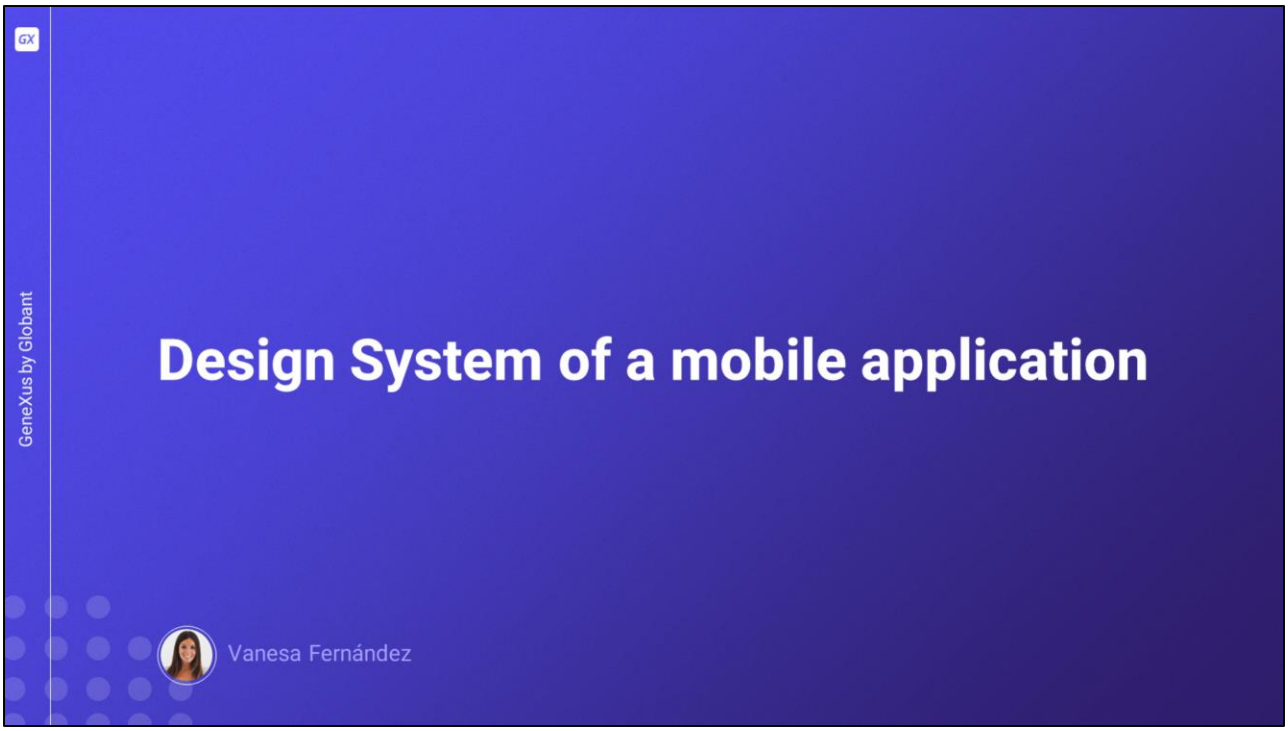
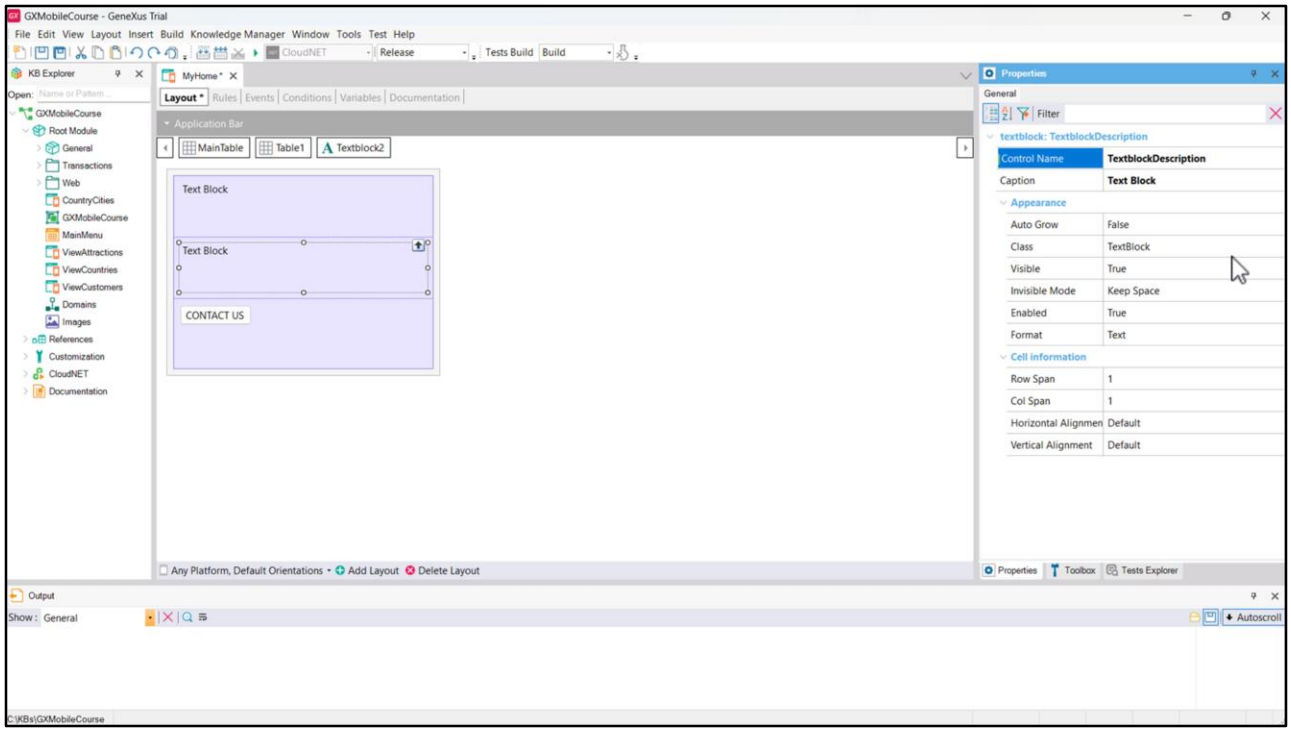
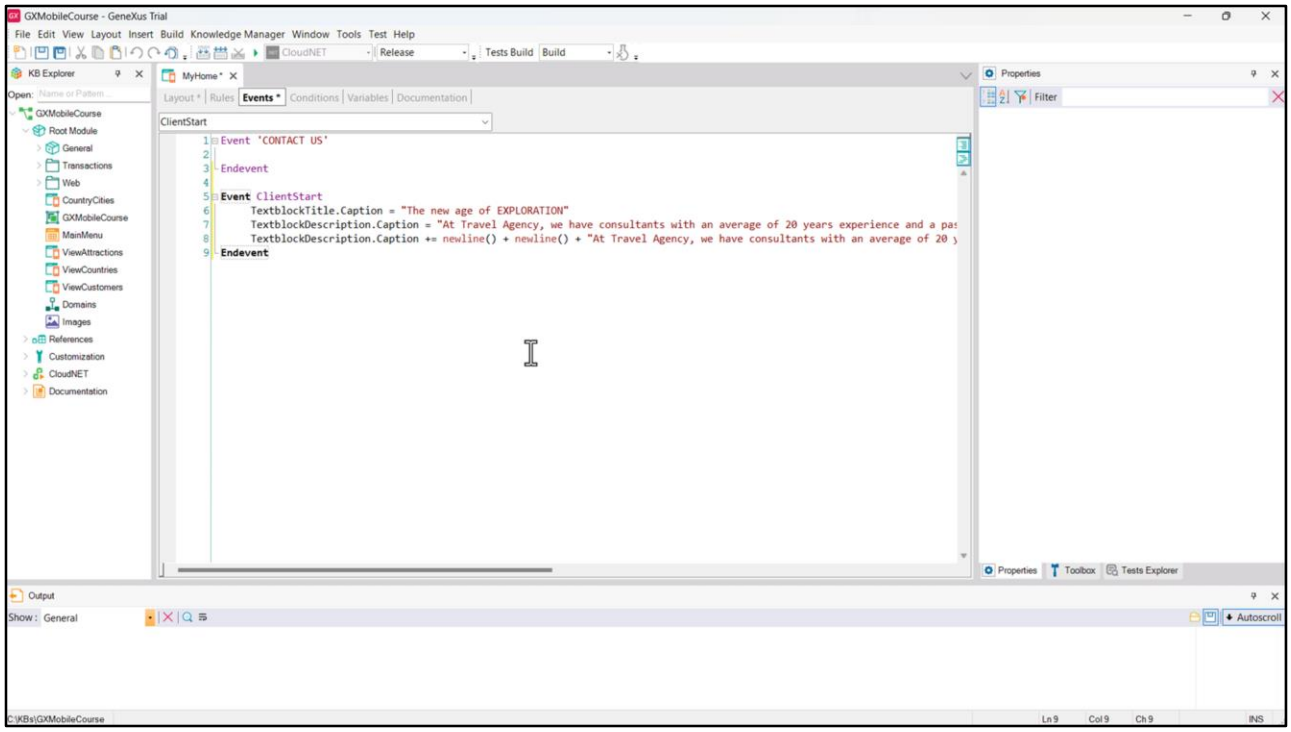


En este video estudiaremos cómo trabajar con el objeto Design System para darle el diseño a nuestra aplicación. Veremos, entre otros, cómo utilizar tokens, estilos, propiedades, incorporar fuentes y trabajar con el modo claro y oscuro.

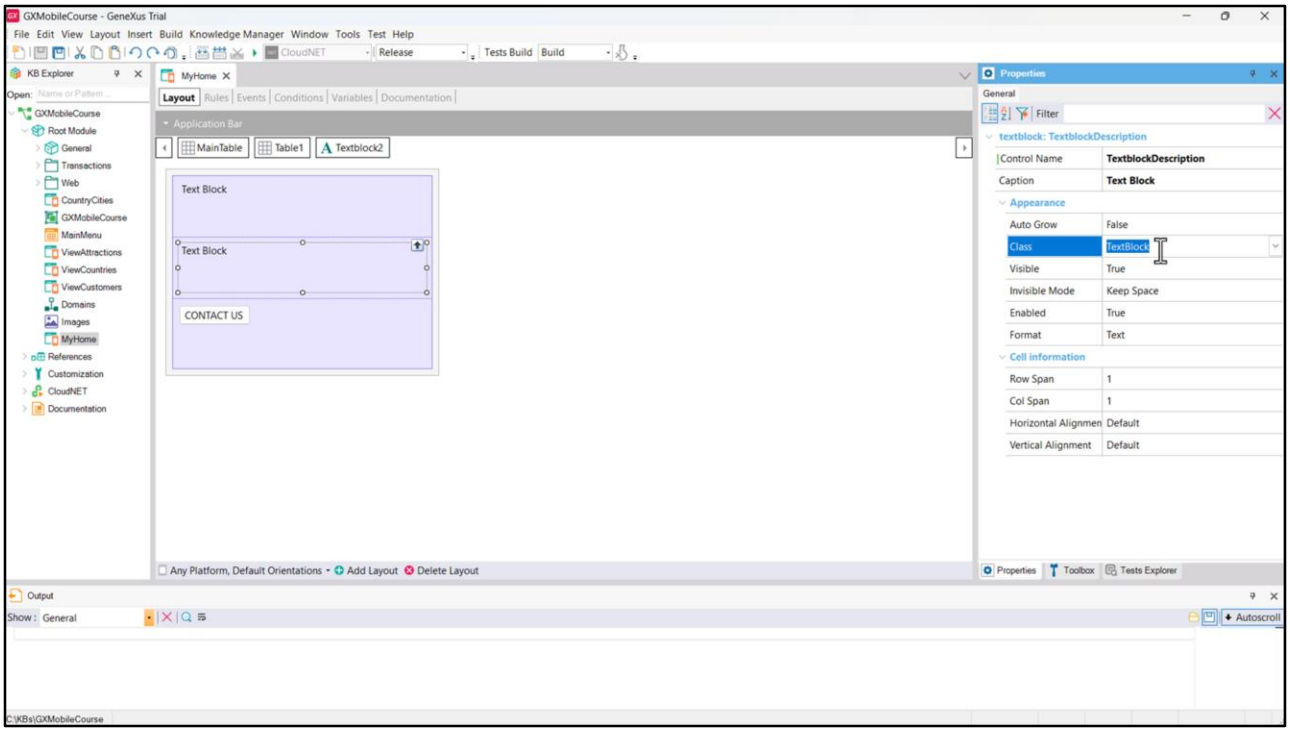


Con el único fin de hacer algunas pruebas, creemos un Panel de nombre MyHome, al que le configuraremos la propiedad Main program en True para poder ejecutarlo fácilmente, ya que no tendrá dependencias.

Agreguemos una tabla, y dentro de ella 3 elementos: 2 textblocks, y un botón, al que le pondremos “CONTACT US”. Cambiemos el nombre del control del primer textblock a TextblockTitle… y el de la segunda a TextblockDescription.

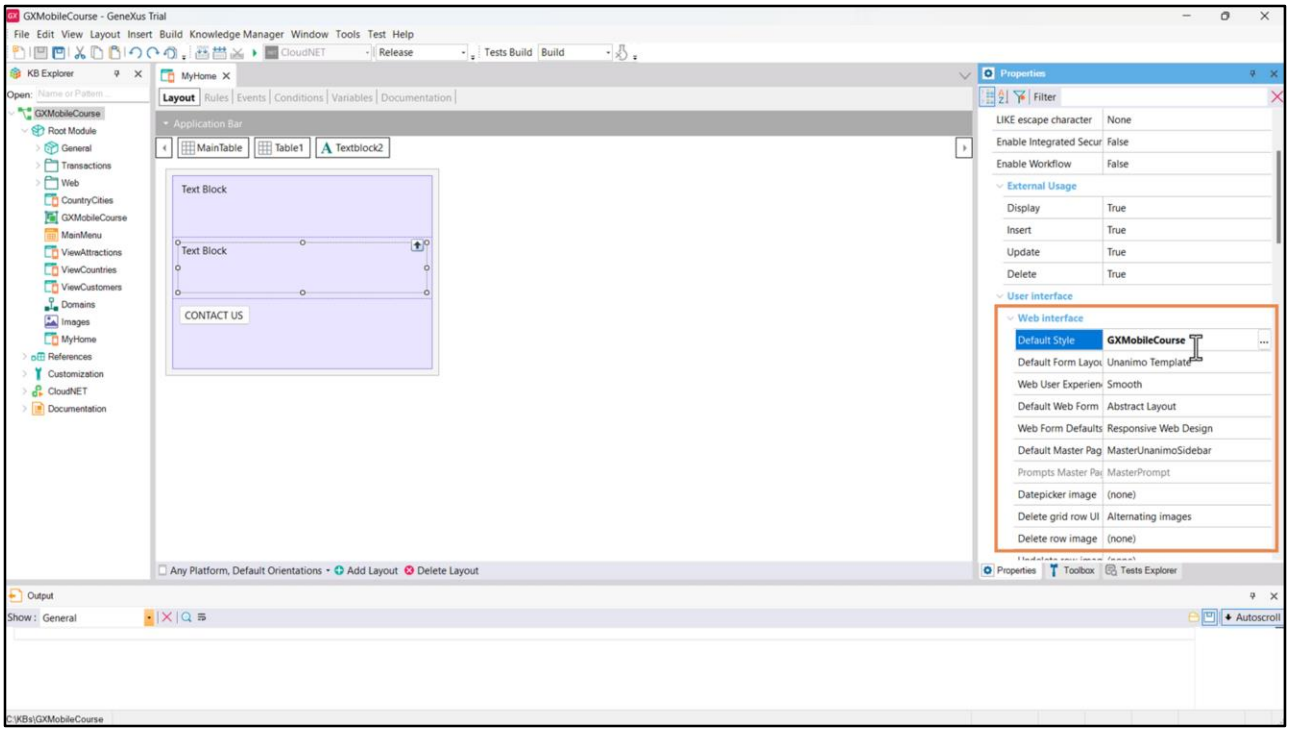


Vamos a los eventos, y agregamos el ClientStart, que se va a ejecutar cuando se abra el Panel. Vamos a cargar los párrafos por separado para poder agregar la nueva línea, que nos permite establecer la separación entre ellos.

