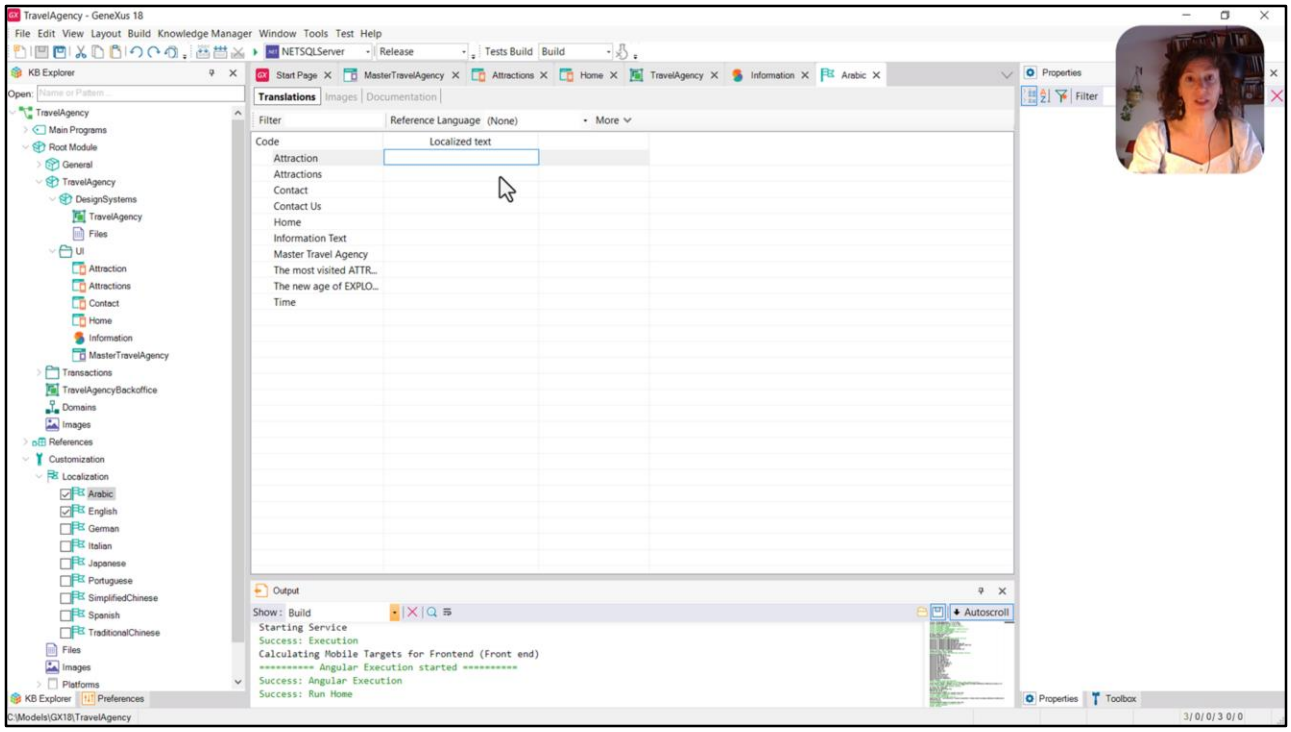
Ahora, si estuviéramos ejecutando la aplicación para hebreo o para árabe, tendría que visualizarse todo esto que estamos mostrando acá como si tuviéramos un espejo que invirtiera completamente la dirección. Por tanto sería en una dirección right to left.

¿Qué es lo que quisiéramos que sucediera en ese caso, es decir, al cambiar la dirección de flujo de la pantalla?

Que la tabla se invirtiera en espejo, donde la primera columna de la izquierda se transforme en la primera de la derecha, la segunda en la segunda, la tercera en la tercera, la cuarta en la cuarta y la quinta en la quinta, en ese orden. A nivel de la ubicación de los controles en nuestros layouts, esto se hará automáticamente.

