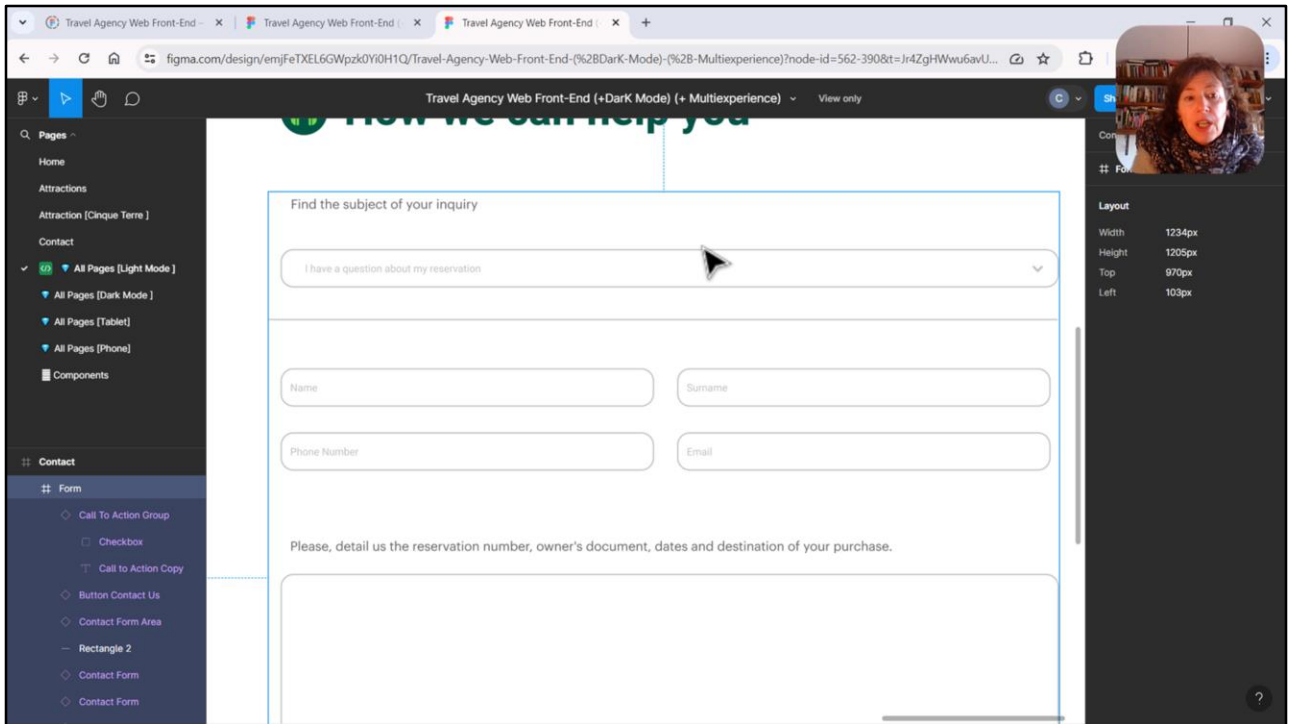


# Contact Panel: Input Fields

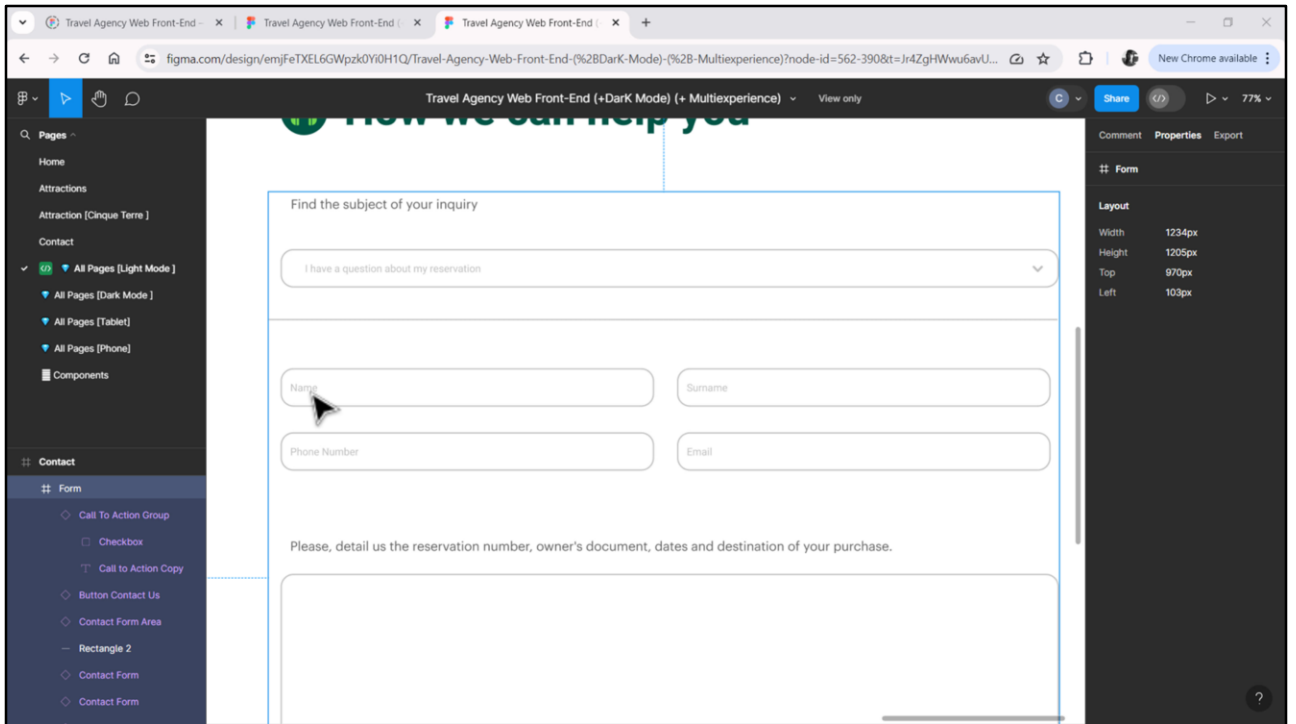


Cecilia Fernández



Para terminar de ver las cuestiones más relevantes en lo que hace al desarrollo de la aplicación Angular para tamaño Desktop nos estaría faltando por ver cómo implementamos los campos de entrada, que en nuestro caso van a estar dados únicamente en la pantalla Contact.

Allí podemos ver que hay primero un combo box para que el usuario seleccione de entre una serie de opciones el tema de su consulta. La descripción que vemos fuera del combo será la etiqueta de la variable. Y vemos que dentro del combo aparece un mensaje de sugerencia. Cuando el usuario presione allí o en la flechita, las opciones se expandirán para que seleccione una.

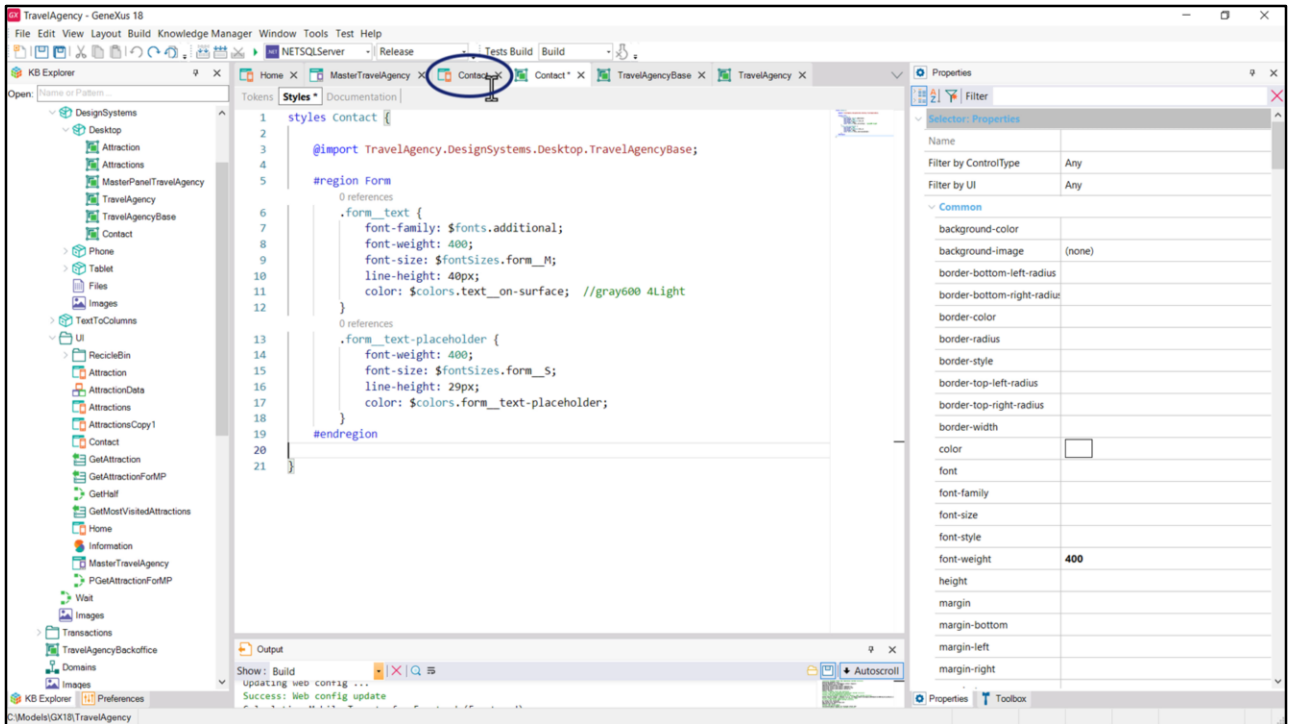


Luego hay cuatro campos de tipo edit, donde el usuario podrá ingresar datos personales. Estos campos no tienen etiqueta descriptiva porque a través del mensaje de invitación a escribir que aparece dentro cuando no se ha ingresado allí ningún valor, ya queda claro de qué se trata.

Cuando el usuario comience a digitar dentro de esos campos, el mensaje de invitación desaparecerá. Reaparecerá sólo si el usuario vacía el campo.

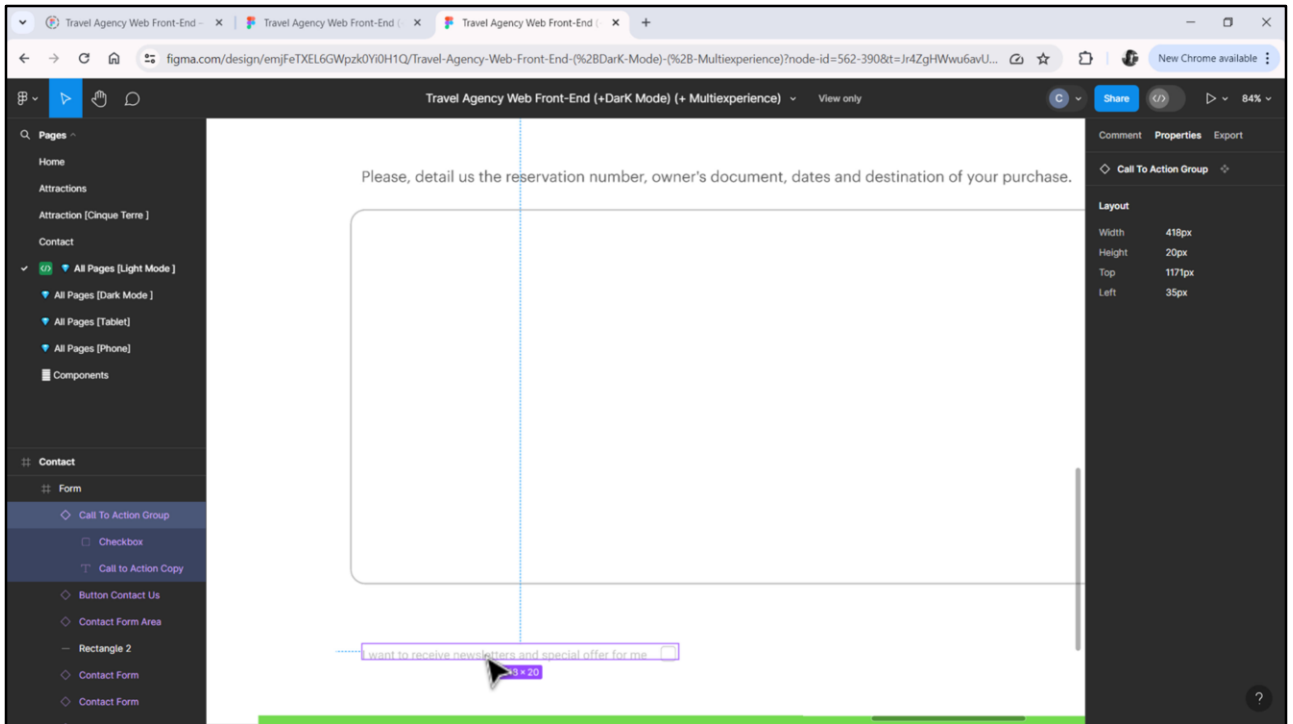
	A1	Name										
10	Banner / H1	Banner	banner_h1	additional	600	Graphik	Semibok	36	-	-		banner_XL
11	Banner / H2		banner_h2	secondary	500	Rubik	Medium	20	-	-		banner_L
12	Card Attraction / H1	Card-Attraction	card-attractions_h1	primary	800	Heebo	ExtraBoli	36	36	20		card-attractions-Big_XL
13			card-attractions-small_h1					36	20	12		card-attractions-Small_XL
14			card-attraction_h1					36	23	24		card-attraction_XL
15	Card Attraction / Location		card-attractions_location	secondary	400	Rubik	Regular	14	14	12		card-attractions-Small_S
16			card-attractions-small_location					14	12	10		card-attractions-Small_S
17	Card Attraction / Rating		card-attractions_rating	secondary	500	Rubik	Medium	38	38	16		card-attractions-Big_M
18			card-attractions-small_rating					38	16	12		card-attractions-Small_M
19			card-attraction_rating					38	21	-		card-attraction_M
20	Form / Regular Text	Form	form_text	additional	400	Graphik	Regular	20	12	12		form_M
21	Form / Place Holder		form_text-placeholder	primary	400	Heebo	Regular	16	10	10		form_S

Si recordamos la etapa de preparación, habíamos ingresado dos clases tipográficas para estos textos: una a la que le habíamos llamado form\_\_text, para los textos de las etiquetas, y otra a la que le habíamos llamado form\_\_text-placeholder, para los textos internos, justamente estos de invitación.



Vean que creé un DSO Contact para dar estilo al panel del mismo nombre, y lo agregué al DSO raíz, TravelAgency.

Me voy a copiar estas clases para nuestro DSO. Si sólo tendré campos de entrada en este panel, entonces podría dejarlas sólo aquí y borrarlas del DSO base.