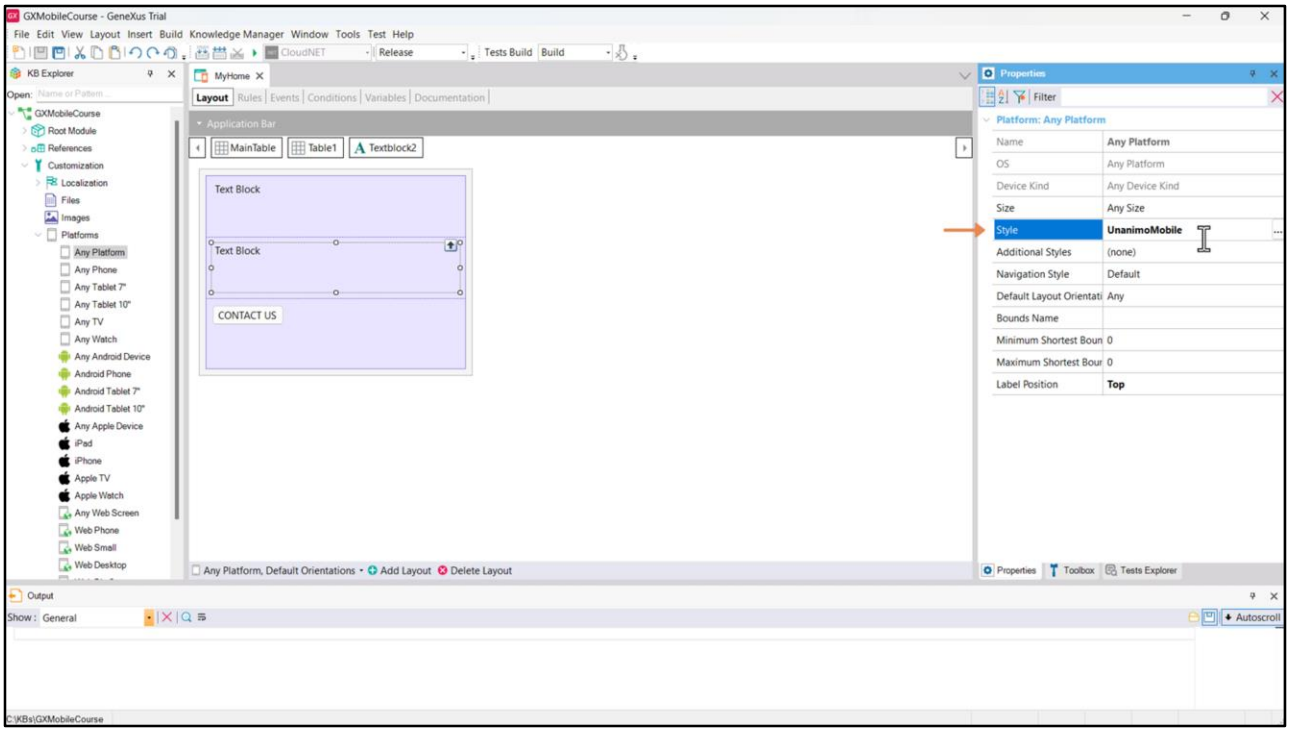


Si editamos un control TextBlock de los que agregamos, vemos que tiene la propiedad Class, que por defecto tiene el valor TextBlock.

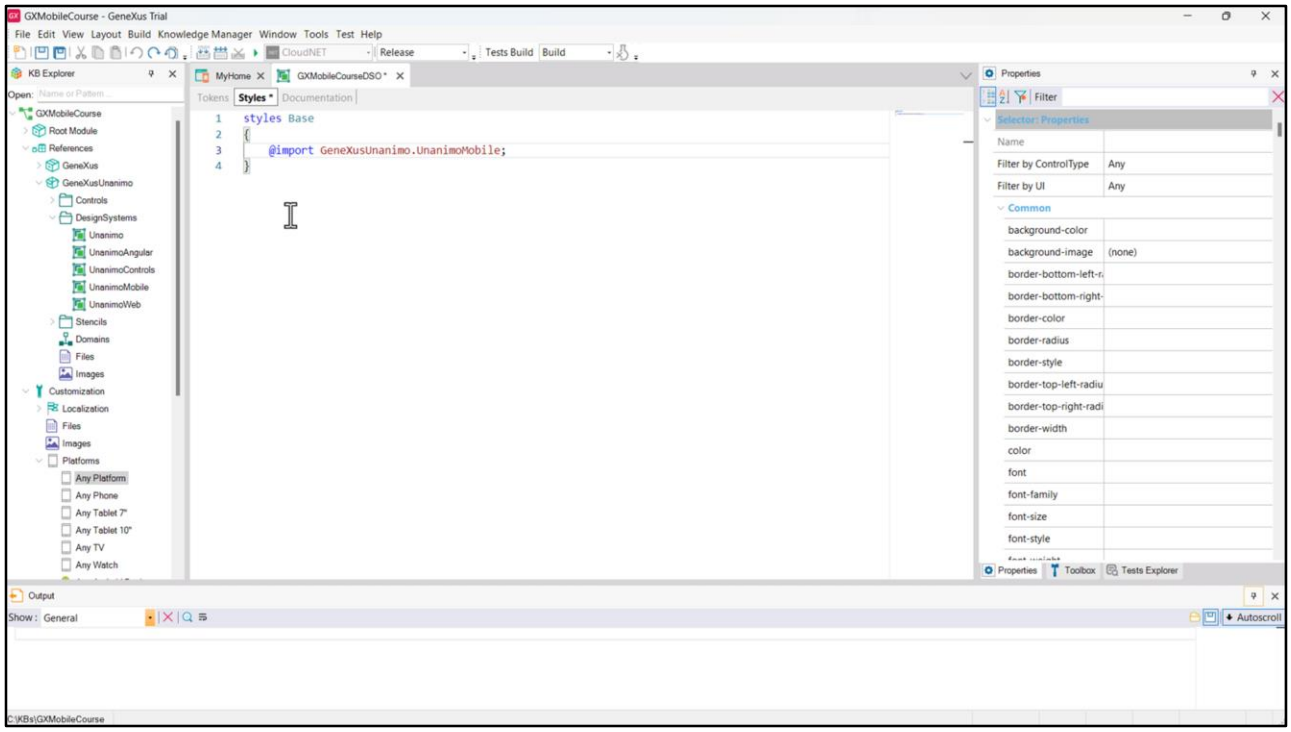
Esas definiciones de las clases, que son las que van a contener las características de cada elemento del layout, van a estar definidas y centralizadas en un objeto: el Design System.



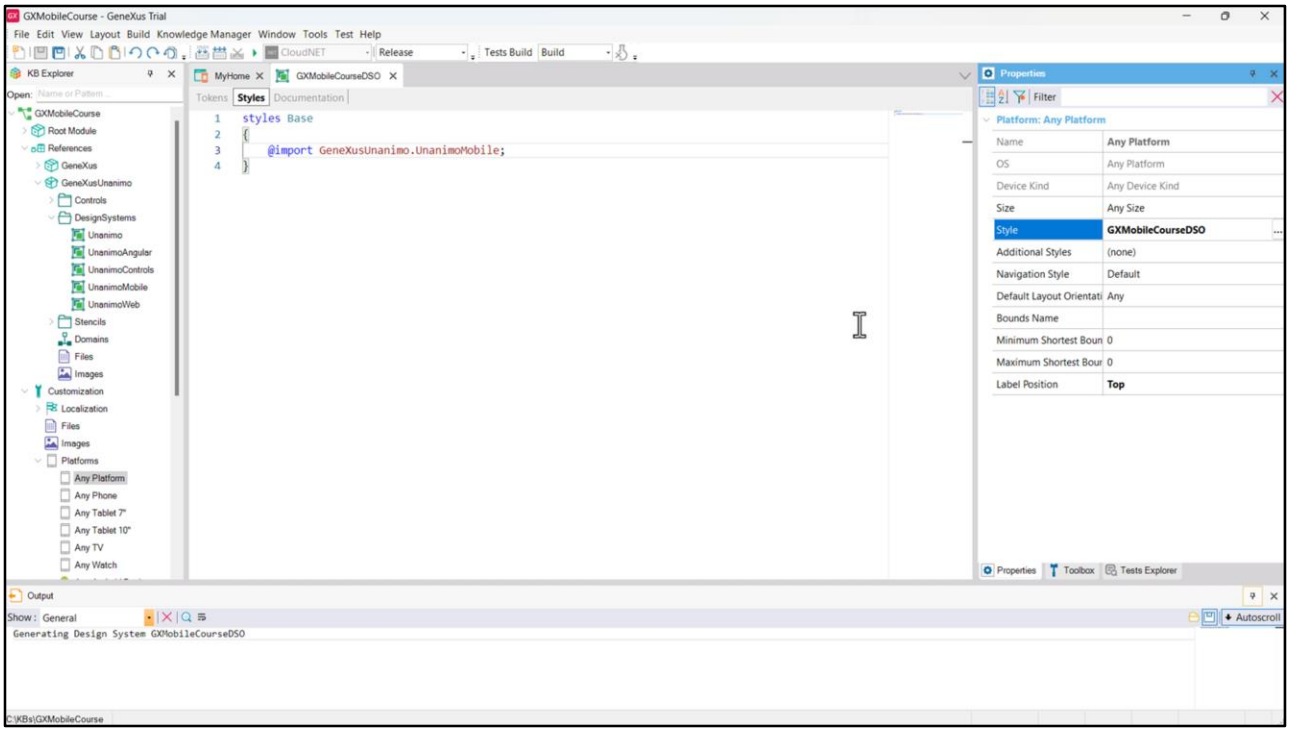
Toda KB se crea por defecto con un design system predefinido que vemos si editamos las propiedades de la versión, en la llamada Default Style. El estilo corresponde al objeto Design System que contiene todas esas definiciones. Por defecto se crea en la KB este Design System object que asume el mismo nombre de la KB. Este DSO está configurado bajo el grupo de propiedades Web interface: esto quiere decir que lo que comanda es el desarrollo de la aplicación cuando es con Web Panels. Como nosotros estamos trabajando en una aplicación móvil nativa, no es nuestro caso, pues estamos utilizando Panels en lugar de Web Panels.



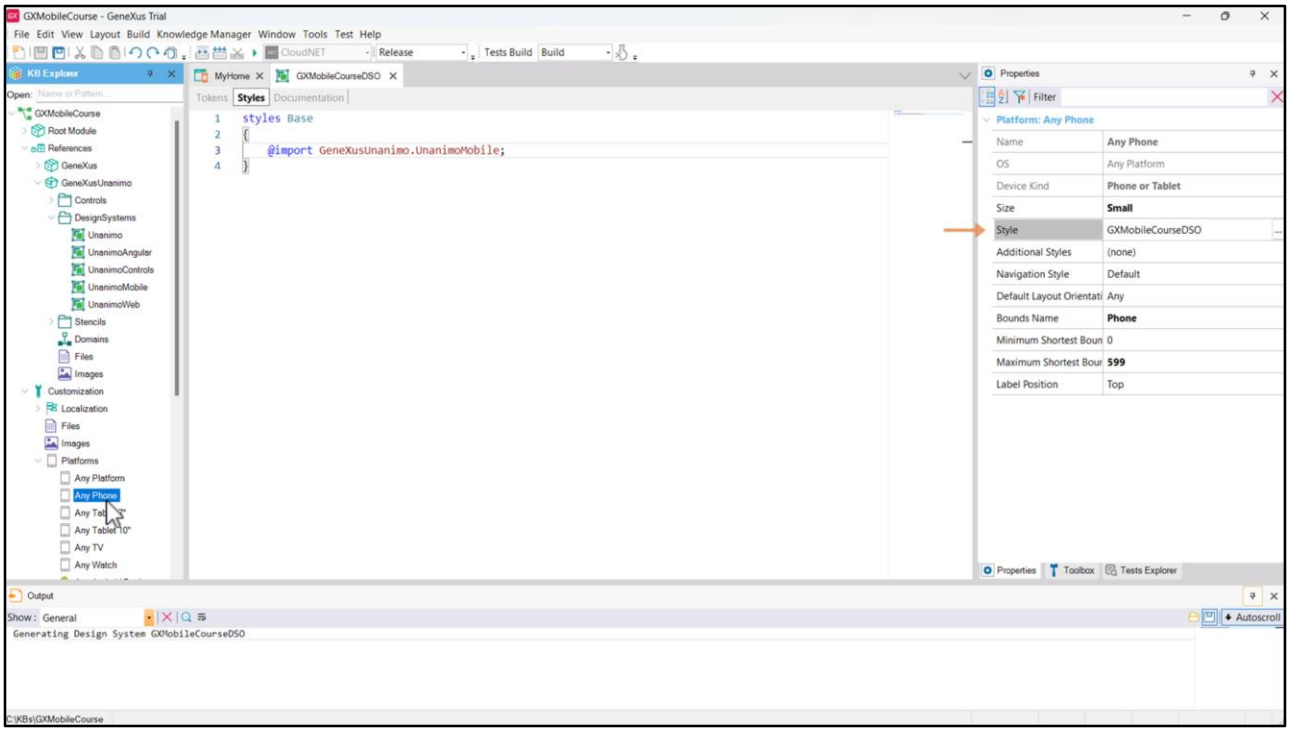
Para comandar el diseño cuando estamos en un ambiente nativo, debemos cambiar donde están las definiciones de las plataformas, bajo el nodo Customization. En el nodo Any Platform, vemos que en la propiedad Style está tomando el objeto Design System de nombre UnanimoMobile. Unanimo ya es un Design System, el predefinido por GeneXus, creado en el nodo GeneXusUnanimo y es el que por defecto tomará el control del diseño de toda la aplicación si no definimos otra cosa.