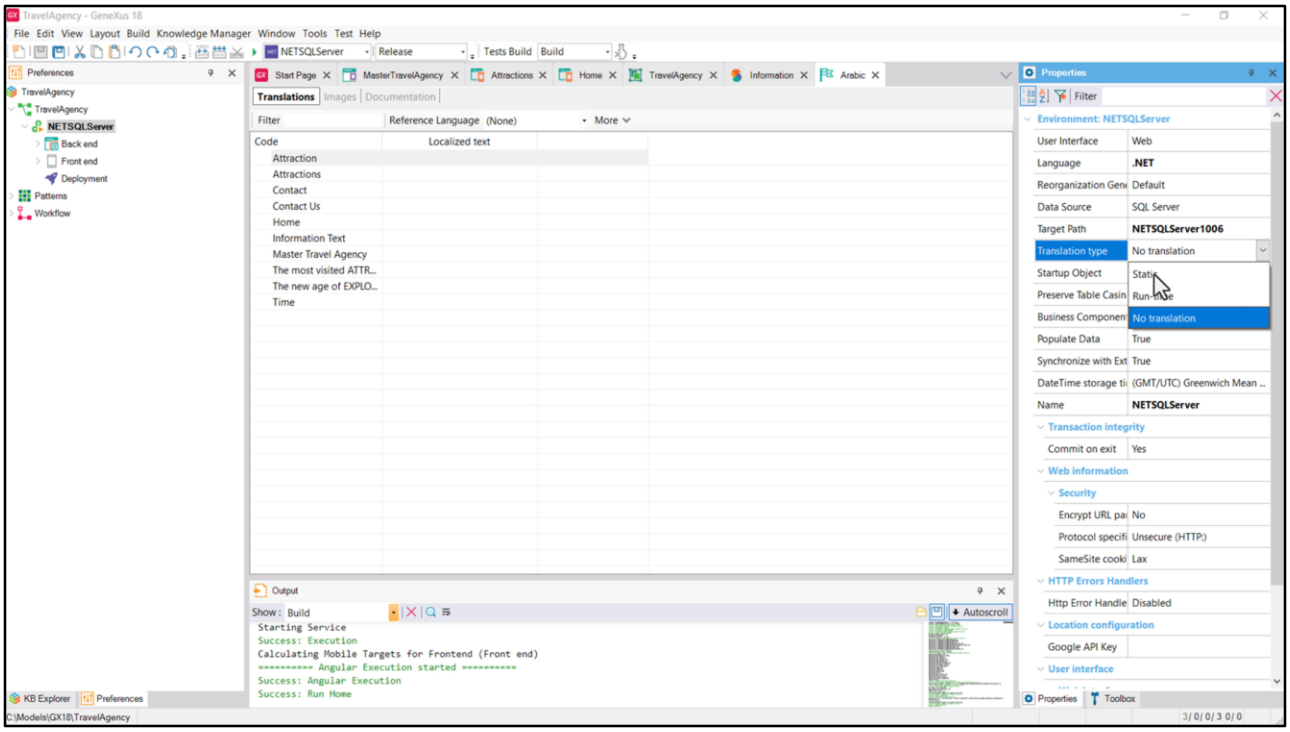


Por supuesto me estoy refiriendo al caso en el que queremos traducir la aplicación. Si venimos a la solapa de Preferences podemos ver que a nivel de la KB tenemos la propiedad Kb Language con el idioma English.



Si venimos al KB Explorer, bajo el nodo Localization, vemos allí que tenemos este objeto English que está coincidiendo con el idioma de la KB. Y es el único que tenemos por el momento incorporado en la KB. ¿Qué es lo que se va a hacer? Se van a recoger de todos los objetos de nuestra KB los textos para ser traducidos.

Entonces, de esta manera, supongamos que queremos traducir la aplicación al árabe. Entonces alcanzará con prender esta propiedad, va a incorporar ese objeto dentro de la KB, va a extraer… voy a abrirlo… todas las constantes, todos los textos que encontró en los distintos objetos, para permitirnos justamente escribir las traducciones de esos textos.