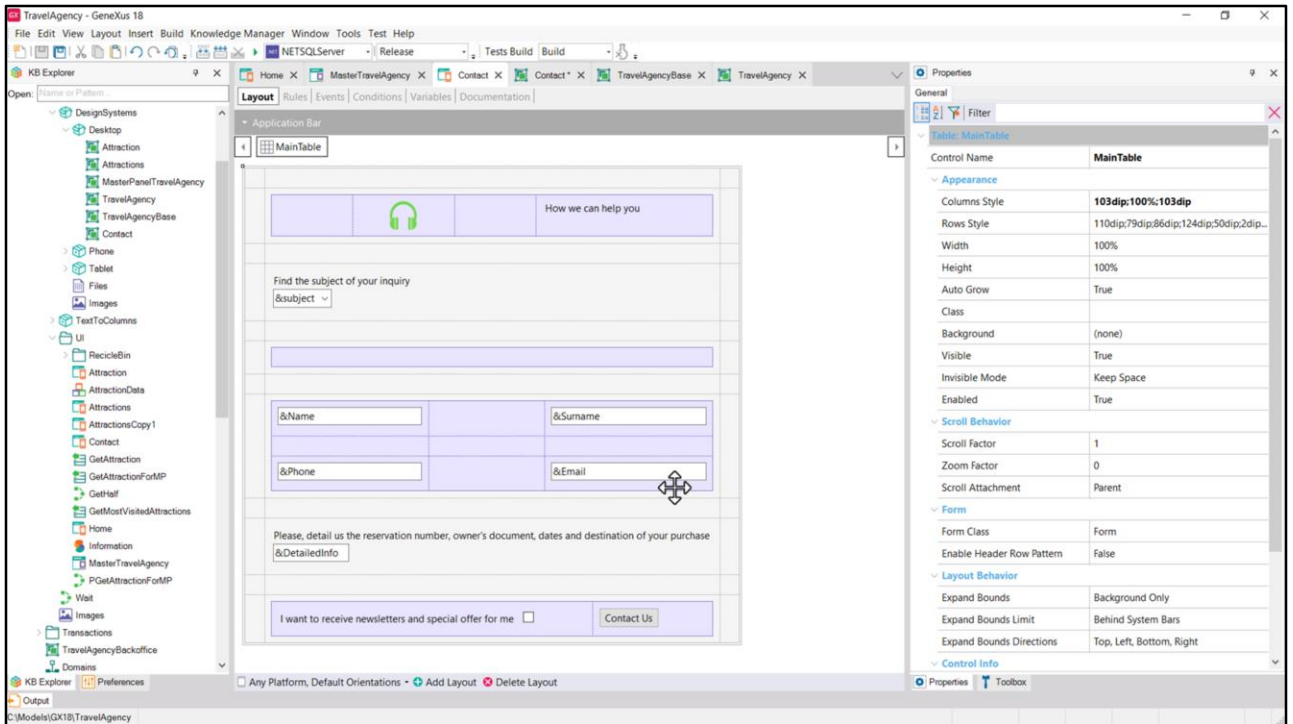


Bueno, si seguimos analizando la página, tengo otro campo Edit con etiqueta arriba, y debajo del todo tengo un campo de tipo Check box, con la etiqueta a la izquierda.

Es importante que implementemos todos estos controles como variables con sus etiquetas, y no como dos controles por separado: un control textblock para la etiqueta y un control variable propiamente dicho para el campo. ¿Por qué? Porque, primero, conceptualmente es lo más adecuado, y segundo, y como consecuencia de ello, porque para la accesibilidad tendrá efectos: si se trata del mismo campo ya se entiende toda la semántica. Si son dos por separado, habrá que relacionarlos y ni siquiera así se logrará lo mismo.

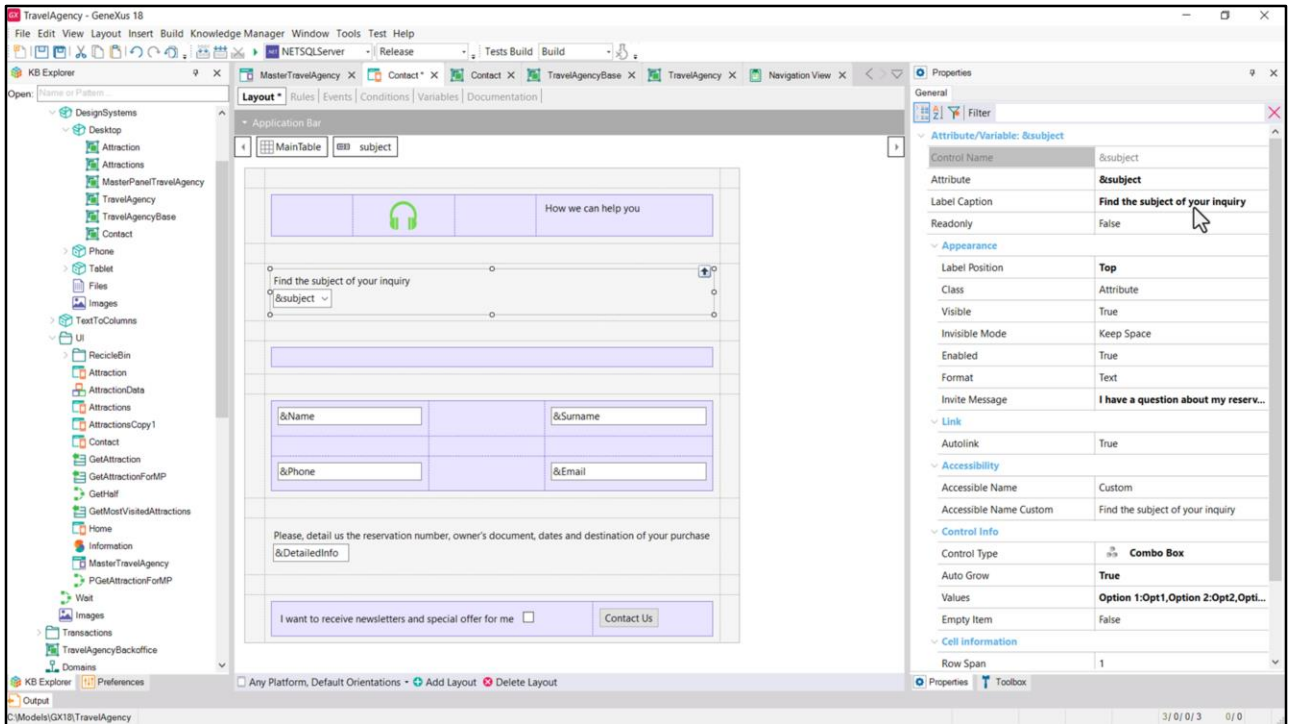
Entonces lo que quiero mostrarles en este video es cómo modelamos la User Interface del control dado que será único, pero tenemos que modelarle dos cosas: el campo en sí y la etiqueta. Pero si el campo es uno sólo, entonces ¿cómo hacemos?

Se los voy a mostrar en detalle para el primer caso y el resto podrán deducirlo ustedes solos. Les voy a dejar de todos modos un xpx con parte de la solución para que ustedes la completen.



Primero veamos que claramente todos los campos irán en una tabla, para conseguir la alineación.

Aquí yo ya creé las variables que serán de entrada y las inserté en las tablas que vemos. Nada de esto requiere conocimientos nuevos, es lo que hemos venido haciendo, así que no vamos a perder tiempo viéndolo otra vez más.



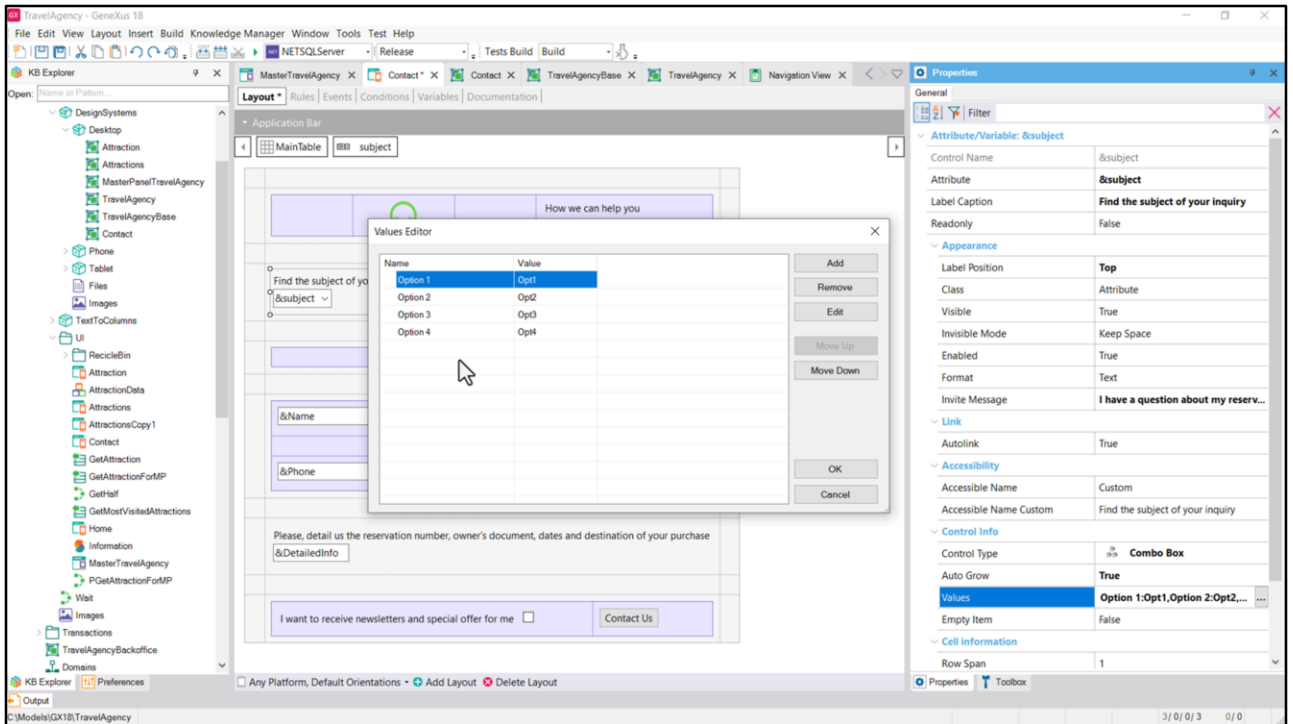
Vamos a dedicarnos ahora a la primera variable. Una vez que la inserté, vean que le coloqué de Label Caption el texto que tomé de la propiedad en Figma.

Por defecto las variables en los paneles son de entrada, y por ello la propiedad Readonly está en False.

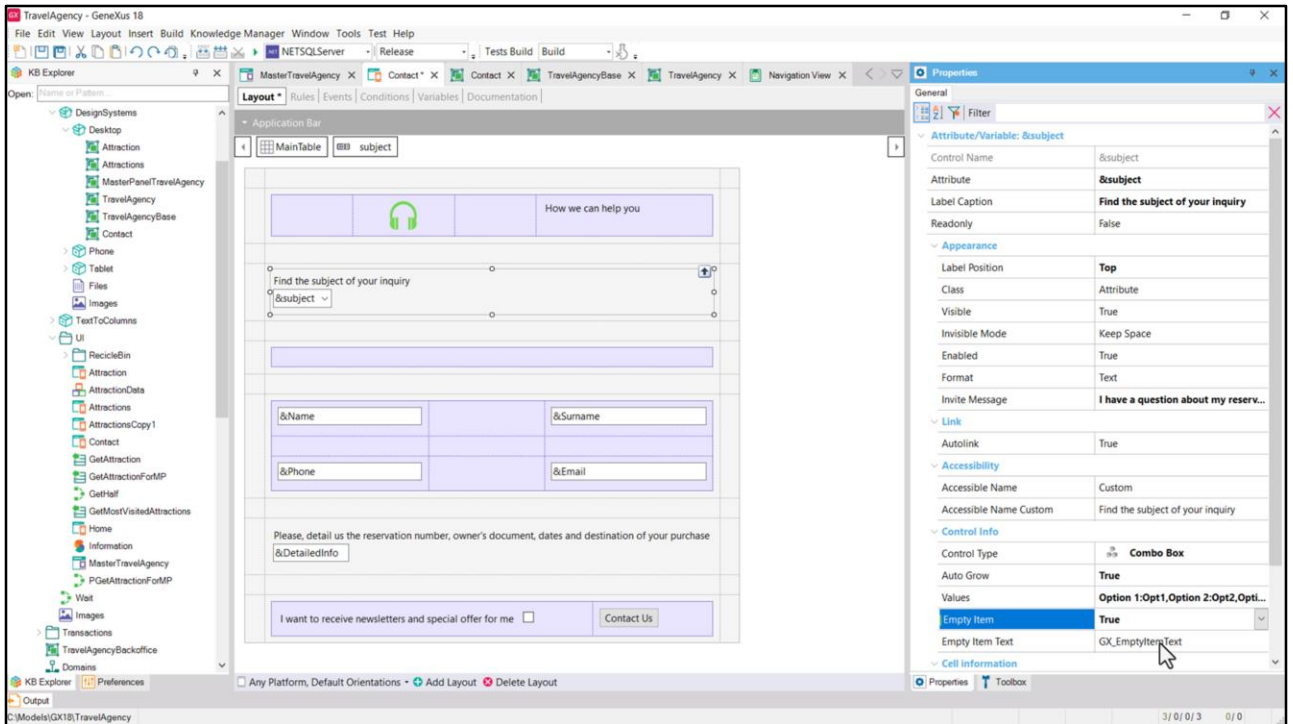
Vean que le cambié el valor por defecto a la propiedad Label Position, para que coloque la etiqueta arriba del campo y no a la izquierda, como es el valor default. Además configuré la propiedad Invite Message, para que este sea el texto que aparezca dentro del campo cuando está vacío.

Y por último, vean que la forma de hacer que la variable en lugar de ser de tipo Edit, que es el valor por defecto (es el de estas de abajo, por ejemplo)... bueno, que en vez de ser, entonces, decía, de tipo edit, sea combo box, es a través de la propiedad Control Type.





Cuando se elije este valor, Combo Box, aparece la propiedad Values para poder justamente proporcionar las opciones, es decir los nombres que verá el usuario en la pantalla para cada opción, y el valor que internamente quedará almacenado en la variable cuando se seleccione la opción correspondiente.



Podríamos agregar un ítem que corresponda al valor vacío o no seleccionado, y a través de esta propiedad le asignamos un texto, que por defecto es el que está codificado en esta constante, y que es "(none)" (lo vamos a ver ahora en ejecución). Podríamos colocar aquí el texto que deseamos para el valor vacío, si no nos sirve ese que nos presenta por defecto GeneXus, que, como la variable es de tipo varchar, corresponderá al valor cadena vacía. Podríamos haber elegido que la variable fuera de valor numérico, por ejemplo, porque igual lo que va a mostrar en la pantalla son los nombres de las opciones, no los valores. El usuario jamás va a ver el valor real. Pero elegí que fuera de tipo varchar por si más adelante quiero que no sea más un combo sino que sea un campo Edit.

Observen también que por defecto la propiedad Accessible Name Custom tomó el valor de la propiedad Label Caption.

Podríamos ejecutar ahora, que todavía no le asigné clases a los controles, para poder apreciar bien después la diferencia.

The screenshot shows a web browser window with the address bar displaying "localhost:51544/TravelAgency/Contact-Level\_Detail". The page features a header image of a bridge at night. Below the image is a green circular chat icon. The main heading is "How we can help you" with a headset icon. A form section titled "Find the subject of your inquiry" contains a text input field with the value "I have a question about my reservation". A dropdown menu is open below the input field, showing the following options: "(None)", "Option 1", "Option 2", "Option 3", and "Option 4".