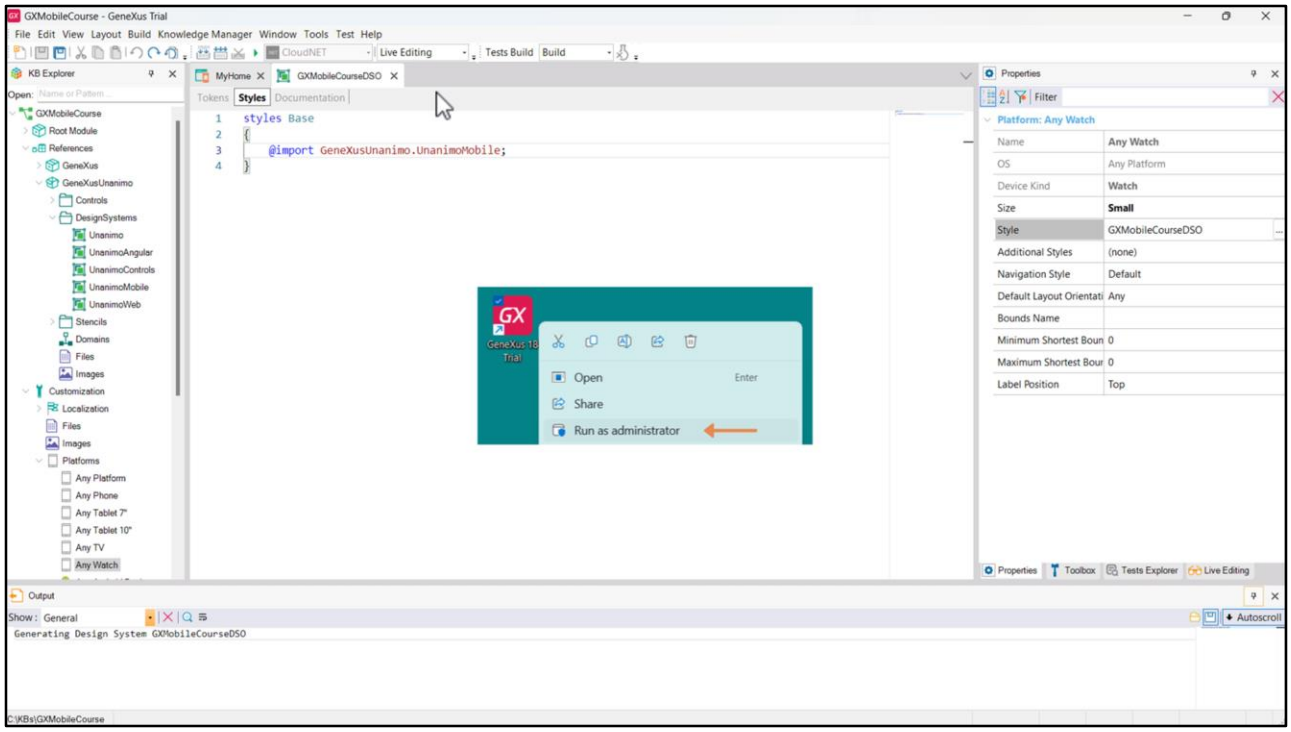
Como nosotros vamos a hacer un desarrollo de cero, crearemos un nuevo objeto de tipo Design System que será el que va a aplicar los estilos a nuestra aplicación nativa. Le llamaremos GXMobileCourseDSO, e indicaremos aquí que este Design System debe heredar de las definiciones que tenga el Design System UnanimoMobile. Para ello, escribimos en la solapa Styles lo siguiente:



Utilizaremos las solapas Tokens y Styles para hacer nuestras definiciones, pero antes, indiquémosle a GeneXus que queremos que nuestra KB tome como Design System al que acabamos de crear. Vayamos a Any Platforms, y cambiemos la propiedad Style.



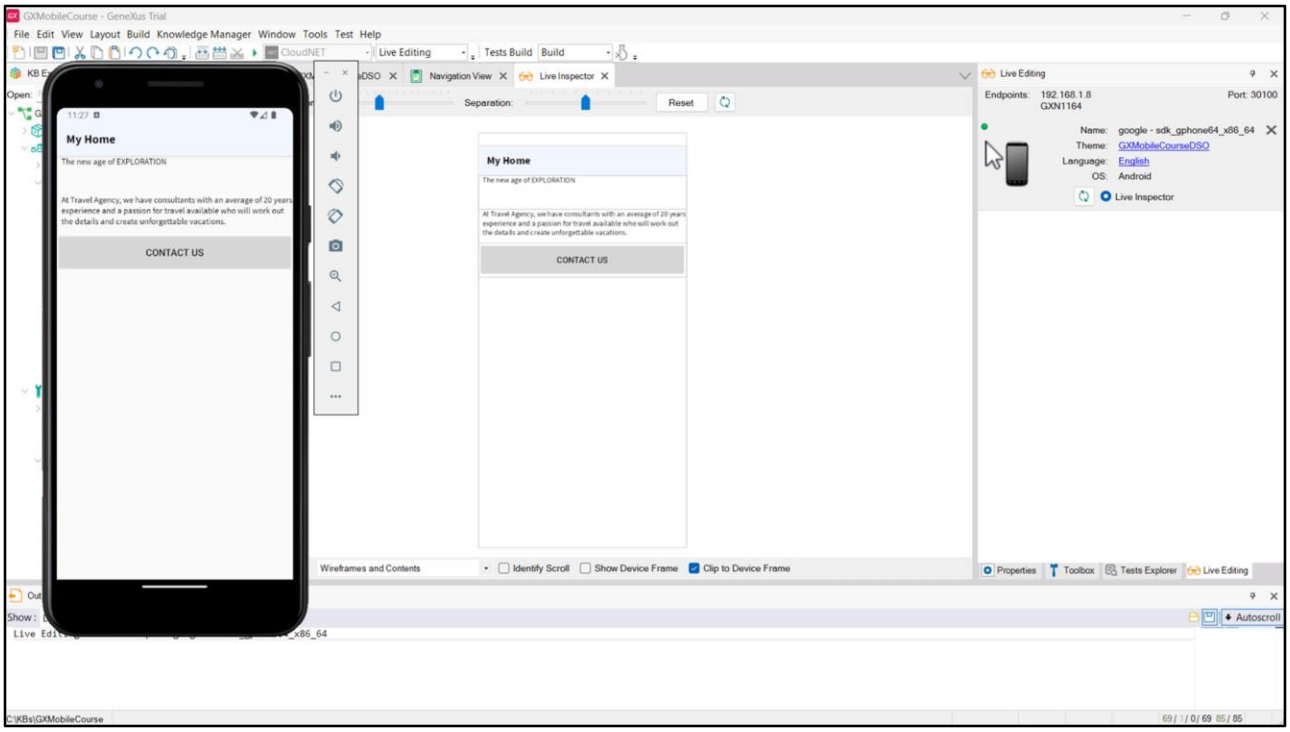
Si observamos en los diferentes tamaños y plataformas, vemos que heredan del que acabamos de configurar, pero esto si queremos lo podemos cambiar, utilizando un Design System object para cada tamaño de pantalla y plataforma.



Antes de ejecutar nuestro Panel para ver cómo se ve antes de empezar a cambiar su estilo, configuremos el uso de la herramienta Live Editing. Pero… ¿qué es Live Editing? Al prototipar una aplicación, una de las tareas que consume más tiempo es perfeccionar el Look&Feel (o User Interface -UI-) y la User Experience -UX-. Para simplificar esta tarea, GeneXus tiene una funcionalidad que permite cambiar el estado de la aplicación en tiempo real desde el IDE sin guardar los objetos modificados.

Para prototipar en modo Live Editing, simplemente cambiamos en este ComboBox el valor Release por Live Editing y ejecutamos nuevamente la aplicación. Si estás utilizando la versión Trial de GeneXus, deberás ejecutar con permisos de Administrador.

En este caso, como es la primera vez que el modo Live Editing es activado, va a ser necesario compilar nuevamente la aplicación, pero para los próximos cambios que efectuemos esto ya no será necesario, y simplemente se verán reflejados los cambios automáticamente en el emulador y en el IDE de GeneXus.

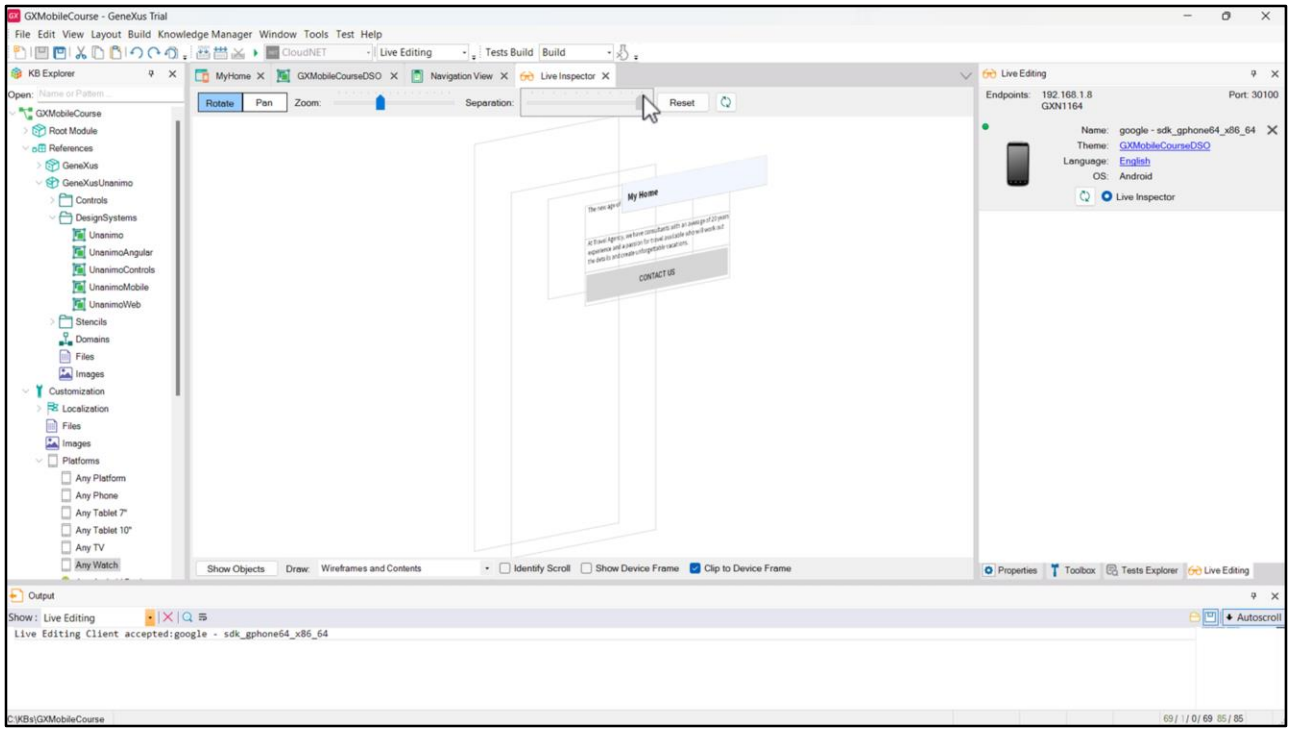


Ejecutemos.

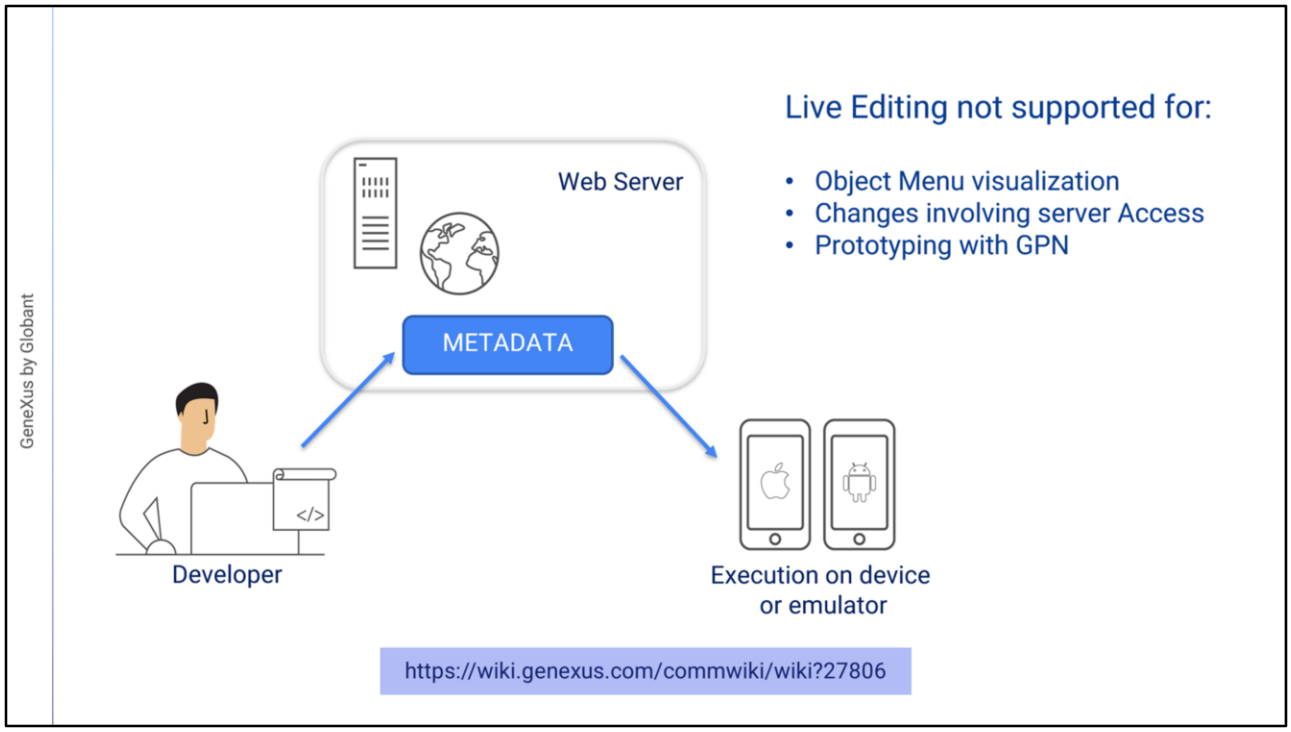
Ya tenemos lista la aplicación nuevamente en el emulador, solo que ahora habilitamos Live Editing.

En el IDE podemos ver que se habilitó una solapa Live Editing, donde tenemos la información de los dispositivos que se encuentran conectados, en nuestro caso el emulador que teníamos en ejecución. Podemos ver datos del dispositivo, como el nombre, el Design System que se está utilizando en la aplicación y el lenguaje. El punto de color verde indica que el dispositivo está conectado al Live Editing.

También podemos ver que se abrió una ventana nueva, llamada Live Inspector. Esta es una de las ventajas principales de Live Editing. Si bien hay una restricción y no podemos ver cuando se está ejecutando un objeto menú, el resto de los paneles sí podremos verlos. Alcanza con pasar el mouse sobre los controles en pantalla para poder ver su nombre y si damos clic podremos ver sus datos principales.