Y entonces, de esa manera, luego vamos a la ventana de Preferences y podemos, a nivel del Environment, establecer el tipo de traducción que queremos realizar. Si estática, donde vamos a indicar entonces cuál es el lenguaje al que queremos traducir la aplicación, de los que tenemos incorporados en la KB; o si la queremos, por el contrario, realizar dinámicamente. Y al hacerlo dinámicamente vamos a poder decirle a nivel de nuestros objetos con qué idioma queremos que se renderice la aplicación cada vez. Por supuesto vamos a tener que agregar alguna programación.

Bien, entonces este es el contexto en el que estamos, ¿no? Lo que yo les decía hace un ratito es que una vez que yo traduzca la aplicación al árabe, automáticamente las columnas de las tablas se van a invertir en espejo. Y también así los textos. Van a aparecer en el idioma que haya elegido, con las traducciones que haya ingresado.

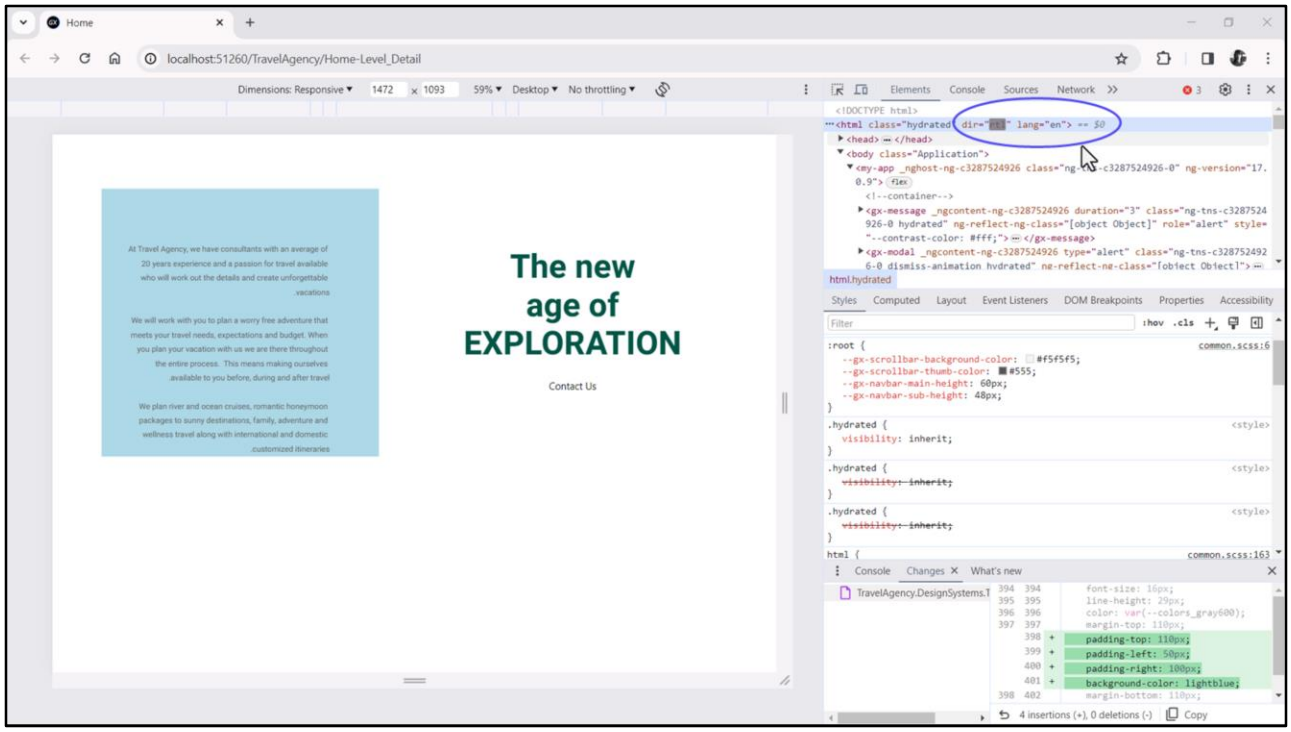
…en realidad se mostraría así.

Ahora bien, concentrémonos en lo que nos interesaba, en el padding. Cuando utilizamos referencias izquierda y derecha para proporcionar el padding, no nos dimos cuenta de que en verdad las referencias correctas serían inicio y fin, en relación a la dirección de flujo y no izquierda y derecha… porque en el flujo right to left no queremos que el padding de la izquierda sea de 50, y el de la derecha de 100, sino al revés.