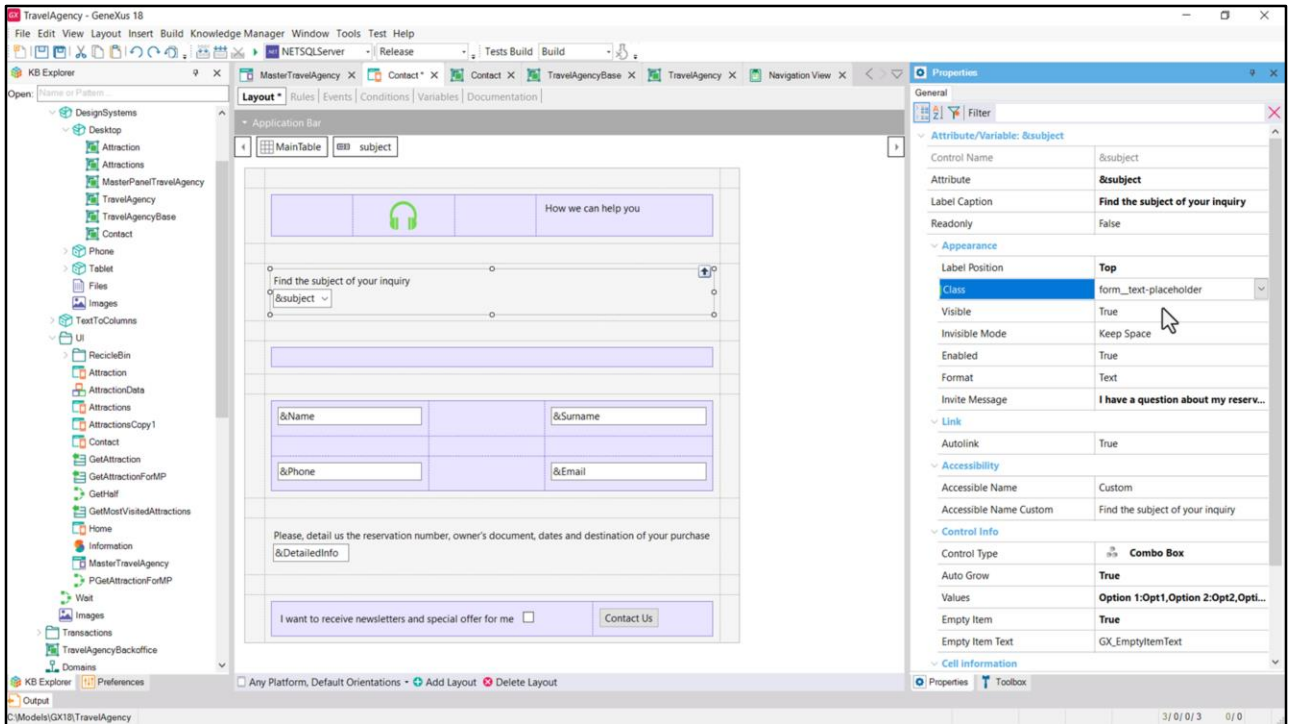
Aquí vemos la etiqueta y vemos el mensaje de invitación y también vemos los nombres de las opciones, y este "None" que les decía.

The screenshot shows a web browser window with the URL `localhost:51544/TravelAgency/Contact-Level_Detail`. The page content includes a green headphones icon, the text "How we ca", and a form with the text "Find the subject of your inquiry" and "I have a question about my reservation". A dropdown menu is open, showing options: "(None)", "Optid1", "Option 2", "Option 3", and "Option 4".

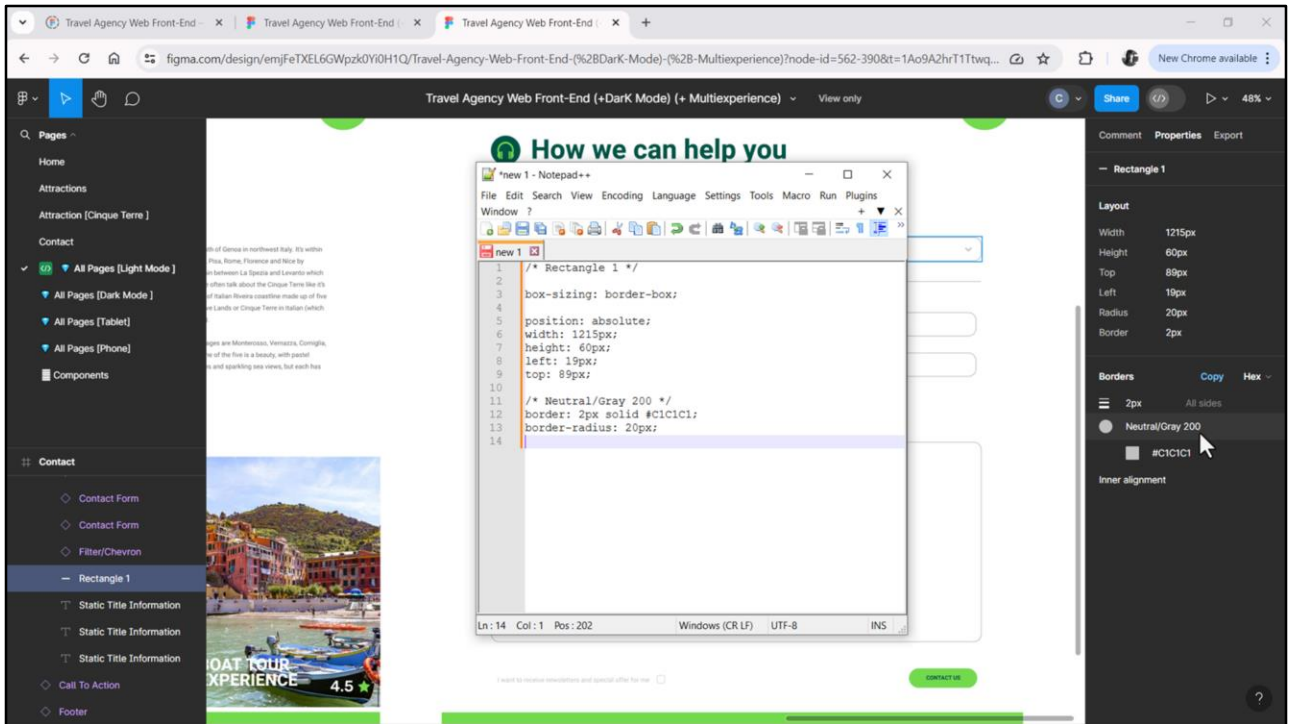
Overlaid on the page is a Google Sheets spreadsheet with the following data:

	A	B	C	D	E
5	Button	Button	button	primary	800
6	Menu Label	Menu	.menu_label	primary	500
7	Copyright	Footer	copyright	secondary	400
8	Card Home / H1	Card-Home	card-home_h1	primary	800
9	Card Home / H2		card-home_h2	secondary	500
10	Banner / H1	Banner	banner_h1	additional	600
11	Banner / H2		banner_h2	secondary	500
12	Card Attraction / H1	Card-Attraction	card-attractions_h1	primary	800
13			card-attractions-small_h1		
14			card-attraction_h1		
15	Card Attraction / Location		card-attractions_location	secondary	400
16			card-attractions-small_location		
17	Card Attraction / Rating		card-attractions_rating	secondary	500
18			card-attractions-small_rating		
19			card-attraction_rating		
20	Form / Regular Text	Form	form_text	additional	400
21	Form / Place Holder		form_text_placeholder	primary	400
22					

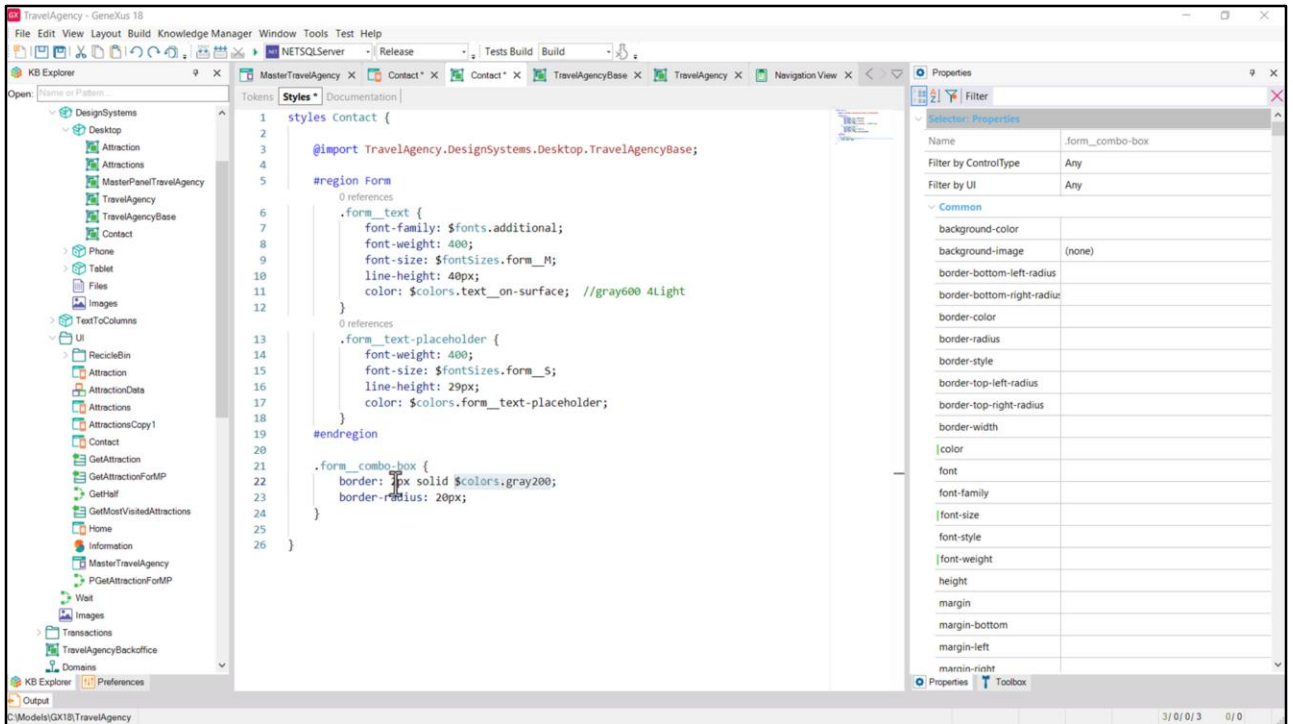
Sabemos que la clase para la tipografía de la etiqueta era la que habíamos llamado `form__text`, y la del campo en sí, que será la de las opciones del combo, la que habíamos llamado `form__text-placeholder`. Luego veremos la del Invite Message.



Entonces a nuestro control le asociaremos esta segunda clase... pero esta es únicamente la tipográfica. No la del control combo en sí mismo. Y todavía no hicimos nada en lo relativo a la etiqueta.

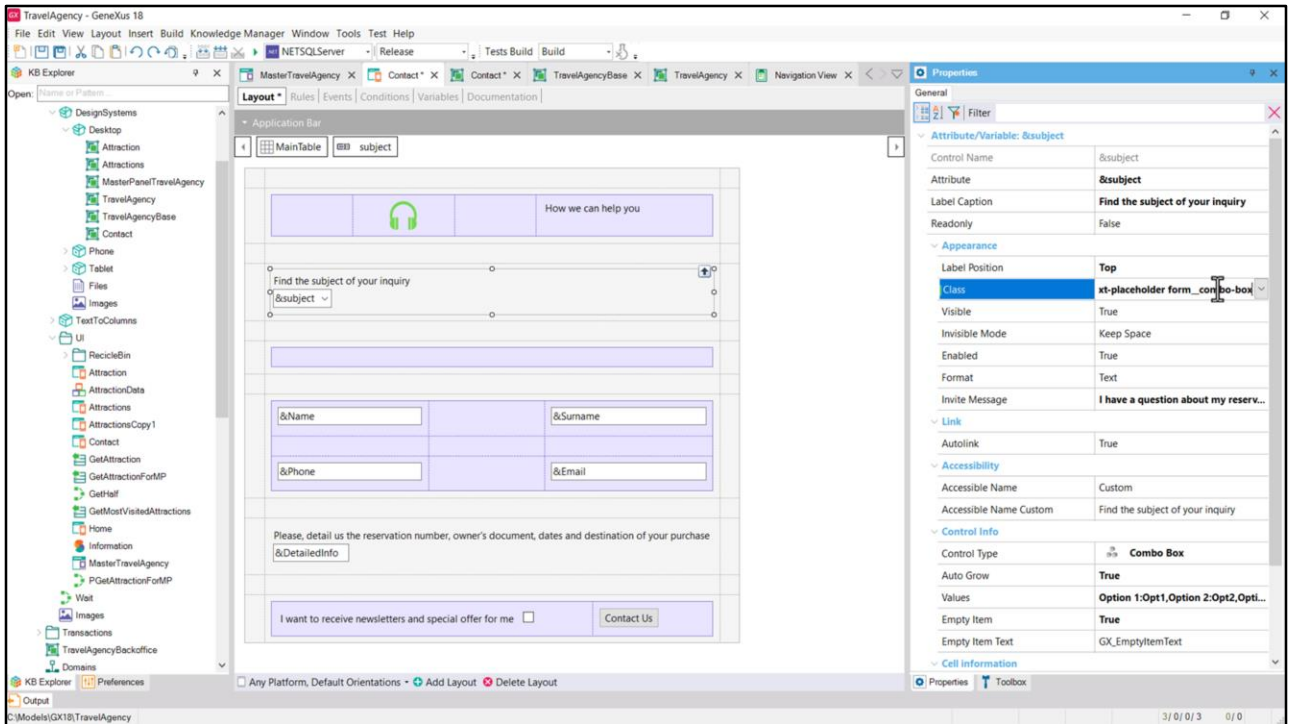


En Figma vemos estas propiedades radius y border... si pedimos el código CSS... tendremos que llevar a GeneXus estas dos propiedades... este es el color de nuestro token gray200...



Así que voy a crear una clase para modelar la parte del combo-box propiamente dicha.

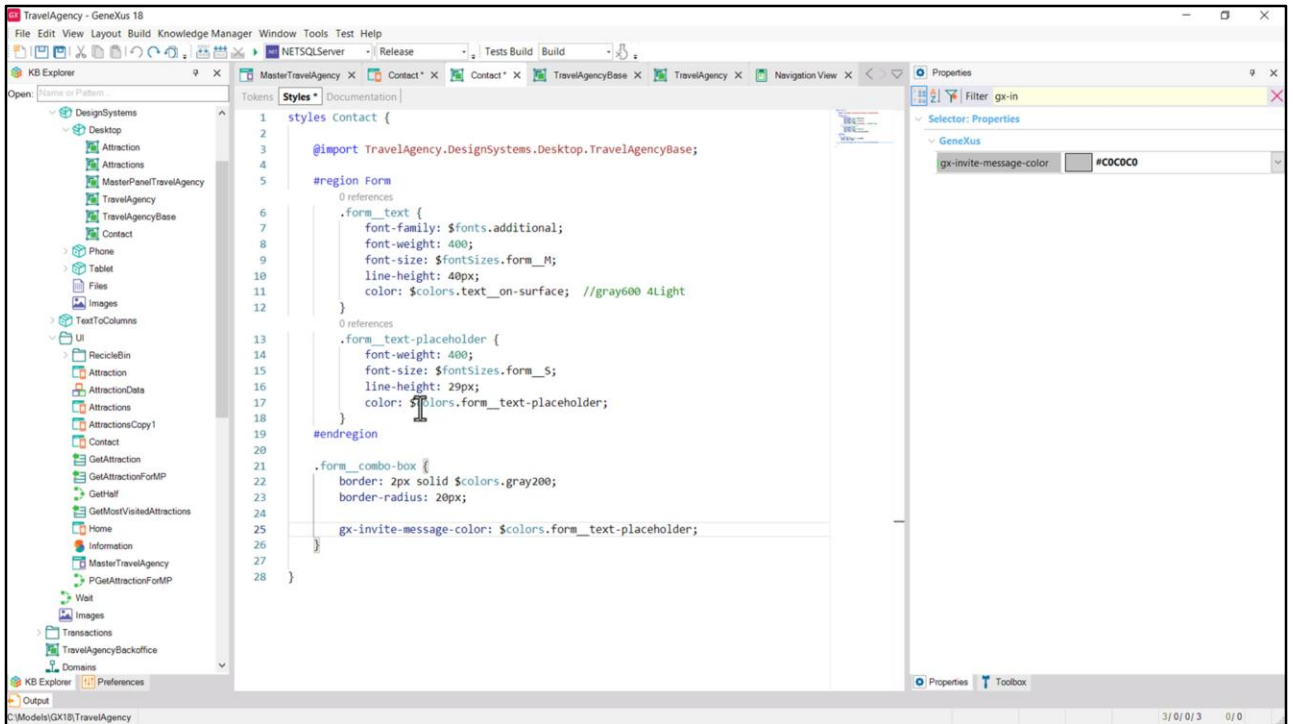
Le llamaré form\_\_combo-box... y aquí copio las dos propiedades y sustituyo aquí por el token de color.



Y por supuesto se la asigno al control combo box. Así que este control va a tener dos clases: la de la tipografía y la que da estilo al combo box propiamente dicho.

