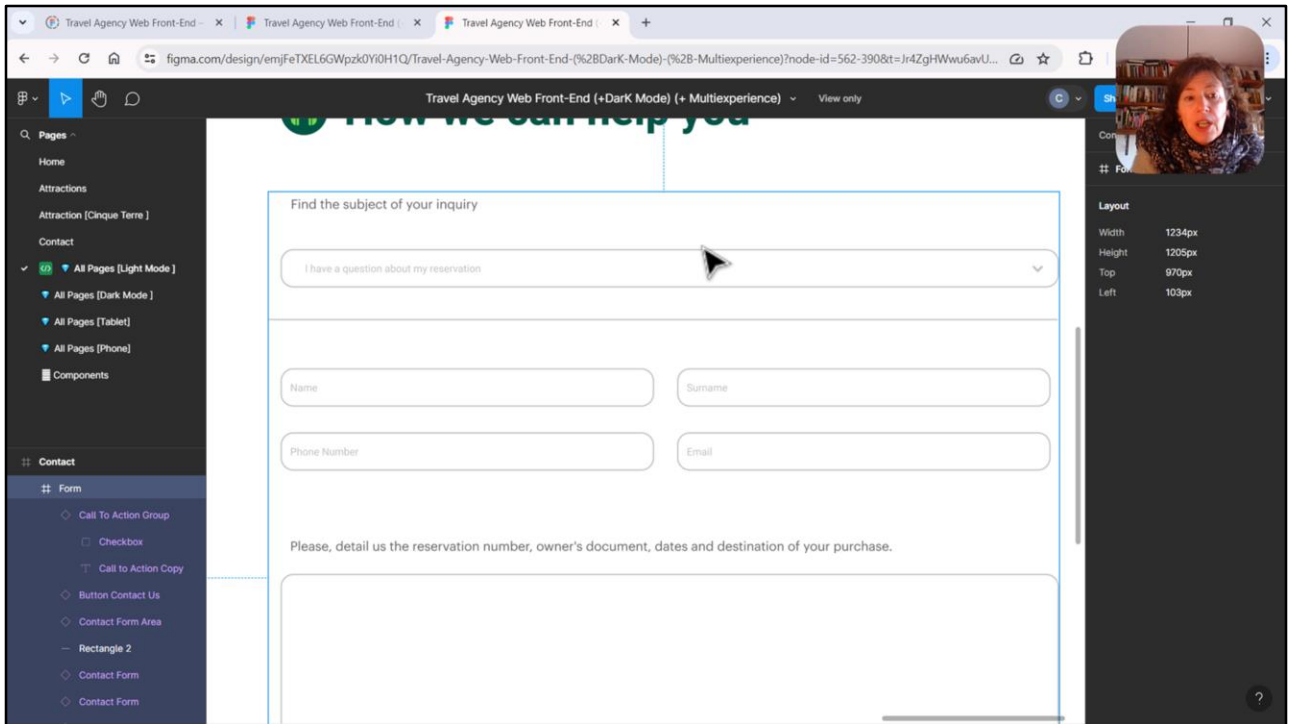


# Contact Panel: Input Fields

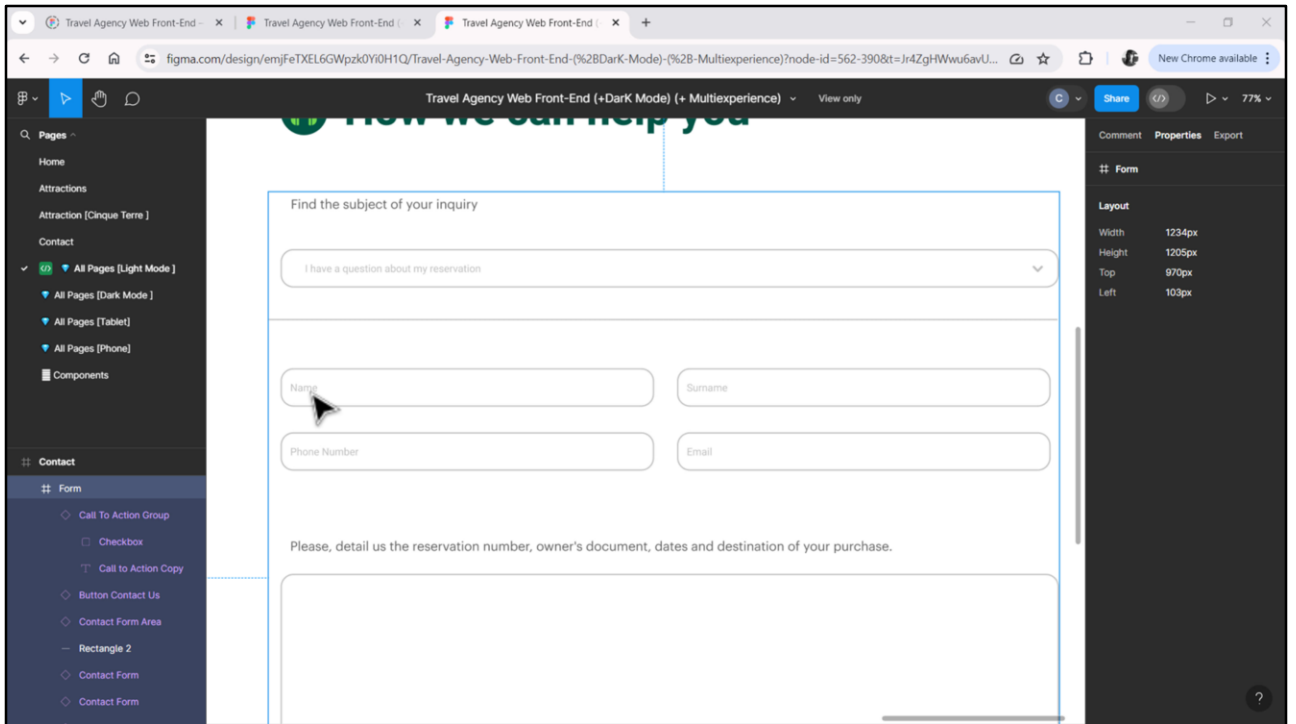


Cecilia Fernández



Para terminar de ver as questões mais relevantes no que diz respeito ao desenvolvimento da aplicação Angular para tamanho Desktop, ainda precisaríamos ver como implementamos os campos de entrada, que no nosso caso estarão apenas na tela Contact.

Lá podemos ver que há primeiro um combo box para que o usuário selecione entre uma série de opções o tema de sua consulta. A descrição que vemos fora do combo será o rótulo da variável. E vemos que dentro do combo aparece uma mensagem de sugestão. Quando o usuário clicar ali ou na seta, as opções serão expandidas para que selecione uma.

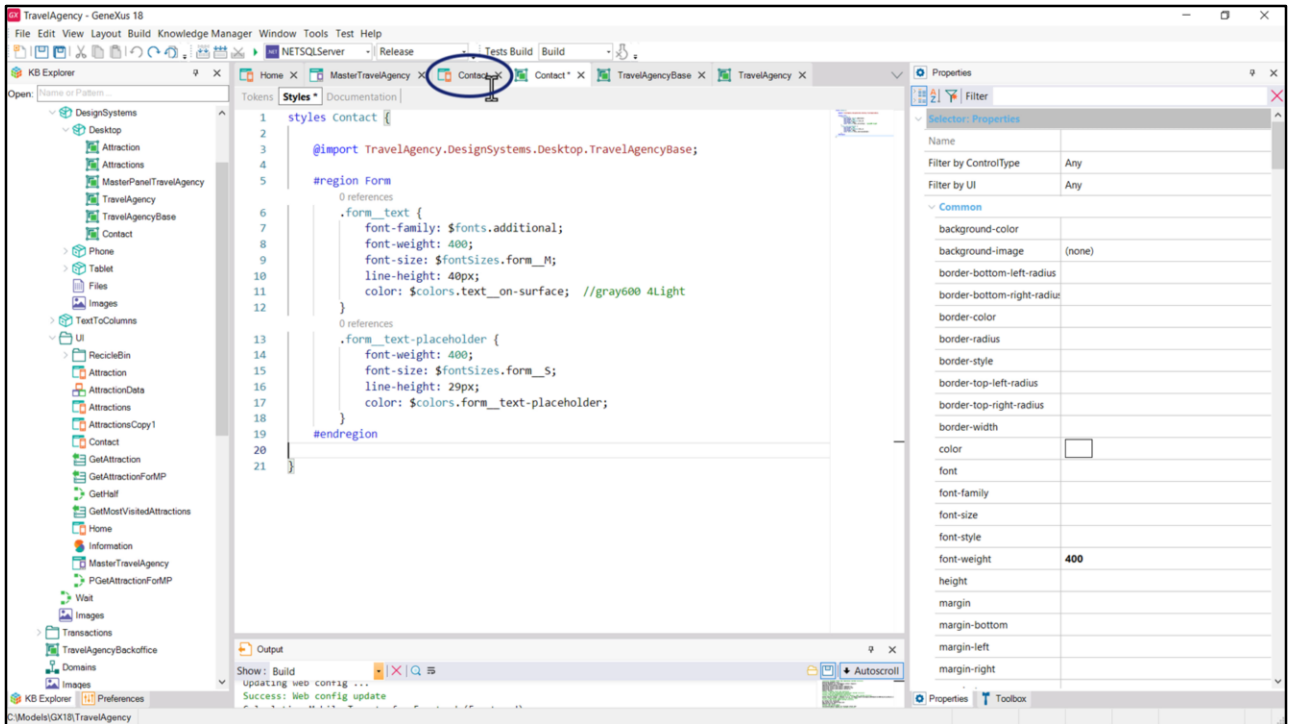


Depois, há quatro campos do tipo edit, onde o usuário poderá inserir dados pessoais. Esses campos não têm rótulo descritivo porque através da mensagem de convite para escrever que aparece dentro deles quando nenhum valor foi inserido, deixa claro do que se trata.