Necesitamos utilizar referencias lógicas, es decir, el padding en la dirección del flujo (inline) respecto al inicio, que sea de 50; y el padding en la dirección del flujo (inline), respecto al fin del elemento de 100. Esta forma lógica valdrá tanto para el flujo left to right, como para el right to left. Es decir, nos estamos independizando de las coordenadas izquierda y derecha, que nos complican justamente cuando queremos hacer esta inversión.

Pero de la misma manera y por consistencia, si cambiamos las propiedades en torno a la dirección horizontal, también tendremos que cambiar las propiedades que tienen que ver con la ubicación arriba abajo… Entonces es que aparecen las propiedades análogas pero lógicas: block-start y block-end.

Por tanto, inline representa al dirección horizontal, la de escritura, y block la vertical.



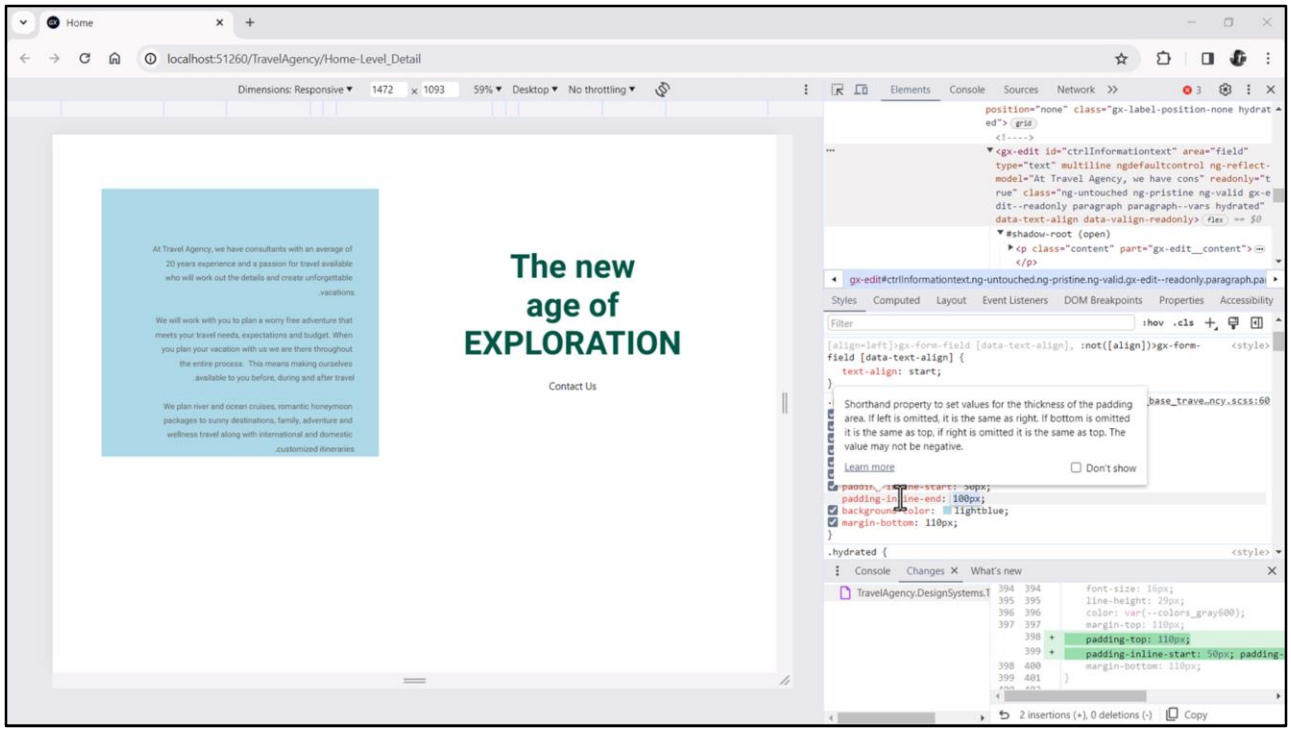
Para que nos termine de quedar claro todo esto vamos a verlo aquí. Tenemos, entonces, la padding-left y la padding-right. Y lo que voy a hacer es venir al tag html a cambiarle la dirección de left to right a right to left, sabiendo que el idioma es inglés, como decíamos.