Si movemos el mouse podemos rotar la imagen en cualquier dirección, vemos unas capas que nos permiten ver dentro de qué control se encuentra situado cada control. Estas capas nos permiten seleccionarlos de una manera mucho más sencilla, ya que en general las secciones o tablas están superpuestas. Con el slider de Separación podemos aumentar o disminuir el espacio entre las capas, con Zoom podemos aumentar o disminuir el Zoom sobre lo que estamos viendo, y si seleccionamos Pan en lugar de Rotate, podemos mover el contenido con el mouse, sin Rotarlo. Con Reset volvemos a la visualización por defecto.

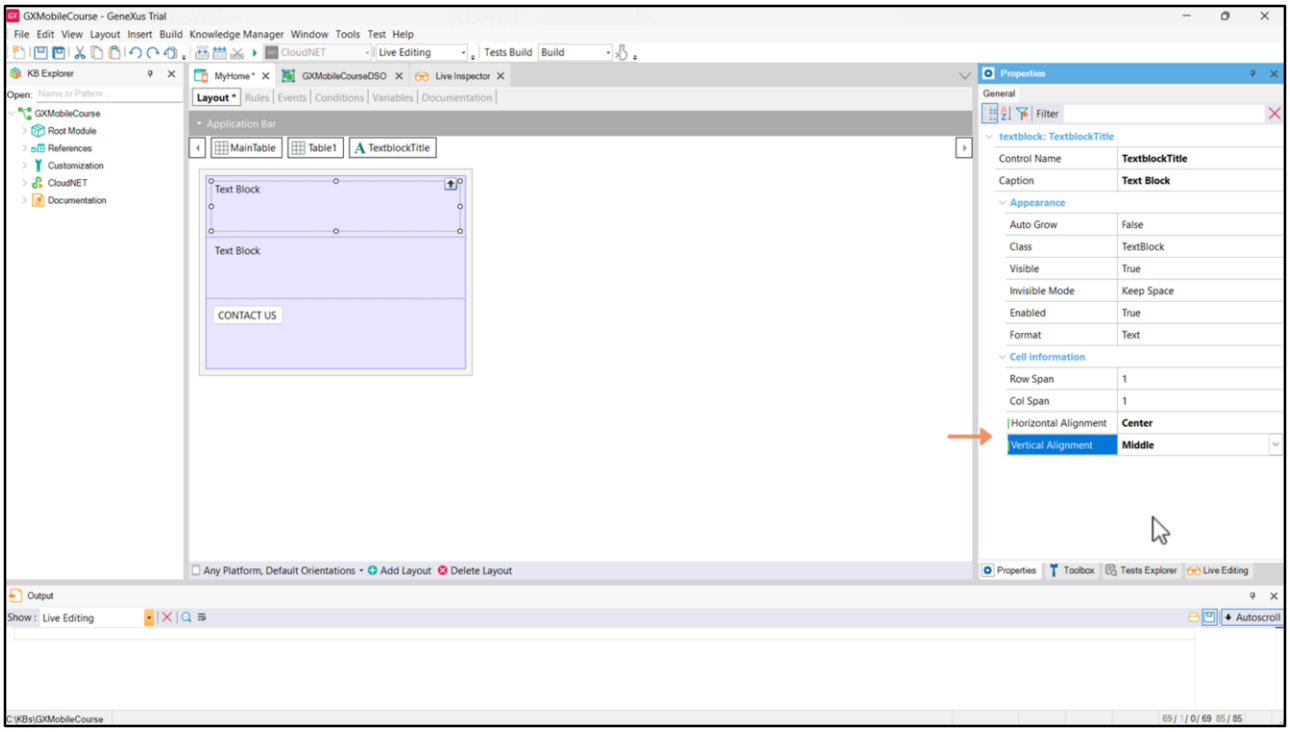


Pero, ¿cómo funciona el Live Editing? Cuando usamos Live Editing el Servidor quedará "escuchando" los cambios que hagamos y replicará estos cambios sobre la Metadata a la que accede el dispositivo móvil (en nuestro caso el emulador) y que se utiliza para dibujar la pantalla en el dispositivo y definir su comportamiento.

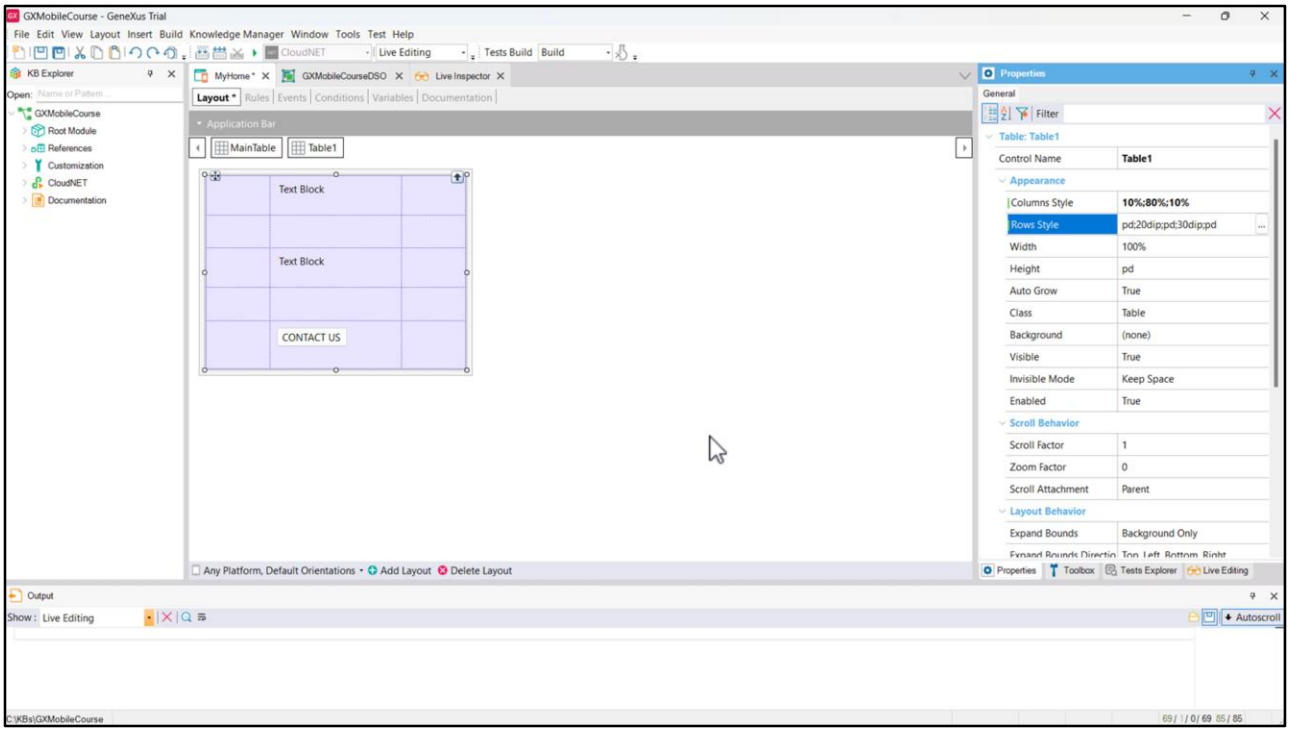
El Live Editing tiene algunas restricciones de uso, por ejemplo:

- No funciona para objetos menú.
- No se pueden visualizar cambios que impliquen accesos al servidor. Por ejemplo, si agregamos un atributo al form, el valor deberá recuperarse de la base de datos en un evento del servidor (Start, Refresh o Load), por lo que será necesario compilar la aplicación nuevamente.
- Como el live editing funciona solamente con la aplicación compilada, no funciona si se está prototipando con el GPN (GeneXus Projects Navigator).

Para conocer más detalles sobre el uso del Live Editing y el Live Inspector, te invito a visitar el contenido del wiki de GeneXus: https://wiki.genexus.com/commwiki/wiki?27806

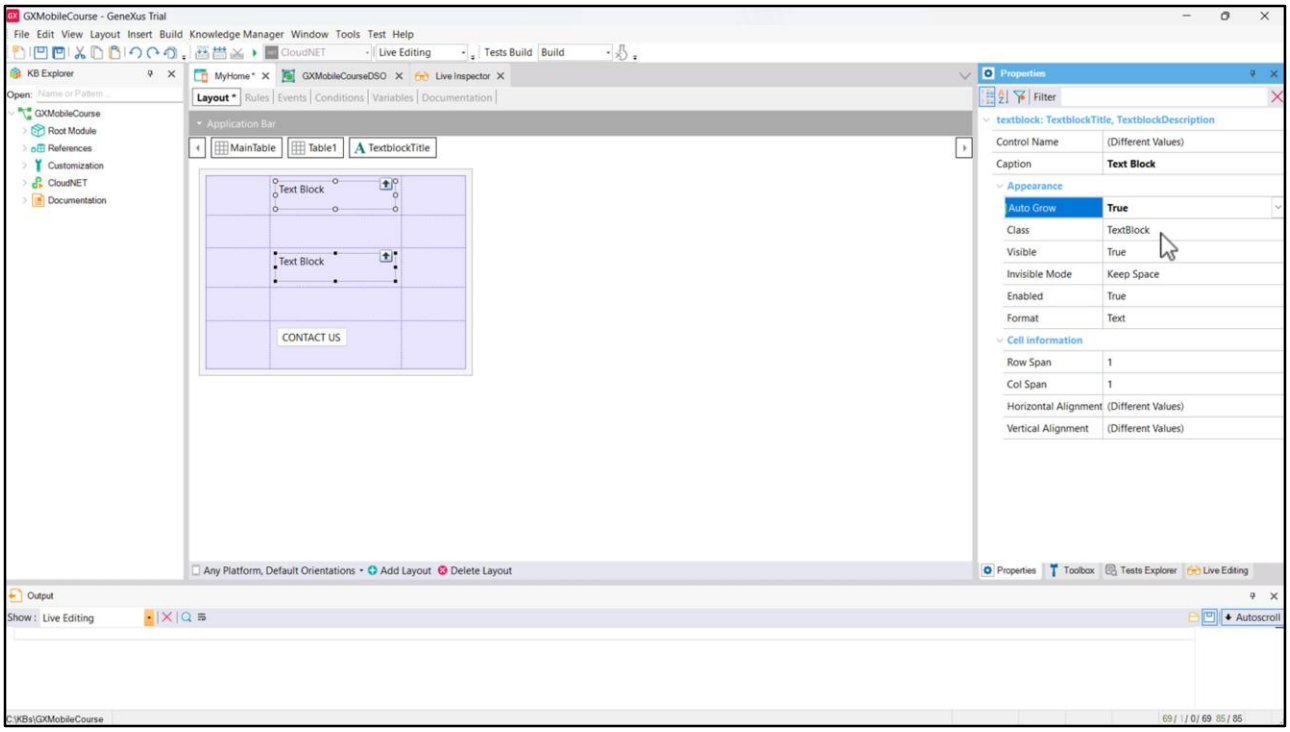


Volvamos a nuestro ejemplo. Como el título lo queremos centrado en la celda que lo contiene, podemos utilizar las propiedades Horizontal Alignment y Vertical Alignment con los valores Center y Middle, respectivamente.

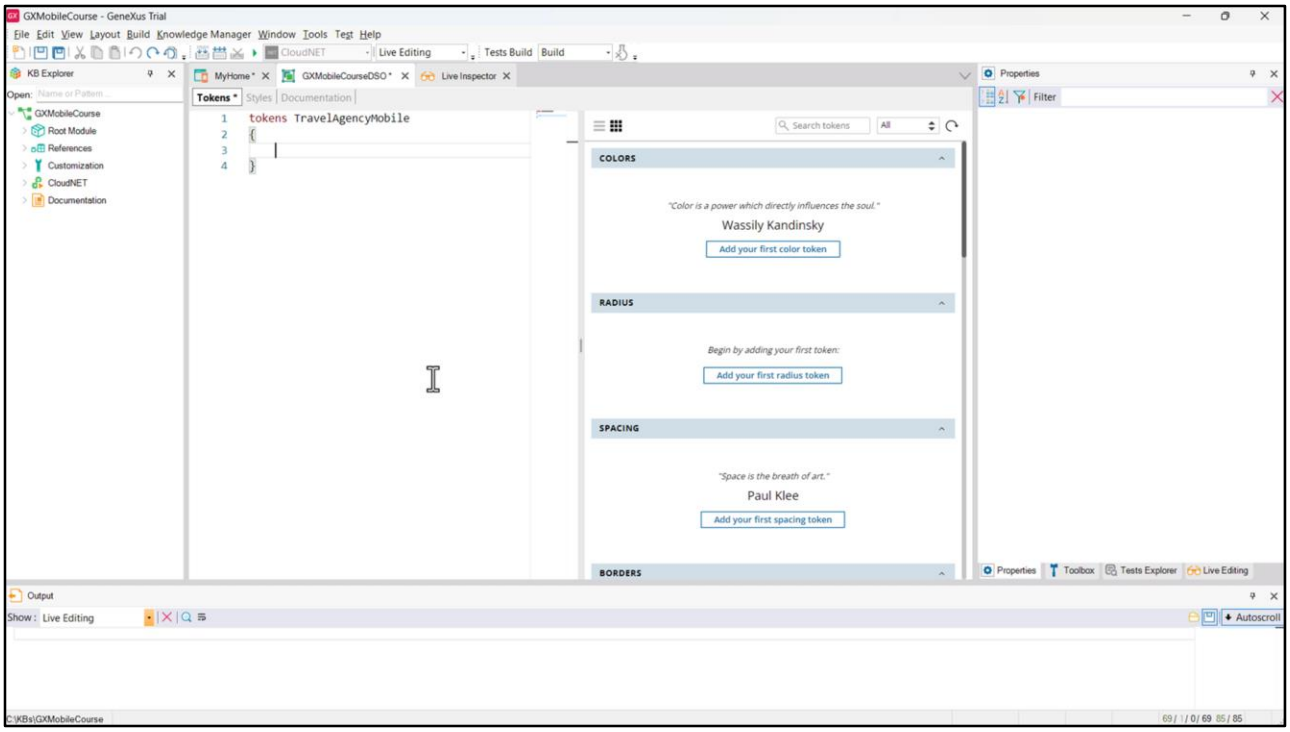


Cambiemos también la propiedad Horizontal Alignment del TextblocDescription, para que el texto se vea justificado.

Agreguemos dos columnas más a la tabla, una a la izquierda, y otra a la derecha del contenido, y dos filas más, una debajo del título, y otra arriba del botón, con el fin de generar espacios que hagan que la información se vea mejor. Si observamos la propiedad Columns Style, vemos que cada columna quedó ocupando un 33% del ancho total de la tabla. Cambiemos los valores para que la primera columna ocupe un 10% del ancho total, la segunda (que es la que contiene el contenido) un 80%, y la última el 10% restante. Ahora, cambiemos los valores de la propiedad Rows Style, para indicar que la segunda y la cuarta columna, que son las separadoras, ocupen 20 y 30 dips respectivamente.