Quando o usuário começar a digitar nesses campos, a mensagem de convite desaparecerá. Reaparecerá somente se o usuário esvaziar o campo.

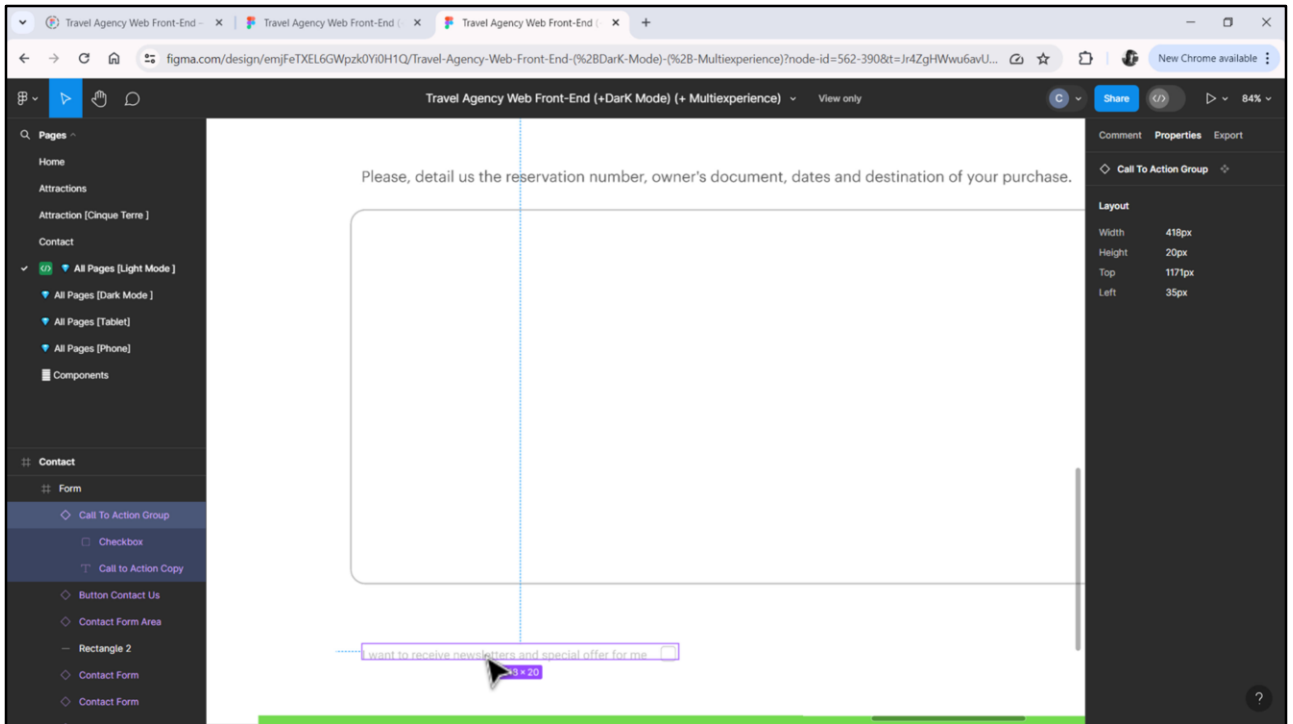
	A1	Name											
10	Banner / H1	Banner	banner_h1	additional	600	Graphik	Semibok	36	-	-		banner_XL	
11	Banner / H2		banner_h2	secondary	500	Rubik	Medium	20	-	-		banner_L	
12	Card Attraction / H1	Card-Attraction	.card-attractions_h1	primary	800	Heebo	ExtraBo	36	36	20		card-attractions-Big_XL	
13			.card-attractions-small_h1					36	20	12		card-attractions-Big_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	

Se nos lembrarmos da etapa de preparação, havíamos inserido duas classes tipográficas para esses textos: uma que chamamos de form\_\_text, para os textos dos rótulos, e outra que chamamos de form\_\_text-placeholder, para os textos internos, justamente esses de convite.



Vejam que criei um DSO Contact para dar estilo ao painel de mesmo nome e o adicionei ao DSO raiz, TravelAgency.

Vou copiar essas classes para nosso DSO. Se apenas terei campos de entrada neste painel, então posso deixá-las apenas aqui e excluí-las do DSO base.

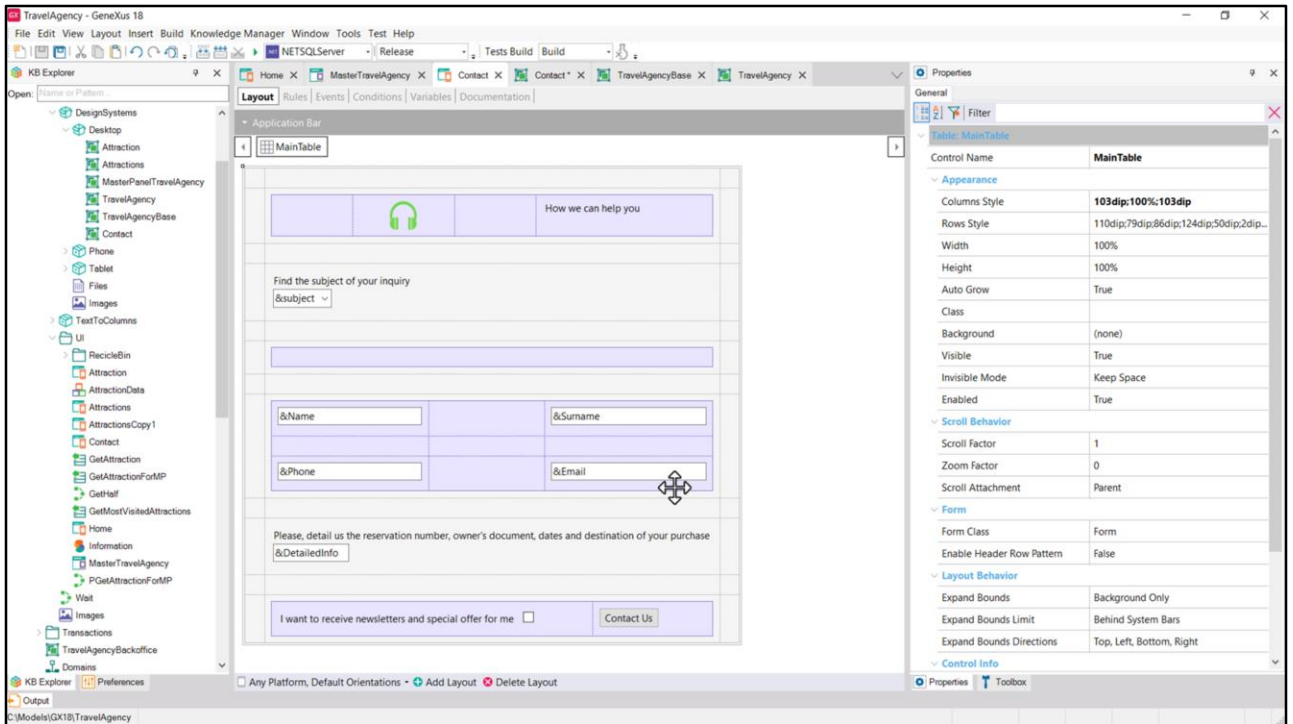


Bem, se continuarmos analisando a página, tenho outro campo Edit com rótulo acima, e abaixo de tudo eu tenho um campo de tipo Check box, com o rótulo à esquerda.

É importante implementar todos esses controles como variáveis com seus rótulos, e não como dois controles separados: um controle textblock para o rótulo e um controle variável propriamente dito para o campo. Por quê? Porque, primeiro, conceitualmente é o mais adequado, e segundo, e por consequência disso, porque terá efeitos para a acessibilidade: se trata-se do mesmo campo, já está entendida toda a semântica. Se são dois separados, terá que relacioná-los e mesmo assim não será alcançado o mesmo resultado.