Entonces ahora cuando presione enter… vemos que efectivamente se invirtió todo lo que queríamos,

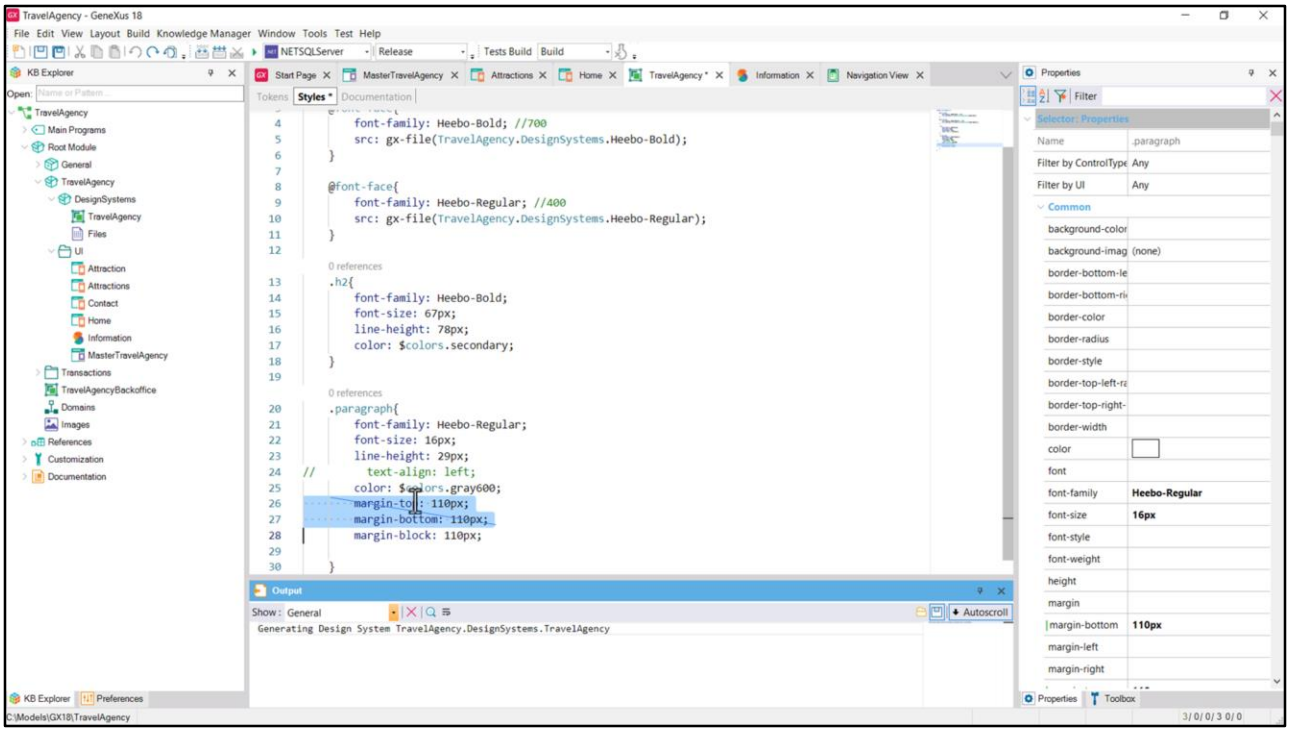
(los textos no porque el idioma es inglés, pero no nos importa, cuando el idioma sea árabe o hebreo también se invertirán)

Entonces, repito todo lo que queríamos se invertirá,

salvo el padding… porque al haber utilizado las propiedades físicas va a respetar que el padding de la **izquierda** sea de 50 y el de la **derecha** de 100. Nosotros no queríamos esto, queríamos que se invirtiera, por tanto…