Podríamos incluir las dos en una, o mantenerlas separadas, si por ejemplo fuera a utilizar solo la tipografía en otro control.

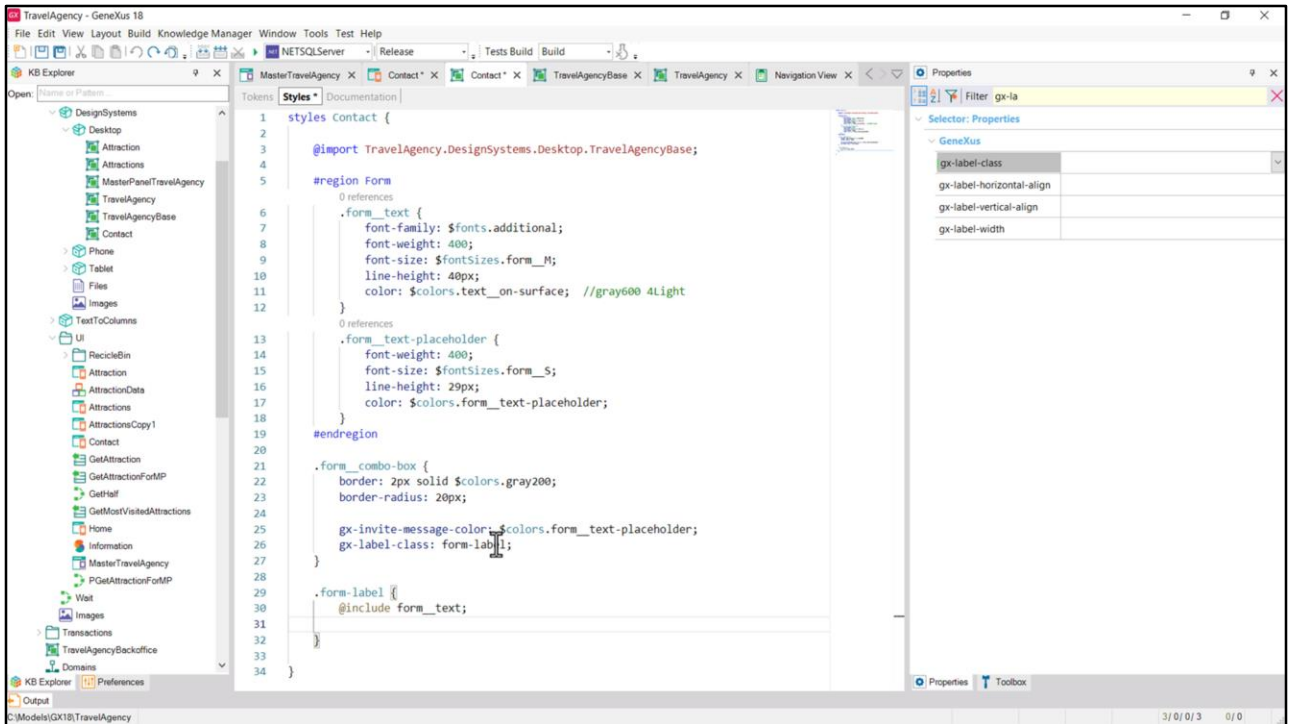




Para indicar el color del Invite Message, tenemos una propiedad GeneXus... vamos a buscarla filtrando por las propiedades para controles de tipo Attribute (que es el mismo tipo que el de las variables).

Filtremos mejor por "gx-in" ... y allí la vemos. Si elegimos desde aquí un color ya lo agrega.

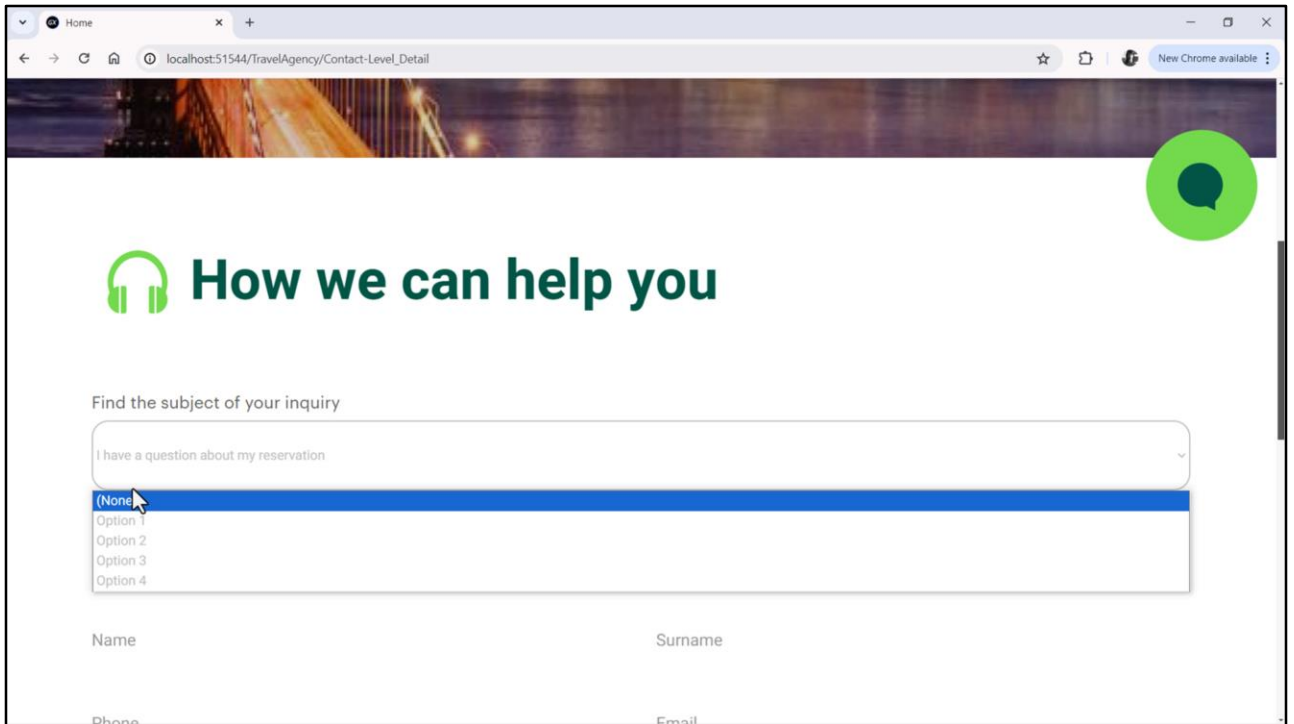
Vamos a colocarle el token de color que utilizamos para la clase tipográfica del control.



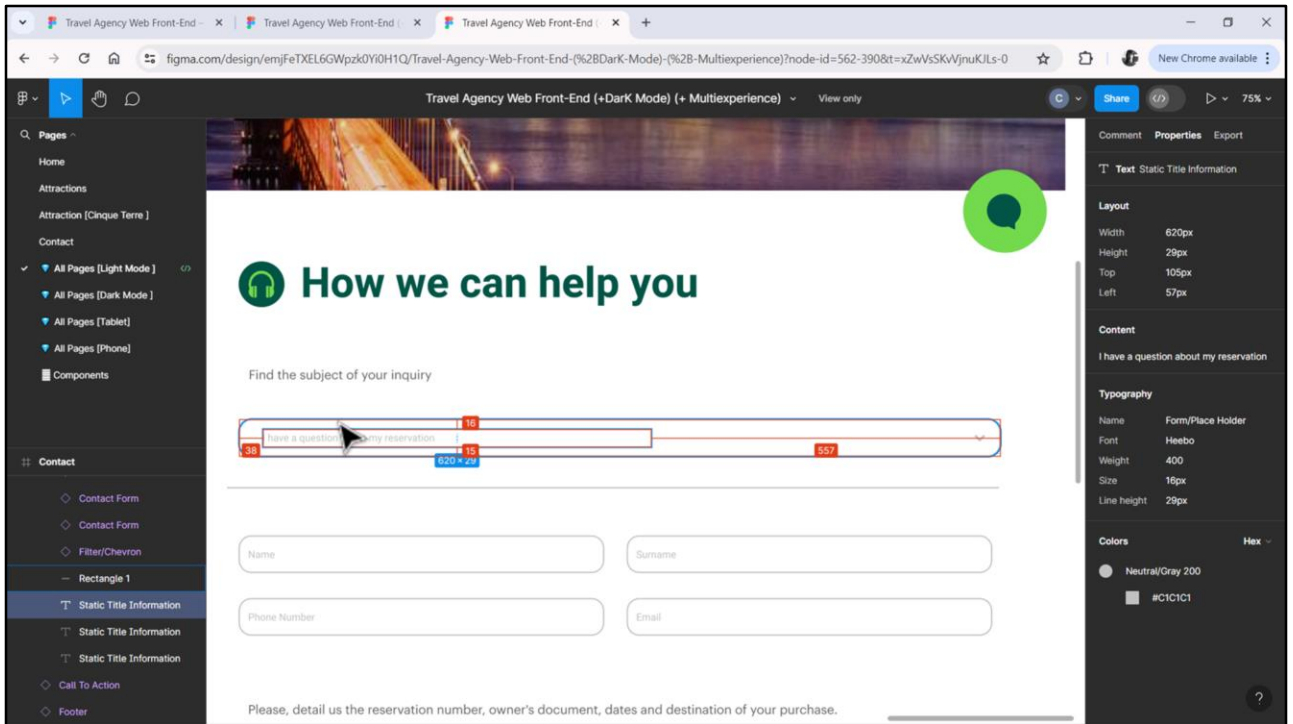
Y ahora sí, asignémosle la clase para la etiqueta... será gx-label-class. Lo que hará será asignarle una nueva clase, a la que llamaré form-label.

Por ahora solamente la definiré incluyendo la clase tipográfica, pero ya veremos que deberemos agregarle más propiedades.

Ejecutemos para ver lo hecho hasta aquí.

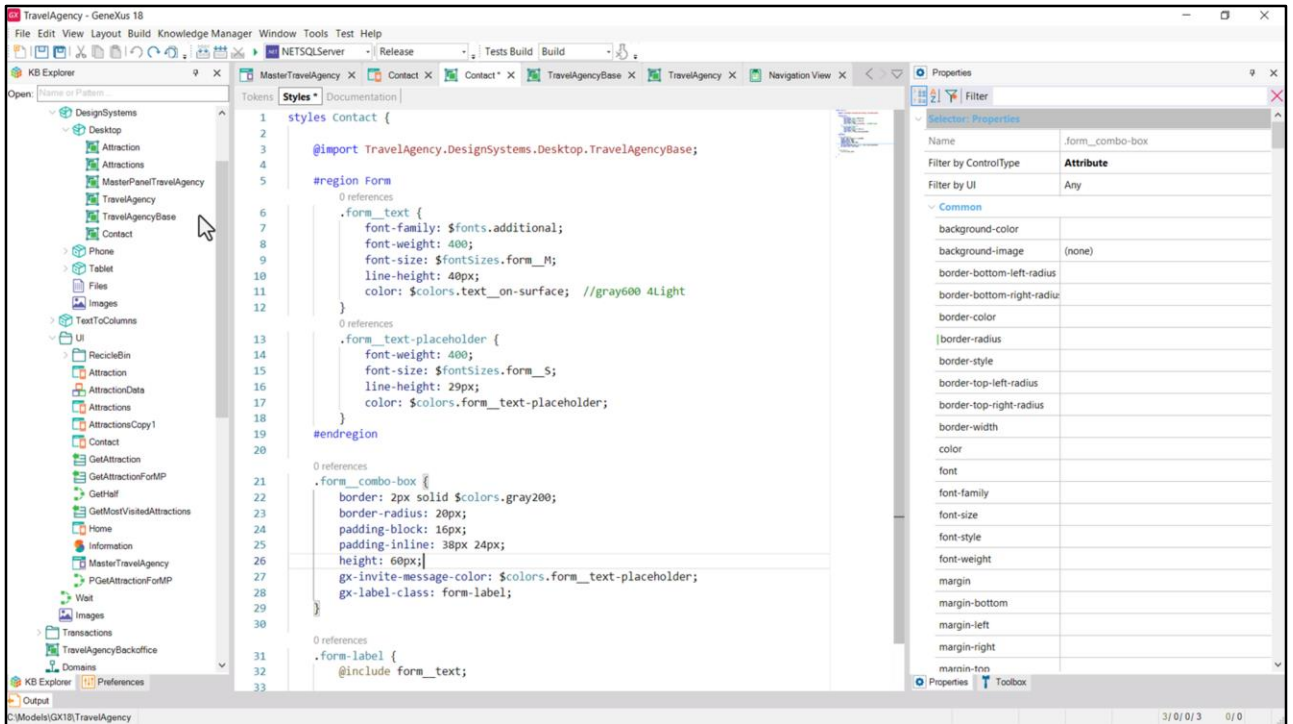


Empieza a tomar forma. Pero observemos el texto del combo. Está pegado al borde izquierdo... y el alto del campo también luce mayor al de Figma.



Si inspeccionamos... vemos que tiene de alto 84 píxeles... pero si vamos a Figma, el alto debería ser de 60.

Y además, vemos que el texto debería ser de 29, con un padding de arriba de 16, y de abajo de 15. De la izquierda de 38... y de la derecha...de 25.



Entonces a la clase deberíamos agregarle estas propiedades:

- padding-block, es decir, el padding en la dirección vertical, de 16 píxeles (el padding de abajo era de 15, pero dejemos los dos iguales, de 16).
- Y padding-inline, es decir en la dirección horizontal, de 38 píxeles del principio, y 24 del fin.
- Y especifiquemos el height de 60 píxeles.

label.form\_\_text-placeholder-label.form\_\_combo-box-label.form\_\_text-placeholder-l... 882.08 x 40

Find the subject of your inquiry

I have a question about my reservation

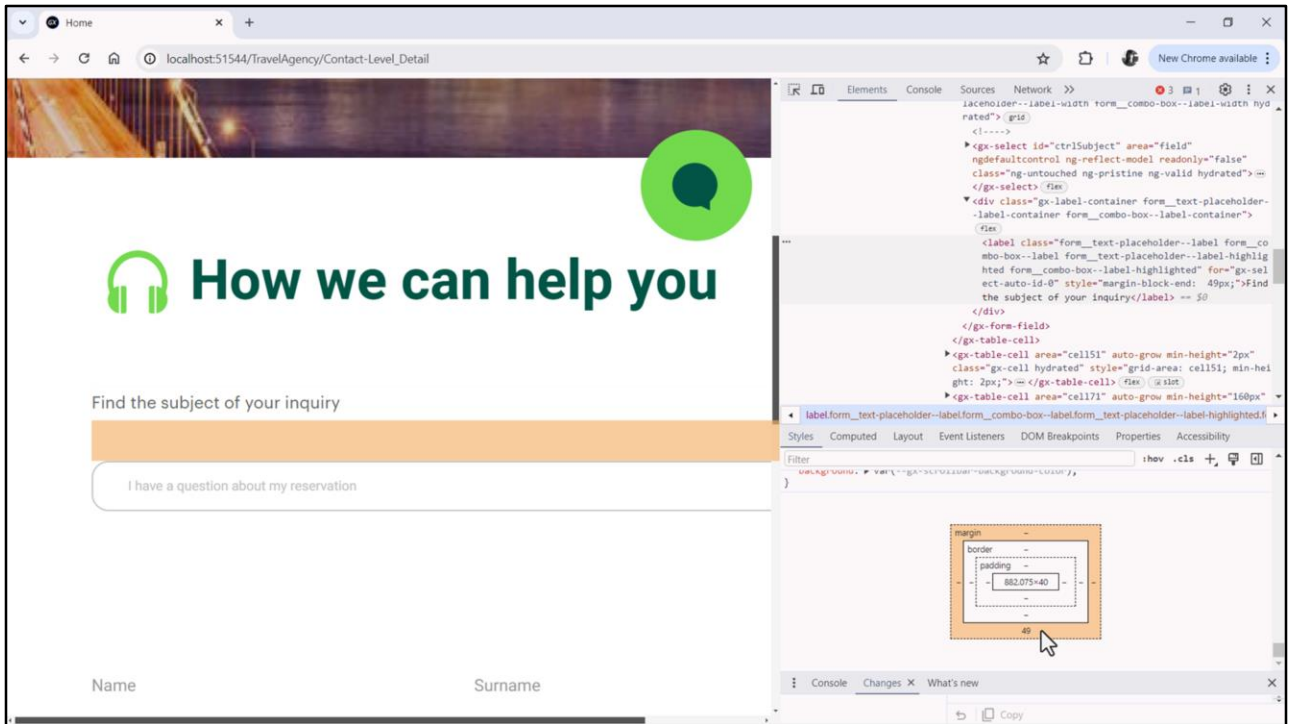
Name Surname

```
label class="form__text-placeholder--label form__co
mbo-box-label form__text-placeholder--label-highlig
hted form__combo-box-label-highlighted" form="gx-sele
ct-aut" style="font-size: 0px;">Find the subject of your inquiry
</label>
</div>
</gx-form-field>
</gx-table-cell>
</table>
</div>
</div>
```

```
label.form__text-placeholder-label.form__combo-box-label.form__text-placeholder-label-highlighted.f
Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility
Filter
element.style {
}
gx-form-field.gx-label-container>label {
  transition-property: background-color, border-color, box-shadow, color, filter,
  opacity, transform;
  transition-duration: 0.25s;
  flex: 1;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
.form-label, .form__combo-box-label {
  font-family: var(--fonts_additional);
  font-weight: 400;
  font-size: var(--fontsizes_form_m);
}
```

Bien, ahora, ¿qué pasa con la etiqueta?

