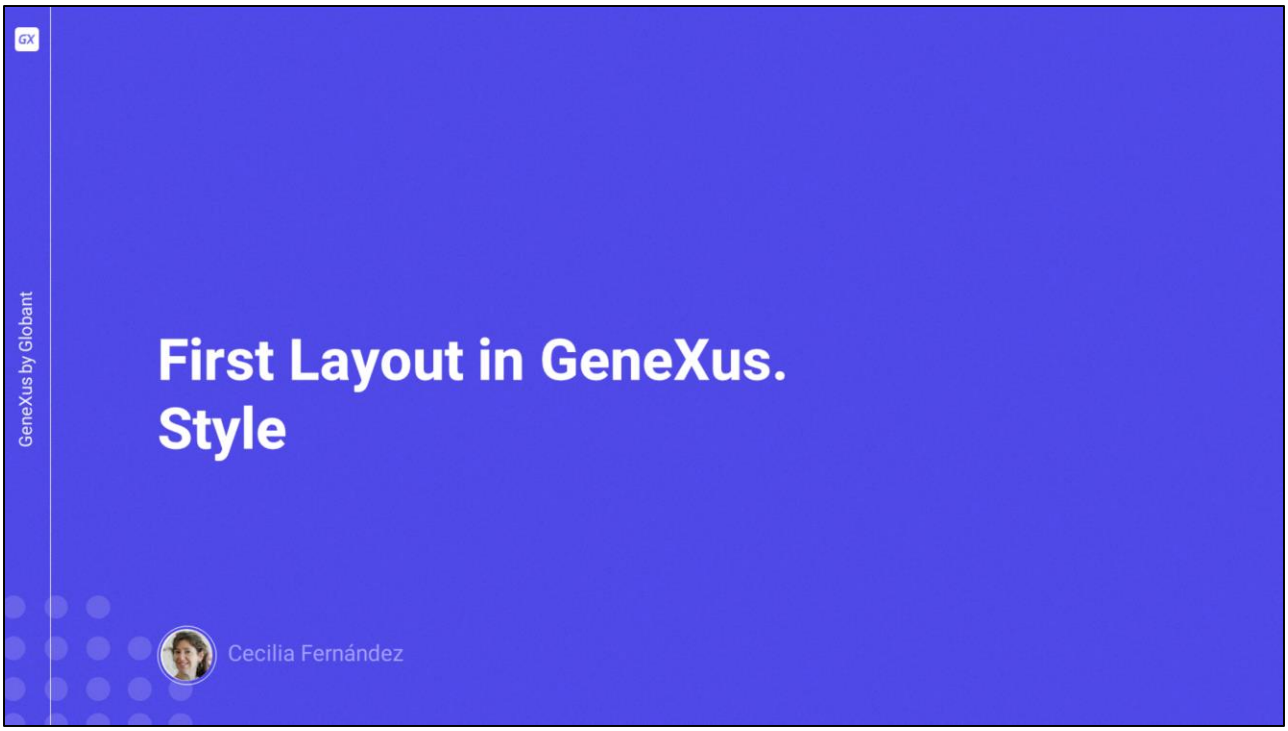
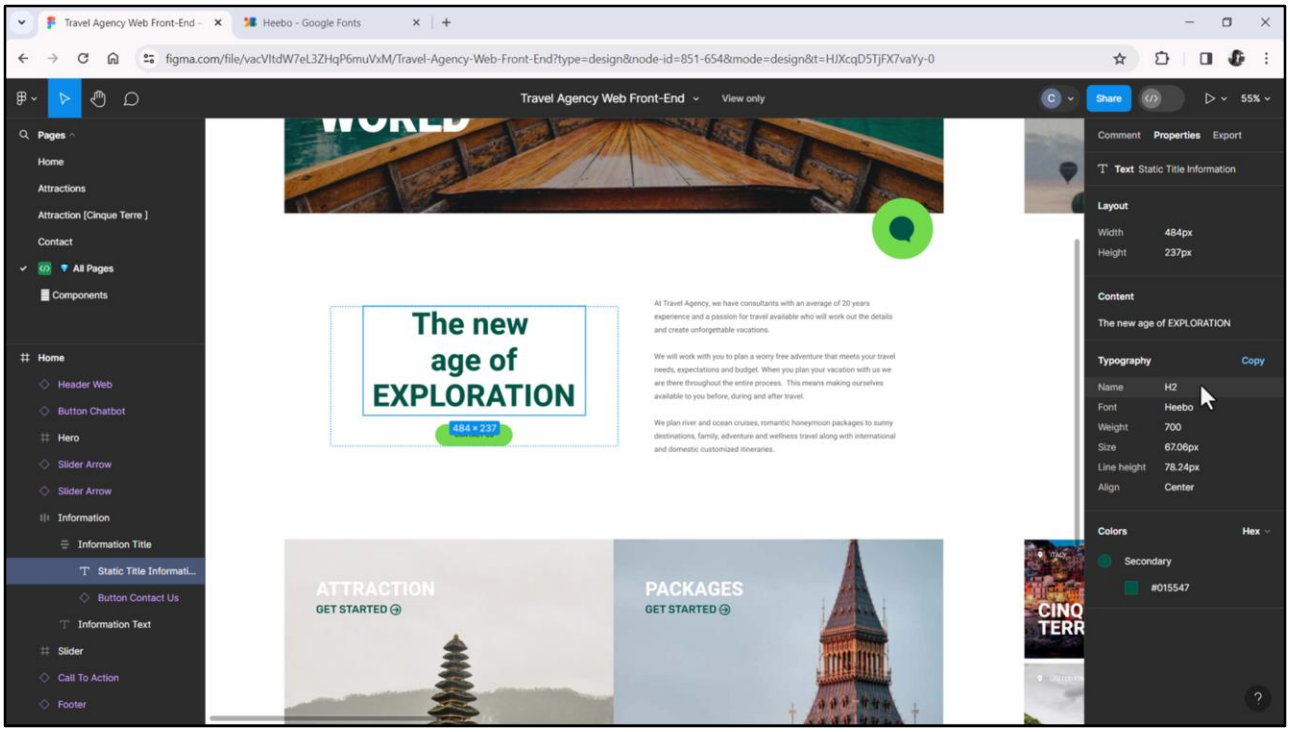
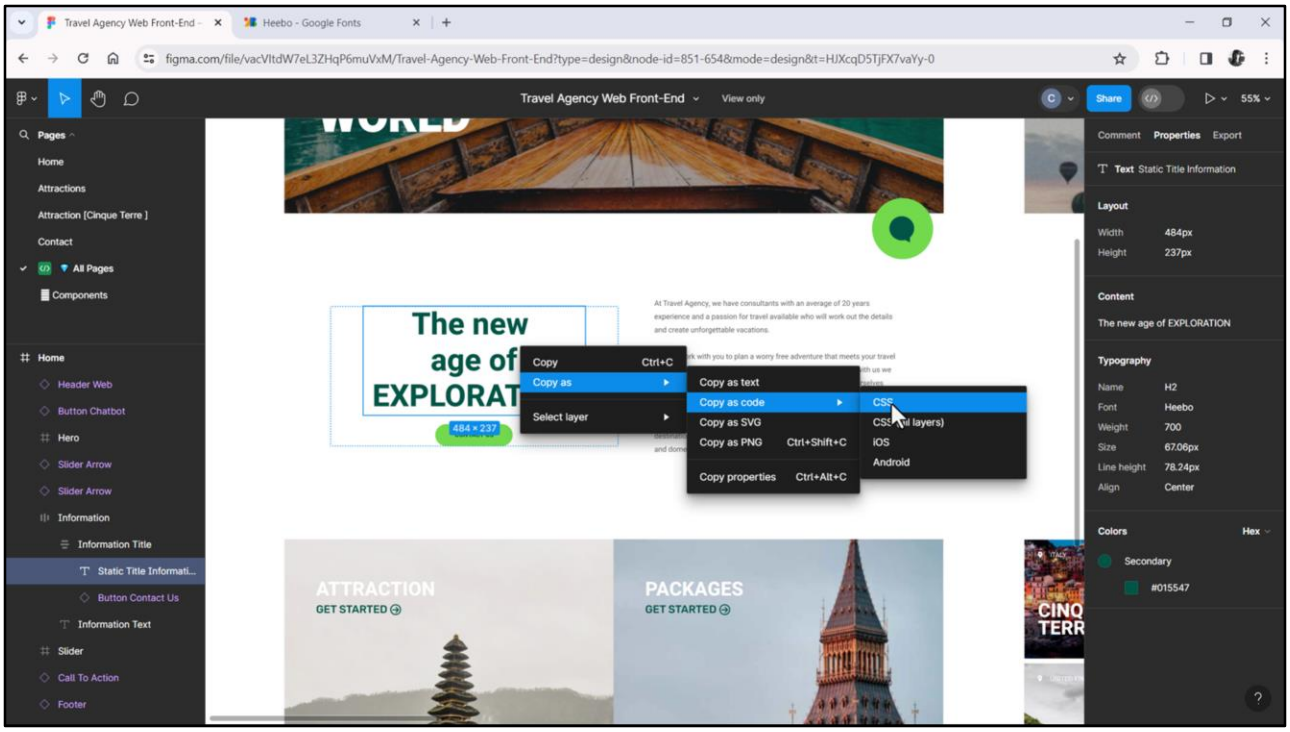


Ahora que ya tenemos la estructura del layout del Stencil, pasaremos a trabajar sobre el Design System object para lograr que luzca ese layout del stencil como queremos.

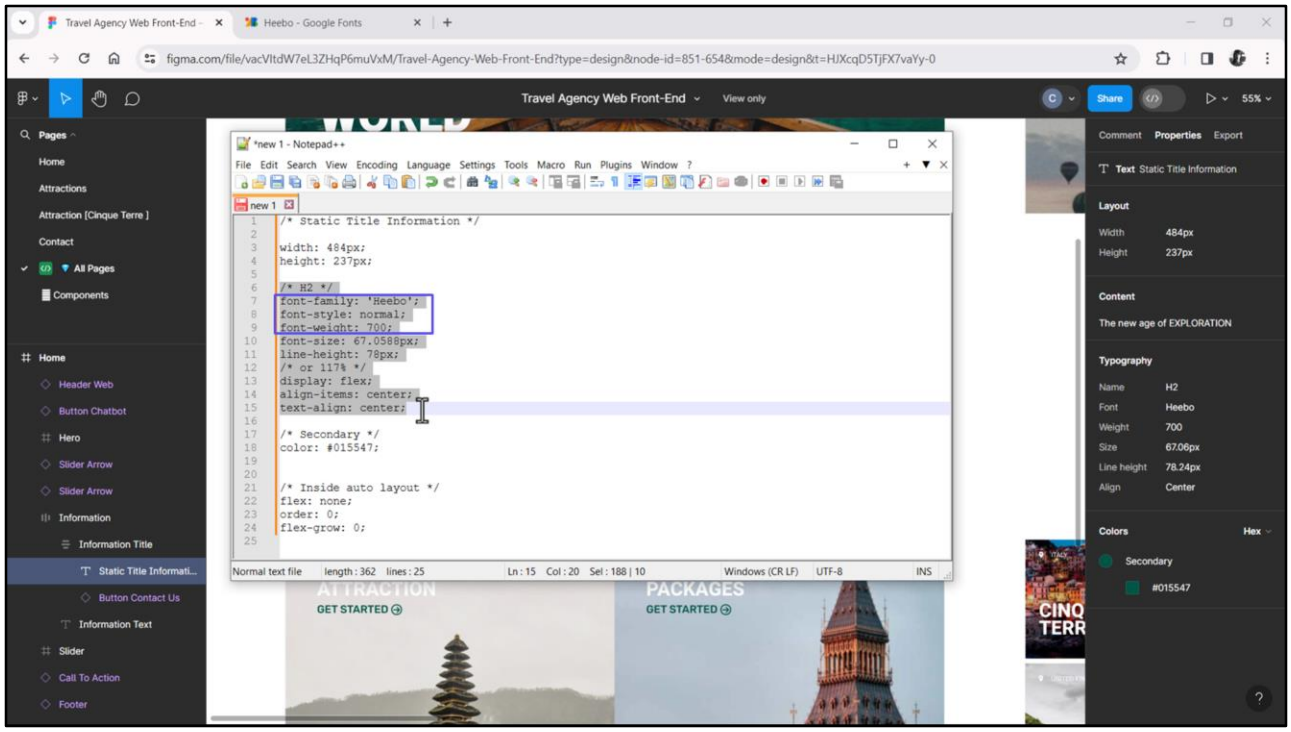


Empecemos por este texto. Si vamos a inspeccionar sus propiedades, aquí tenemos las relativas a los tamaños en el layout, ya las vimos en los videos anteriores. Aquí tenemos el contenido, y aquí nos muestra las características relacionadas con la tipografía y con los colores.

Si observamos las características tipográficas vemos que nuestra diseñadora les ha dado un nombre, H2. Esto, recordemos lo que Chechu nos contó, significa que creó un estilo tipográfico en Figma. Con estas características: esta familia de fuentes, con este peso, este tamaño, este alto de línea y esta alineación.