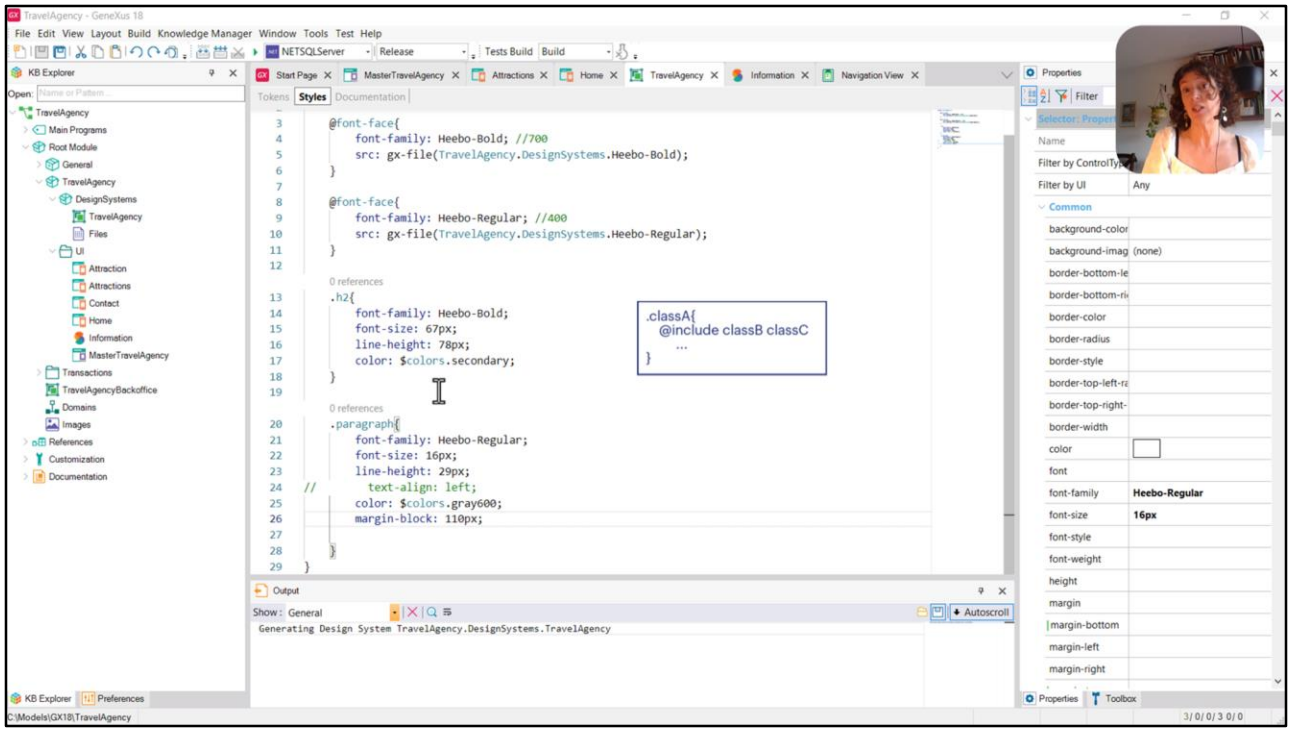


…vamos a modificar las propiedades físicas cambiándolas por las lógicas… así que en lugar del padding-left vamos a usar padding-inline-start de 50… y para el padding right vamos a usar padding-inline-end. Y ahora sí.



Bueno, todo esto que vimos es para recomendar fuertemente no utilizar las propiedades físicas sino utilizar las lógicas. En el caso anterior veíamos el ejemplo con padding pero es exactamente lo mismo con margin.

Acá estamos usando las físicas, entonces la sugerencia es utilizar en lugar de éstas… la margin-block sería en este caso… start y end… O, si quiero utilizar la notación abreviada, solamente block, que en este caso me va a convenir porque además son exactamente el mismo valor, así que puedo colocarlo una única vez. Y quitar estas dos.