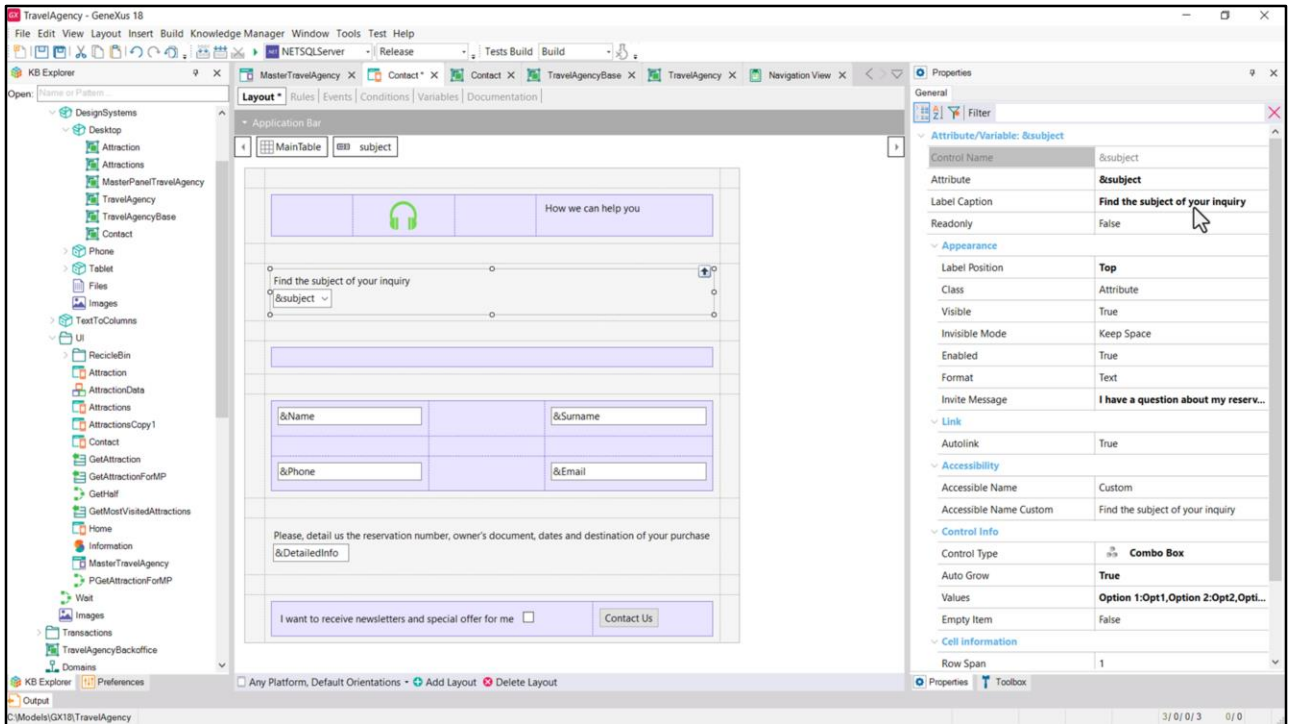
Então, o que eu quero mostrar neste vídeo é como modelamos a User Interface do controle, dado que será único, mas temos que modelar duas coisas: o campo em si e o rótulo. Mas se o campo é um só, então como fazemos isso?

Vou mostrar em detalhes o primeiro caso, e o restante vocês podem deduzir sozinhos. De qualquer forma, vou deixar um xpx com parte da solução para que vocês possam completá-la.



Primeiro, vamos ver que claramente todos os campos vão em uma tabela, para conseguir o alinhamento.

Aqui já criei as variáveis que serão de entrada e as inseri nas tabelas que vemos. Nada disso requer conhecimentos novos; é o que temos feito, então não vamos perder tempo vendo mais uma vez.



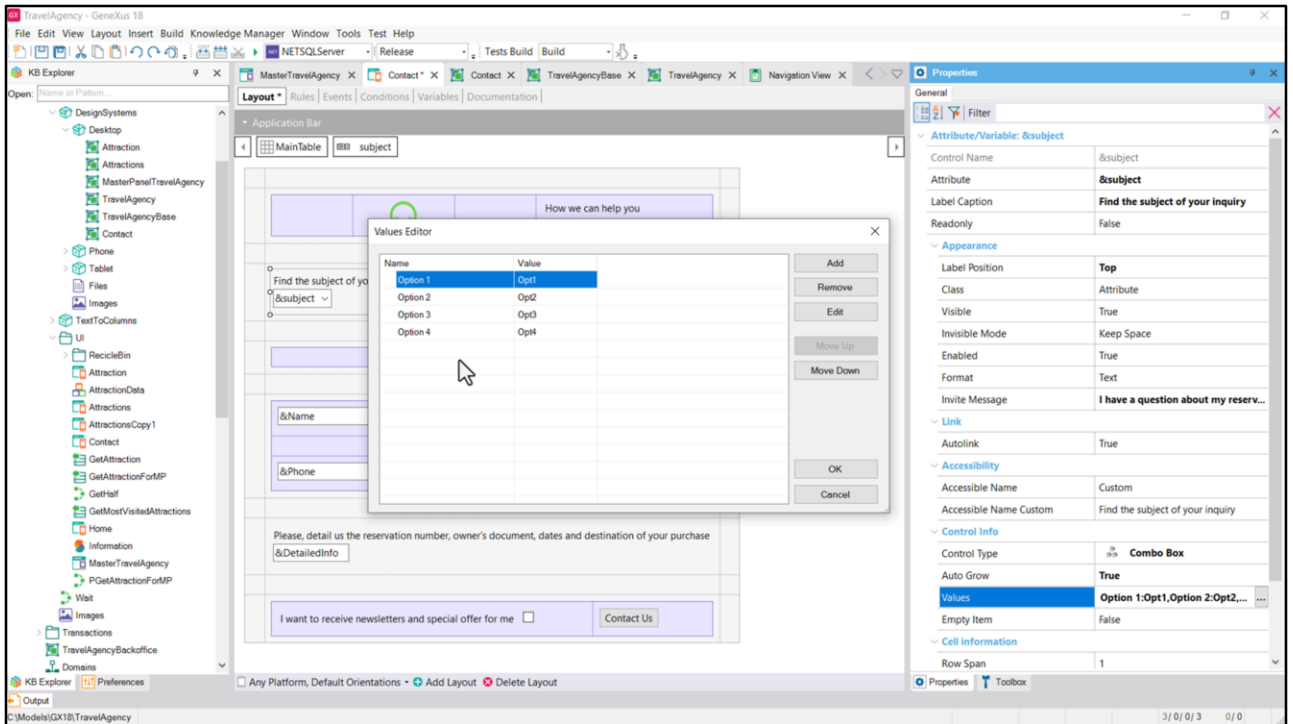
Vamos nos dedicar agora à primeira variável. Depois de inseri-la, veja que coloquei como Label Caption o texto que peguei da propriedade em Figma.

Por padrão, as variáveis nos painéis são de entrada, e é por isso que a propriedade ReadOnly está como False.

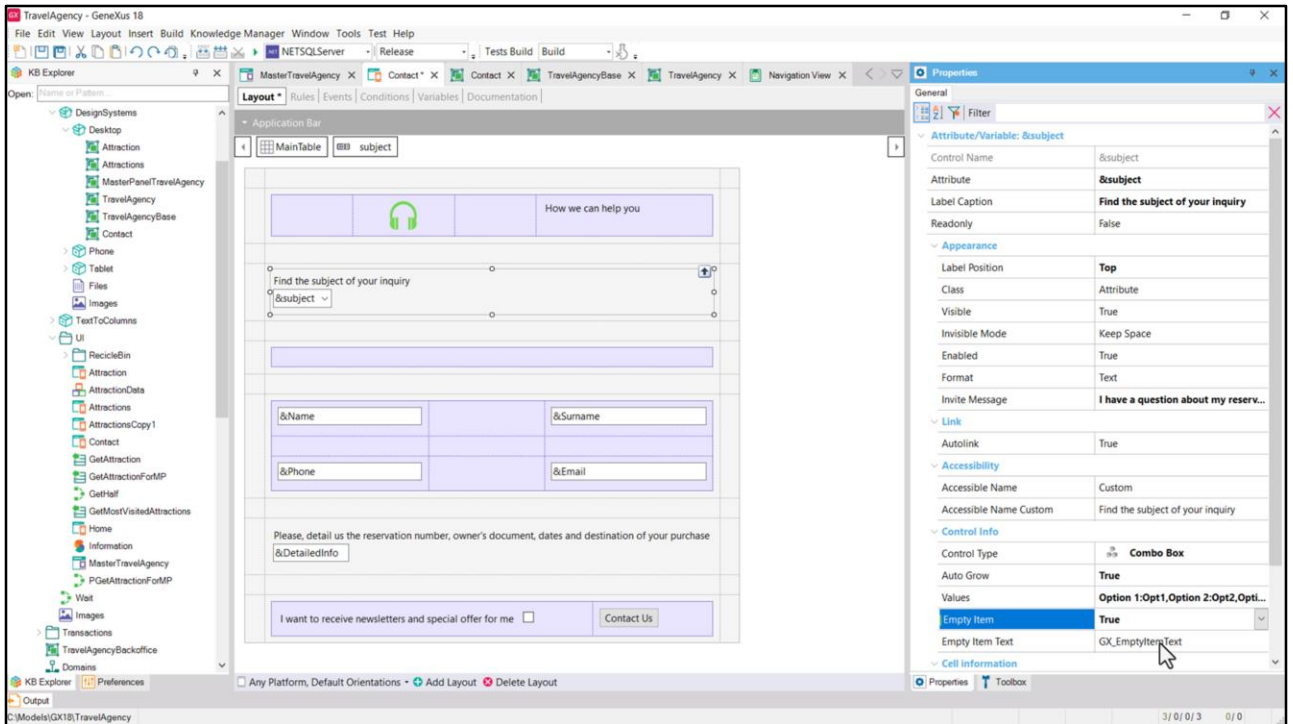
Observem que alterei o valor padrão da propriedade Label Position, para que coloque o rótulo acima do campo e não à esquerda, como é o valor default. Também defini a propriedade Invite Message para que este seja o texto que aparece dentro do campo quando estiver vazio.

E por último, veja que a maneira de fazer com que a variável em vez de ser do tipo Edit, que é o valor padrão (é o destas abaixo, por exemplo)... bem, em vez de ser, então, eu estava dizendo, do tipo edit, seja combo box, é através da propriedade Control Type.