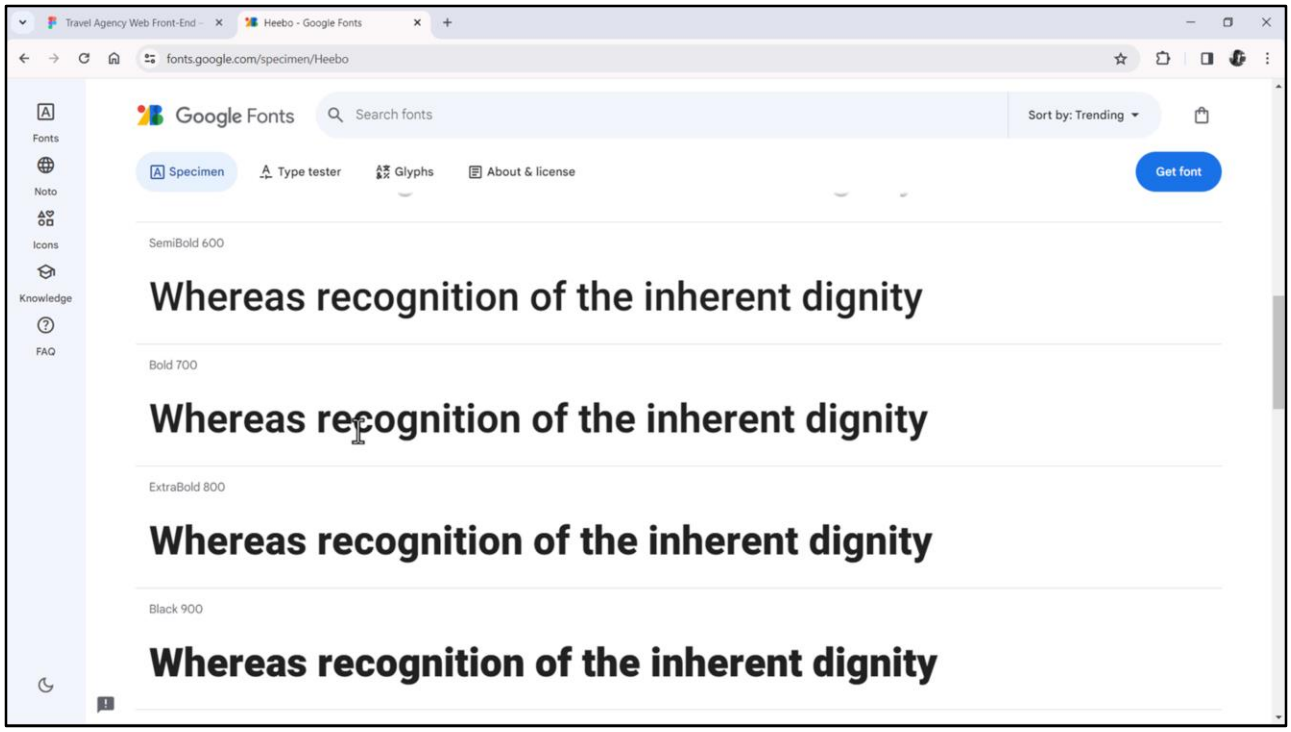
Vamos a querer tener estar propiedades en formato CSS, porque es el que utilizaremos en GeneXus en nuestro Design System Object. Las copiamos…



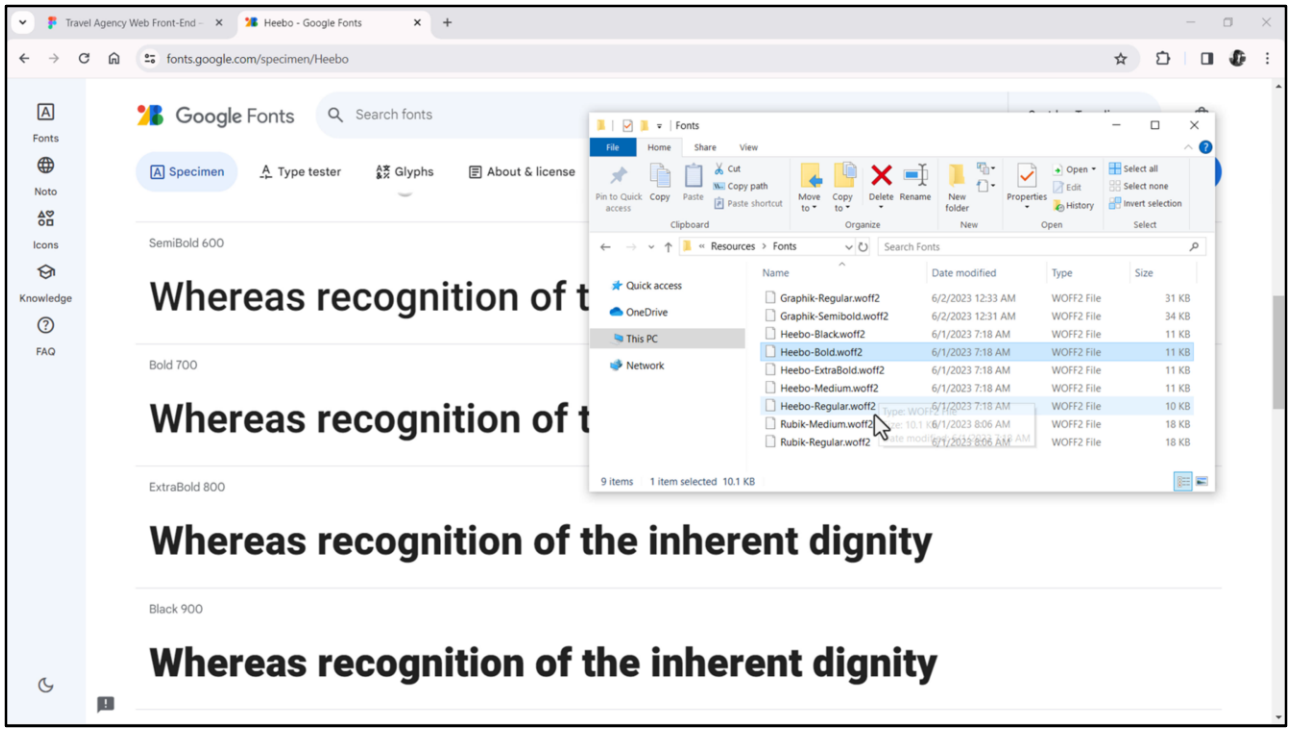
…y peguémoslas para verlas en un archivo de texto.

Nos interesan, fundamentalmente, las que tienen que ver con el estilo H2.

Acá vemos estas primeras tres que tienen que ver con la familia de fuente. Heebo es una fuente web que mejora la legibilidad en pantallas digitales, pero no es de las fuentes estándar de los navegadores, por lo que tendremos que incluirla en nuestro CSS.



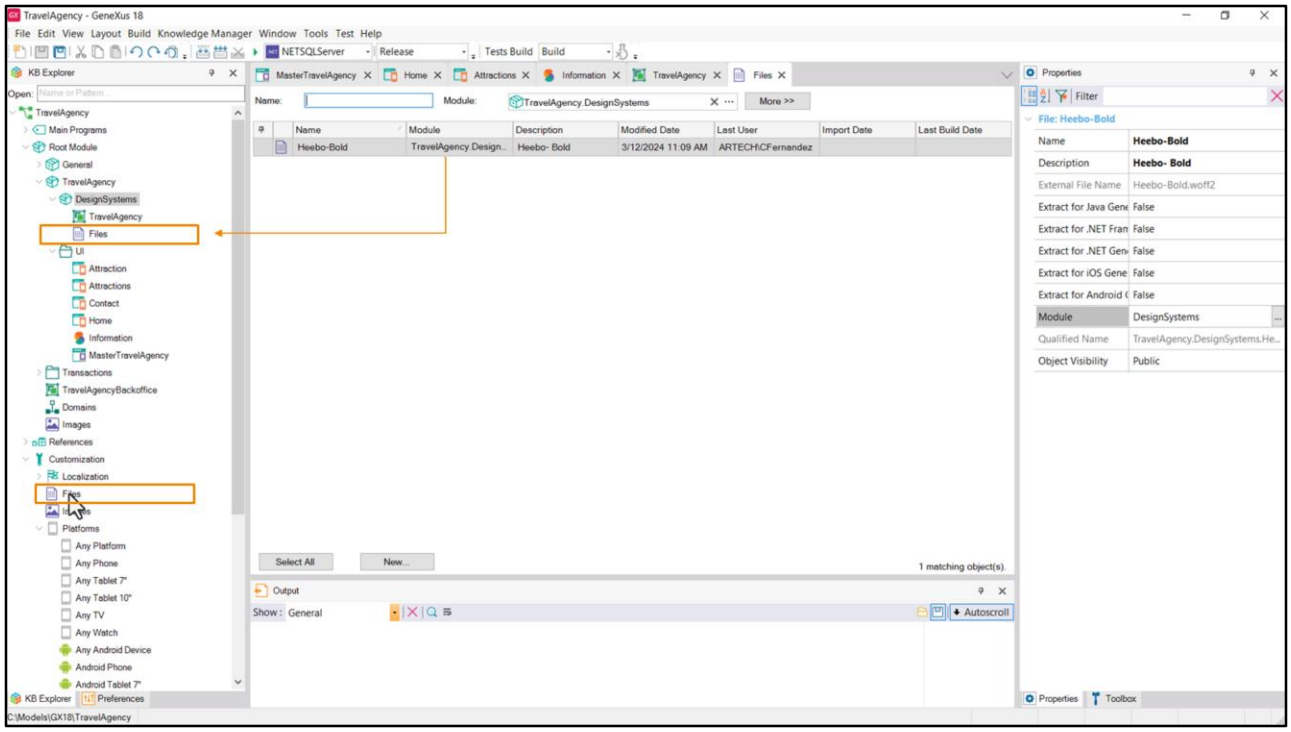
La pueden googlear y descargarla. Aquí la ven con sus distintos pesos. La que necesitamos para el estilo H2 es la de peso 700, es decir, esta, que es la bold.



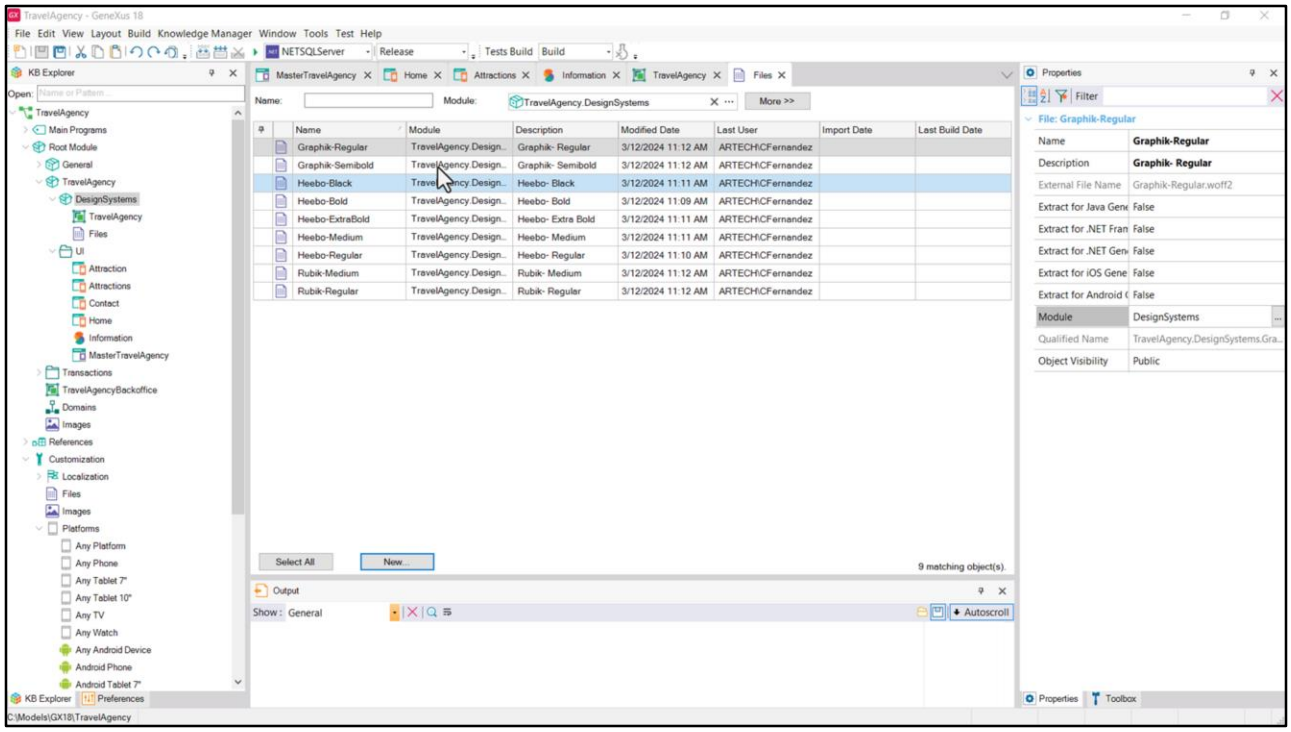
Yo me descargué en esta carpeta la Regular de peso 400; la Medium de peso 500; la Bold, como sabemos, de peso 700; la Extra Bold de peso 800; y la Black de peso 900. Además me descargué estas otras fuentes que luego utilizaremos.

Y me las descargué en formato woff2, que es un formato moderno, soportado por Angular, y que es el que ocupa menos espacio. La contra es que no es soportado por Android ni iOS.