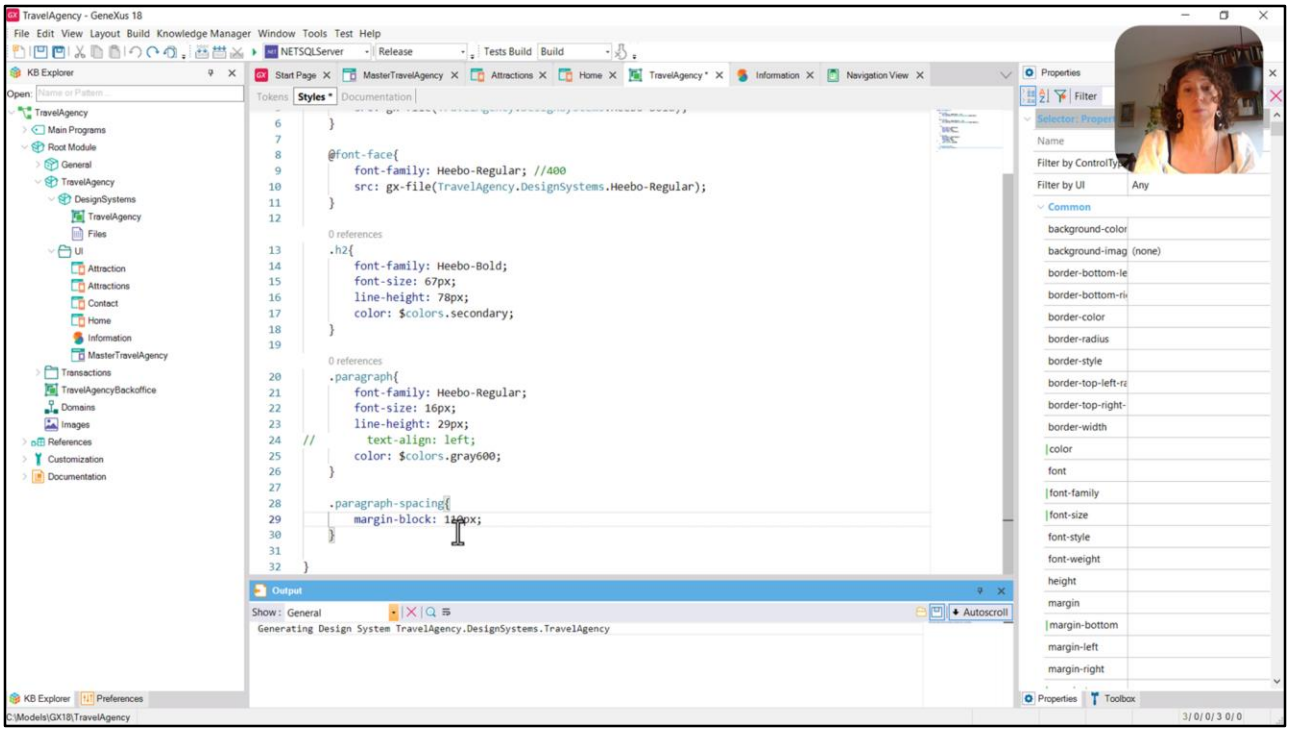


Un par de observaciones antes de terminar…

Las clases son independientes del tipo de control al que se terminen aplicando. Por independientes me refiero a que yo no le tengo que decir en ningún momento que esta clase se aplique a tal o cual tipo de control. Va a depender del tipo de control al que yo le asocie esta clase si todas las propiedades van a tomar efecto o no, o algunas no. Van a tomar efecto aquellas que tenga sentido que tomen efecto, y las demás se ignorarán.