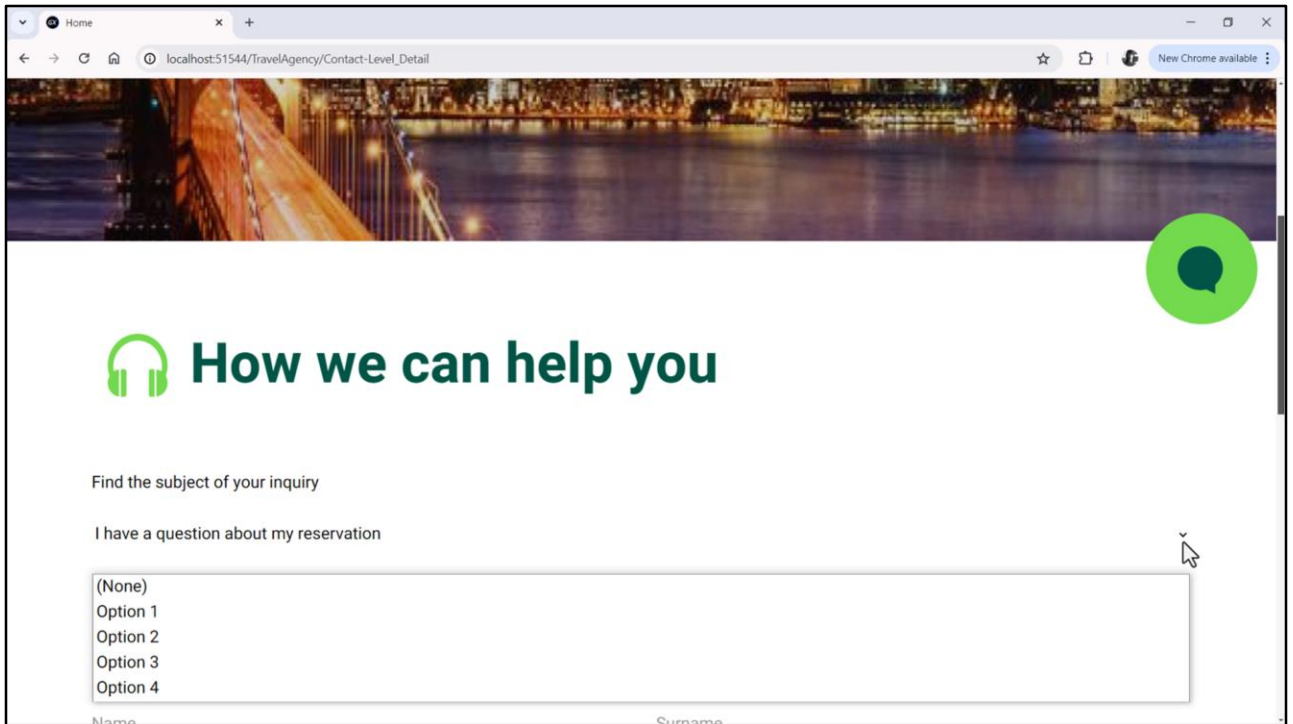
Quando esse valor é escolhido, Combo Box, aparece a propriedade Values para ser possível, justamente, fornecer as opções, ou seja, os nomes que o usuário verá na tela para cada opção, e o valor que ficará internamente armazenado na variável quando for selecionada a opção correspondente.



Poderíamos adicionar um item que corresponde ao valor vazio ou não selecionado, e através desta propriedade atribuímos a ele um texto, que por padrão é o que está codificado nesta constante, e que é "(none)" (veremos isso agora em execução). Poderíamos colocar aqui o texto que queremos para o valor vazio, caso o apresentado por padrão pelo GeneXus não seja adequado, que, como a variável é do tipo varchar, corresponderá ao valor string vazia. Poderíamos ter escolhido que a variável fosse de valor numérico, por exemplo, porque igual ao que será exibido na tela, são os nomes das opções, não os valores. O usuário nunca verá o valor real. Mas escolhi que fosse de tipo varchar, caso mais tarde eu queira que ele não seja mais um combo, mas sim um campo Edit.

Observem também que, por padrão, a propriedade Accessible Name Custom assumiu o valor da propriedade Label Caption.

Poderíamos executar agora, já que ainda não atribuí classes aos controles, para que possamos ver a diferença mais tarde.



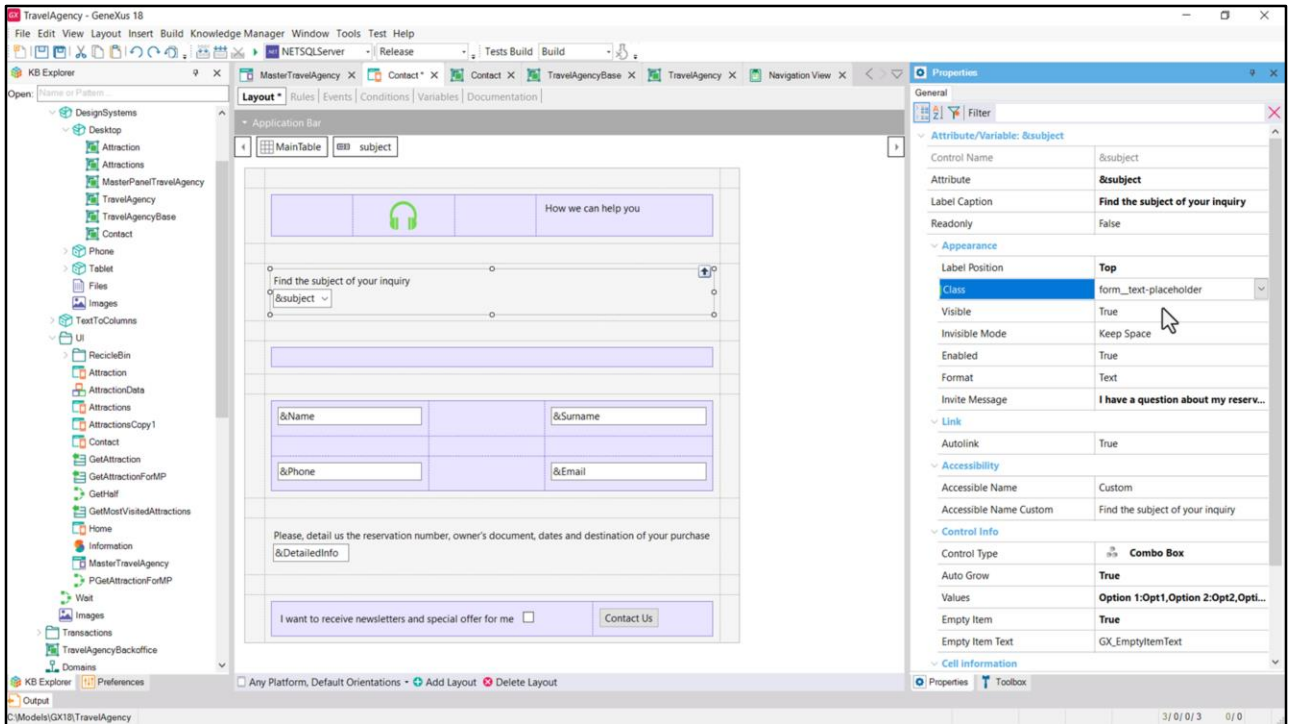
The screenshot shows a web browser window with the address bar displaying "localhost:51544/TravelAgency/Contact-Level\_Detail". The page features a header image of a city at night with a bridge over water. Below the image is a green circular chat icon. The main heading is "How we can help you" with a green headphones icon to its left. Underneath, there is a text prompt "Find the subject of your inquiry" followed by the selected option "I have a question about my reservation". A dropdown menu is open, showing the following options: "(None)", "Option 1", "Option 2", "Option 3", and "Option 4". At the bottom of the page, the labels "Name" and "Surname" are partially visible.

Aqui vemos o rótulo e vemos a mensagem de convite e também vemos os nomes das opções, e esse "None" que eu estava falando.