Si la inspecciono veo que tiene 40 de alto, que veremos que es el valor de line-height de la tipografía, sin márgenes ni paddings.



Vean qué pasa si por ejemplo le coloco a este elemento (es sólo para probar rápido) un margin-block-end de 49. Allí vemos los 49 del margen y los 40 de alto...

localhost:51544/TravelAgency/Contact-Level\_Detail

# How we can help you

Find the subject of your inquiry

I have a question about my reservation

Name Surname

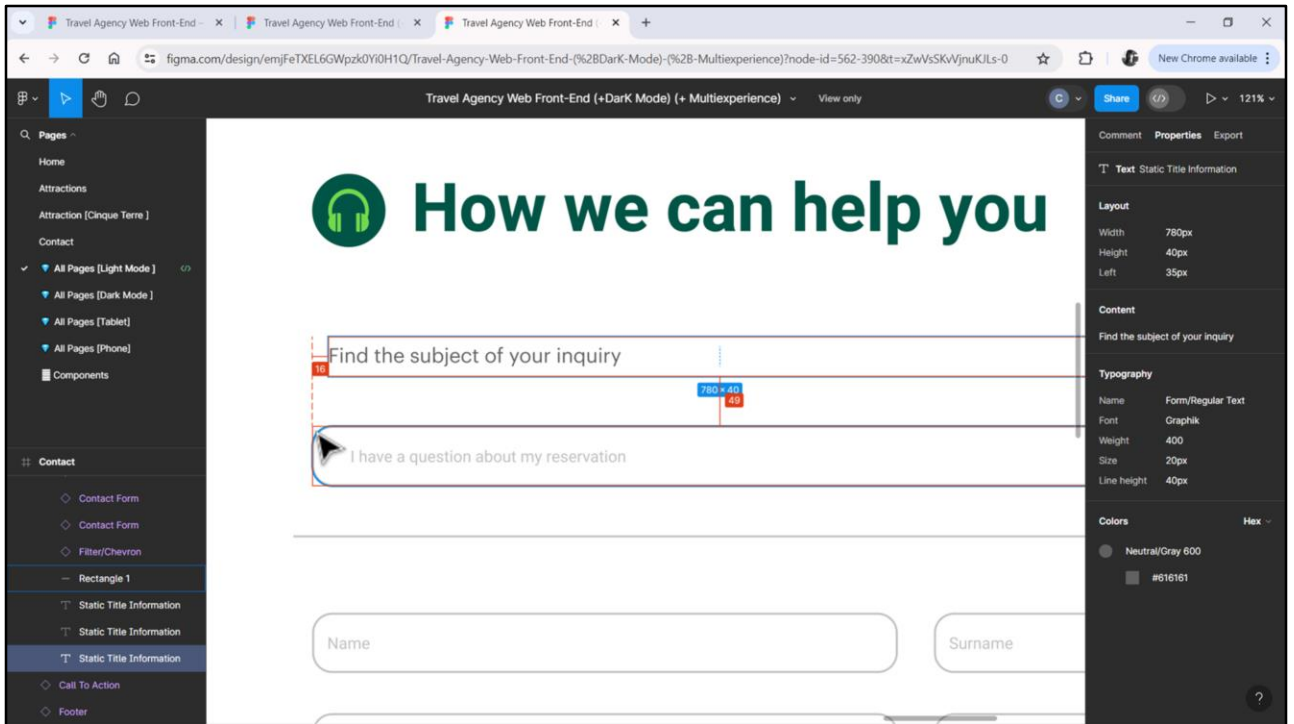
```
element.style {  
  margin-block-end: 49px;  
  margin-inline-start: 16px;  
}
```

```
gx-form-field>gx-label-container>label {  
  transition-property: background-color, border-color, box-shadow, color, filter,  
  opacity, transform;  
  transition-duration: 0.25s;  
  flex: 1;  
  white-space: nowrap;  
  overflow: hidden;  
  text-overflow: ellipsis;  
}
```

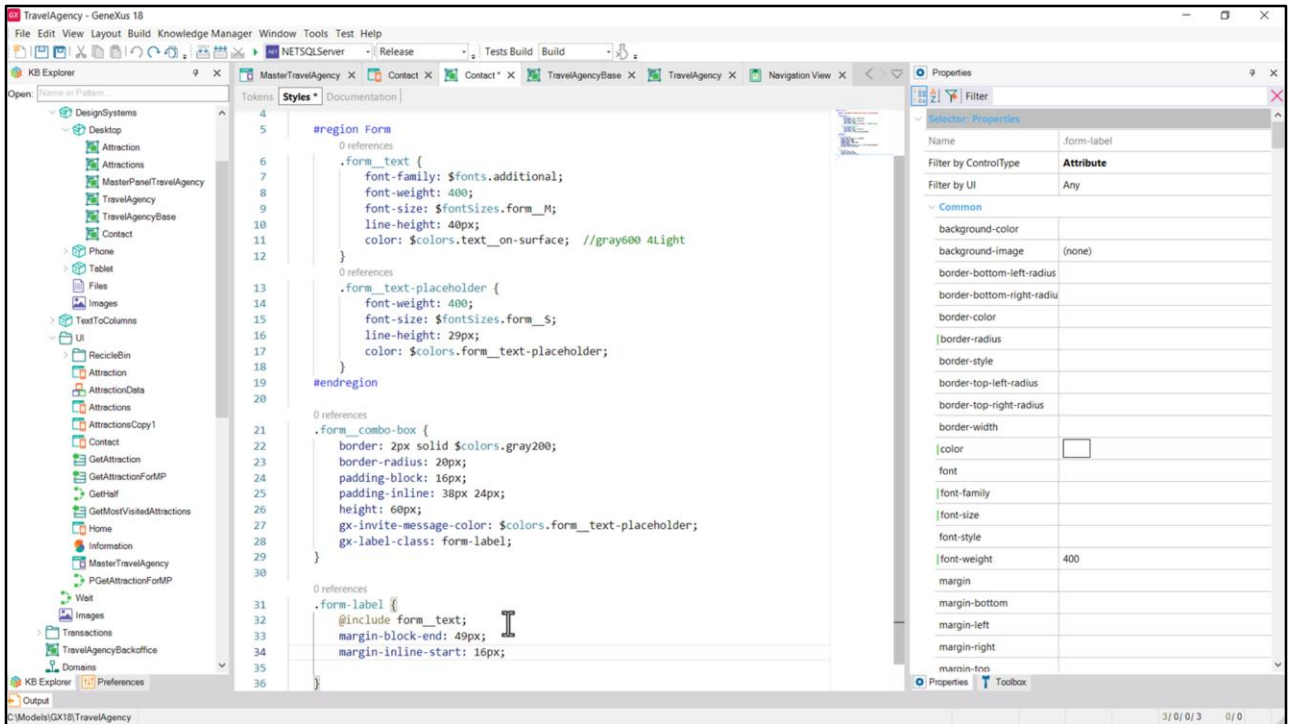
```
.form-label, .form__combo-box--label {  
  font-family: var(--fonts_additional);  
}
```

Y si le agregamos un margin inline start de 16...





Coincide exactamente con lo que vemos en Figma.





Entonces, a la clase de la etiqueta le agregamos las dos propiedades.

Home x +

localhost:51544/TravelAgency/Contact-Level\_Detail

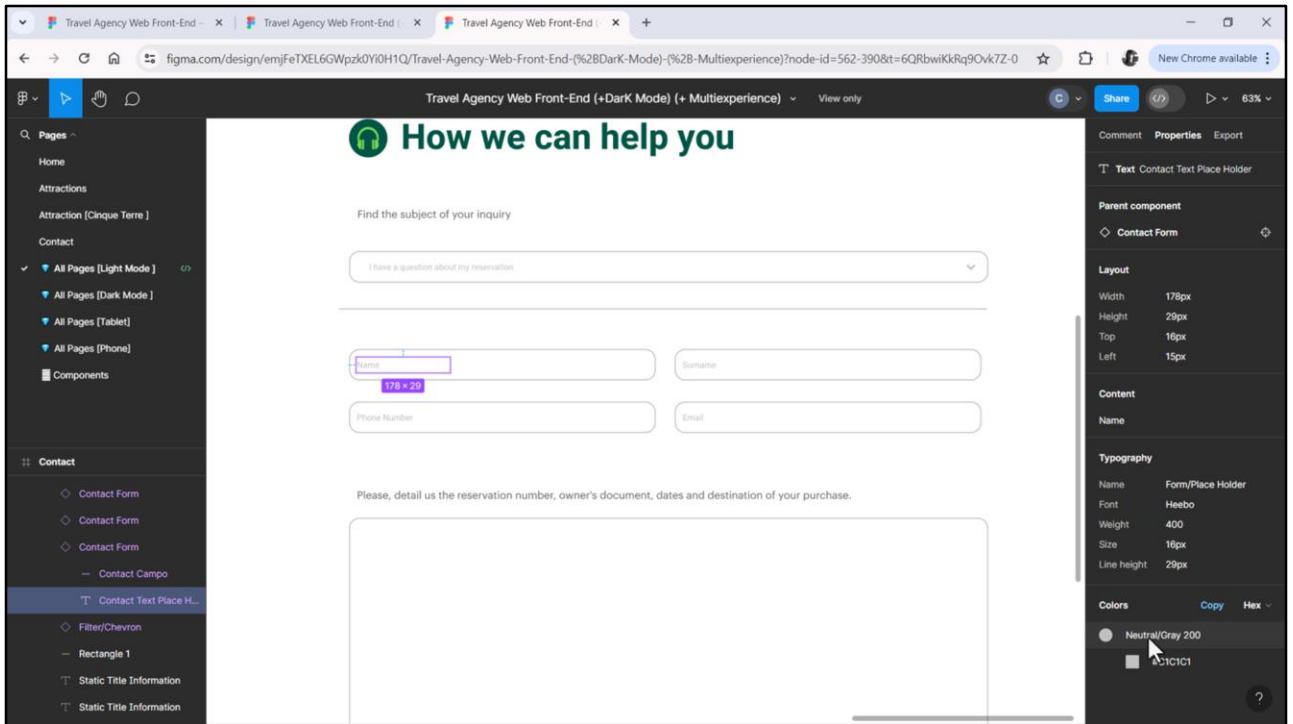
New Chrome available



# How we can help you

Find the subject of your inquiry

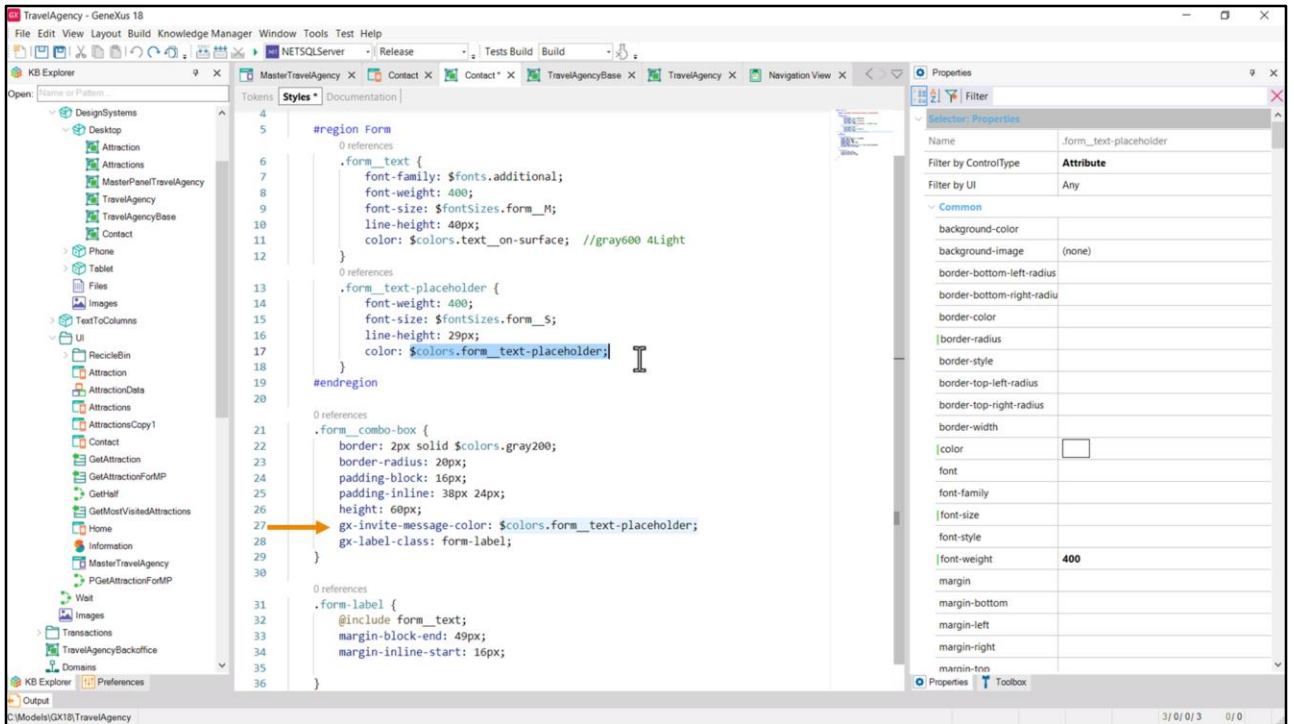
I have a question about my reservation



No fui muy precisa antes: en verdad en el diseño de Chechu no vemos por ningún lado la tipografía correspondiente a los valores que el usuario ingresará en los campos.