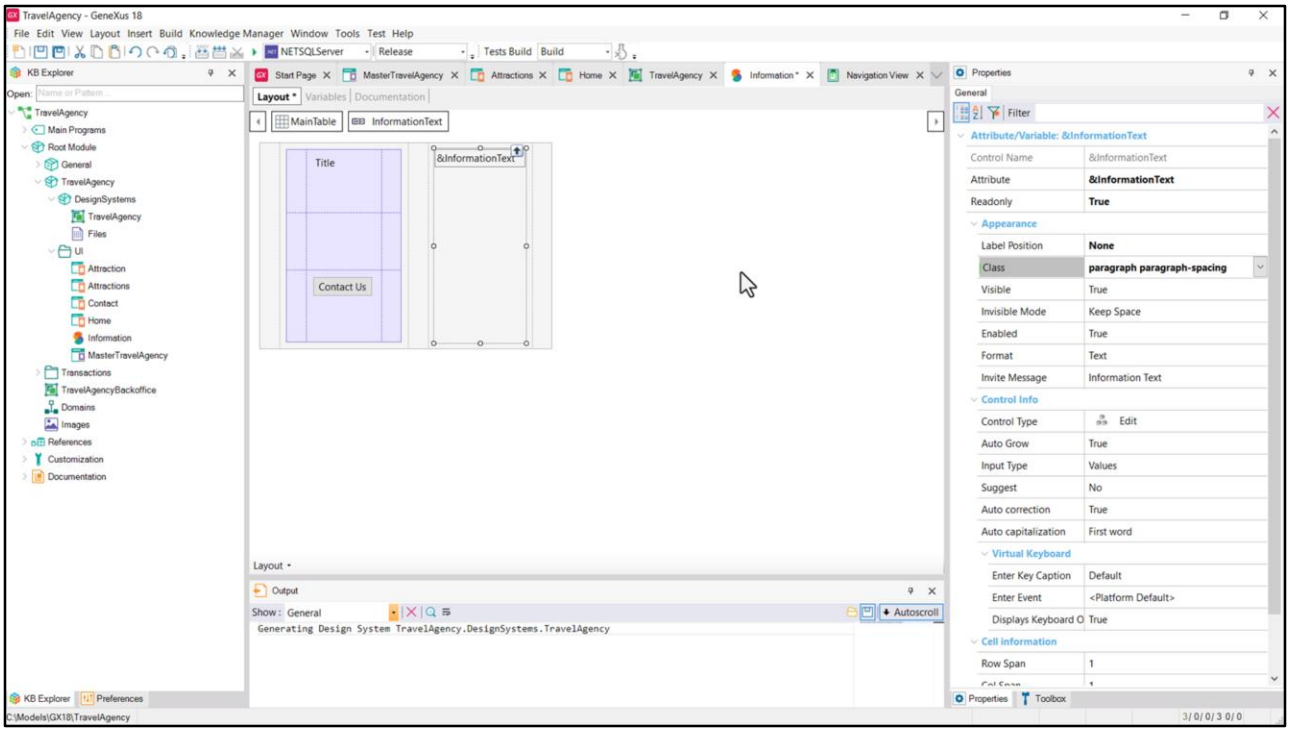
En este caso, por ejemplo, nosotros aplicamos la clase h2 a un control de tipo textblock y la clase paragraph a un control de tipo atributo/variable, que claro, en este caso son tipos de control muy parecidos. Pero eso no es obligatorio.

Por otro lado, para darle un estilo determinado a un control, no es necesario colocar todas las características de estilo dentro de la misma clase. Las clases se pueden componer, ya sea definiendo dentro de la solapa de styles clases compuestas por otras clases, y eso se va a hacer con la regla include; o asignándole al control más de una clase.



Por ejemplo, supongamos que queremos dejar separada lo que son las características tipográficas de nuestro parágrafo, de lo que son las características que tienen que ver con el espaciado.

Entonces lo que podemos hacer es quitar esta propiedad de aquí, la del margen, dejar entonces la clase paragraph limpia en cuanto a lo tipográfico… y seleccionar otra clase, a la que por ejemplo le puedo llamar así… y donde sí, allí, en esa clase, es que defino la característica del margen.



Y luego lo que hago (voy a grabar) es venir a mi stencil, y en lugar de que el control variable tenga solamente la clase paragraph le voy a agregar también esta otra. Entonces a partir de ahora nuestro control va a tener dos clases.

Bueno, y este es el tipo de decisiones que vamos a tener que ir tomando: si queremos aislar, separar, entonces, algunas propiedades en clases diferentes, para poder luego componerlas entre sí.

Bueno, visto todo esto, ya estamos en condiciones de empezar a ver cómo damos estilo al botón. Nos vemos en el próximo video.


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