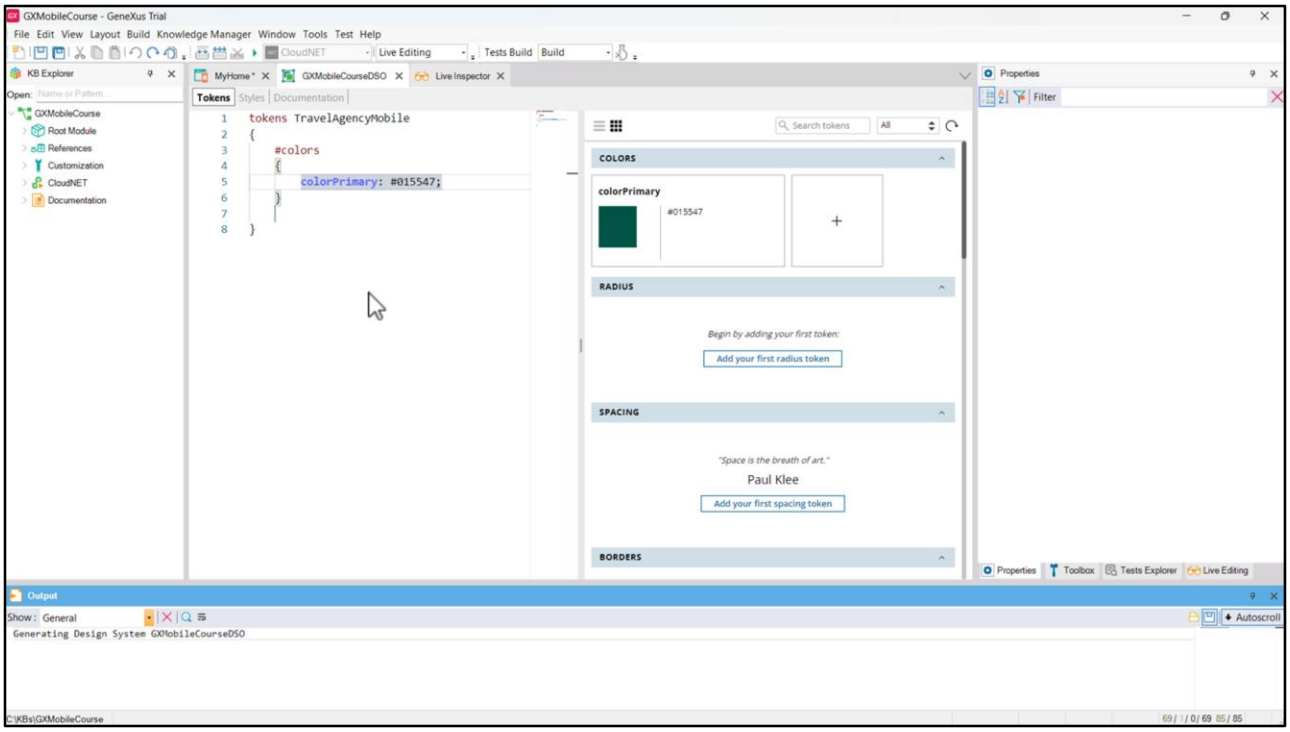


También establezcamos la propiedad Auto Grow de las celdas que contienen los Textblocks en True, para que, si no alcanza el espacio predefinido que ocupa cada celda, se expandan de modo de que todo el contenido quede visible.



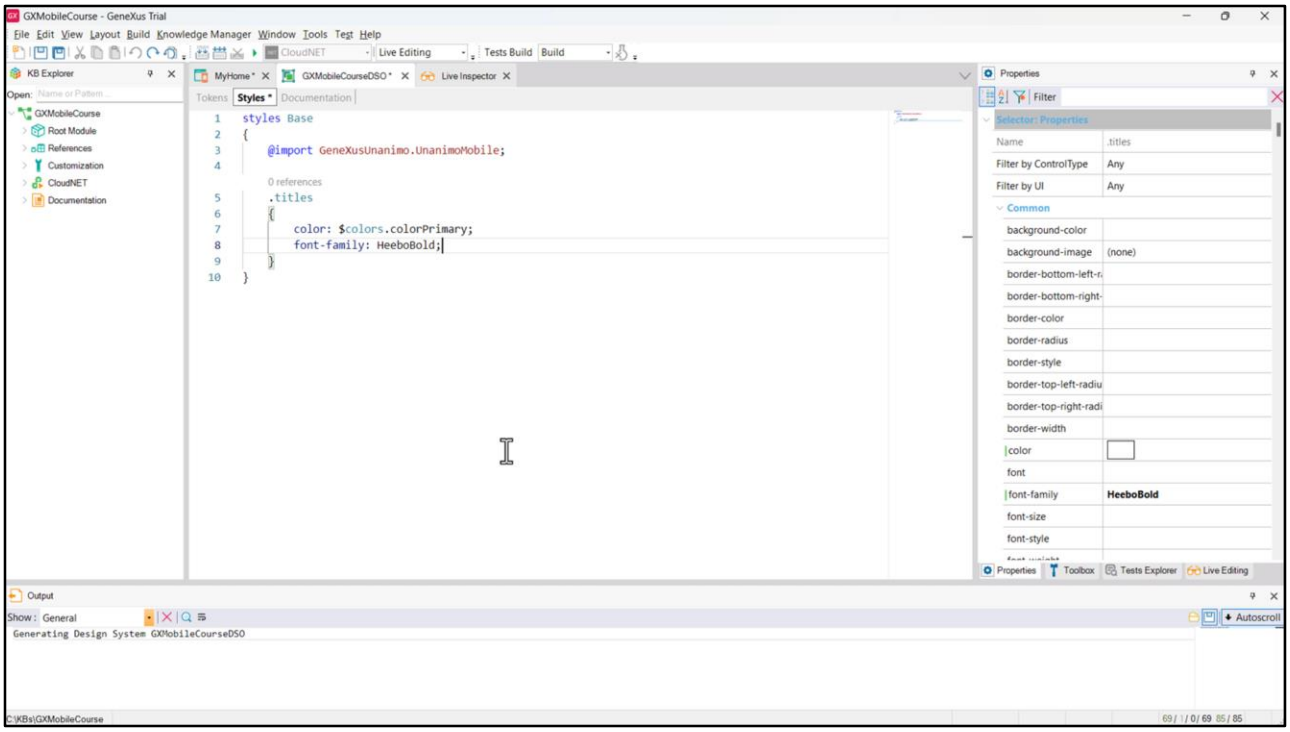
Empecemos ahora a cambiar el estilo a los controles de nuestro Panel. Empecemos con el título: vamos a cambiar el color y tipo de fuente. Podríamos también definir en nuestro DSO la alineación centrada, pero no será necesario ya que la configuramos por medio de las propiedades, y estas pertenecen a un orden jerárquico superior al DSO: es decir que si estableciéramos la alineación centrada en nuestro DSO, los valores de las propiedades les pasarían por encima.

Vamos a ponerle un nombre al conjunto de tokens que crearemos, con el fin de que nuestro DSO nos quede ordenado: pongámosle TravelAgencyMobile.



Empecemos por definir una constante de color: le pondremos un nombre que lo identifique, por ejemplo, Primary, pero para que nos queden todos los colores fácilmente identificables con solo mirar el código, mantendremos la misma nomenclatura para toda constante de color que creemos, y su nombre empezará con la palabra *color*. Si ya conocemos el valor hexadecimal del color (por ejemplo, porque el equipo de diseñadores nos lo pasó en un archivo de Figma) lo ponemos aquí. Usaremos para el título este verde oscuro.

Vemos que a la derecha tenemos otro editor que está sincronizado con el de la izquierda, es decir que podemos trabajar en uno o en el otro y lo que hagamos en uno, se verá reflejado en el otro.



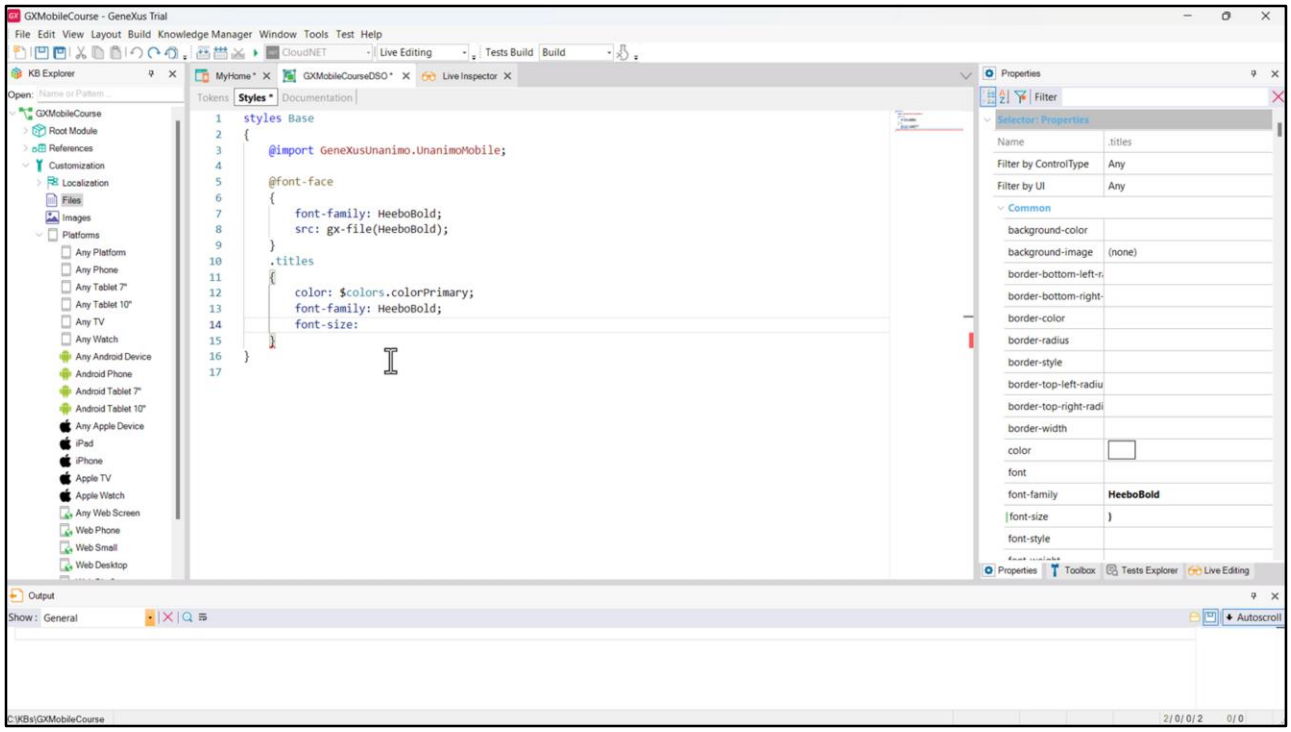
En la solapa Styles, crearemos la clase que va a dar estilo al contenido.

Le pondremos el nombre *titles*, y agregaremos las propiedades CSS necesarias para establecer las características de esta clase.

El color que va a aplicar, es el del token que acabamos de definir, por lo tanto escribimos:
color: $(que nos permite referenciar un token) colors (porque va a ser un token de color).colorPrimary;
Con esto estamos indicando que todo control que tenga esta clase asociada, tendrá este color.

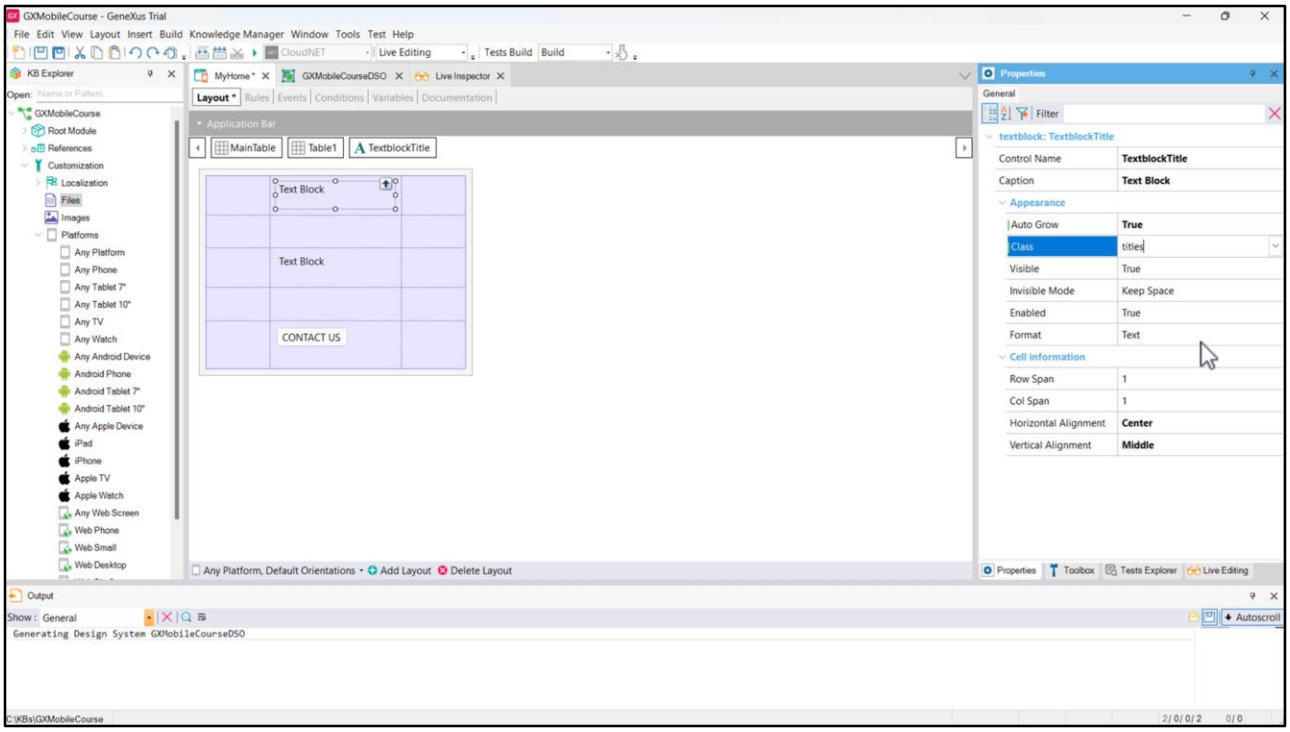
Especifiquemos ahora la familia de fuente a ser utilizada:
font-family: HeeboBold;

Esta familia de fuentes no forma parte de las predefinidas, es decir, las que entienden por defecto los navegadores. Esto quiere decir que debemos definir en el DSO cuál es esa fuente, de dónde se obtiene. Para incluirla vamos a tener que importar el archivo de la fuente en la KB para poder referenciarlo en la propiedad font-family del DSO.