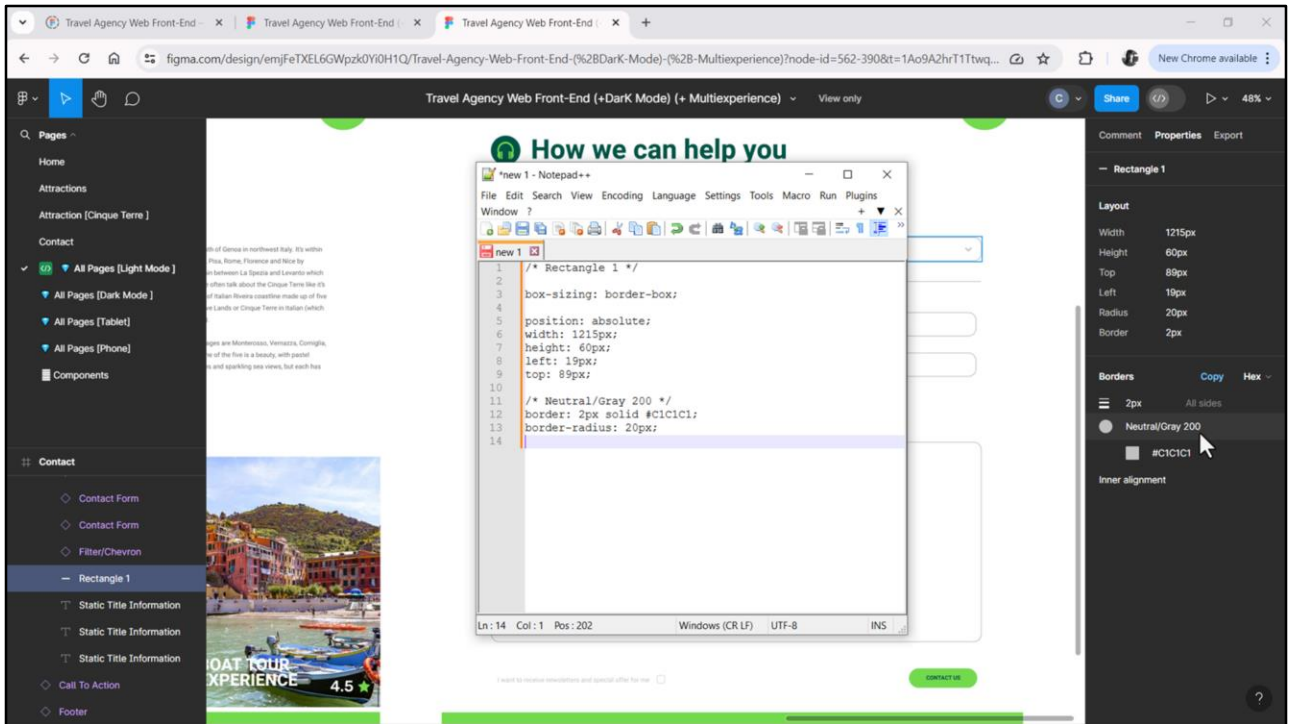
The screenshot shows a web browser window displaying a contact level detail page for 'Tokens Travel Agency'. The page features a green header with a headphones icon and the text 'How we can help you'. Below this, there are two lines of text: 'Find the subject of your inquiry' and 'I have a question about my reservation'. A dropdown menu is open, showing options: '(None)', 'Option 1', 'Option 2', 'Option 3', and 'Option 4'. A spreadsheet is overlaid on the page, showing a list of text styles and their corresponding class names. The spreadsheet has columns for ID, Component, Class Name, Style Name, and Weight. The class name 'form\_\_text-placeholder' is highlighted in blue, and a mouse cursor is pointing at it.

ID	Component	Class Name	Style Name	Weight
5	Button	button	primary	800
6	Menu Label	.menu_label	primary	500
7	Copyright	copyright	secondary	400
8	Card Home / H1	card-home_h1	primary	800
9	Card Home / H2	card-home_h2	secondary	500
10	Banner / H1	banner_h1	additional	600
11	Banner / H2	banner_h2	secondary	500
12	Card Attraction / H1	card-attractions_h1	primary	800
13		card-attractions-small_h1		
14		card-attractions_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-attractions_rating		
20	Form / Regular Text	form_text	additional	400
21	Form / Place Holder	form__text-placeholder	primary	400
22				

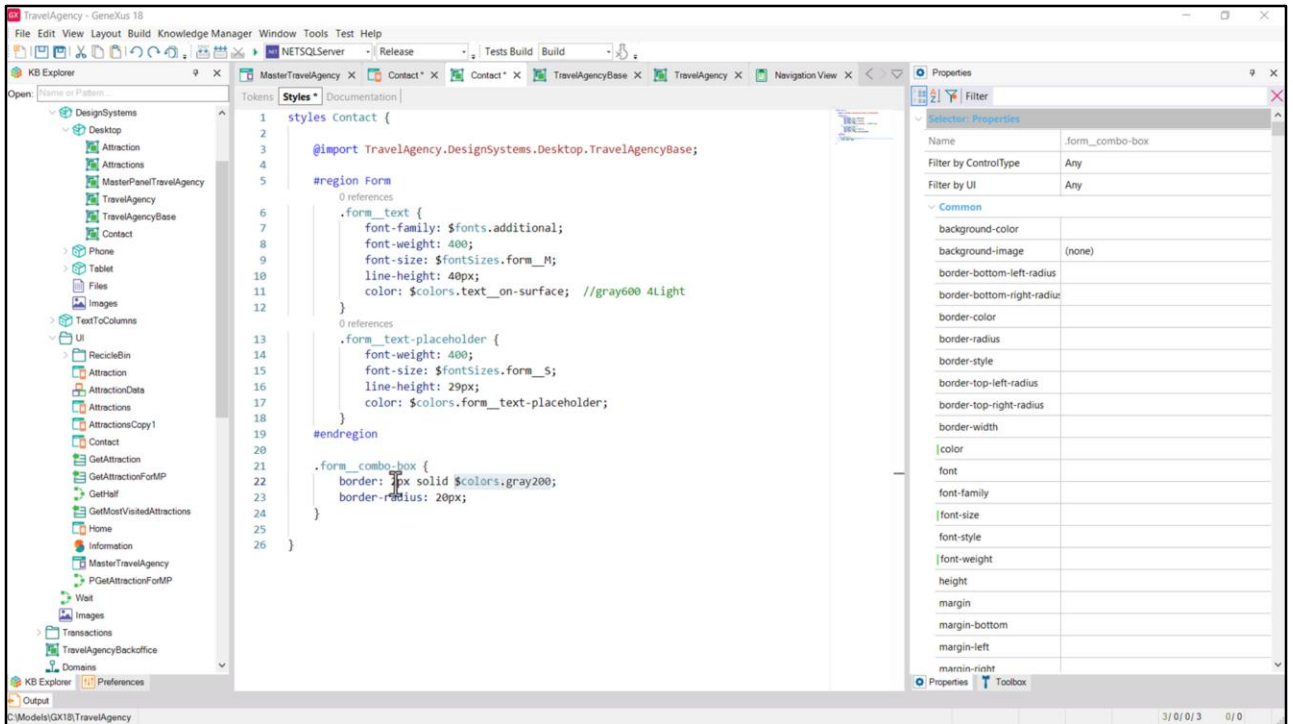
Sabemos que a classe para a tipografia do rótulo era a que chamamos de form\_\_text, e a do campo em si, que será aquela das opções do combo, a que chamamos de form\_\_text-placeholder. Depois veremos a da Invite Message.



Então, ao nosso controle, associaremos esta segunda classe... mas esta é apenas a tipográfica. Não a do controle combo em si. E ainda não fizemos nada em relação ao rótulo.

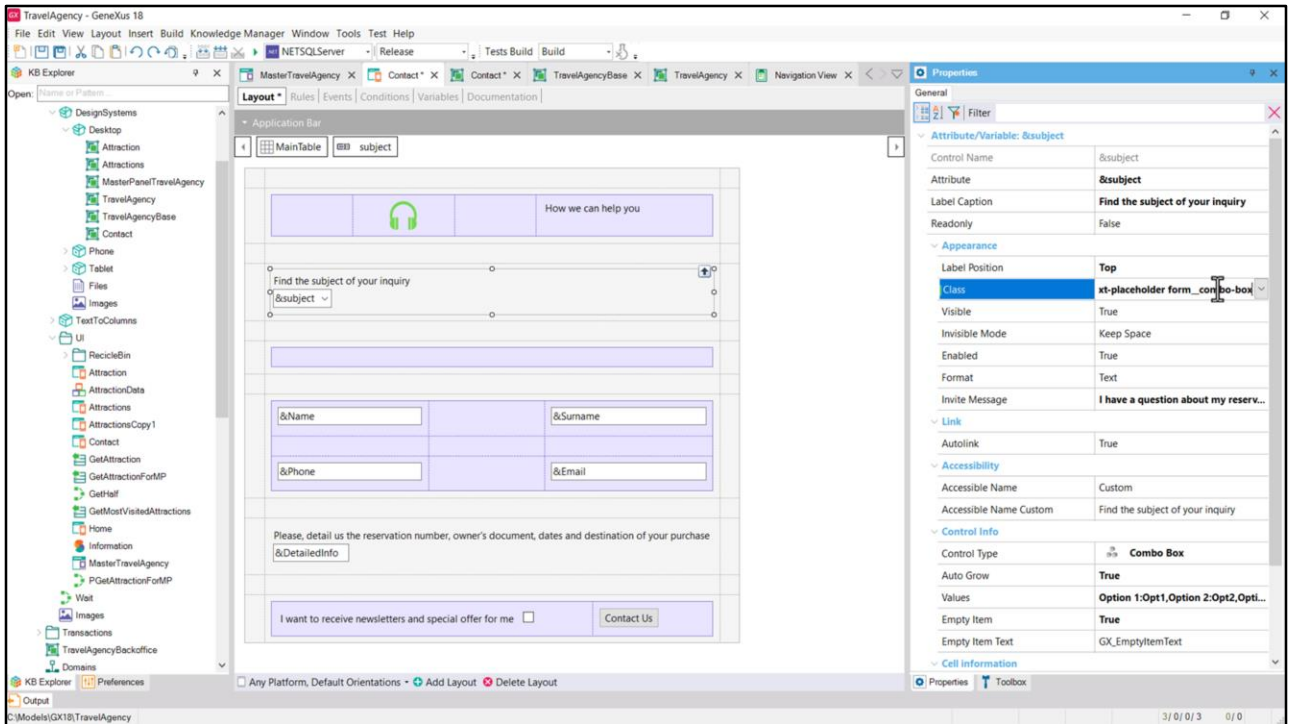


No Figma vemos essas propriedades, radius e border... se pedirmos o código CSS... teremos que levar para o GeneXus essas duas propriedades... essa é a cor de nosso token gray200...