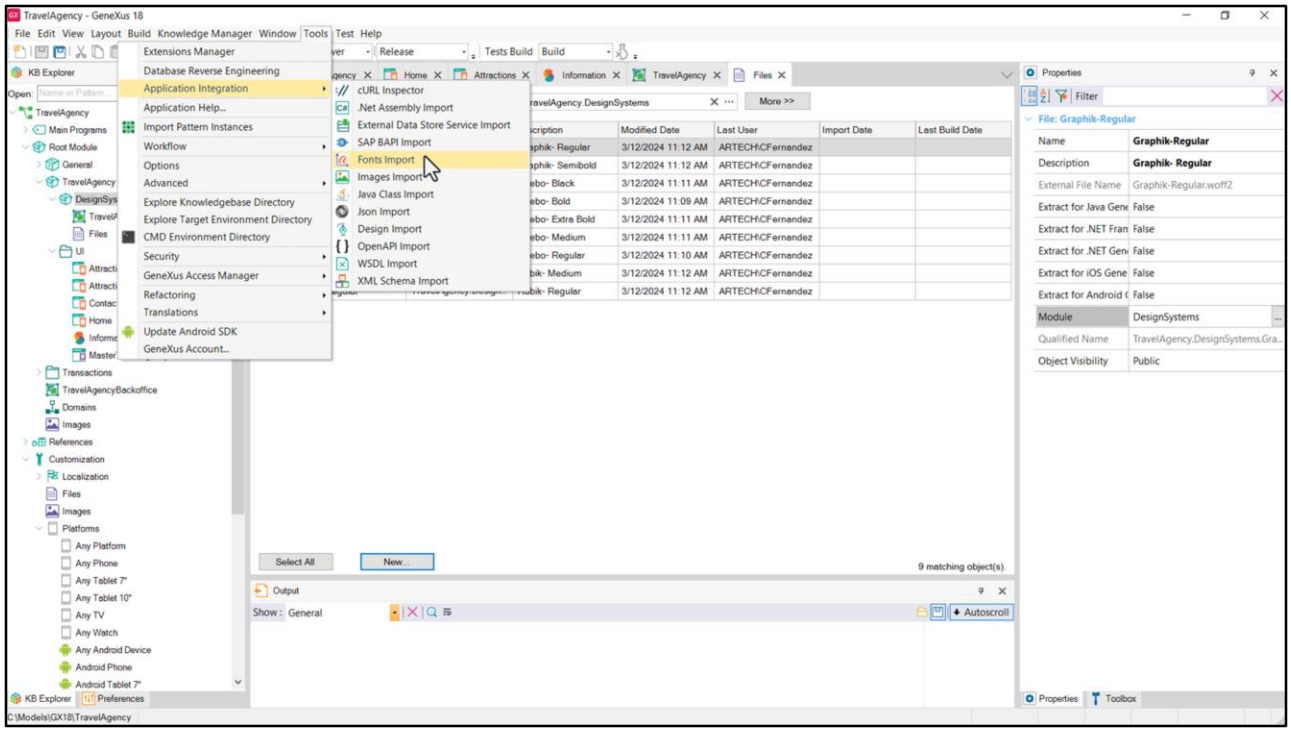
Bueno, entonces lo primero que haremos será insertar estos archivos de fuentes en la KB.



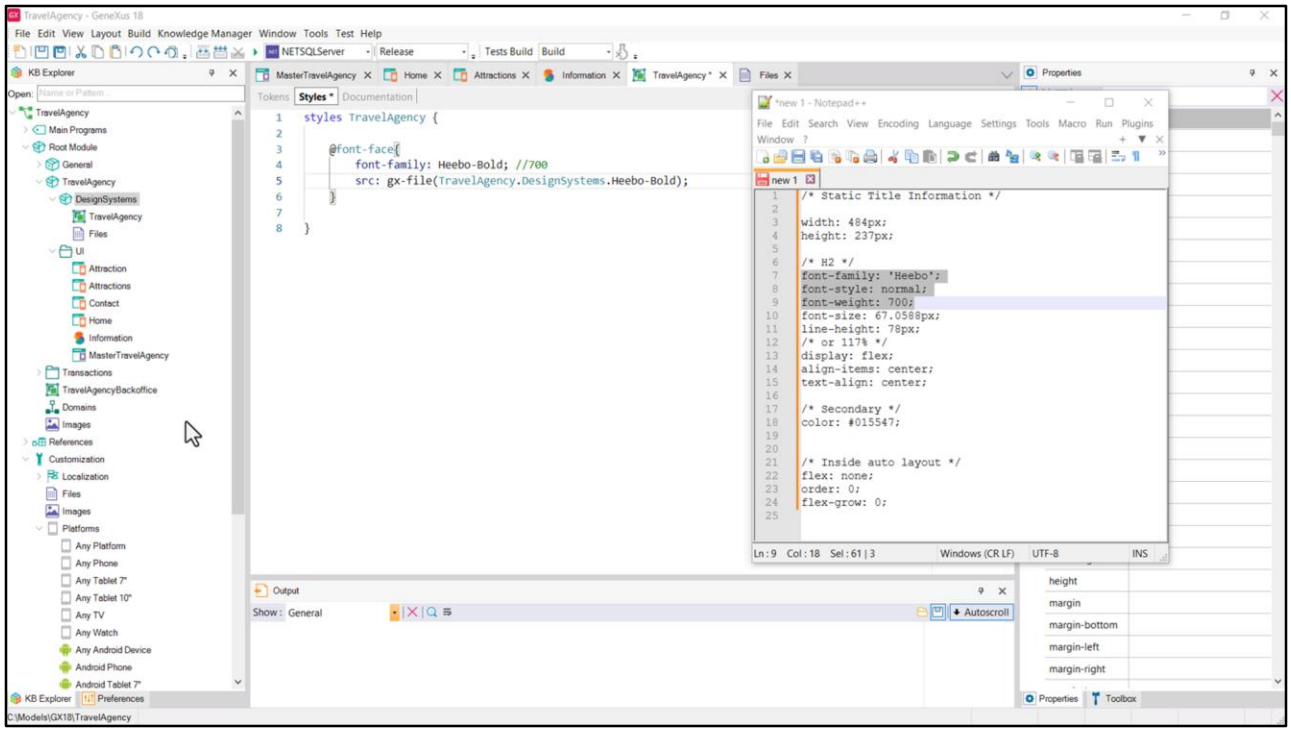
Podemos insertarlos de a uno, manualmente… dándoles un nombre y aquí eligiendo el archivo de nuestra carpeta…
Vemos que al insertarlo dentro del módulo DesignSystems se muestra separado de los demás archivos de la KB…



Y luego podemos seguir así, insertando cada uno por separado… hasta completarlos todos…



…O la otra opción, mucho más sencilla, es insertarlos todos juntos de esta manera… ahí indicamos el folder… y presionando Ok se insertarían todos en una sola operación (y si nos quedan en el módulo Root luego le cambiamos el módulo a cada uno a través de la propiedad).

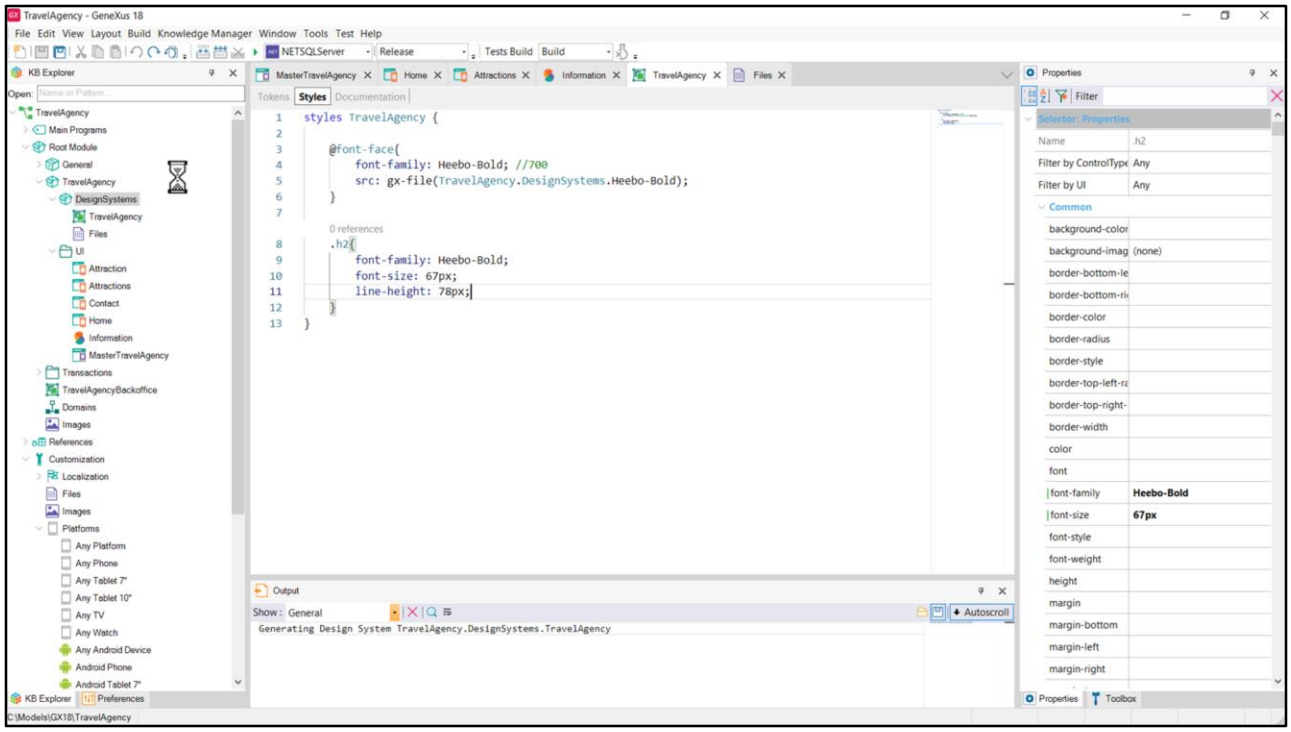


Bien, ya tenemos los archivos de fuentes en la KB. Empecemos por agregar el que corresponde a la fuente Heebo-Bold a nuestro Design System Object.

Lo primero será declarar el conjunto de estilos.

Y ahora sí, escribamos la regla font-face para agregar justamente la fuente.

Con la propiedad font-family le damos el nombre con el que nos referiremos a esta fuente en todo el DSO (voy a escribir como comentario el peso correspondiente, que es 700); y con esta otra propiedad le indicamos el source, es decir, dónde se encuentra. Para ello utilizamos la función de GeneXus “gx-file”… su parámetro es el nombre del archivo de la fuente, calificado con el módulo donde se encuentra.

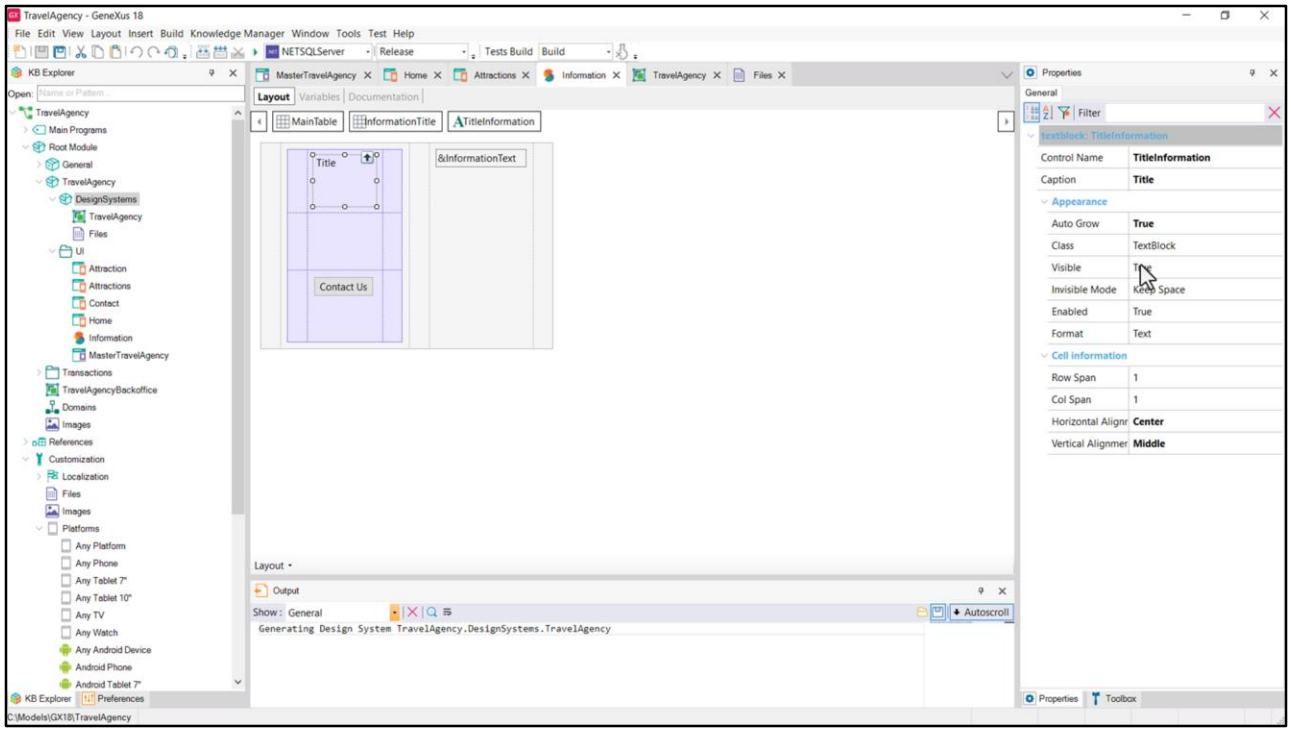


Ya estamos en condiciones de especificar nuestra primera clase, a la que llamaremos igual que nuestra diseñadora llamó al estilo tipográfico: h2.