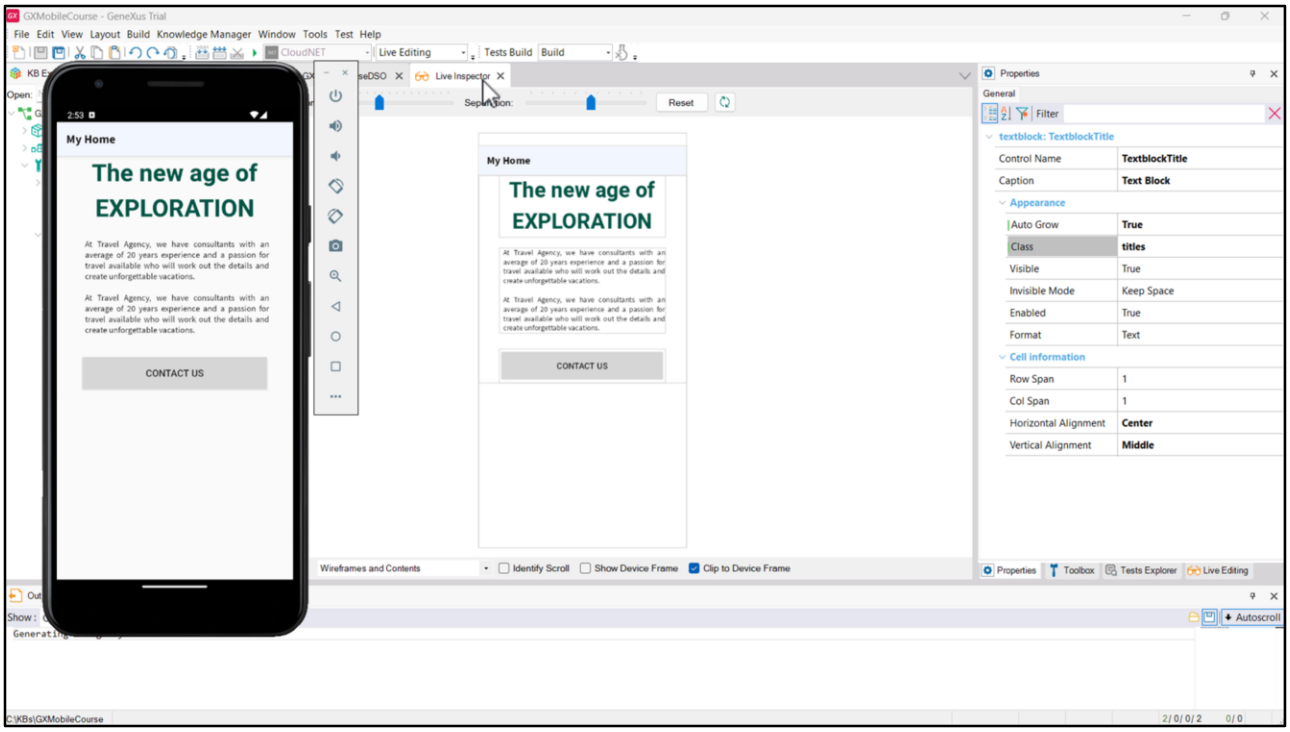


Vamos a Customization → Files → New file, creamos el archivo llamado HeeboBold y lo elegimos desde su ubicación. Aquí ponemos el nombre que tendrá la familia de fuentes, y en la propiedad src indicamos de dónde se toma ese archivo. Con la función gx-file() recuperamos el archivo accesible en nuestra KB. Con esto simplemente le estamos diciendo al DSO que incluya esta fuente, pero después donde se usa, es en la referencia de la clase mediante la propiedad font-family que definimos para nuestra clase titles.
font-family: HeeboBold;

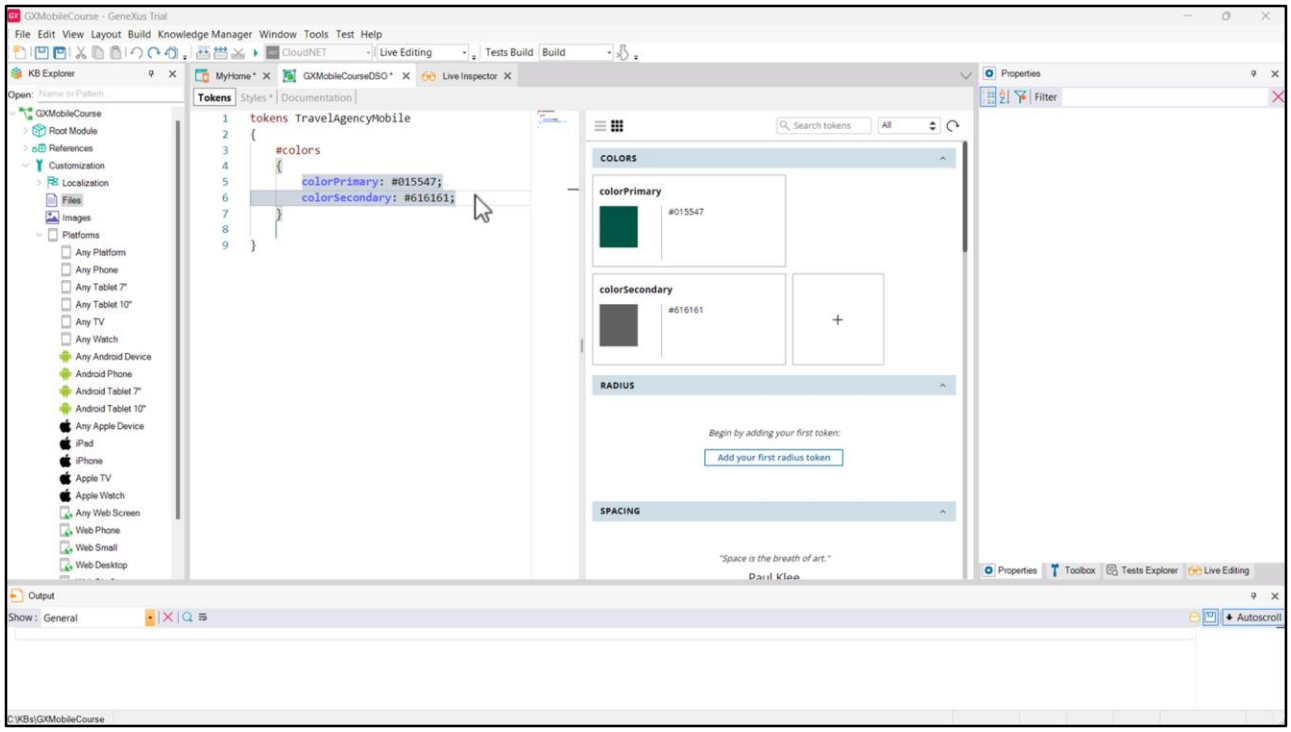
Con la propiedad font-size indicamos el tamaño que queremos para el título.



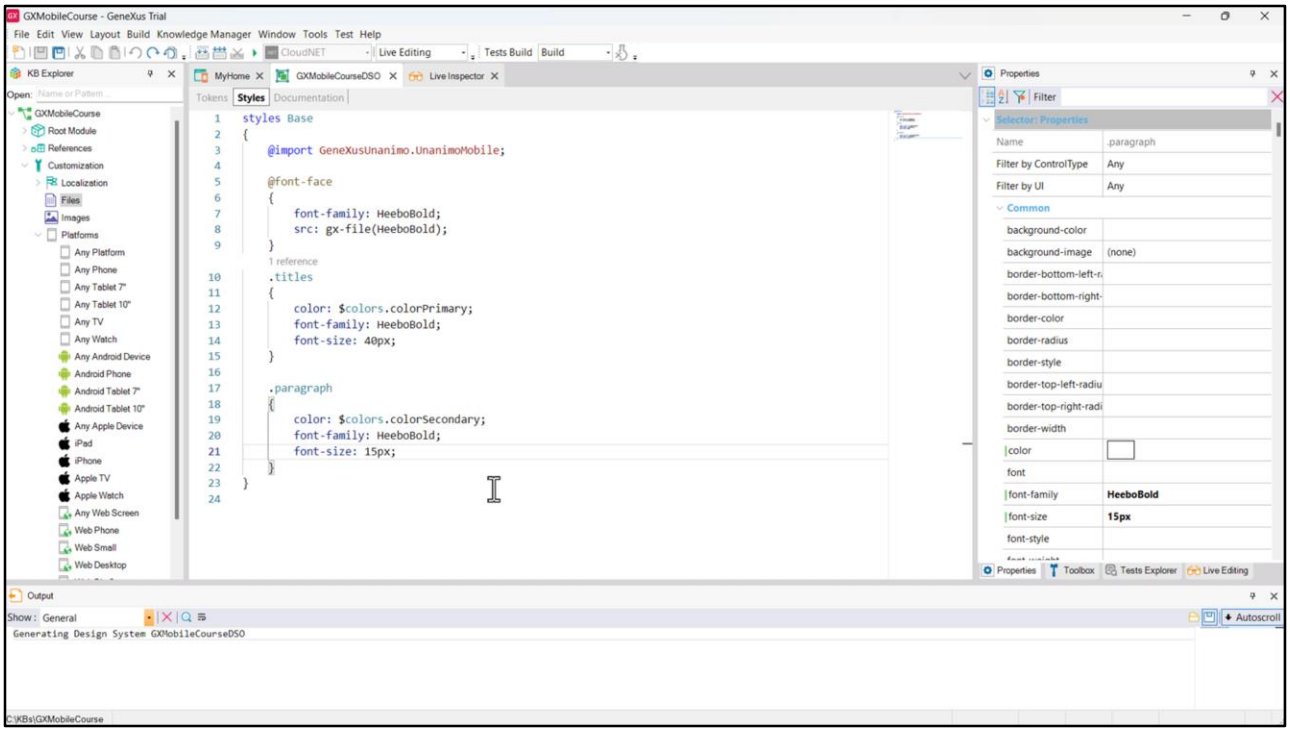
El próximo paso, es indicarle a nuestro control Textblock que tome la clase *titles* que acabamos de definir.



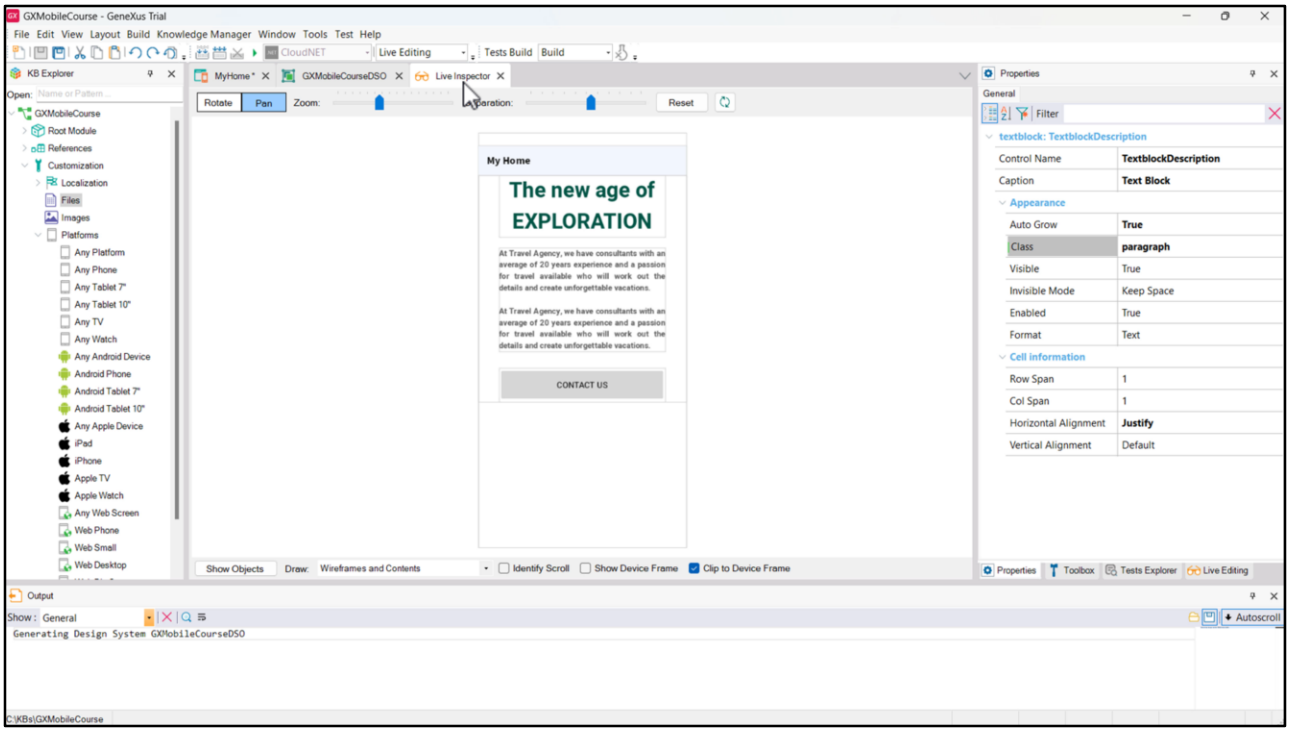
Ya podemos ver en el Live Editing los efectos de lo que hicimos en nuestro DSO.



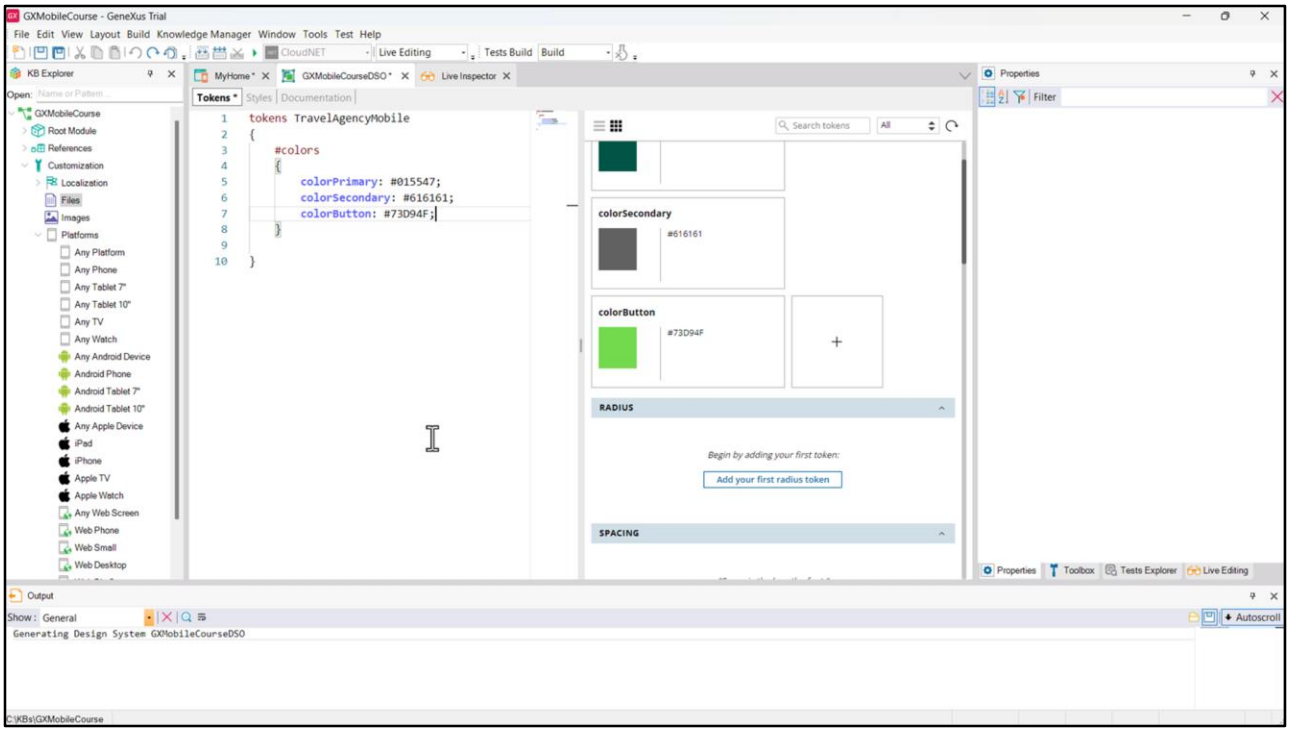
Hagamos lo mismo para el texto informativo. Creemos otro token de color al que llamaremos colorSecondary, esta vez de un gris oscuro…