Então vou criar uma classe para modelar a parte do combo box propriamente dita.

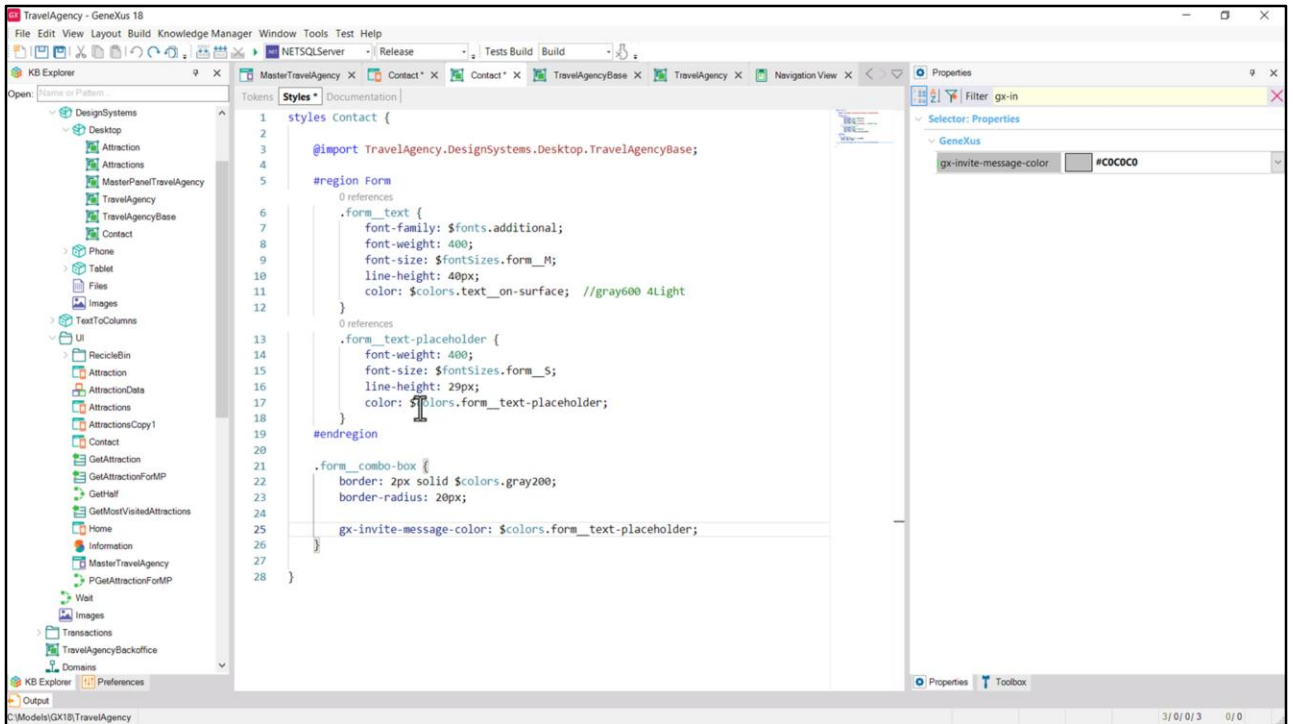
Vou chamá-la de form\_\_combo-box... e aqui copio as duas propriedades e substituo aqui pelo token de cor.



É claro, a atribuo ao controle combo box. Então esse controle terá duas classes: a da tipografia e a que dá estilo ao combo box propriamente dito.

Poderíamos incluir as duas em uma, ou mantê-las separadas, se por exemplo fôssemos utilizar apenas a tipografia em outro controle.

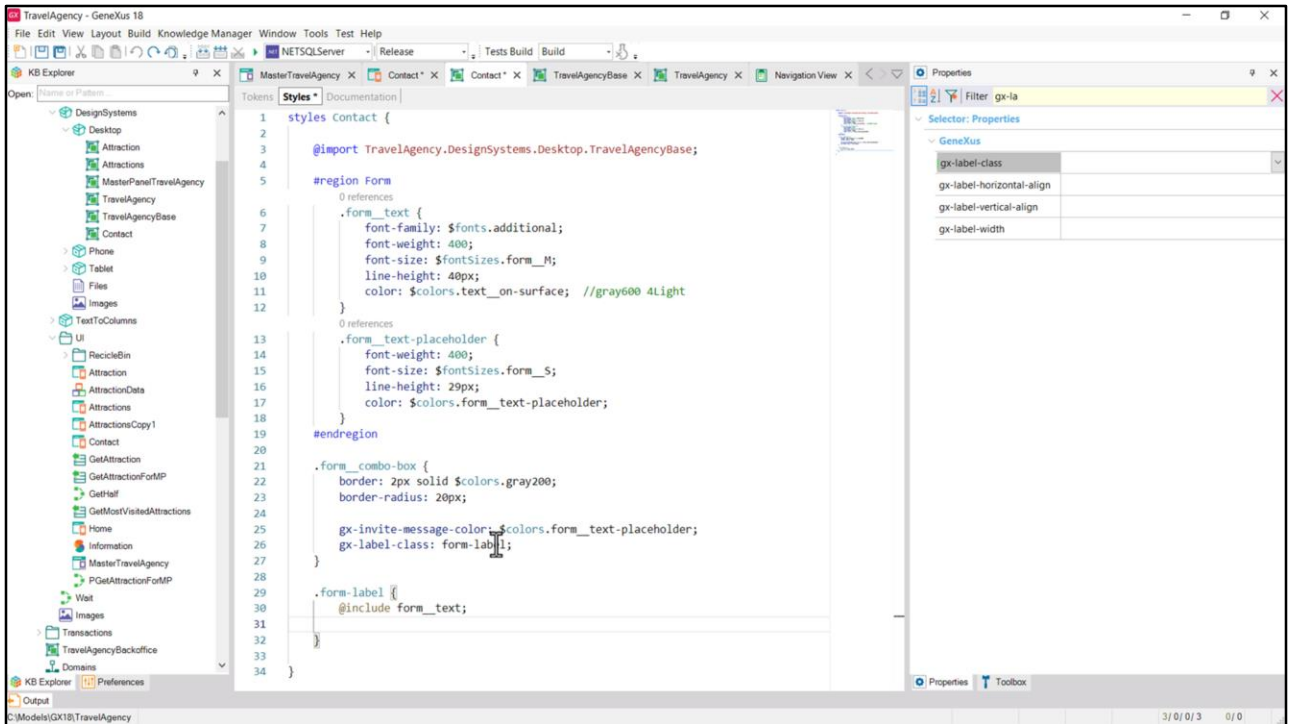




Para indicar a cor da Invite Message, temos uma propriedade GeneXus... vamos procurá-la filtrando as propriedades para controles do tipo Attribute (que é o mesmo tipo das variáveis).

Vamos filtrar melhor por "gx-in" ... e ali a vemos. Se escolhermos uma cor daqui, já a adicionará.

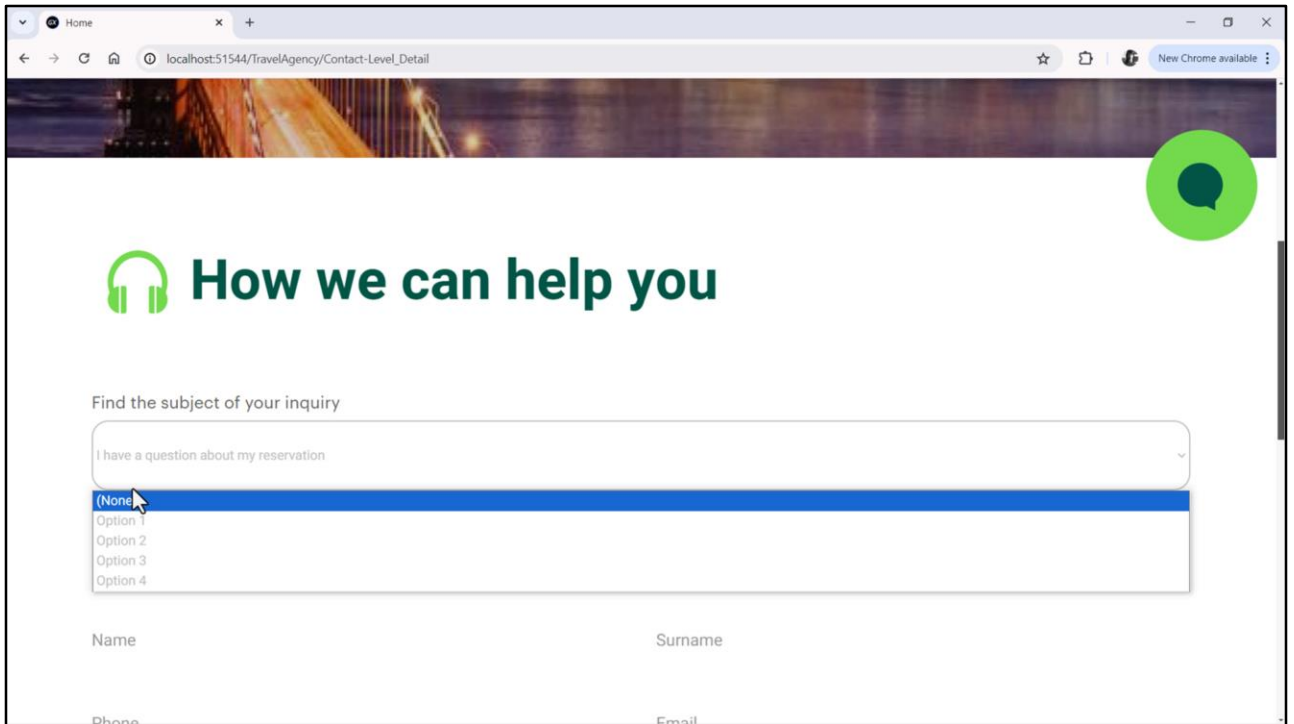
Vamos colocar o token de cor que utilizamos para a classe tipográfica do controle.