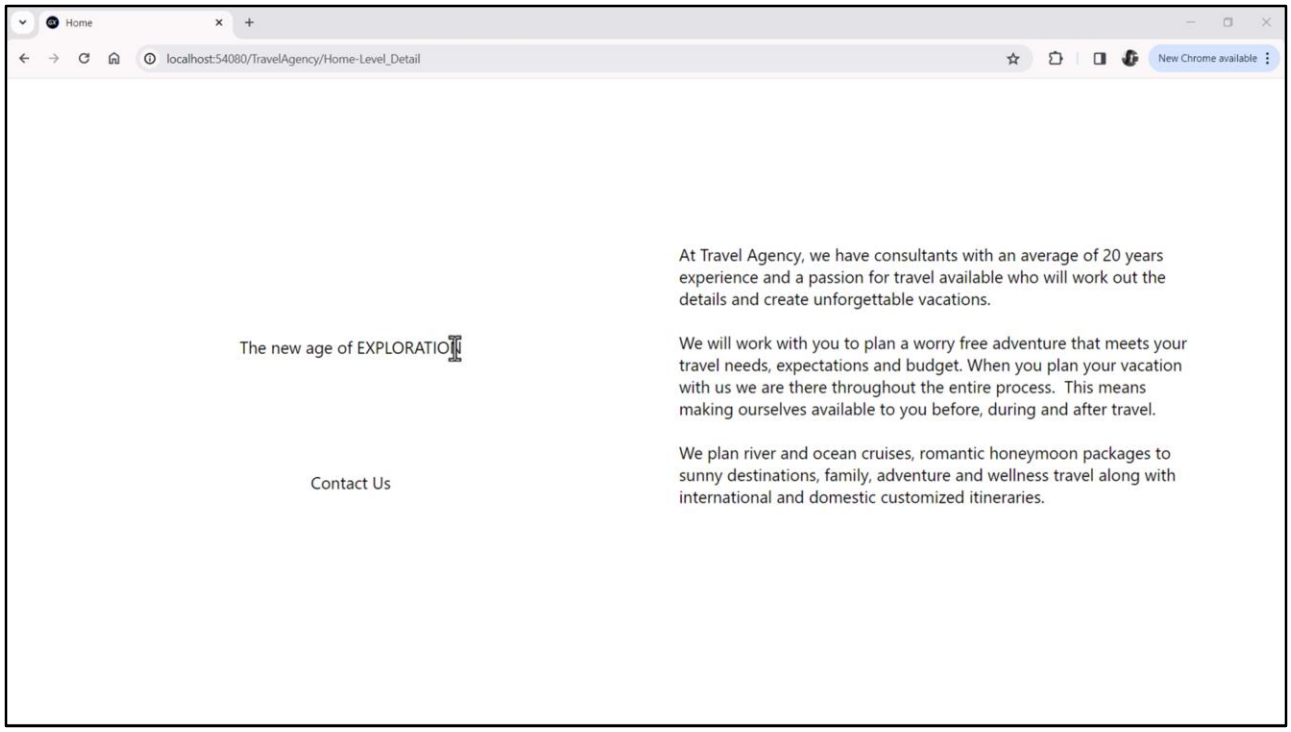
Recordemos que las clases de un DSO son en extremo parecidas a las de CSS, ya que para el caso Web se transformarán efectivamente en CSS. Por eso para seleccionarlas primero debemos escribir punto y a continuación el nombre de la clase.

Las primeras tres propiedades se transformarán en una, la font-family, que aquí referirá a este nombre, justamente. No necesitamos variar ni el estilo ni el peso de la fuente, porque estamos usando una con exactamente ese estilo y peso. Siempre usaremos la fuente exacta, para evitar deformaciones y es por eso que integramos un archivo de fuente para cada peso.

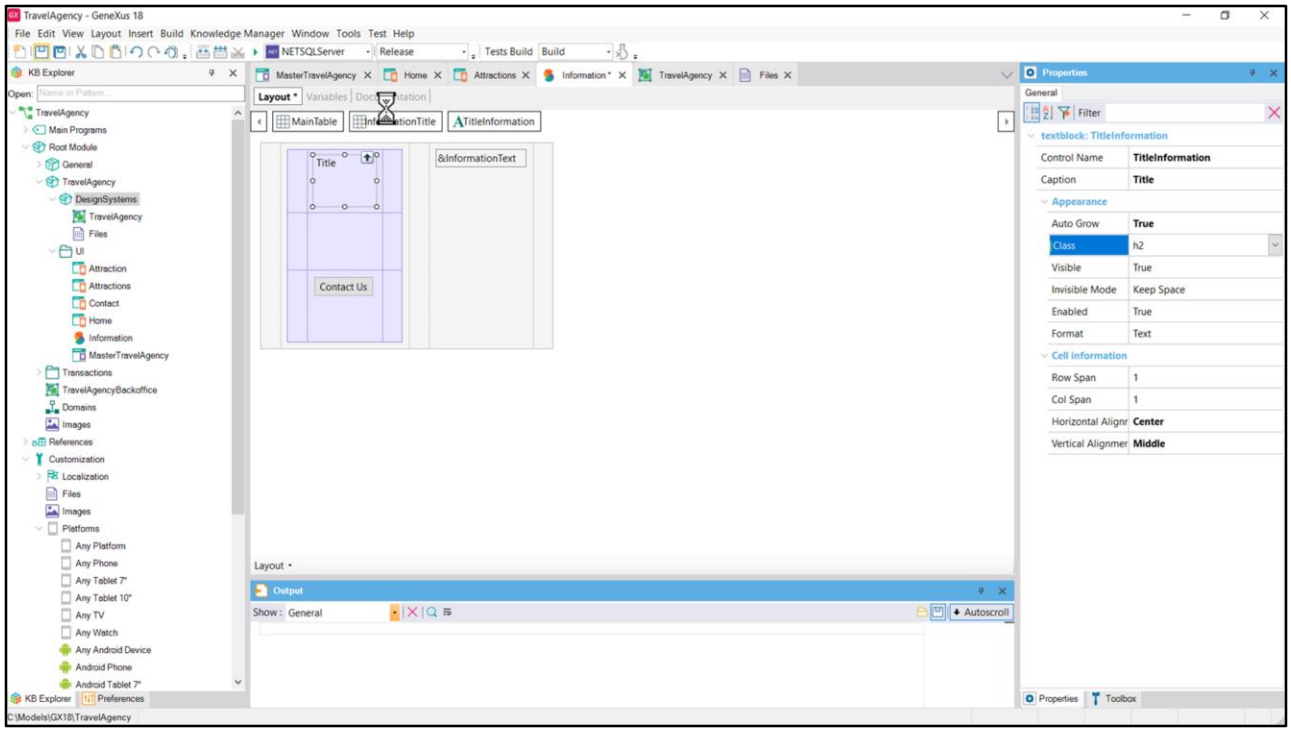
Luego sí utilizaremos tal cual estas otras dos propiedades, quitando los decimales.



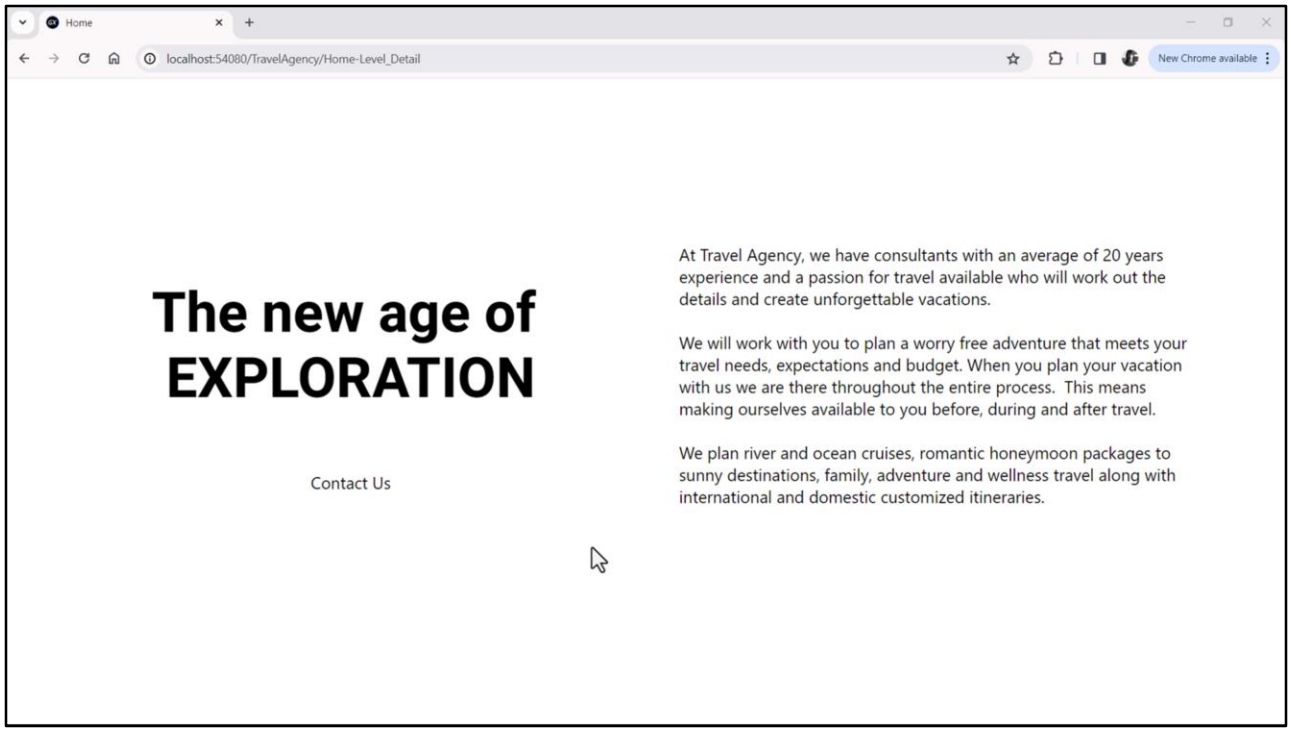
Grabemos y antes de probar anda observemos qué clase tiene asociada este Textblock. La clase de este nombre.



Antes de cambiársela por la nueva, vamos a ejecutar la aplicación, para poder comparar.

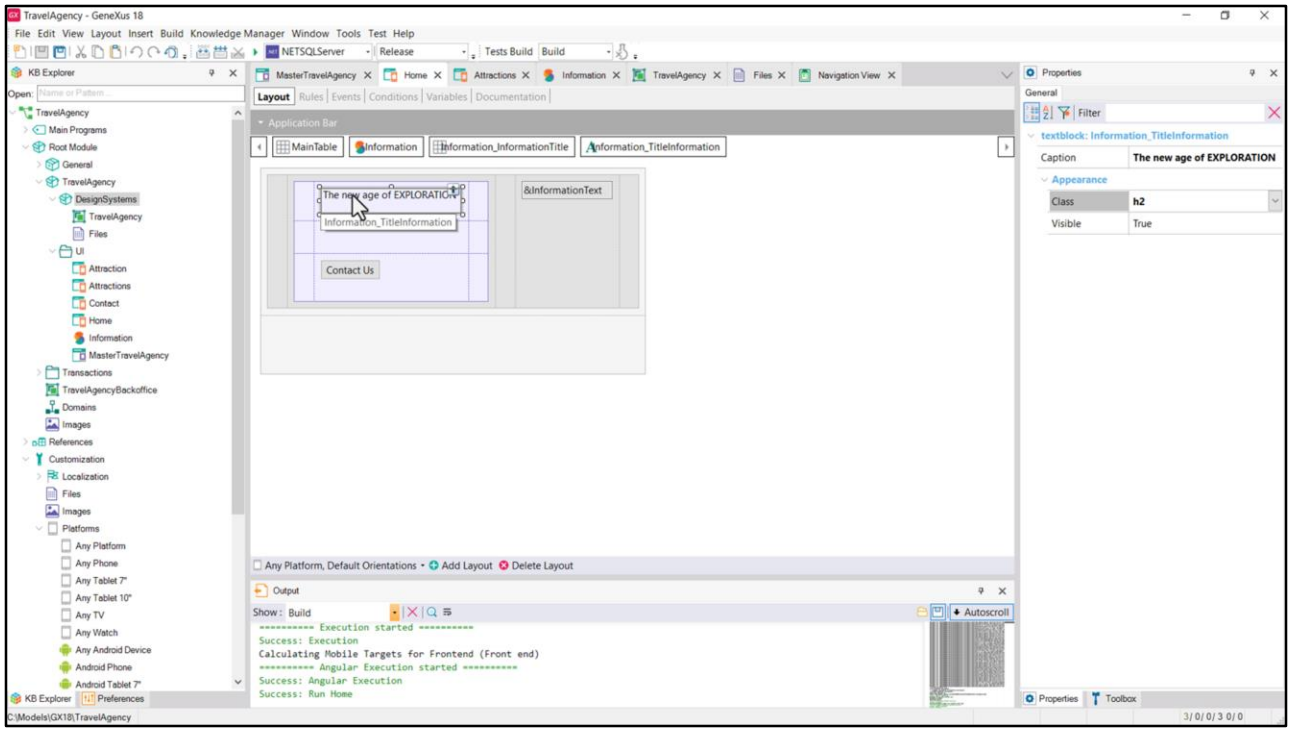


Ahora sí, vayamos al stencil… y vamos a cambiarle la clase de la TextBlock por la nueva, h2. Grabemos. Y ejecutemos, Run.



Y aquí lo vemos claramente.