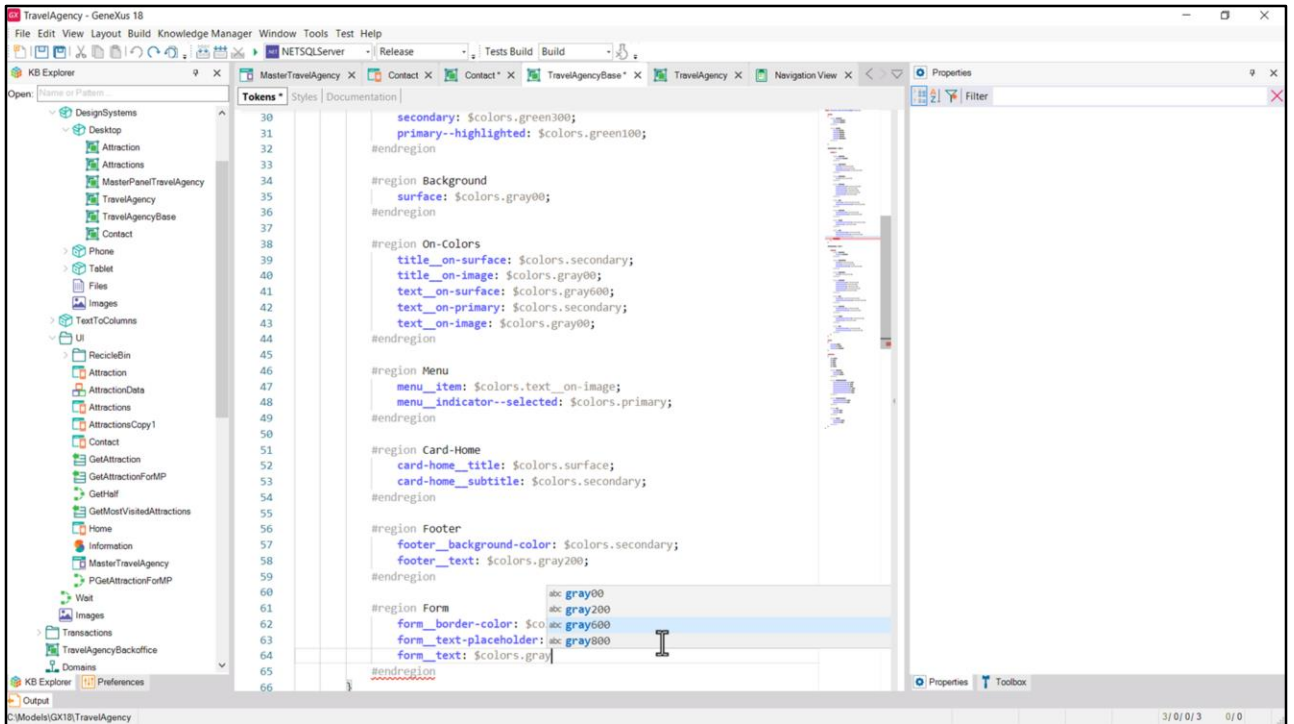
Nuestra clase `form__text-placeholder` en verdad corresponde a los mensajes de invitación.

En general el mensaje de invitación tiene la misma tipografía que el contenido del campo, salvo en lo que hace al **color**. En general el mensaje de invitación es mucho más claro, suele ser un gris claro.

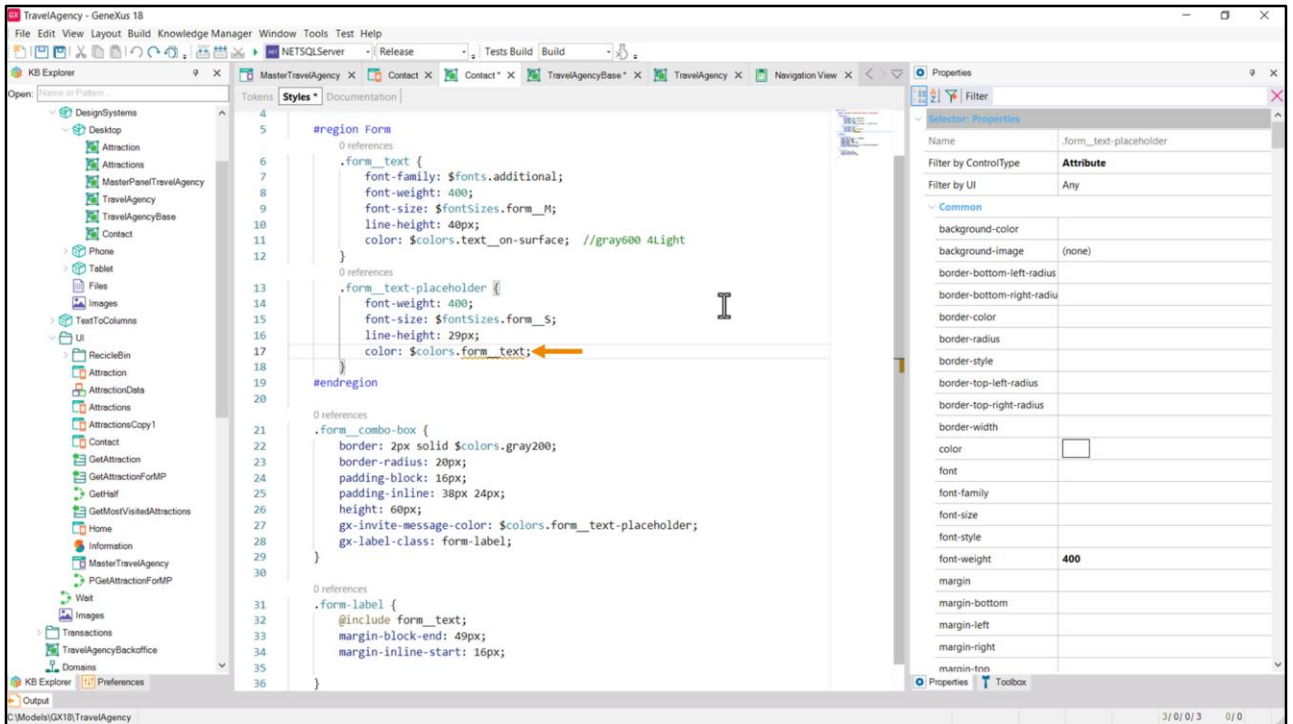


Y es por ello que existe la propiedad para poder indicarle el color.

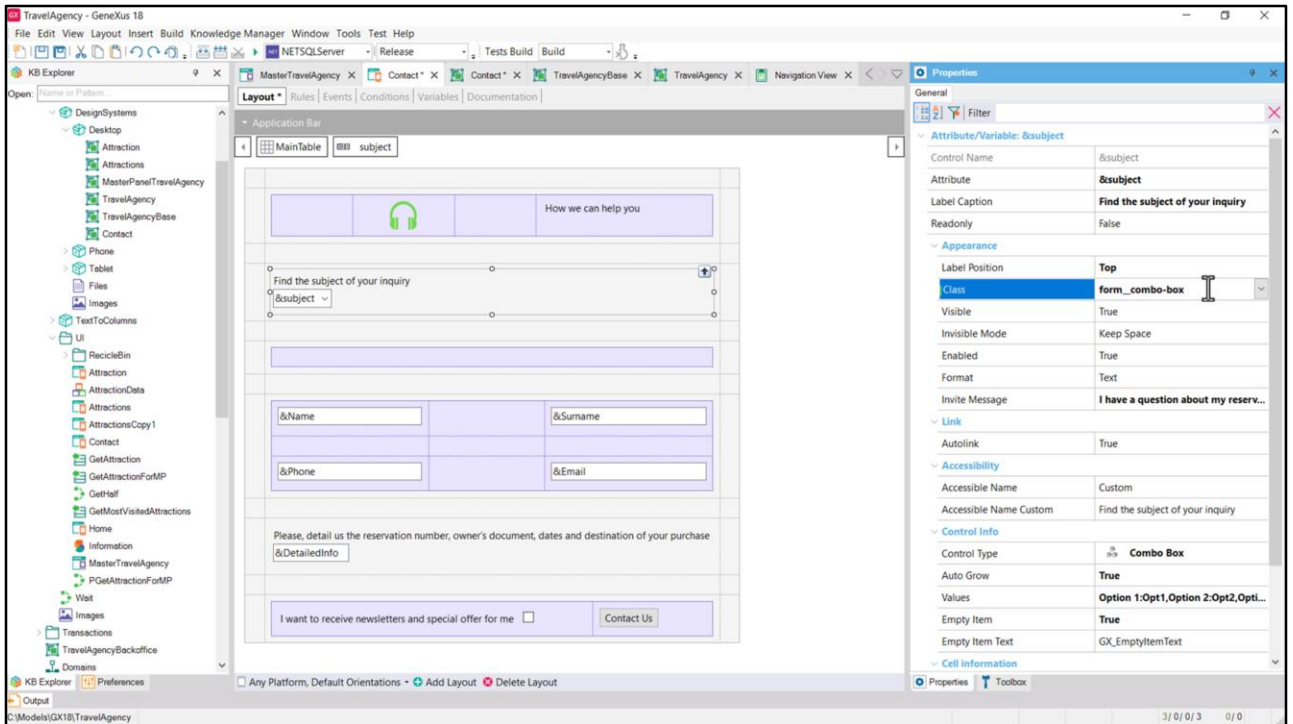
Así que este valor es correcto, pero lo que no sabemos, entonces, es el color que debería tener el contenido del campo. Habría que consultar esto con la diseñadora.



Posiblemente esto nos llevaría a definir otro token de color para el contenido... pongámosle por ejemplo este valor...

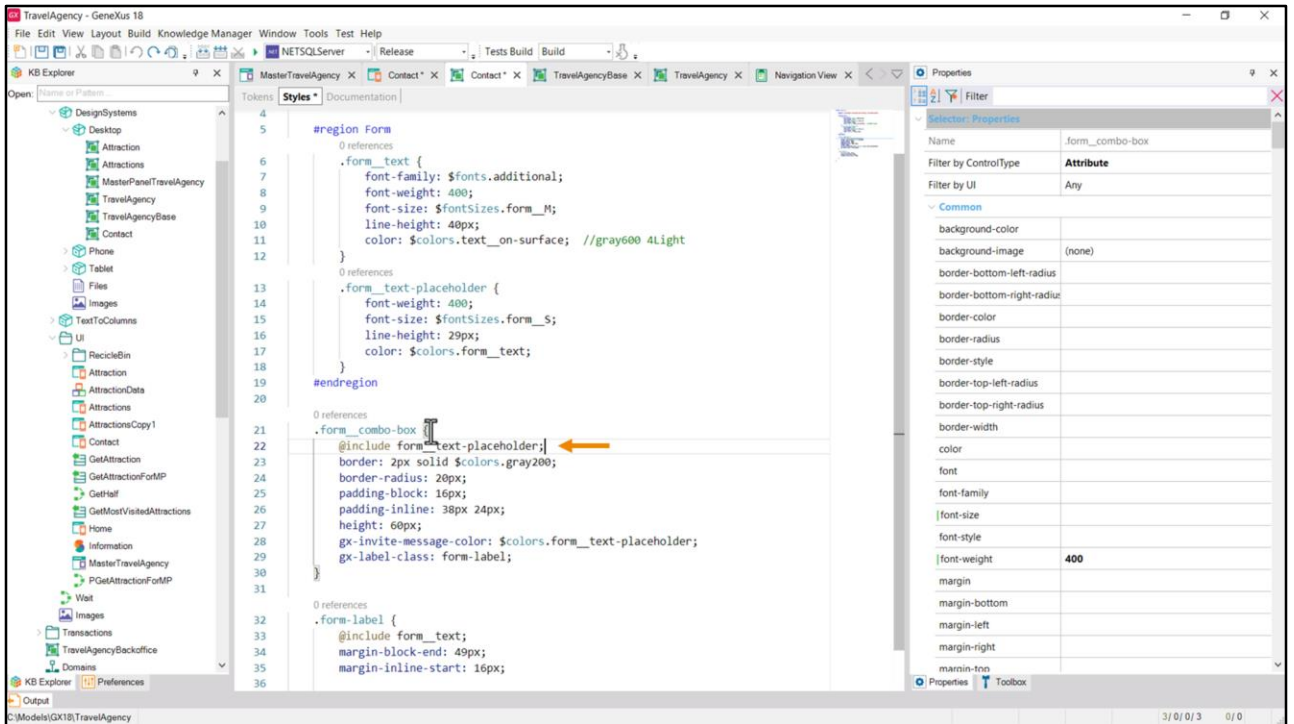


...Y aquí referenciaríamos a ese token.

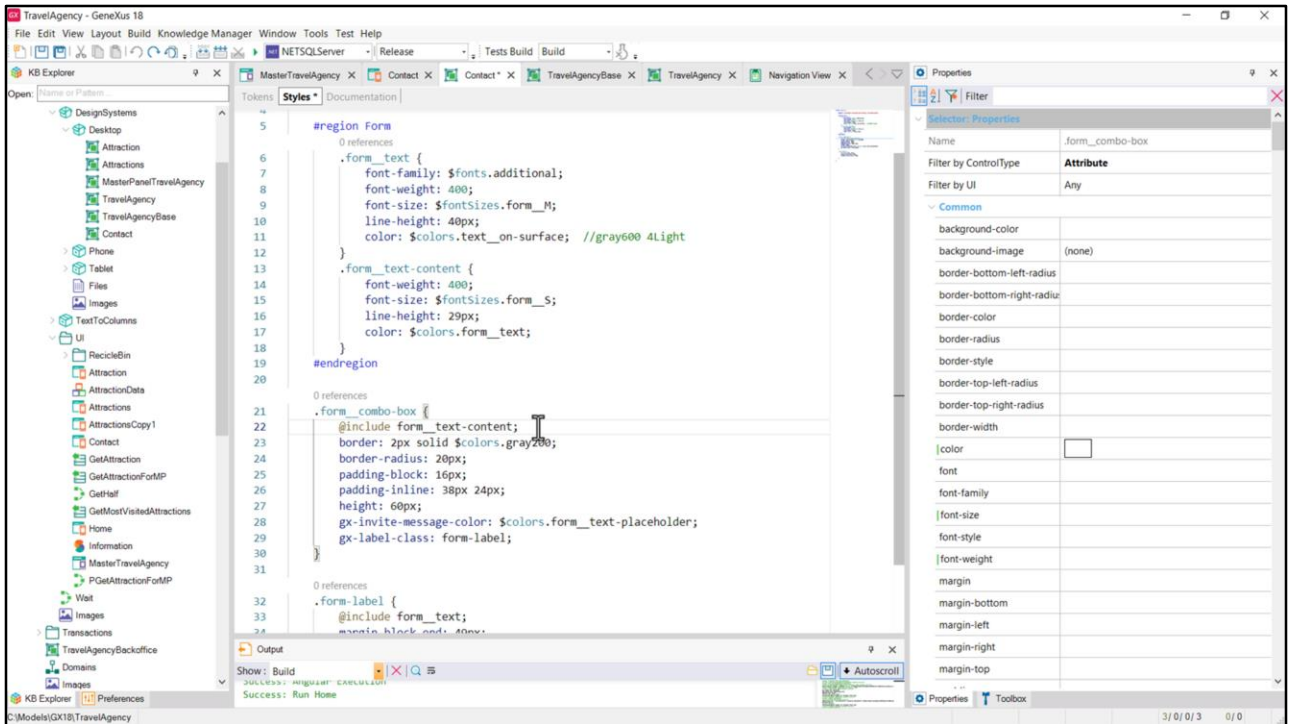


Para ordenar un poco mejor todo esto, yo le dejaría asociado al combo-box una única clase que contenga todo... y no dos...