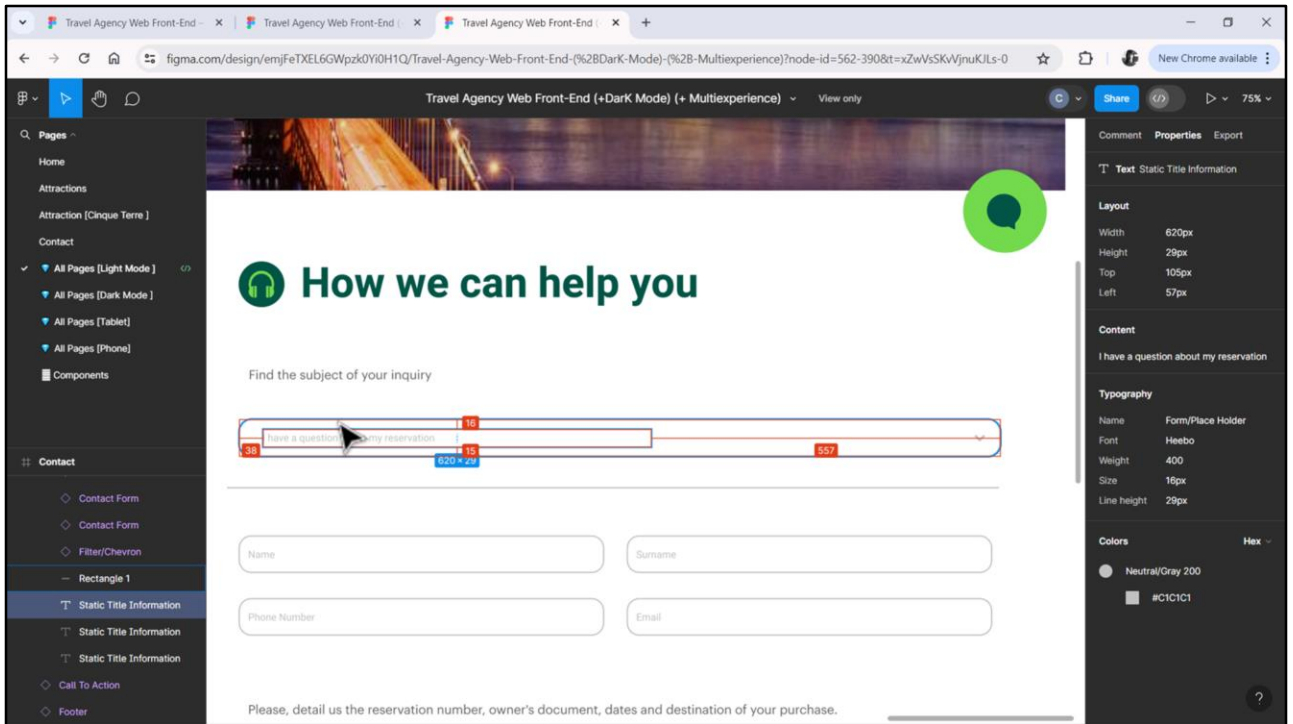
E agora sim, vamos atribuir a classe ao rótulo... será `gx-label-class`. O que farei será atribuir a ele uma nova classe, que chamarei de `form-label`.

Por enquanto, vou apenas defini-la incluindo a classe tipográfica, mas veremos que precisaremos adicionar mais propriedades a ela.

Vamos executar para ver o que fizemos até aqui.

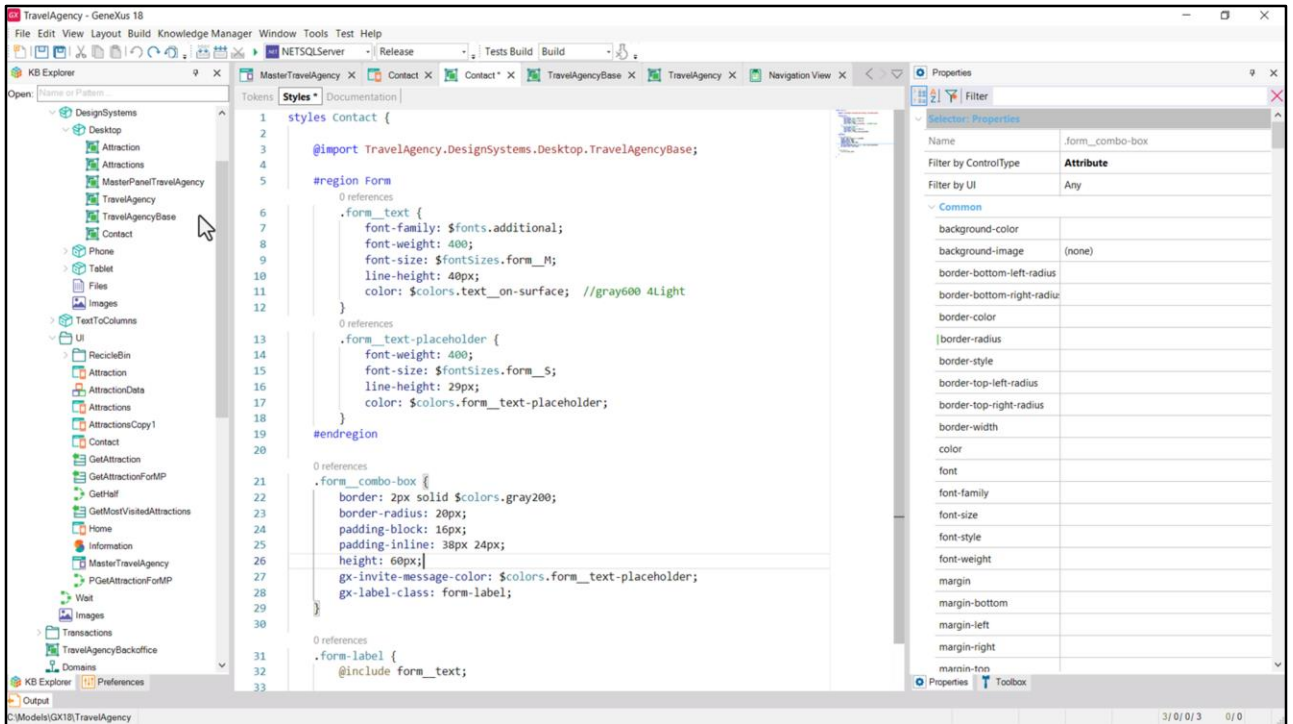


Começa a tomar forma. Mas observemos o texto do combo. Está colado na borda esquerda... e a altura do campo também parece maior que a do Figma.



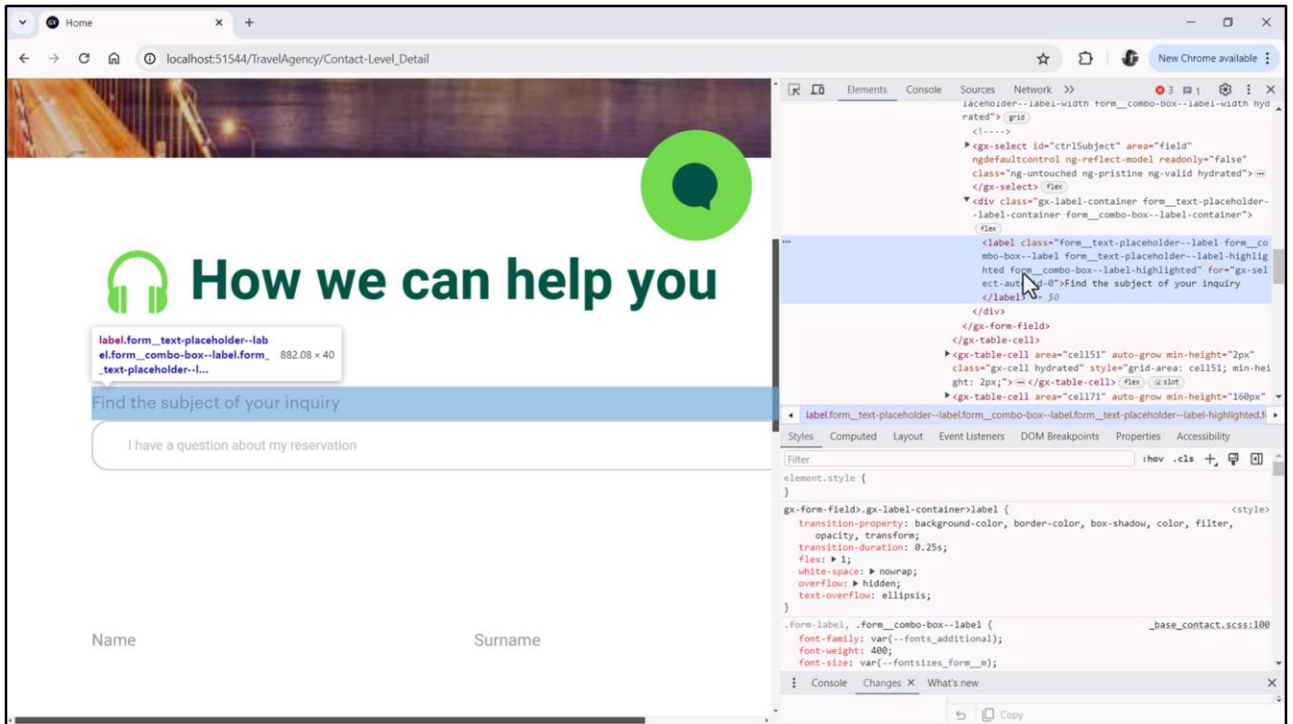
Se inspecionarmos... vemos que tem 84 pixels de altura... mas se formos ao Figma, a altura deveria ser de 60.