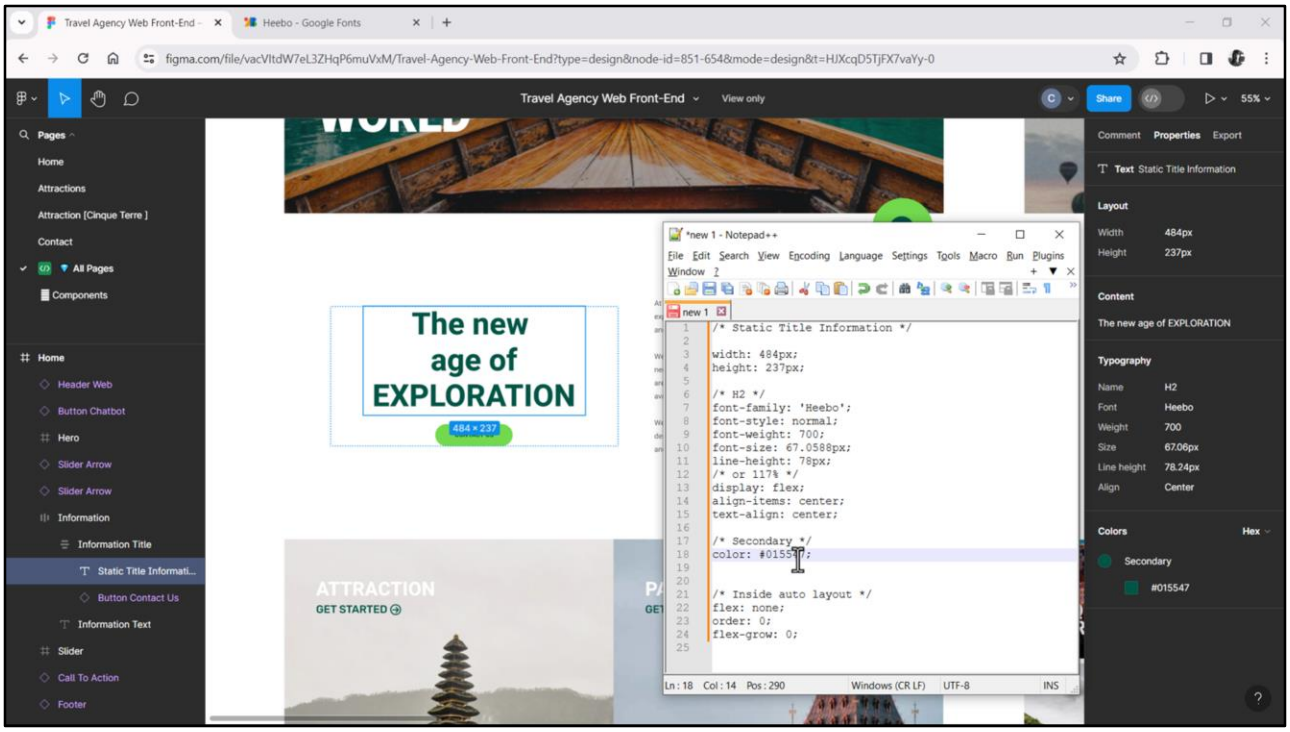
(Si uds están prototipando en la nube, lo que no es tan recomendable durante el tiempo de desarrollo, y no ven el cambio en su ejecución, vuelvan a cargar limpiando el caché –en Windows con Ctrl+F5).



Un par de observaciones antes de seguir:

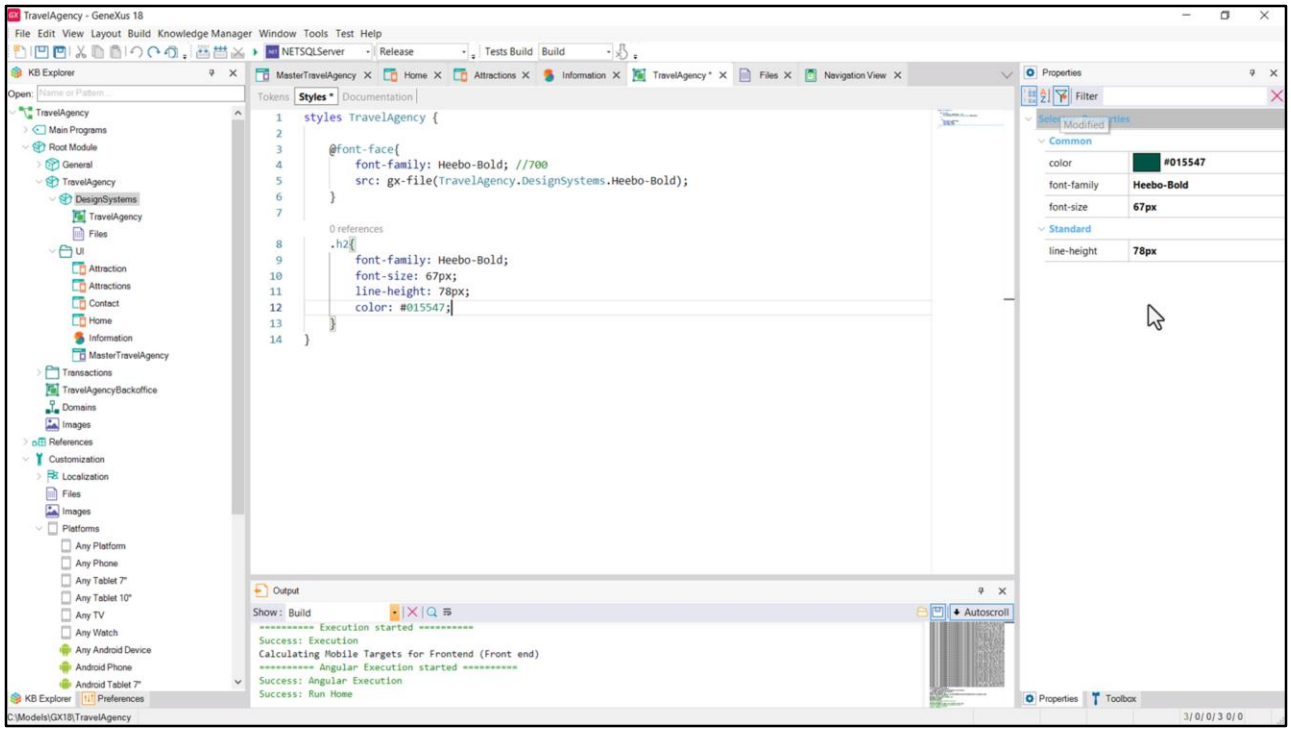
El textblock del Stencil tenía antes de este cambio un nombre de clase (TextBlock justamente) que no está especificada en el DSO, y eso no trajo ningún problema. Tomó los defaults del navegador.

Por otro lado, al haberle cambiado la clase, automáticamente ese cambio se efectuó en los dos paneles en los que ese stencil está instanciado… Por eso lo vemos acá… y también lo veremos en este otro caso.



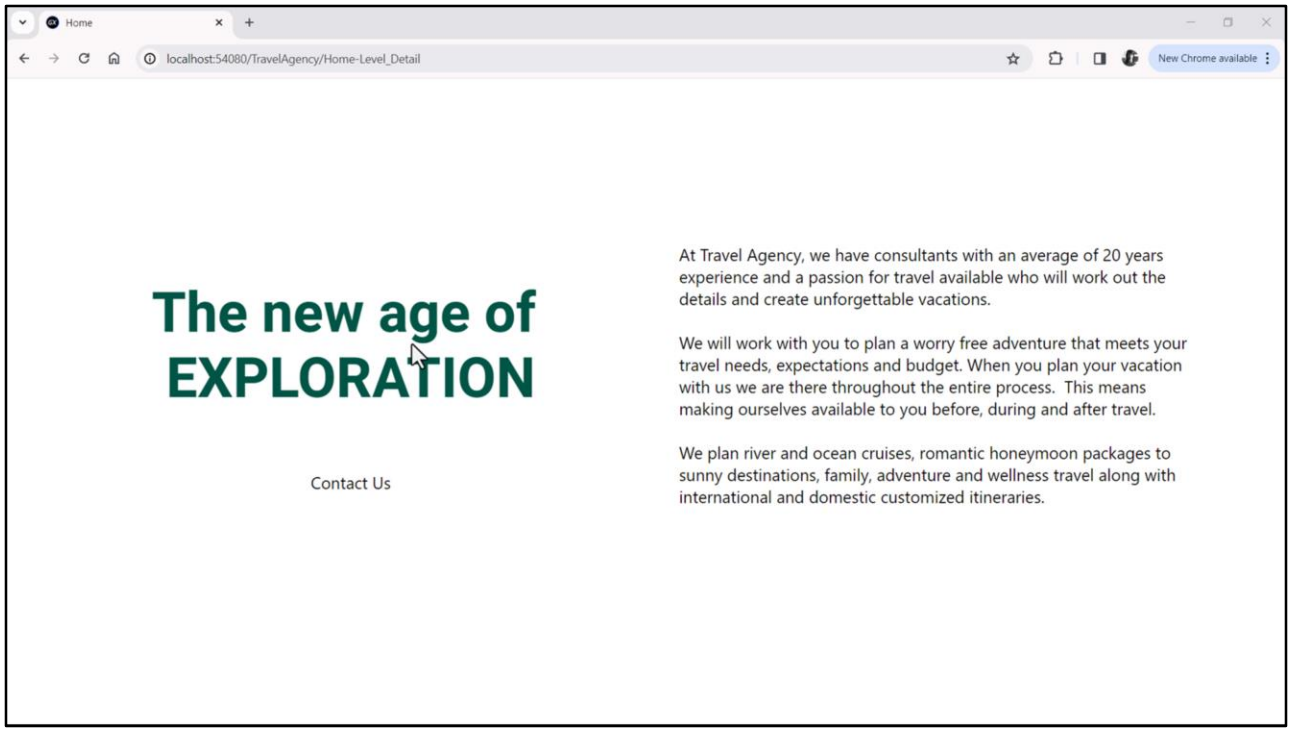
Bien, ahora, ¿qué pasa con el color del texto? Vemos que será este tipo de verde, el que tiene este valor hexadecimal, al que Chechu había denominado Secondary, creando un estilo con este nombre, un estilo de color. Si vemos las propiedades CSS que habíamos copiado, tenemos justamente así indicada la de color, con el valor hexadecimal.

Podríamos sin pensar mucho copiar esta propiedad CSS…

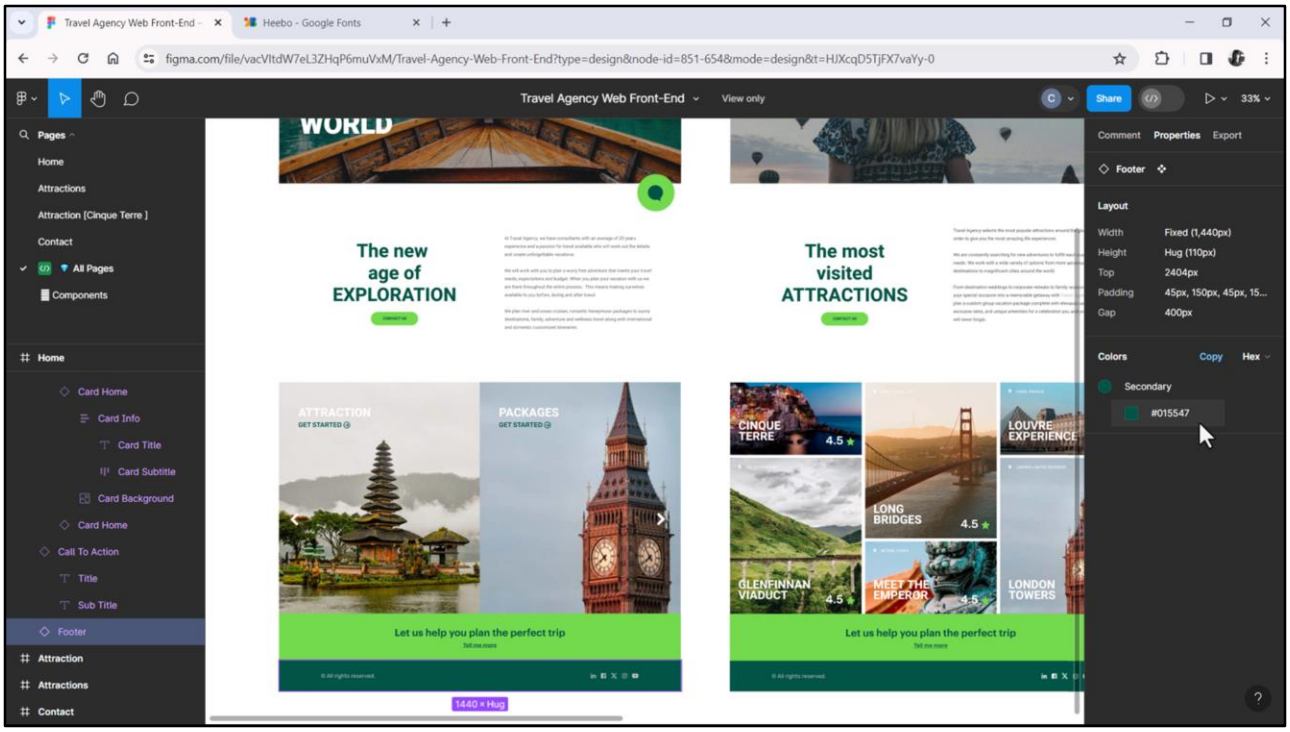


… y venir a nuestro DSO y pegarla. Vemos cómo en la ventana de propiedades muestra todas las propiedades posibles de escribir dentro de una clase y aquí están mostrándose las que utilizamos. Podemos filtrar por las que hemos modificado para tenerlas a la vista.

Grabemos y probemos esto en ejecución.