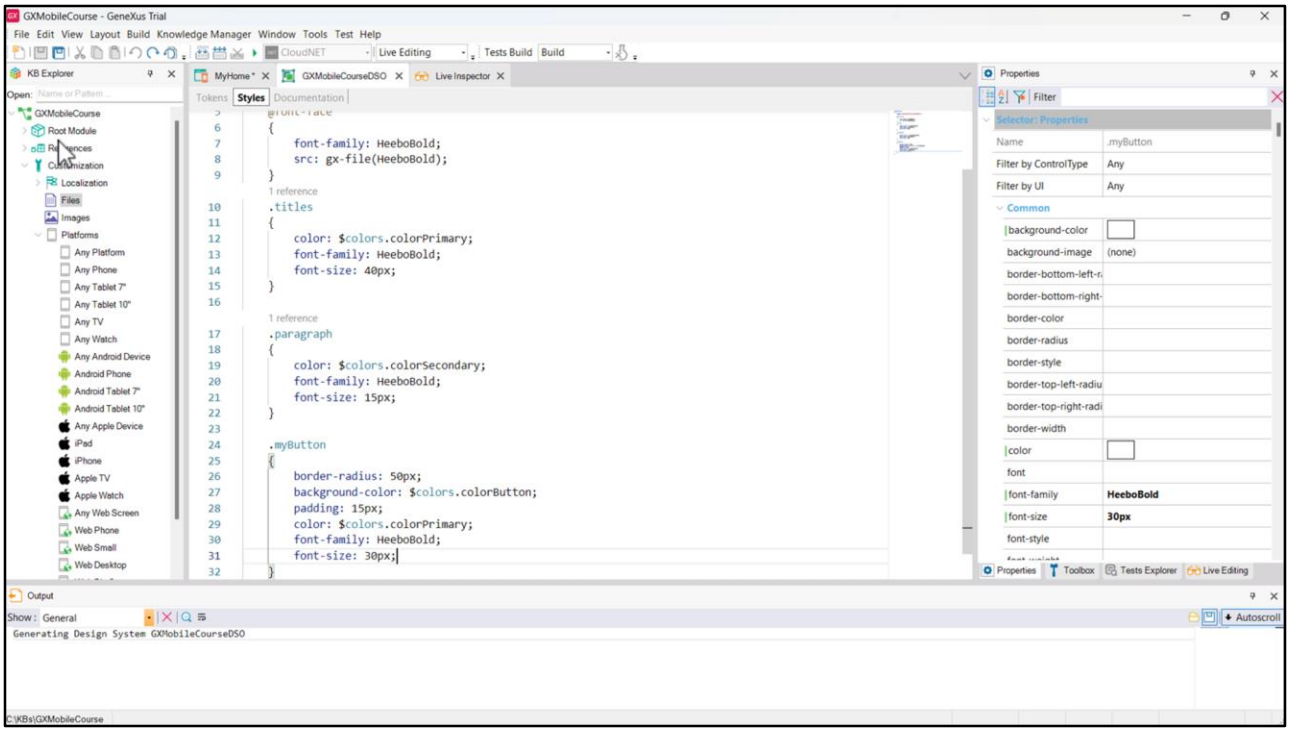
… y una clase de nombre *paragraph* a la que le establecemos las siguientes propiedades:



Asignémosle al control TextblocDescription la clase *paragraph*, y veamos cómo quedaron los cambios.



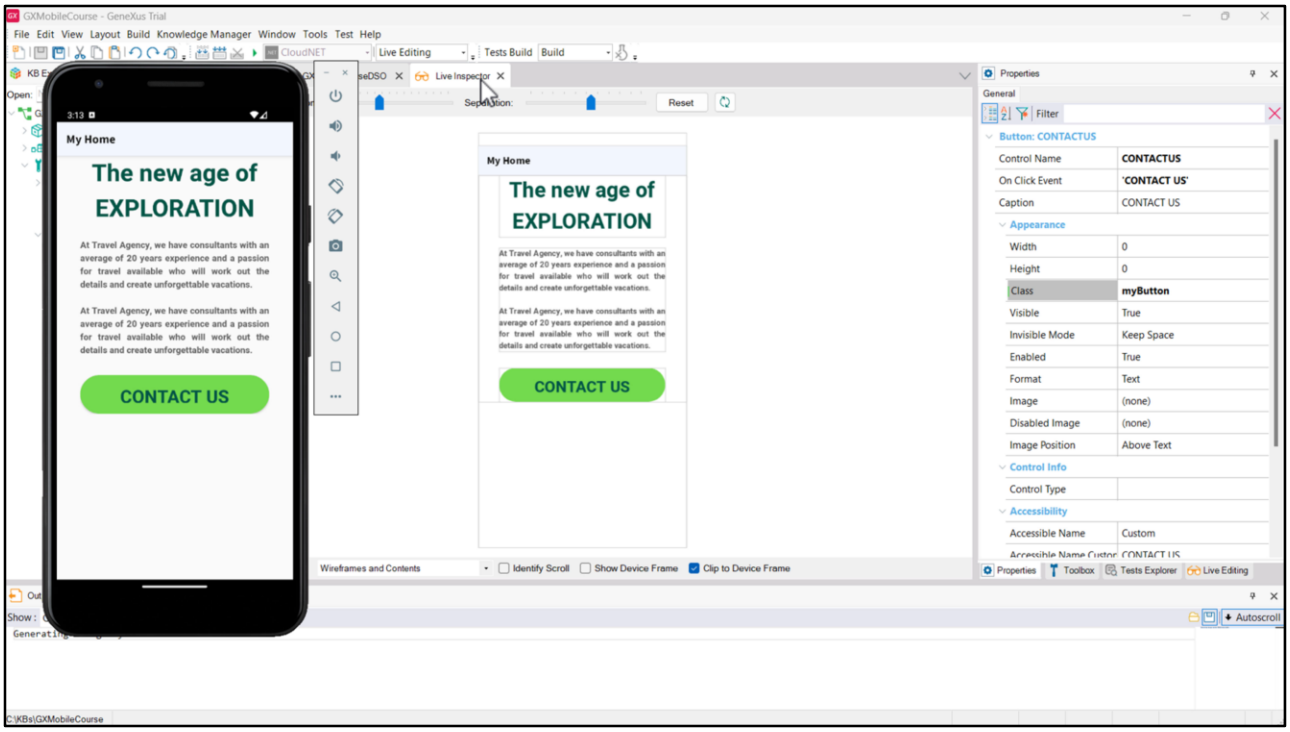
Ahora, nos queda darle estilo al botón. Para eso, creemos un nuevo token de color llamado colorButton para asociar este tono de verde,



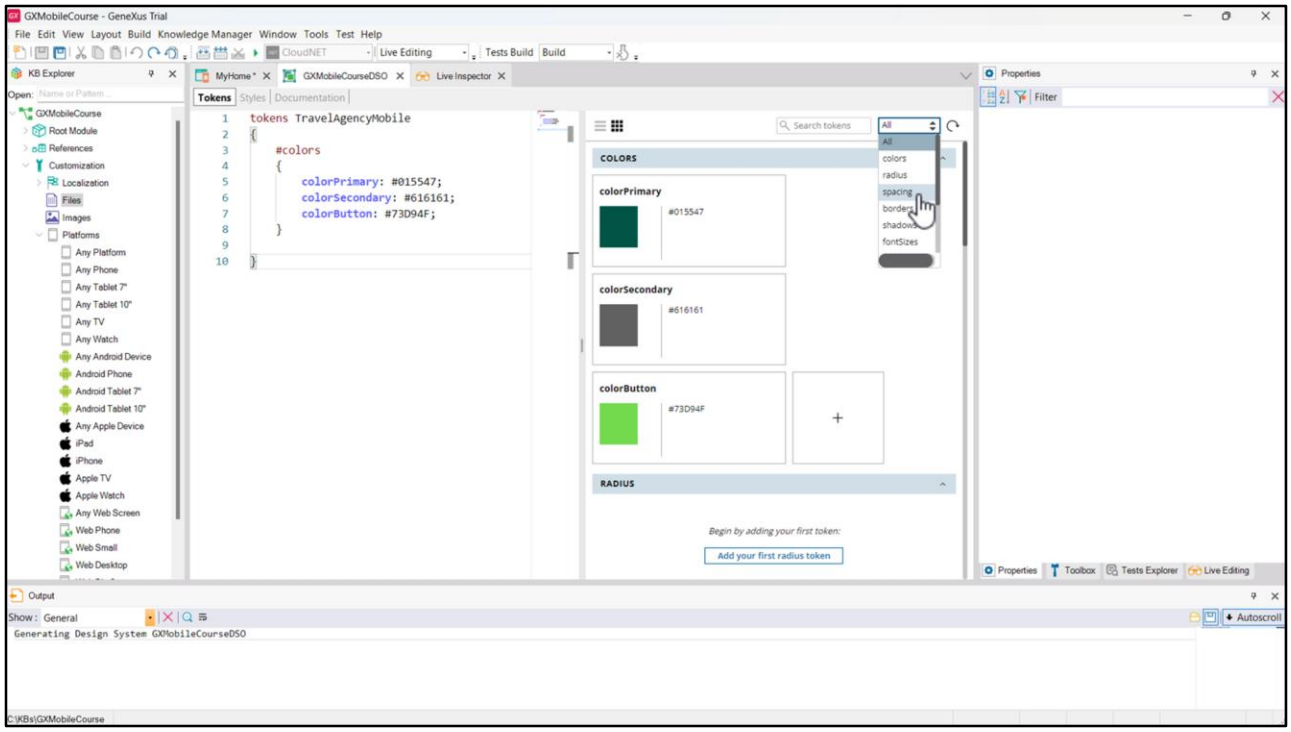
y comencemos a definir las propiedades de la clase *myButton*.

La propiedad border-radius sirve para dar el redondeado a los bordes; la propiedad background-color permite establecer el color de fondo; la propiedad padding, permite generar un espaciado entre el texto del botón y los bordes del mismo.

Ahora completemos con el estilo del texto, que es lo mismo que ya hicimos antes…

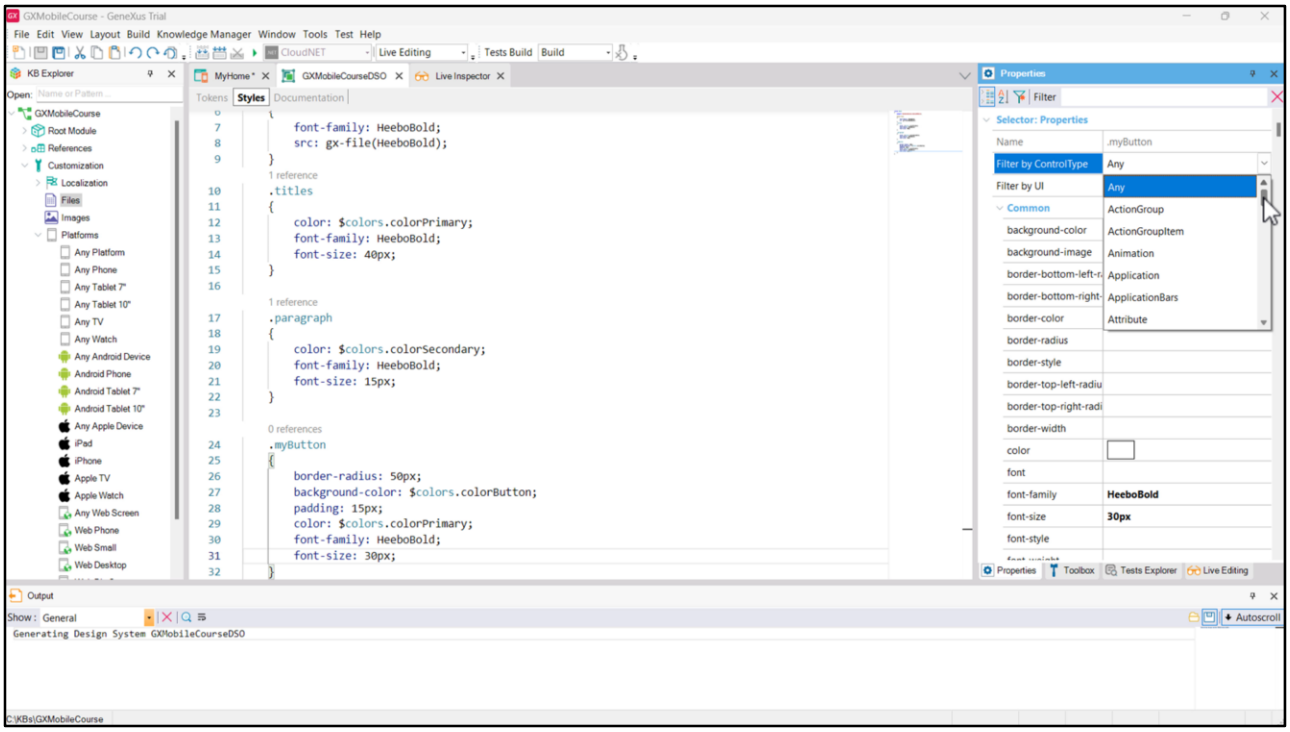


… y asociemos nuestra clase al botón del Panel.