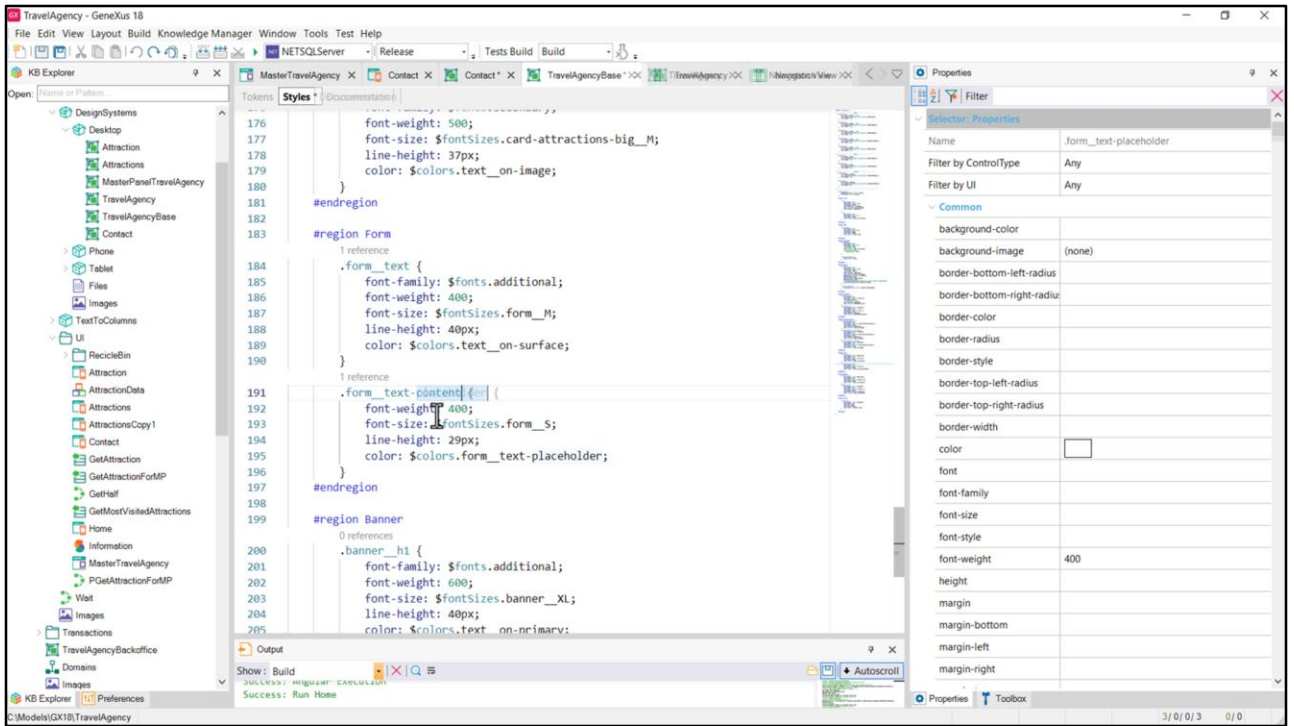


Y entonces, en esta clase, incluyo la tipografía.

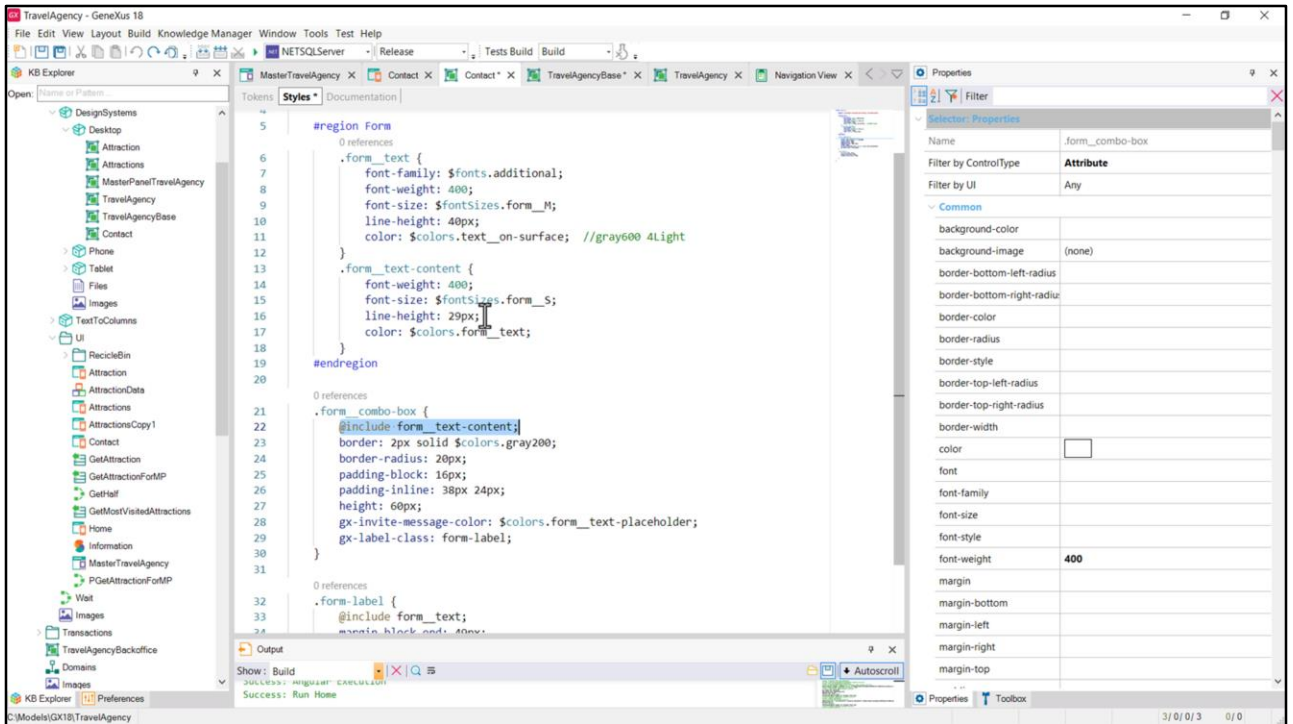


Llegado a este punto nos damos cuenta de que el análisis inicial que hicimos de las clases tipográficas que íbamos a necesitar para los campos de entrada no fue nada fino, y en verdad esta clase más bien debería llamarse form\_\_text-content...

Como es la primera vez que la usamos, es el momento de cambiarle el nombre.

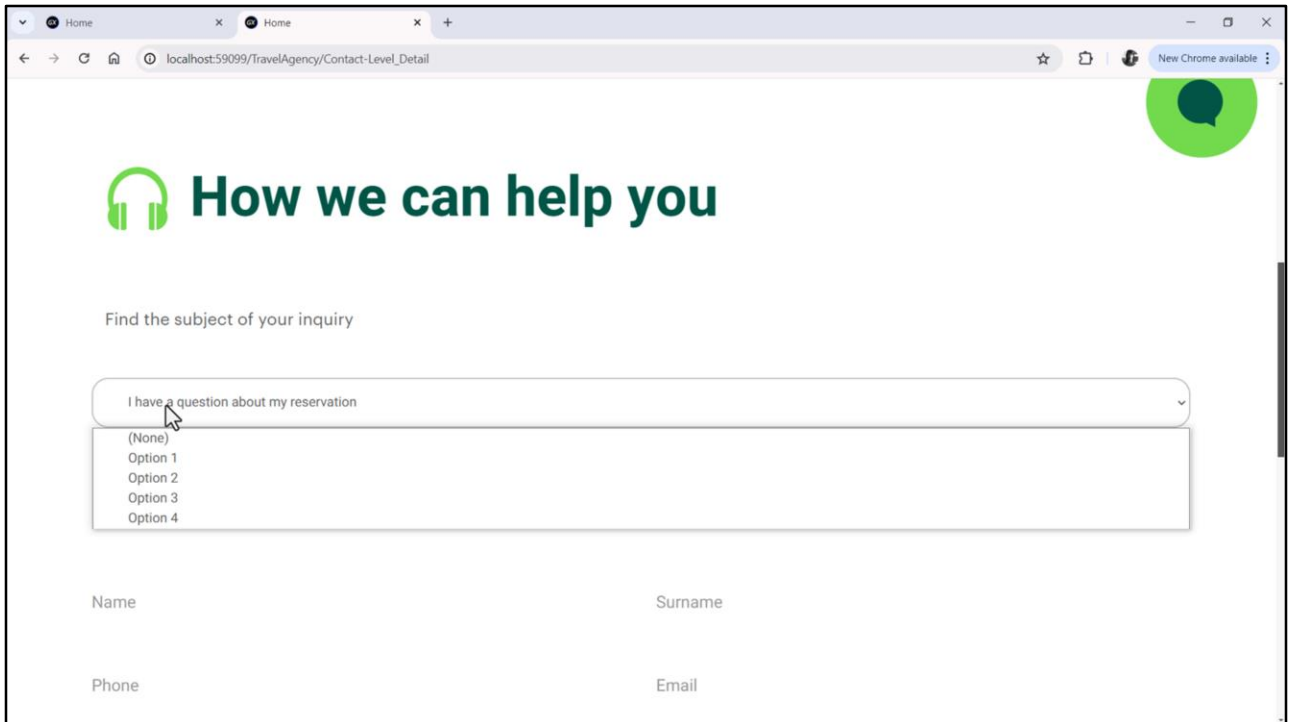


Si no la borramos del DSO base, entonces cuidado porque la tenemos repetida.

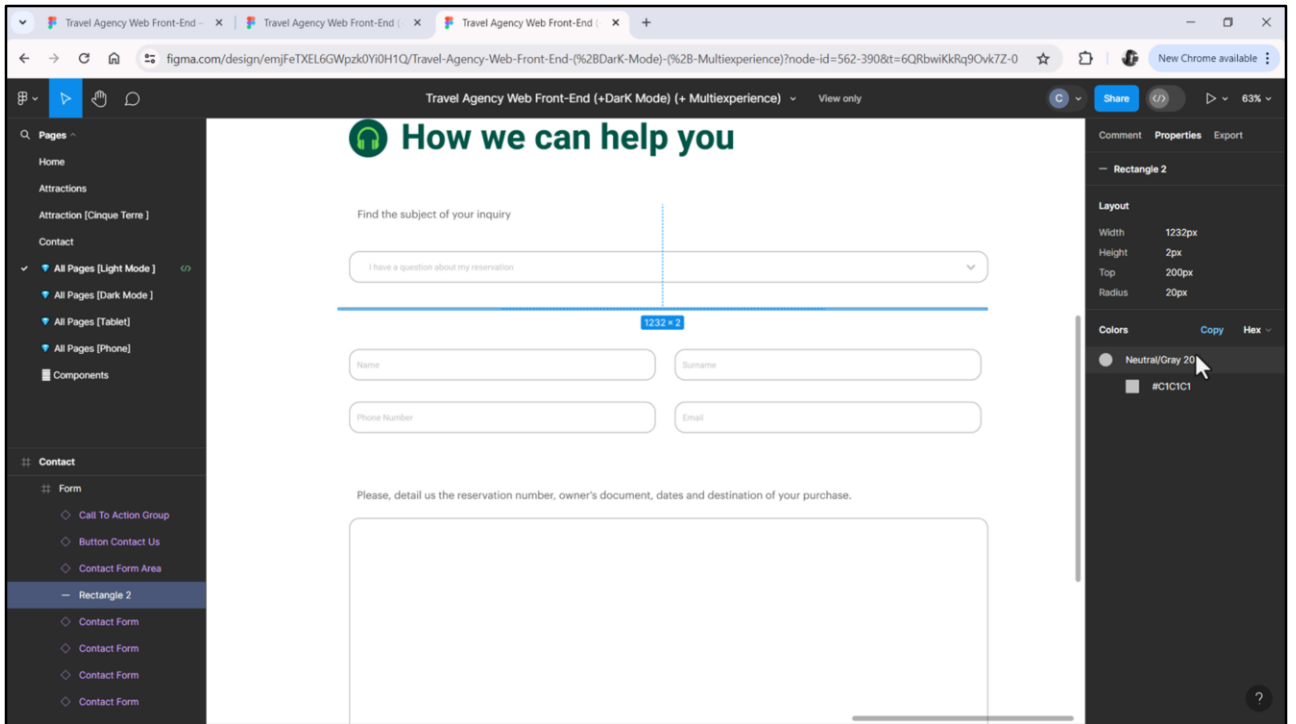


La copié a este porque recuerden que si quiero utilizar la regla include, por ahora es necesario que la clase a ser incluida esté en el mismo DSO.

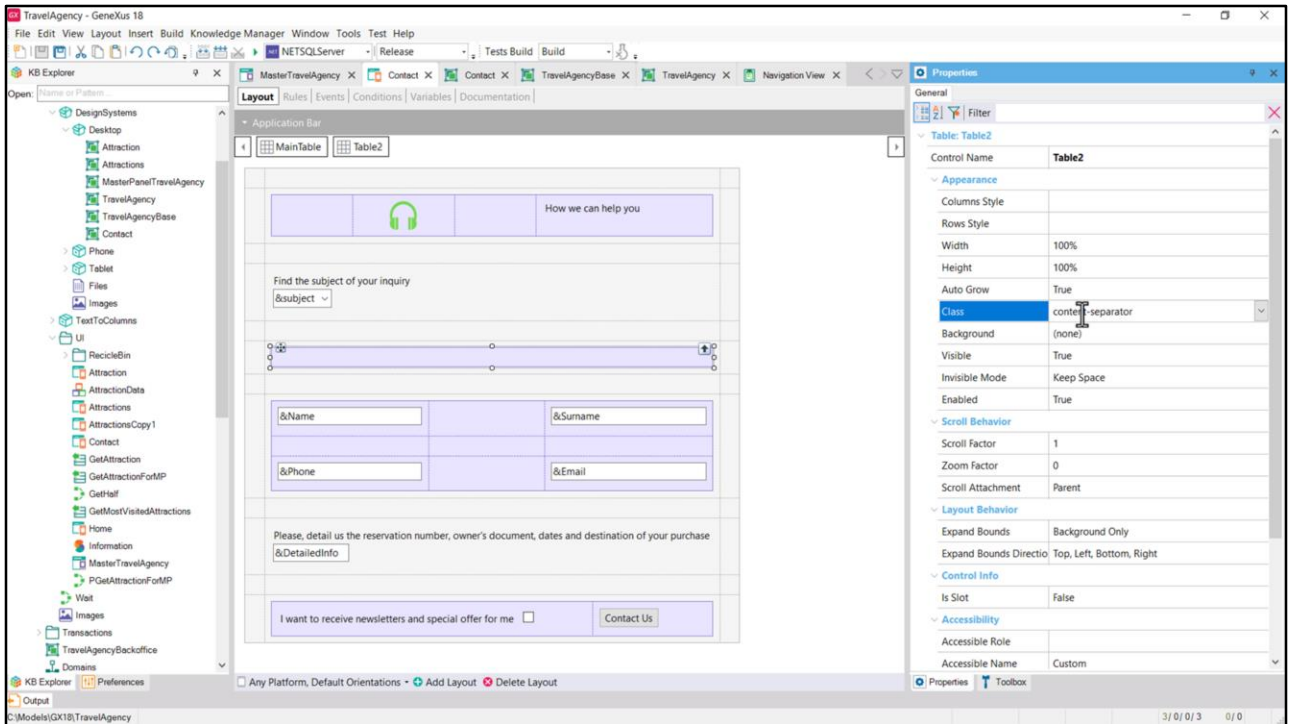
Con nuestra solución anterior (con las dos clases separadas para el control) no era necesario tenerla aquí.