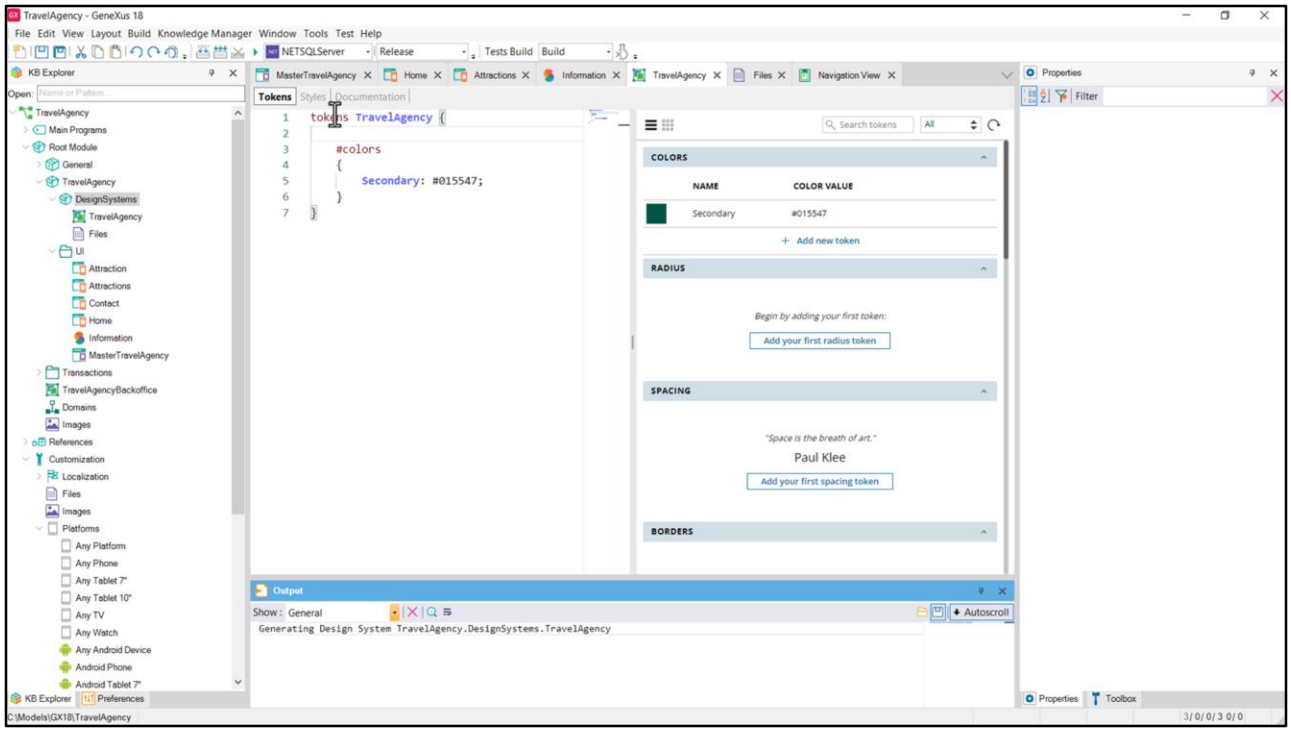


Y allí lo vemos, tal como esperábamos.



Ahora bien, ese exacto color no solamente se va a utilizar aquí para este texto, sino para este otro, para este fondo, para este otro texto, para este otro… es decir, se va a repetir en muchos lugares.

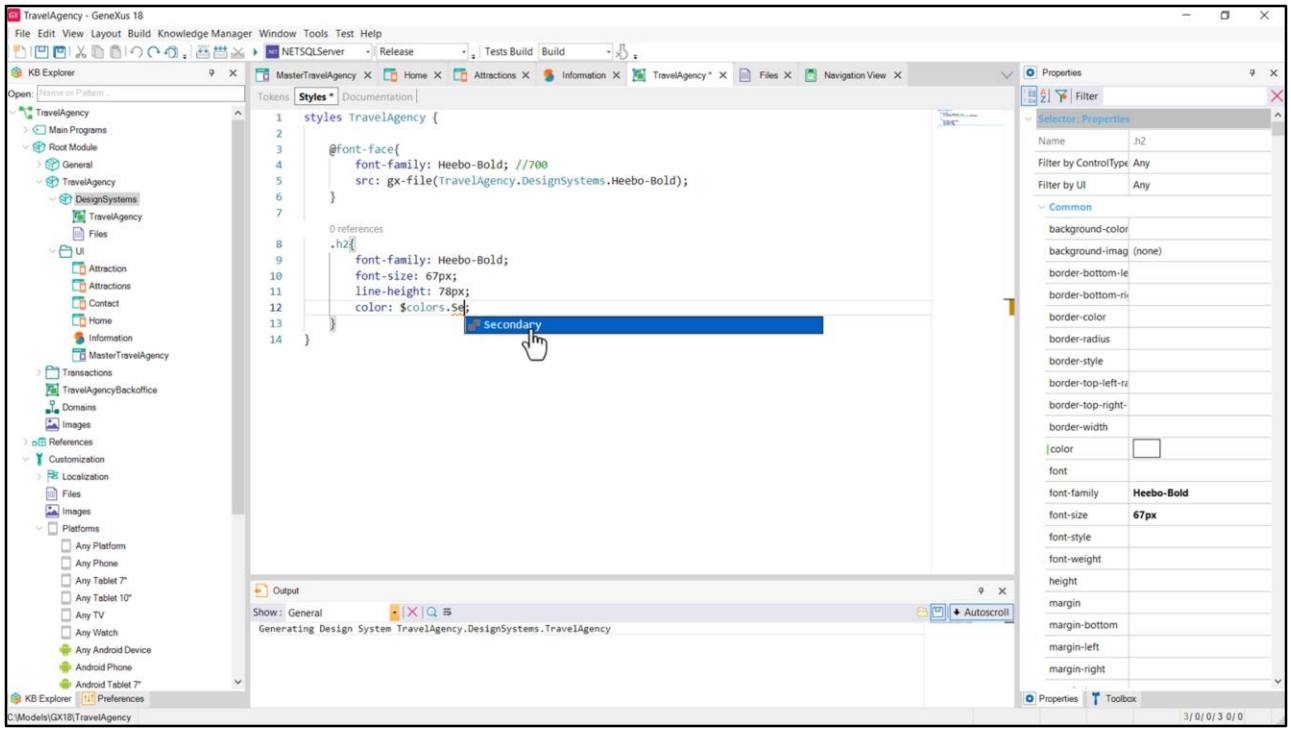
Es que este valor responde a un concepto semántico, a un color semántico, es el color secundario de la aplicación, siendo el primario este otro verde. Es por eso que Chechu lo definió como un estilo de color y nosotros lo que haremos será definirlo como un token en nuestro DSO.



Será un token de color… que llamaremos Secondary… y su valor será ni más ni menos que este de aquí.

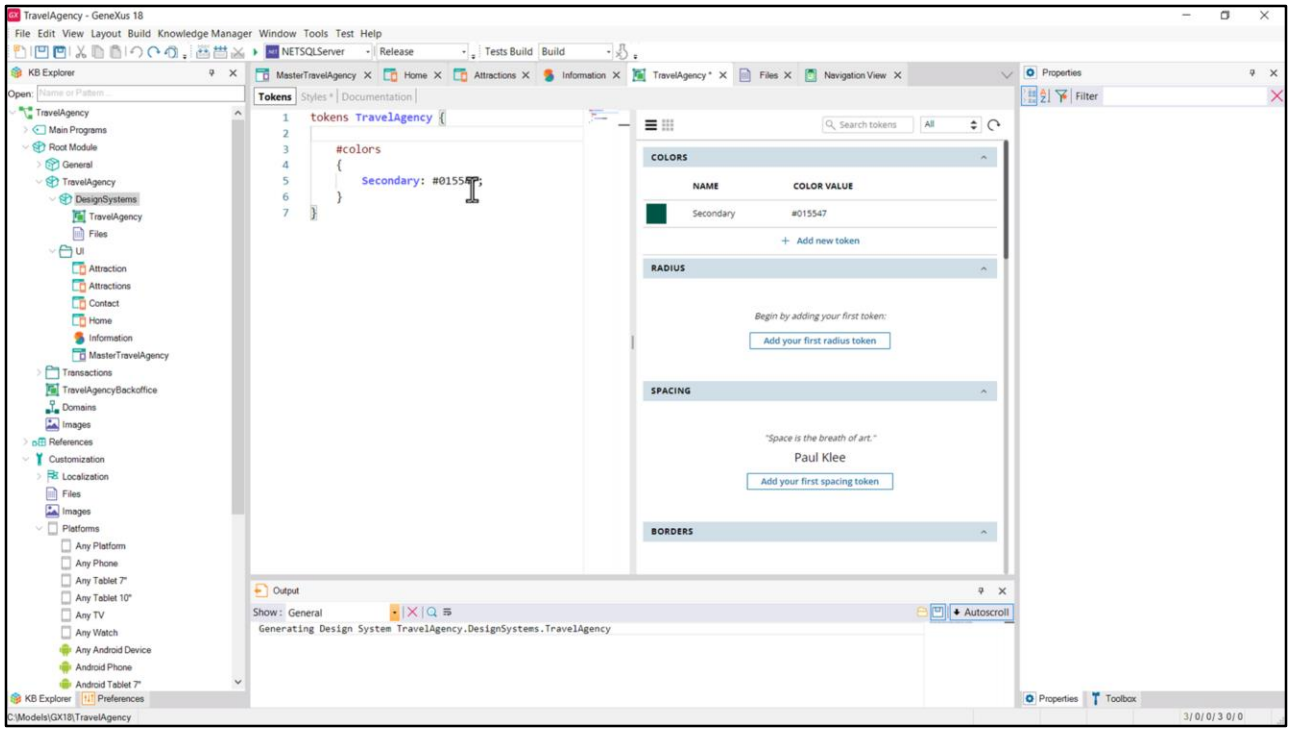
El editor de tokens es doble, recordemos. Podemos editarlo directamente como texto o podemos editarlo desde el editor gráfico.

Y ahora grabemos…



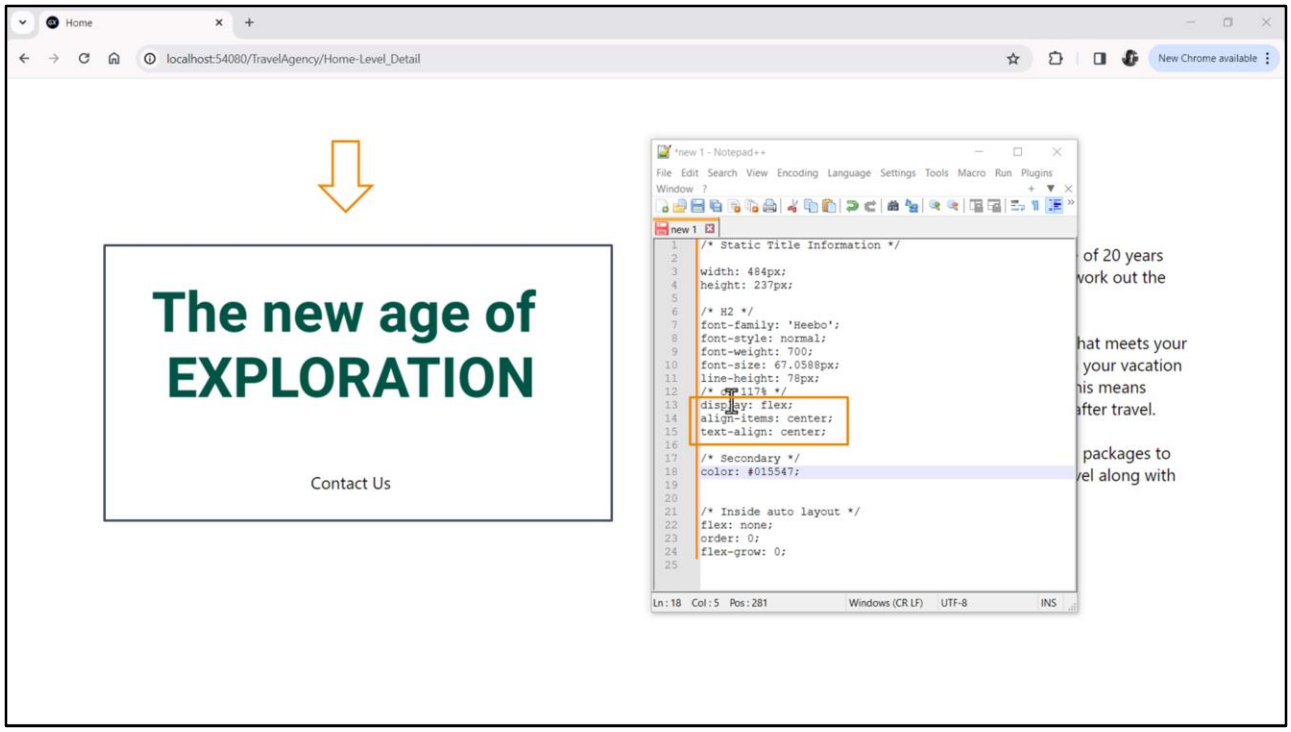
…y vayamos a la solapa de estilos y modifiquemos el valor por el Token. Con el signo de pesos estamos diciendo que estamos queriendo referenciar aquí un token, un token de color… (ahí está, tiene que quedar así), y al colocar el punto vemos que al comenzar a digitar las primeras letras del token ya me ofrece los que empiezan con esas dos letras.

Bien, de esta manera, entonces, le estamos diciendo que la propiedad de color tome su valor a partir del token de color de este nombre, Secondary.