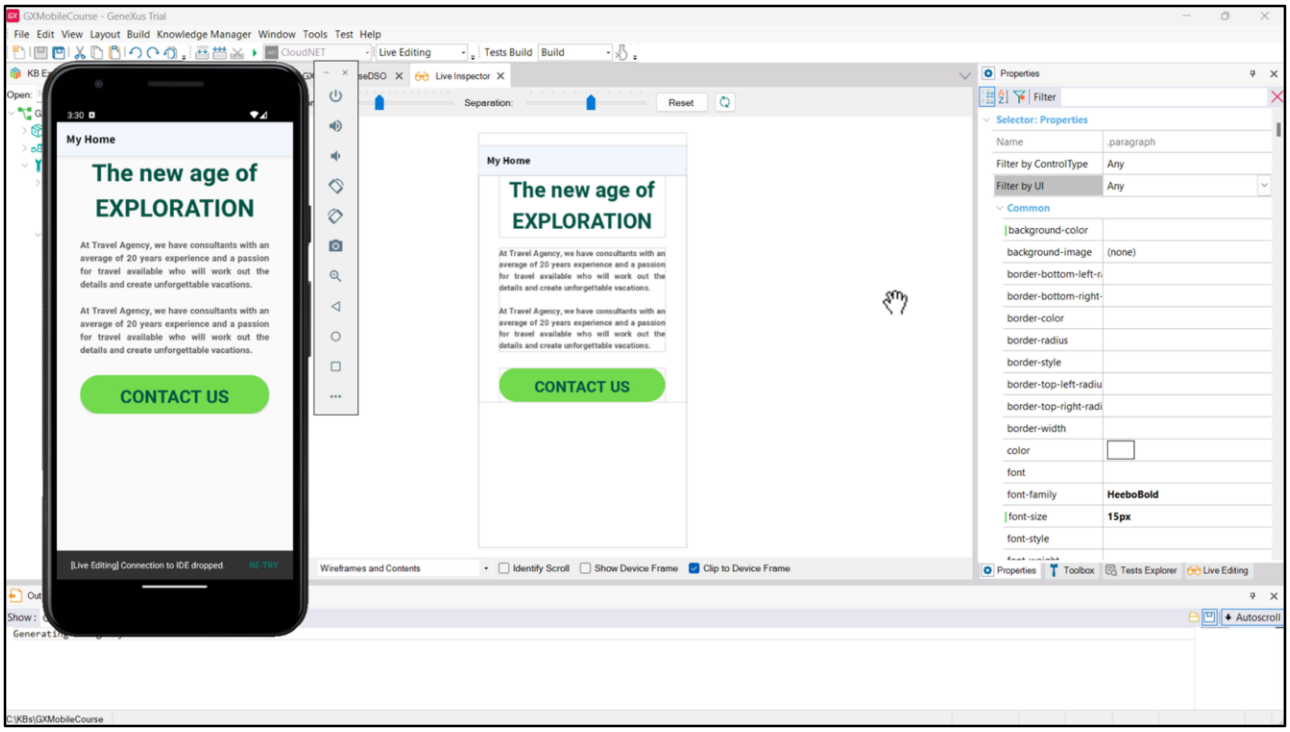


No solamente podemos utilizar tokens de color que son los más evidentes, sino que también podemos definirlos para otro tipo de cosas, por ejemplo: para dar espaciado, para definir los tamaños de fuentes, etc.

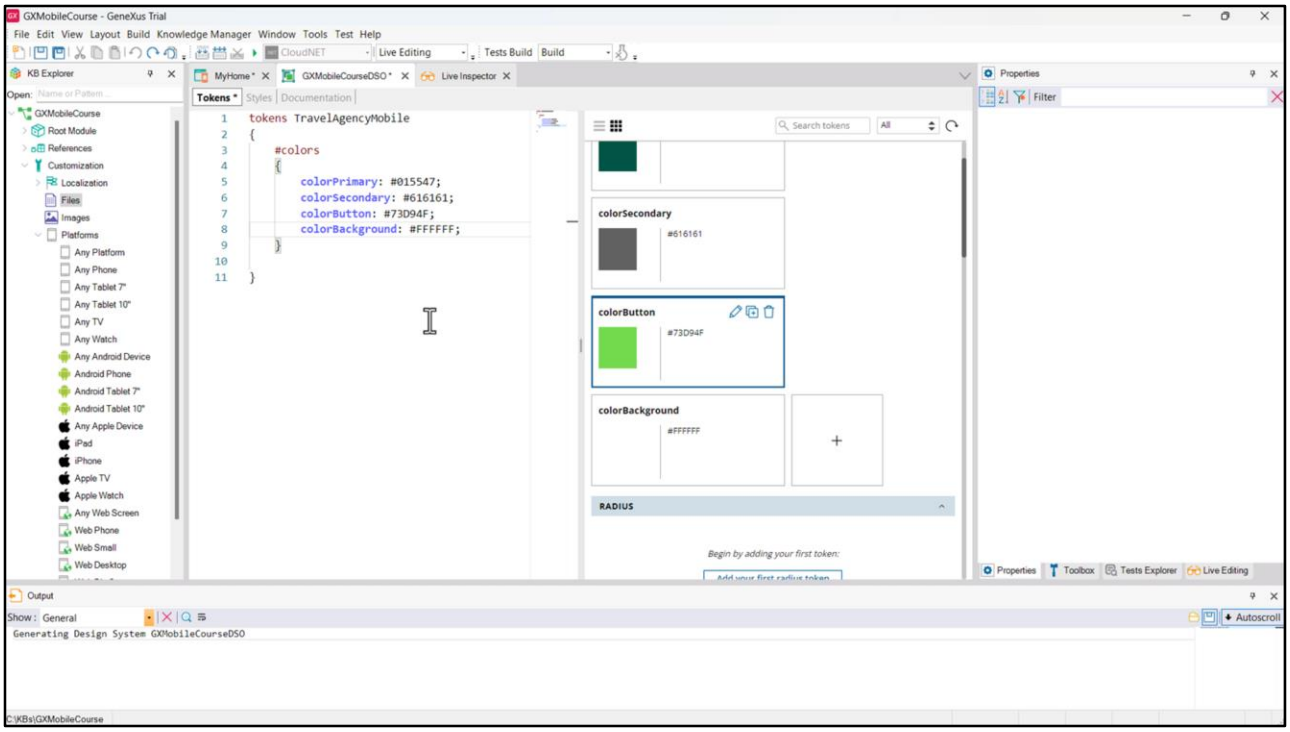
Hay propiedades que son comunes a varios controles (como la border-radius, background-color, color, font-family, font-size, entre otras).



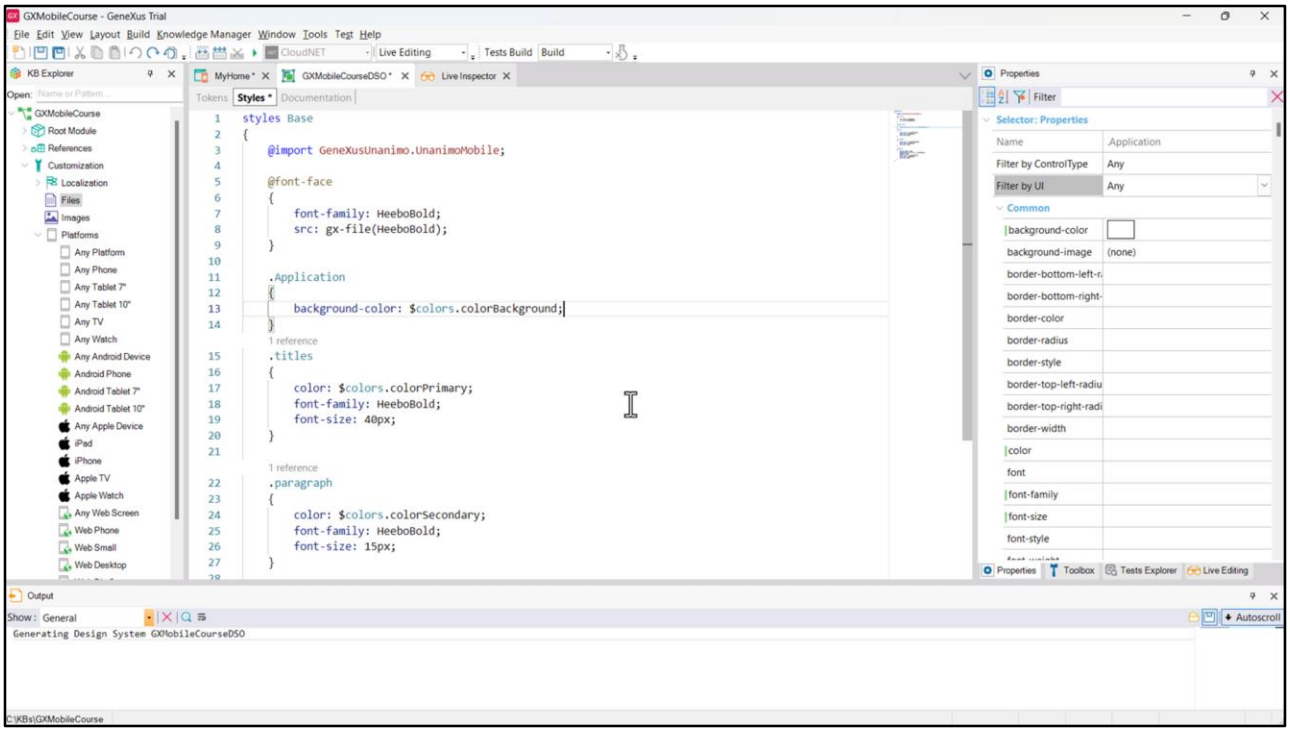
Cuando estamos trabajando en la solapa Styles, la ventana de propiedades nos provee la opción de filtrar por tipo de control, donde veremos y podremos configurar únicamente aquellas propiedades que apliquen para el control seleccionado, y filtrar por UI, donde podremos elegir el tipo.



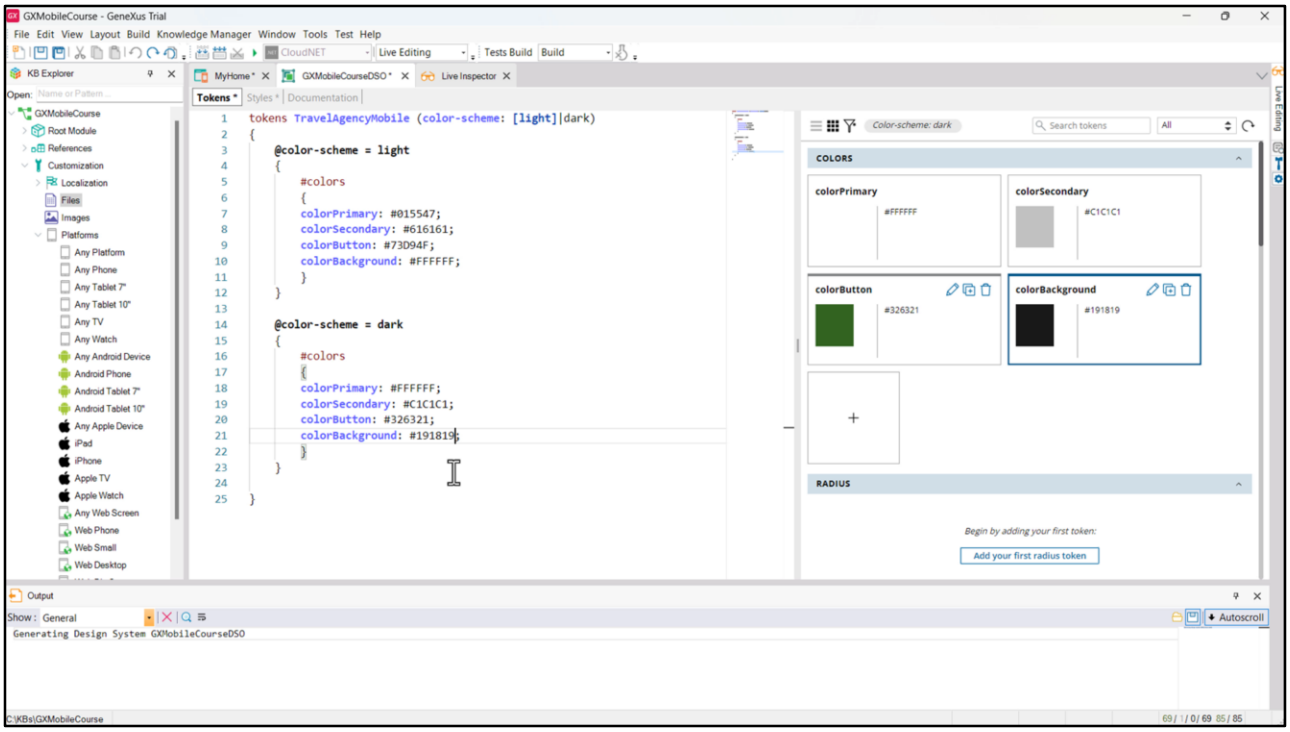
Hasta aquí, hemos diseñado la modalidad *light* de nuestra aplicación, pero también podemos diseñar la modalidad *dark*. Para eso, deberemos pensar en variar el color de fondo, el color de los textos y también del botón. En el caso del fondo, en la modalidad *light* va a asumir el color blanco que ya tenemos, y en el caso de la modalidad *dark* asumirá un gris muy oscuro. Para el título, en la modalidad *light*, será el verde oscuro que definimos como color primario, y en la modalidad *dark* será el blanco, para que contraste con el fondo oscuro. De manera análoga, definiremos los colores para el texto de la descripción y para el botón en la modalidad *dark*.



Especifiquemos el color de fondo de la aplicación: creemos la clase *Application* y definamos el color, que recuperaremos de un token de nombre colorBackground.



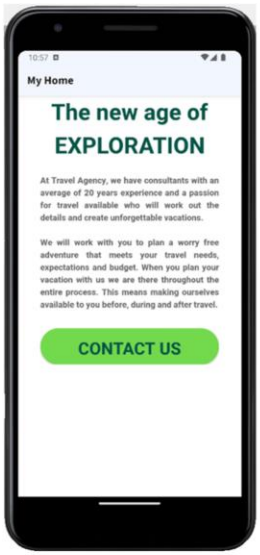
Al llamar así a la clase, será la que se aplique en el tag body de todo HTML de manera automática.

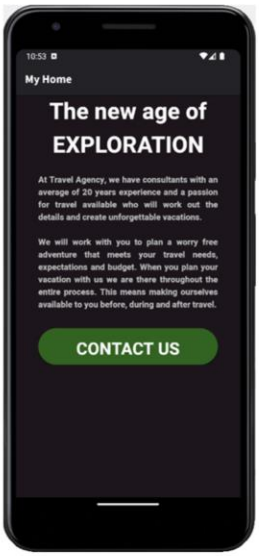


Para especificar los modos, debemos utilizar las opciones: color-scheme es quien permite indicar el esquema de colores, *light* o *dark*. Al escribir el *light* entre corchetes, estamos indicando que queremos que este sea el modo por defecto. Luego, hacemos variar los tokens de color según cuál sea el color-scheme. Para la modalidad *light* utilizaremos estos colores, y para la *dark* copiamos los de la *light* y simplemente vamos cambiando los colores que deben aplicar: el título será blanco, la descripción será de un gris claro, el botón de un verde diferente al que usábamos en el modo claro, y el fondo de un gris bien oscuro.

GeneXus by Globant

Light Mode vs Dark Mode






```
@color-scheme = light
{
    #colors
    {
    colorPrimary: #015547;
    colorSecondary: #616161;
    colorButton: #73D94F;
    colorBackground: #FFFFFF;
    }
}
```

```
@color-scheme = dark
{
    #colors
    {
    colorPrimary: #FFFFFF;
    colorSecondary: #C1C1C1;
    colorButton: #326321;
    colorBackground: #191819;
    }
}
```

De esta manera, nuestra aplicación quedará adaptada a las necesidades de todos los usuarios.

Hemos visto una introducción de todo lo que podemos hacer usando el Design System Object. En un próximo video, estudiaremos cómo importar a nuestra KB un diseño desde Figma creado por los diseñadores.


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