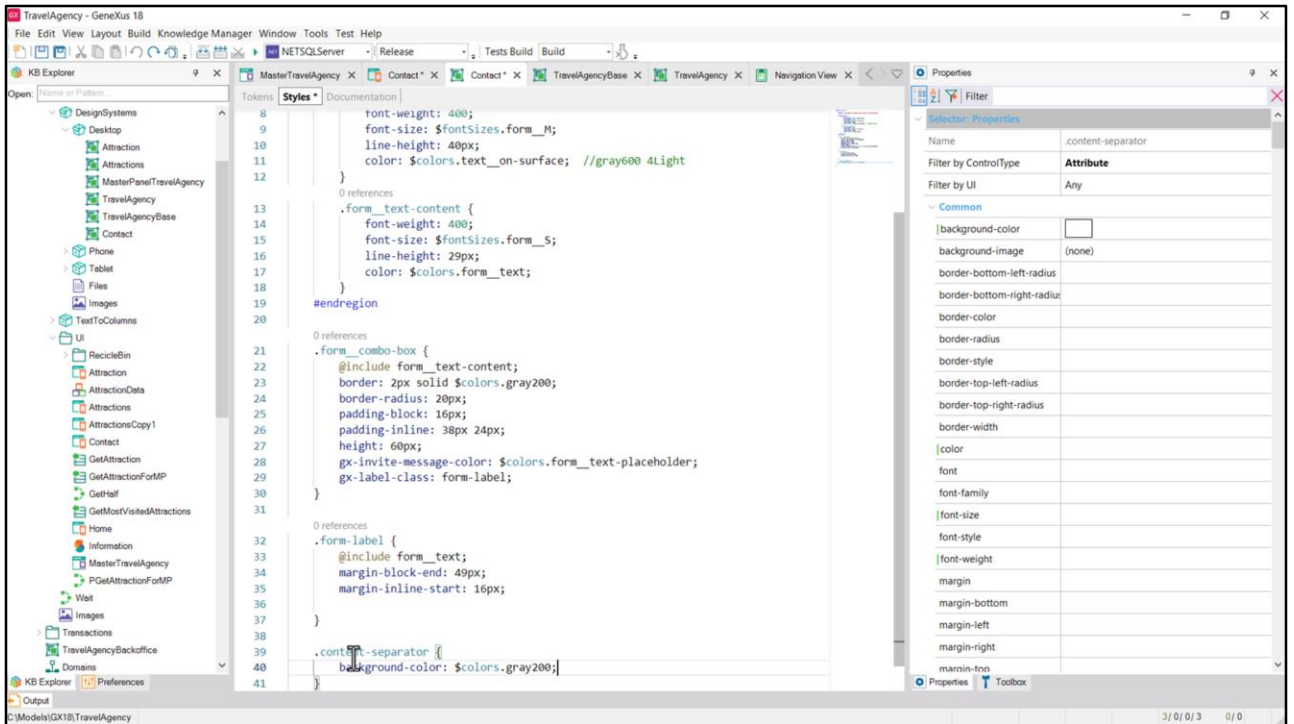
Si ahora ejecutamos... tenemos un problema, y es que no estamos viendo el Invite Message con el color grey200, el gris clarito, sino con el mismo que la clase tipográfica. Esto es porque por ahora solamente está valiendo la propiedad `gx-invite-message-color` para los campos Edit. Es decir, si nos funcionará para todos estos otros campos, a los que aún no les hemos personalizado el estilo, está todo lo default, pero es lo que nos queda por hacer.



Para implementar la línea de separación que vemos en Figma, de 2 píxeles y este color...

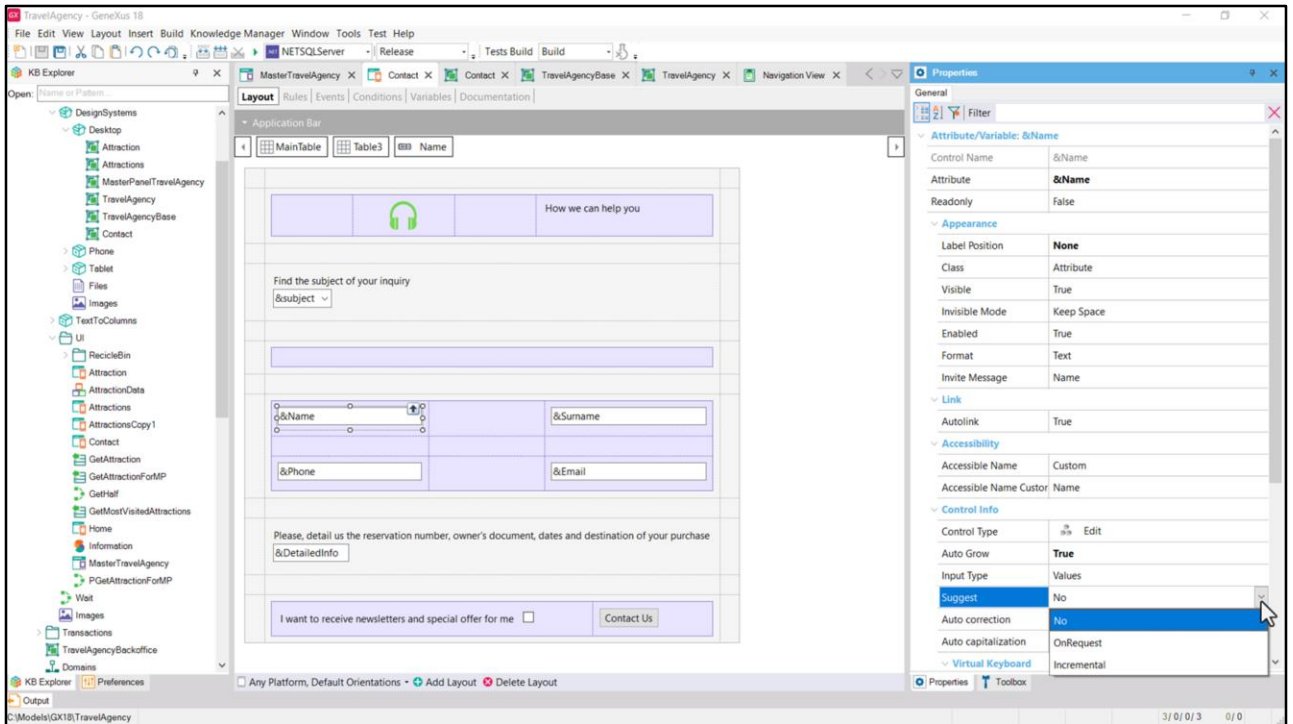


Es que coloqué esta tabla de 2 dips de alto, a la que voy a asignarle esta clase content-separator...



...donde al background-color le daré este valor.

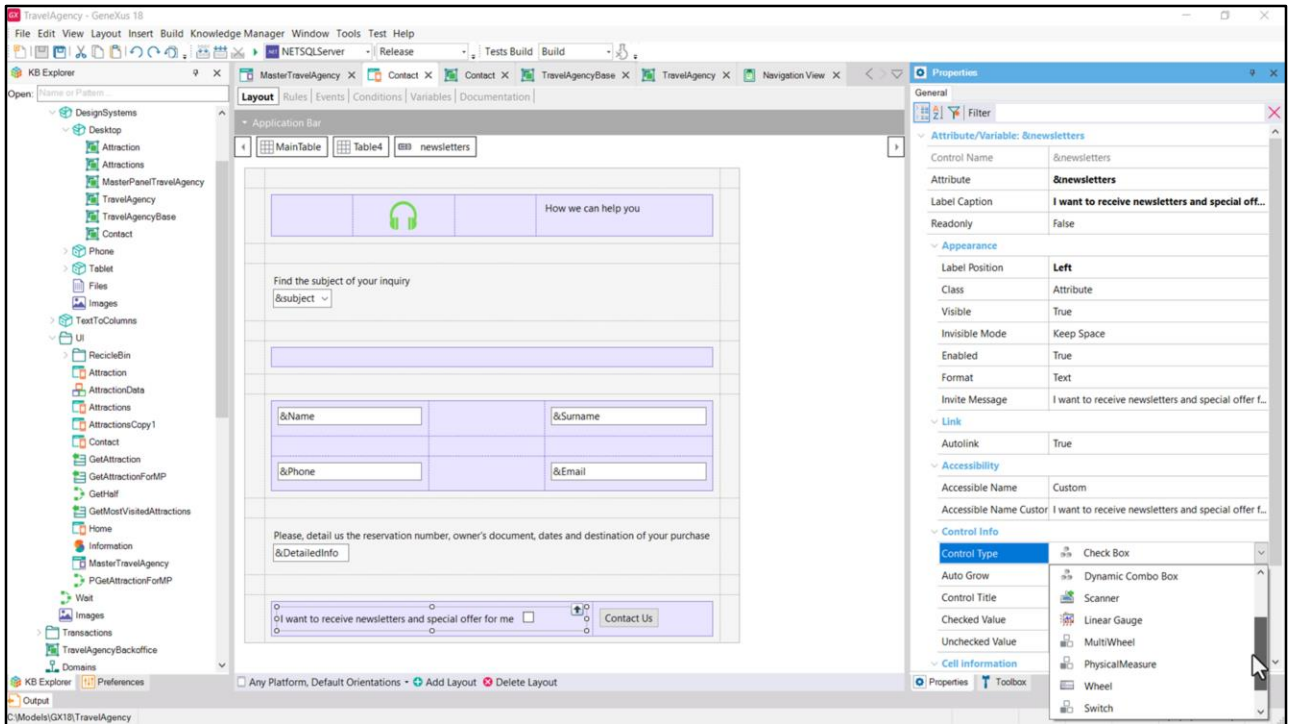




Ahora habría que seguir haciendo lo que hicimos para este campo combo box, para los demás...

Se los dejaré como tarea para ustedes.

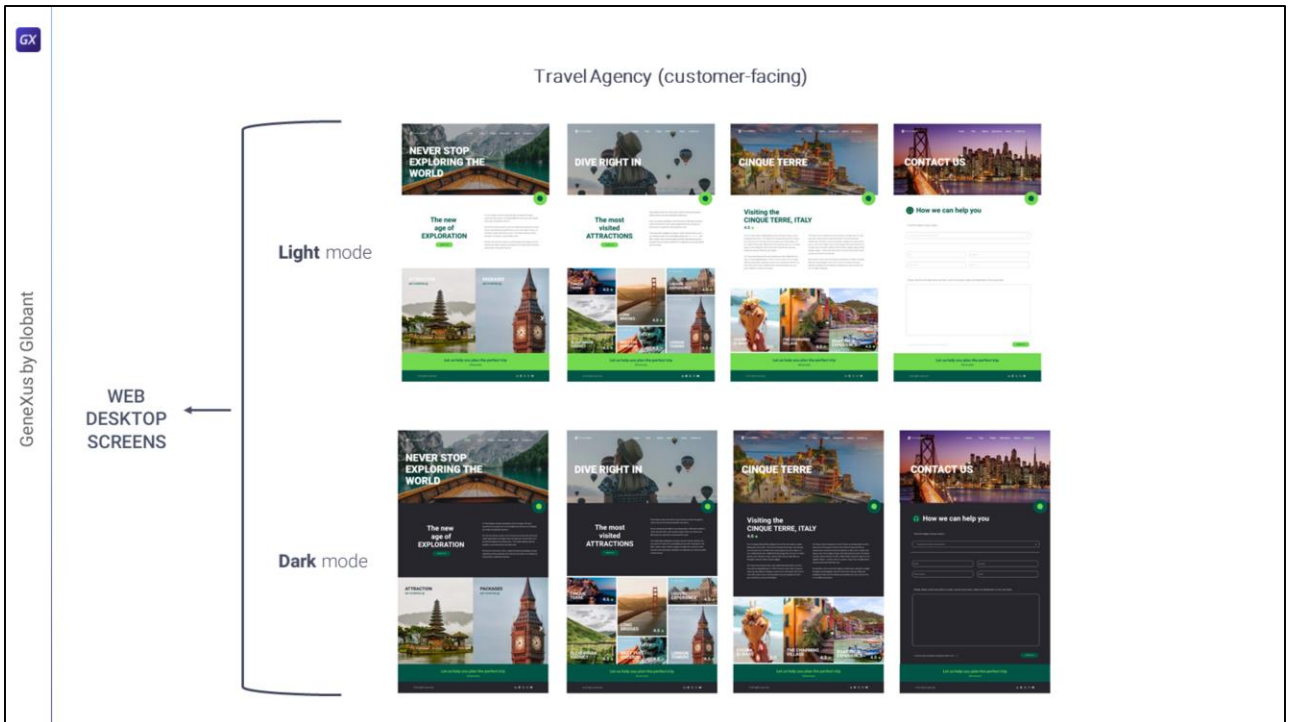
Veán que estos son campos de tipo Edit sin Label y con Invite Message... Para los controles de tipo Edit, se pueden definir una serie de cosas: por ejemplo la posibilidad de sugerir a medida que el usuario va digitando... busquen en el wiki de GeneXus e investiguen todas las opciones.



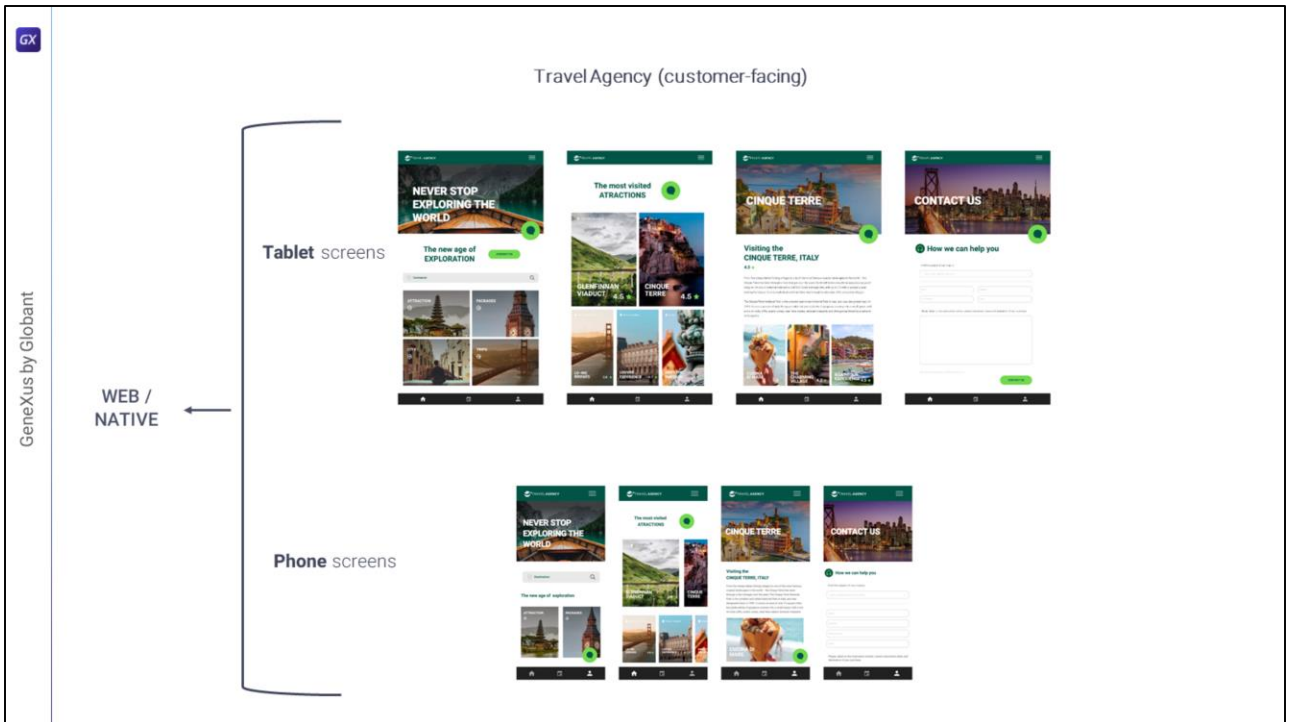
Esta otra variable también es Edit pero tiene label arriba, y no tiene Invite Message...

Y por último esta otra es de tipo Checkbox, con la etiqueta a la izquierda.

Observen que entre los tipos de control para las variables tenemos muchas otras alternativas, que tienen que ver con la User Interface, y que también les invito a investigar.



Con esto doy por cerrado lo más relevante que quería presentarles en lo que hace al desarrollo de la aplicación Angular para tamaño Desktop. Decíamos desde un principio que para tamaño Desktop solamente vamos a tener que implementar esta aplicación, la Angular.



Ya luego, cuando entremos en los demás tamaños de pantalla, tanto Tablet como Phone, ya se nos divide el mundo en dos paradigmas: el de Angular (Web) y el nativo. Es decir, vamos a tener que implementar allí las dos vertientes. Y en el mundo nativo la vertiente para Android y la vertiente para Apple.

Sobre ello vamos a hablar en el próximo módulo, a modo de introducción y de análisis general de cómo tendríamos que pensar, entonces, cómo nos convendría pensar las pantallas y la aplicación, en definitiva, para poder ser lo más eficientes posible y reutilizar lo máximo posible. Ya ahora no solamente entre las pantallas de la misma plataforma, sino también para poder compartir el mismo desarrollo y diseño para todas las plataformas, lo cual es todo un desafío. Pero sobre eso hablaremos en lo que sigue. Los espero.

GX

GeneXus by Globant

**GeneXus**<sup>™</sup>  
by Globant

[training.genexus.com](https://training.genexus.com)