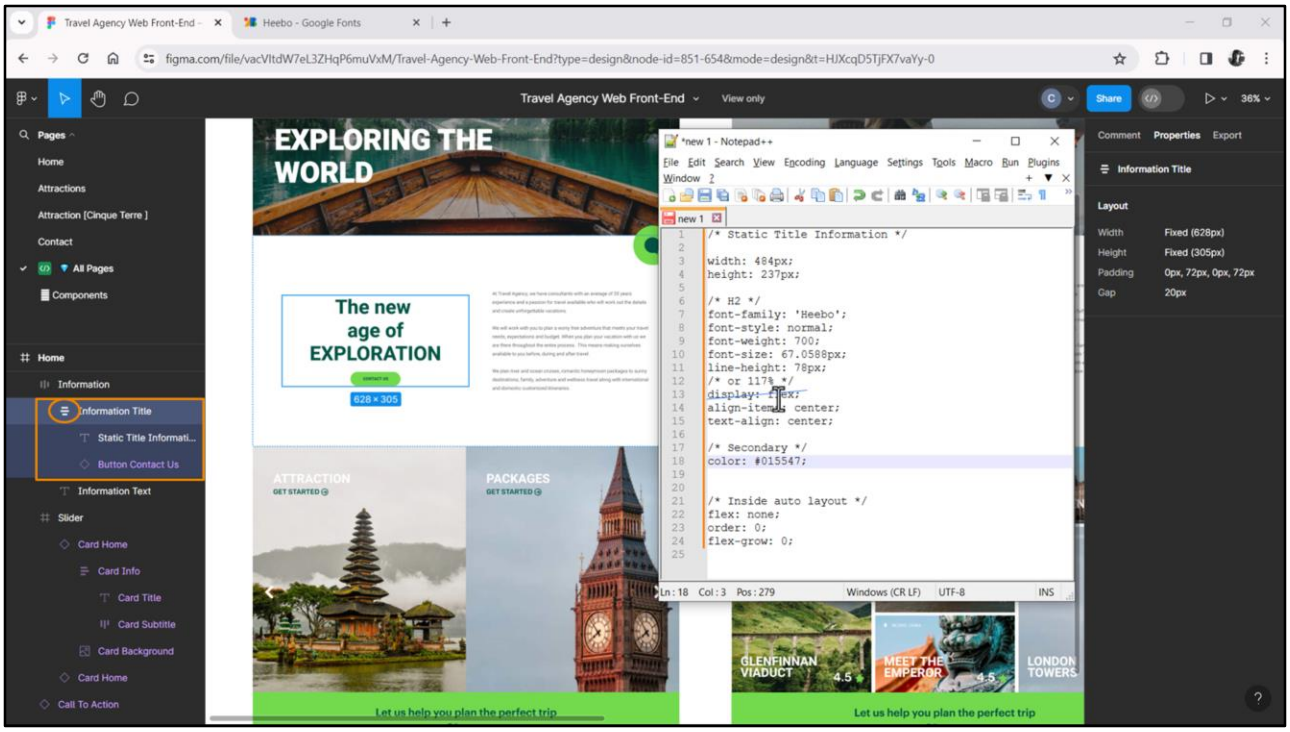
Y eso va a hacer que si después, más adelante, Chechu quiere cambiar ese color de este valor por cualquier otro valor, simplemente vendremos a cambiarlo aquí. Y no tendremos que tocar en absoluto la clase. Bien, grabemos.

Y en ejecución no vamos a notar absolutamente ningún cambio porque seguiremos viendo ese color verde.



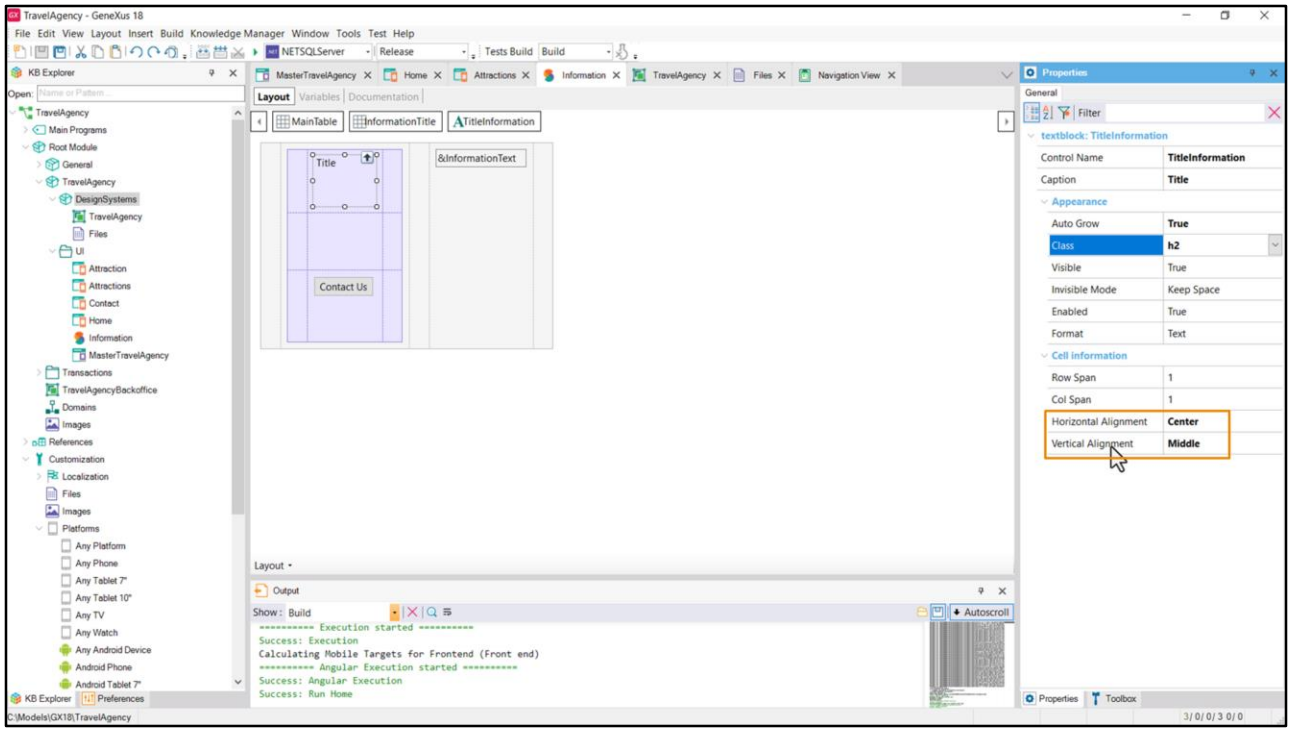
Bien, ¿y qué pasa con estas tres propiedades que no copiamos a nuestra clase h2?

Las primeras dos tienen que ver con el contenedor que se utilizó en Figma para colocar dentro estos dos textos. Y en particular este.



Se trata de un grupo o un frame que se catalogó como Auto Layout y del que vamos a hablar luego, pero que, ya podemos adelantar, se implementa con un contenedor Flex. Nosotros fuimos por otro lado, que fue elegir una tabla para implementar este contenedor.

Por tanto la primera propiedad, display, no la necesitamos para nada. Y estas dos propiedades nosotros las implementamos…

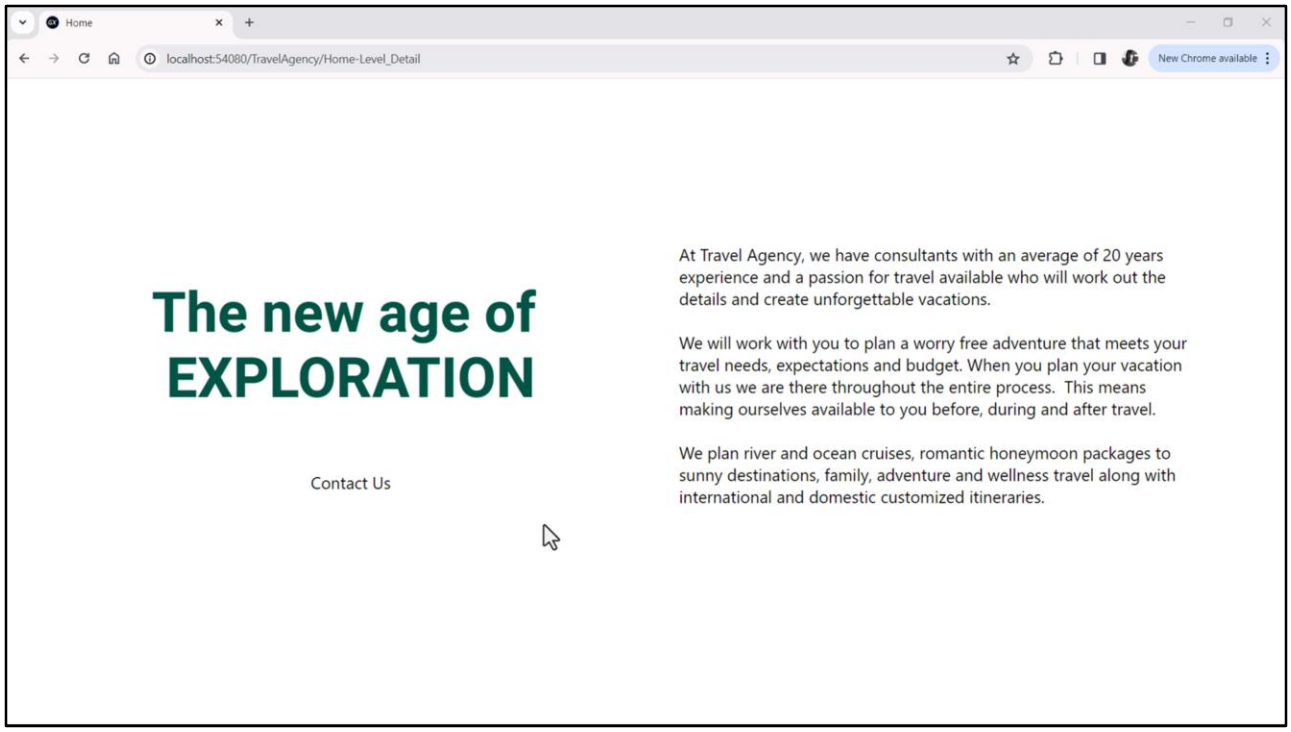


… a través de las alineaciones horizontal y vertical... de los dos elementos.

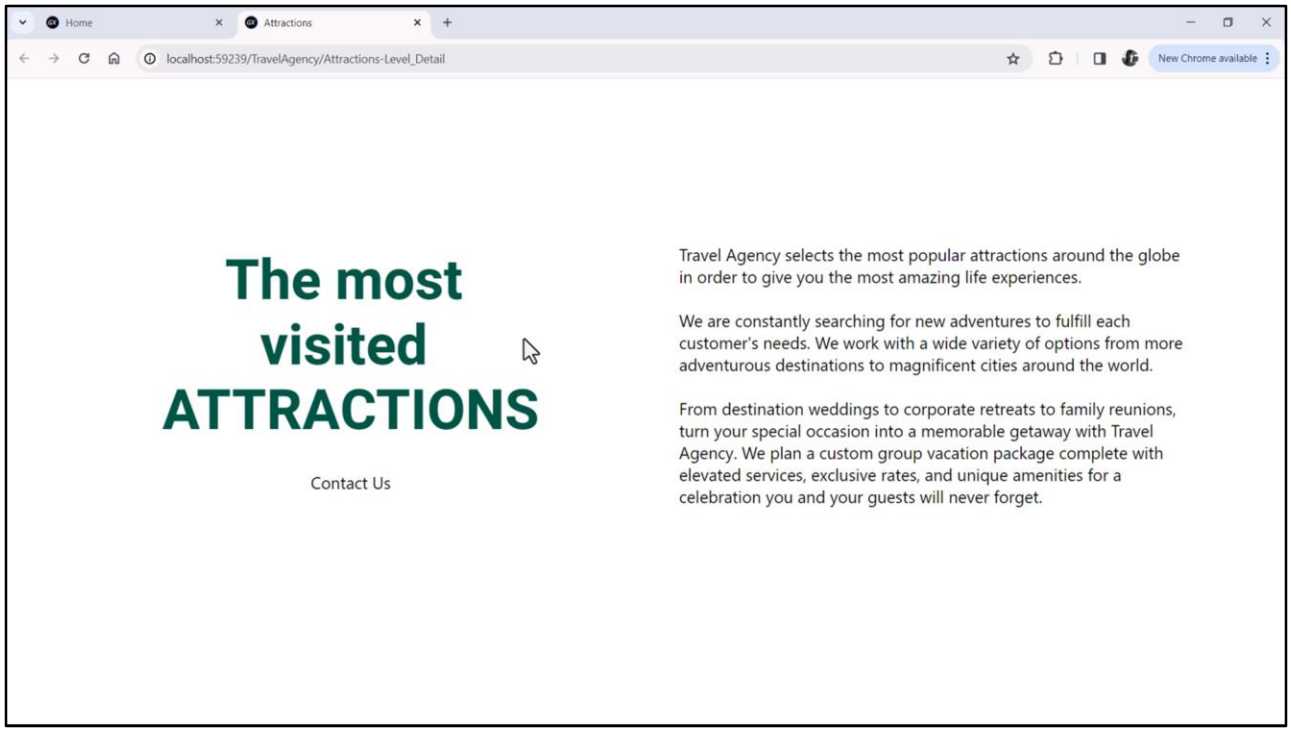
Así que no necesitamos copiar ninguna de estas propiedades a nuestra clase.

De esta manera nos quedó bien separado lo que es el estilo propiamente del control de lo que es más estructural y que tiene que ver con la alineación del control en el layout. Esta es una posibilidad, claro, y no necesariamente es la mejor.

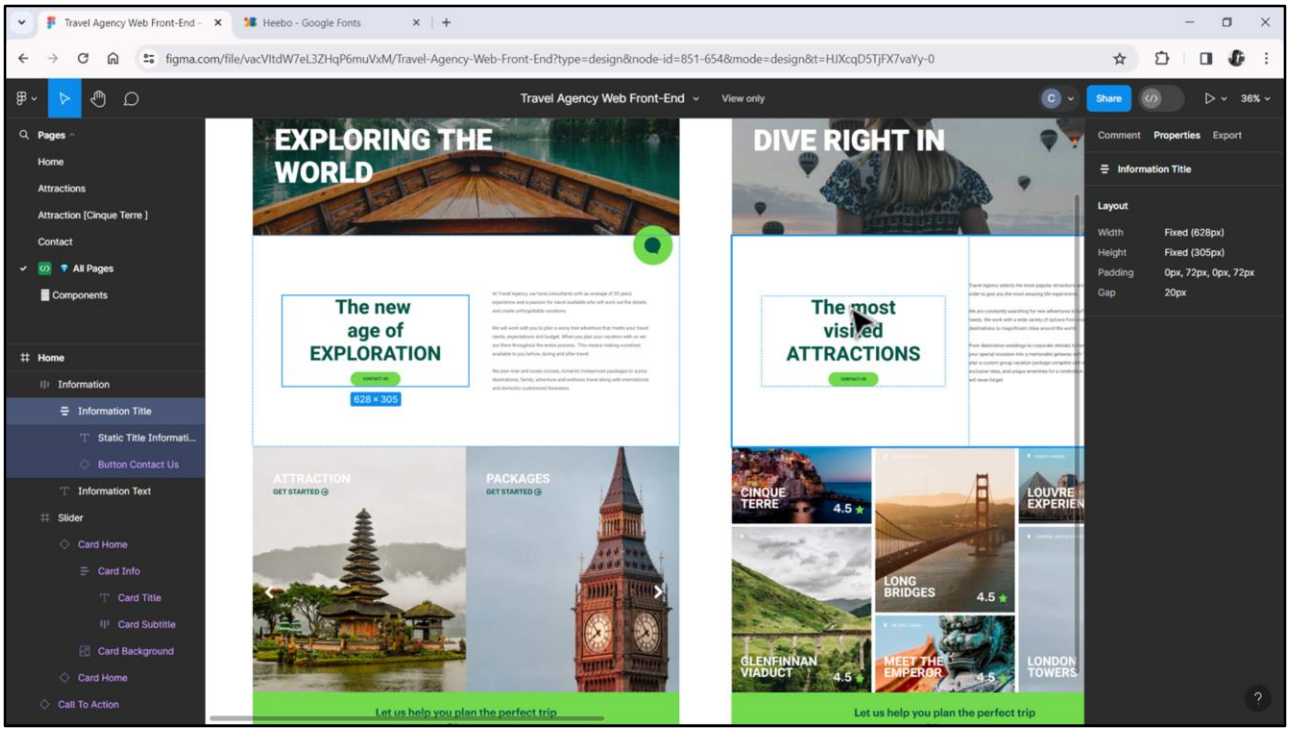
En la implementación que estamos llevando a cabo, la parte del espaciado y de las alienaciones las estamos resolviendo en el propio layout, y el resto del estilo lo estamos manejando en el DSO. Un DSO es más fácil de cambiar que un layout, por lo que tomar estas decisiones tiene sus pros y sus contras.



Una última cosa antes de terminar, en ejecución el texto de Home, por su tamaño, se está viendo en dos líneas… veamos qué pasa con el de Attractions.

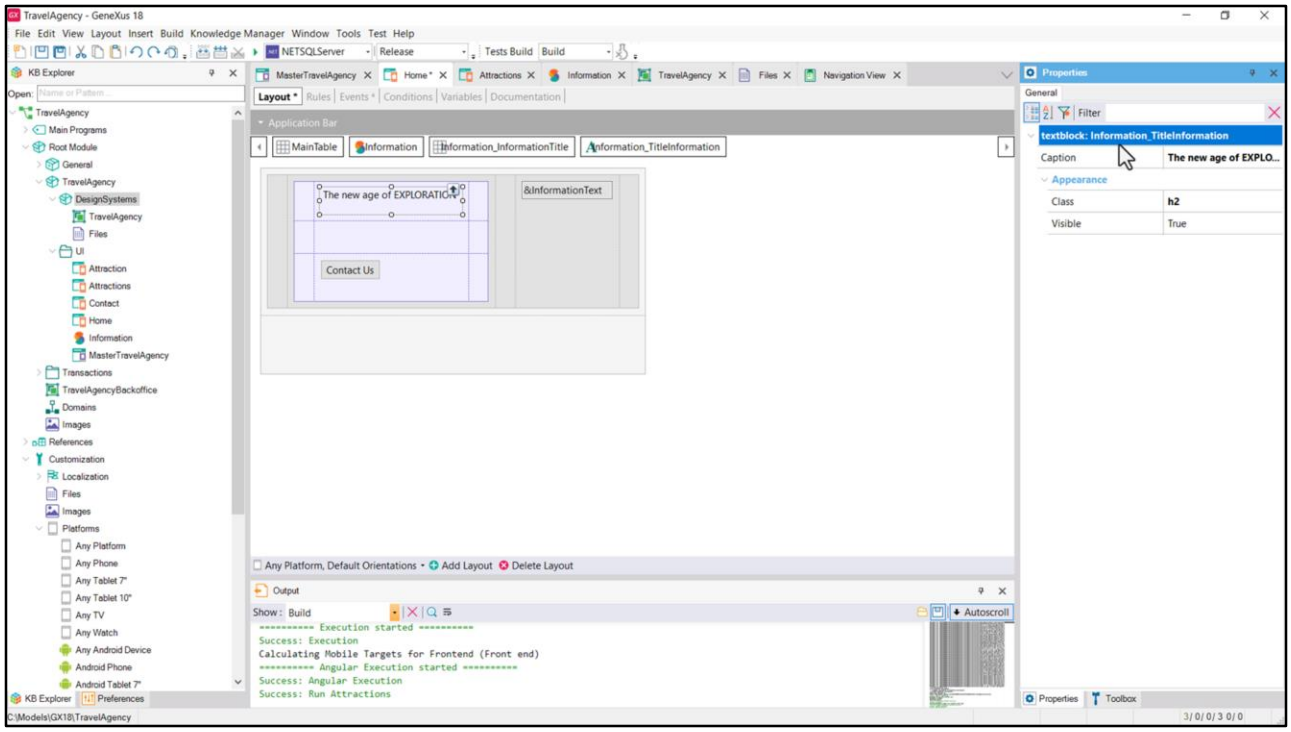


Este sí se ve en tres…

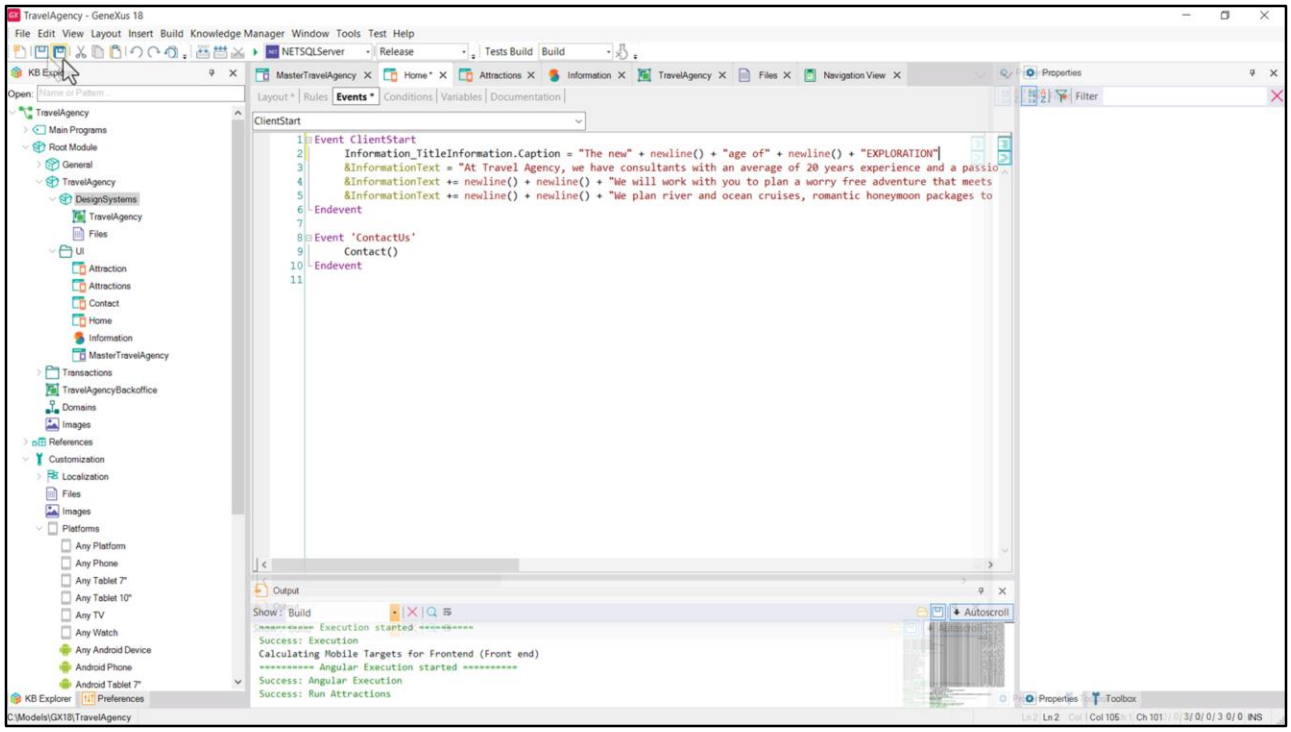


… tal como vemos en el archivo en Figma.

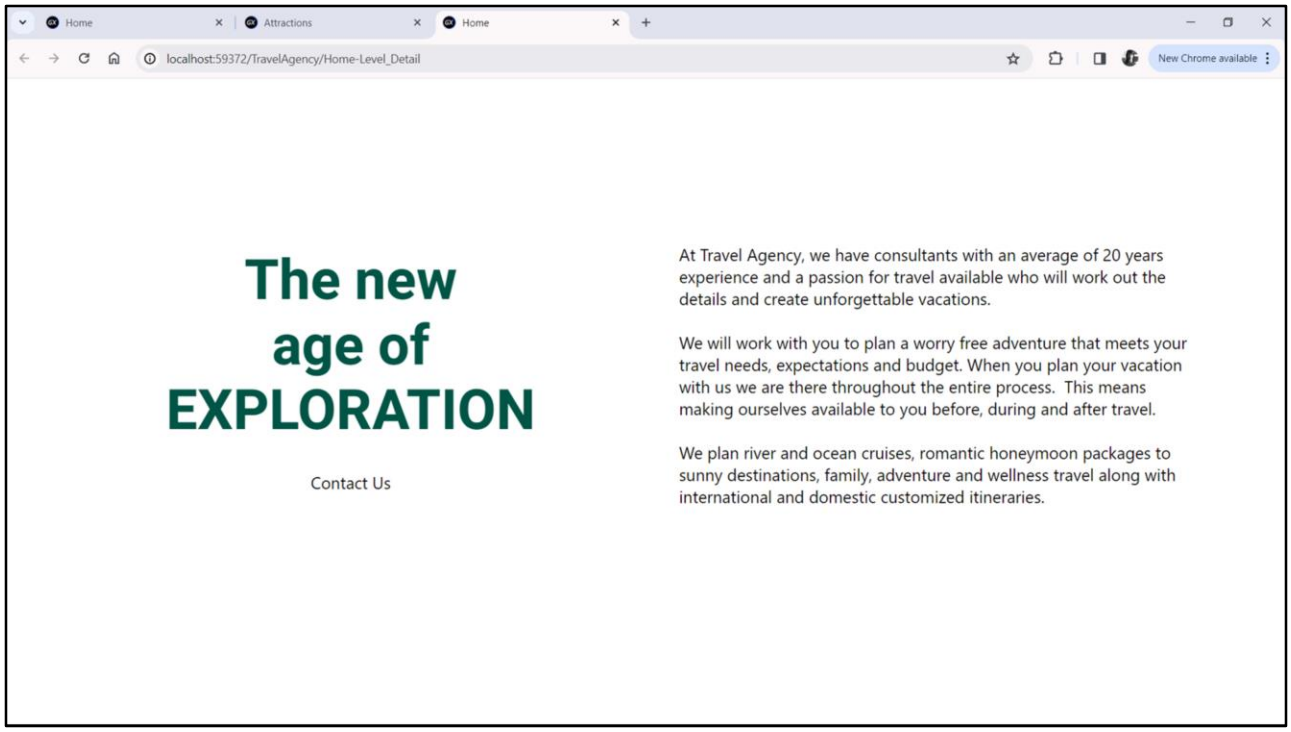
Entonces lo que tendríamos que hacer es separar en tres líneas este texto.



Una forma de hacerlo es: vemos acá el nombre del textblock, que es el que proviene del stencil, Information, infraguión y el nombre del control en el stencil, TitleInformation.



Entonces vengo al Home y en los eventos, en el ClientStart, al Information Title Information punto Caption le asigno “The new”…
Grabo y ejecuto.



Y aquí lo vemos.

Bueno, ahora pasemos a dar estilo al párrafo y al botón… en el video siguiente.


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