Além disso, vemos que o texto deveria ser de 29, com um padding de cima de 16, e de baixo de 15. Da esquerda de 38... e da direita... de 25.



Então, à classe, deveríamos adicionar estas propriedades:

- padding-block, ou seja, o padding na direção vertical, de 16 pixels (o padding de baixo era de 15, mas vamos deixar os dois iguais, de 16).
- E padding-inline, ou seja, na direção horizontal, de 38 pixels do início e 24 do final.
- E vamos especificar a height de 60 pixels.



Certo, agora, o que acontece com o rótulo?

Se eu o inspecionar, vejo que tem 40 de altura, que veremos ser o valor de line-height da tipografia, sem margens ou paddings.

The image shows a web browser window displaying a contact form on a travel agency website. The page title is "How we can help you" and the form asks for the "subject of your inquiry". The form contains a text input field with the placeholder text "I have a question about my reservation". Below the form are input fields for "Name" and "Surname".

The browser's developer tools are open on the right side, showing the DOM tree and the style pane. The selected element is a `label` with the following style properties:

```
margin-block-end: 49px;
```

The style pane also shows a visual representation of the margin and padding for the selected element, with a value of `margin-block-end: 49px` and a height of `40px`.

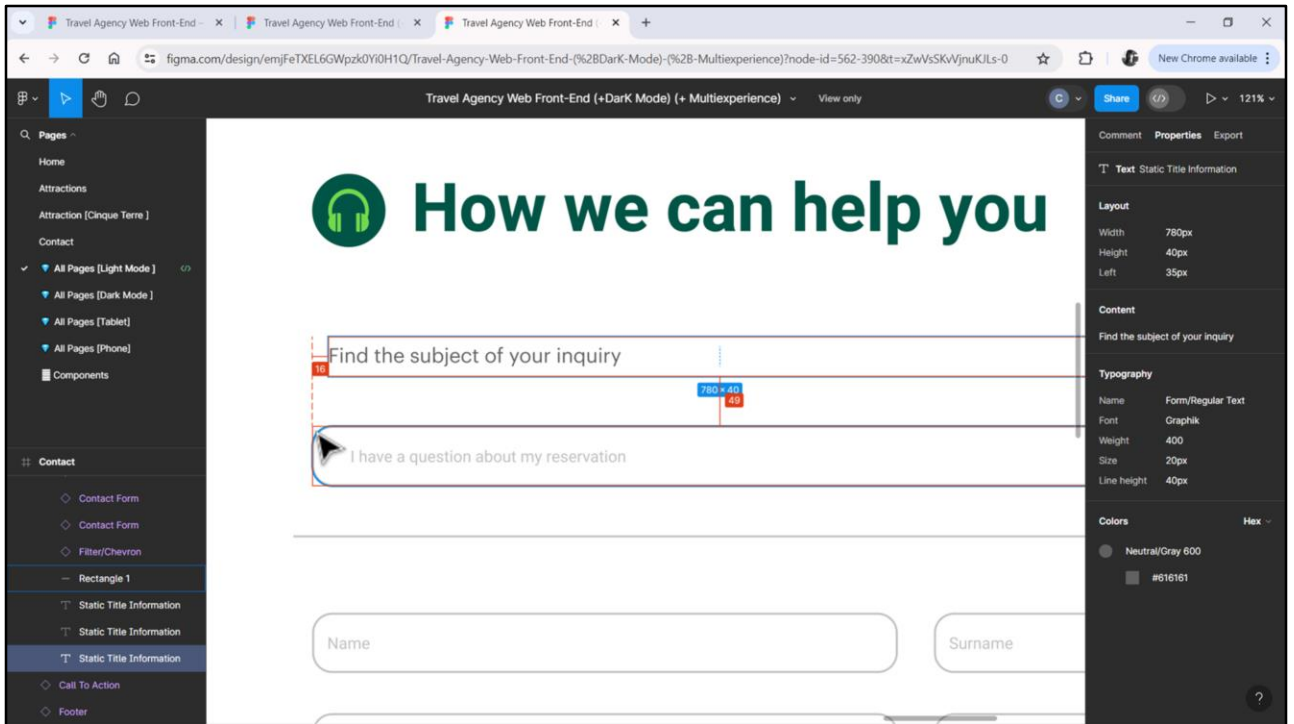
Vejam o que acontece se, por exemplo, coloco para este elemento (apenas para testar rápido) um `margin-block-end` de 49. Ali vemos os 49 da margem e os 40 de altura...

The image shows a web browser window with a contact form on the left and its developer tools on the right. The browser's address bar shows the URL `localhost:51544/TravelAgency/Contact-Level_Detail`. The page features a green speech bubble icon and the heading "How we can help you". Below the heading is the text "Find the subject of your inquiry" and a text input field containing "I have a question about my reservation". At the bottom of the form are labels for "Name" and "Surname". The developer tools on the right show the DOM tree with the selected element being a label. The "Styles" pane shows the following CSS rules:

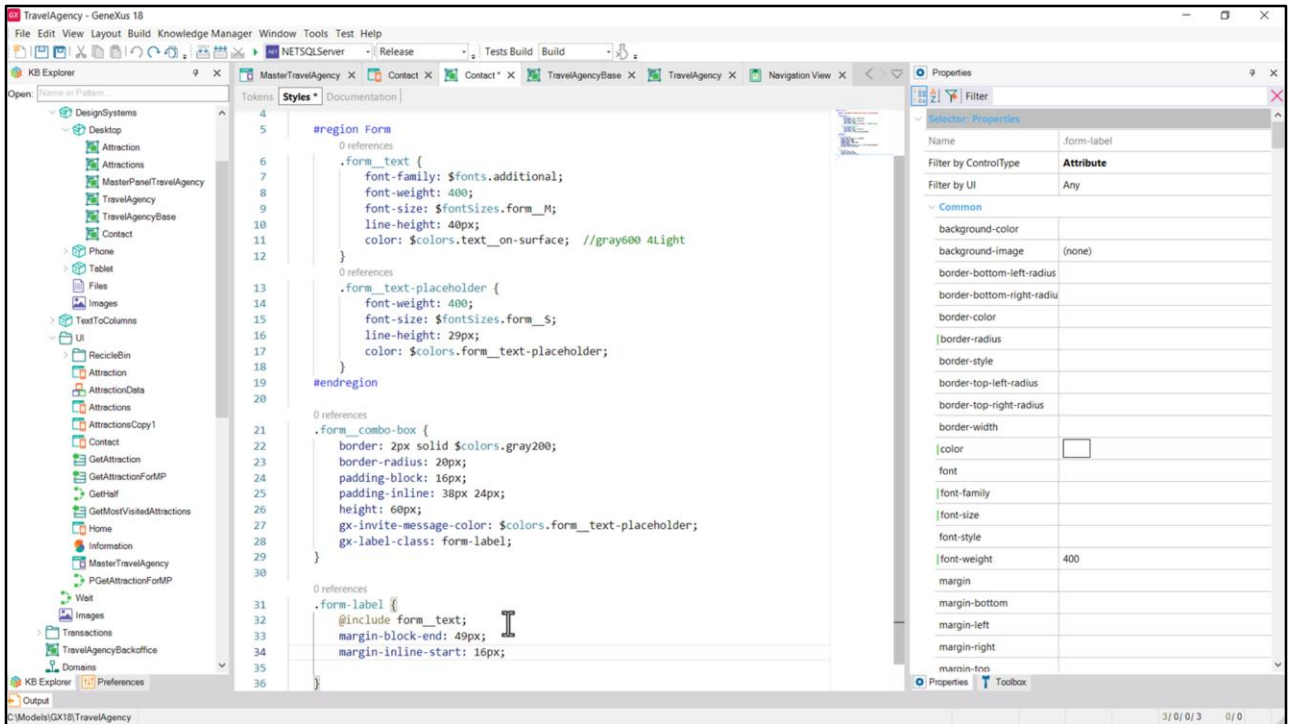
```
element.style {  
  margin-block-end: 49px;  
  margin-inline-start: 16px;  
}  
gx-form-field>gx-label-container>label {  
  transition-property: background-color, border-color, box-shadow, color, filter,  
  opacity, transform;  
  transition-duration: 0.25s;  
  flex: 1;  
  white-space: nowrap;  
  overflow: hidden;  
  text-overflow: ellipsis;  
}  
.form-label, .form__combo-box--label {  
  font-family: var(--fonts_additional);  
}
```

E se adicionarmos um margin inline start de 16...





Corresponde exatamente ao que vemos no Figma.





Então, à classe do rótulo adicionamos as duas propriedades.

Home x +

localhost:51544/TravelAgency/Contact-Level\_Detail

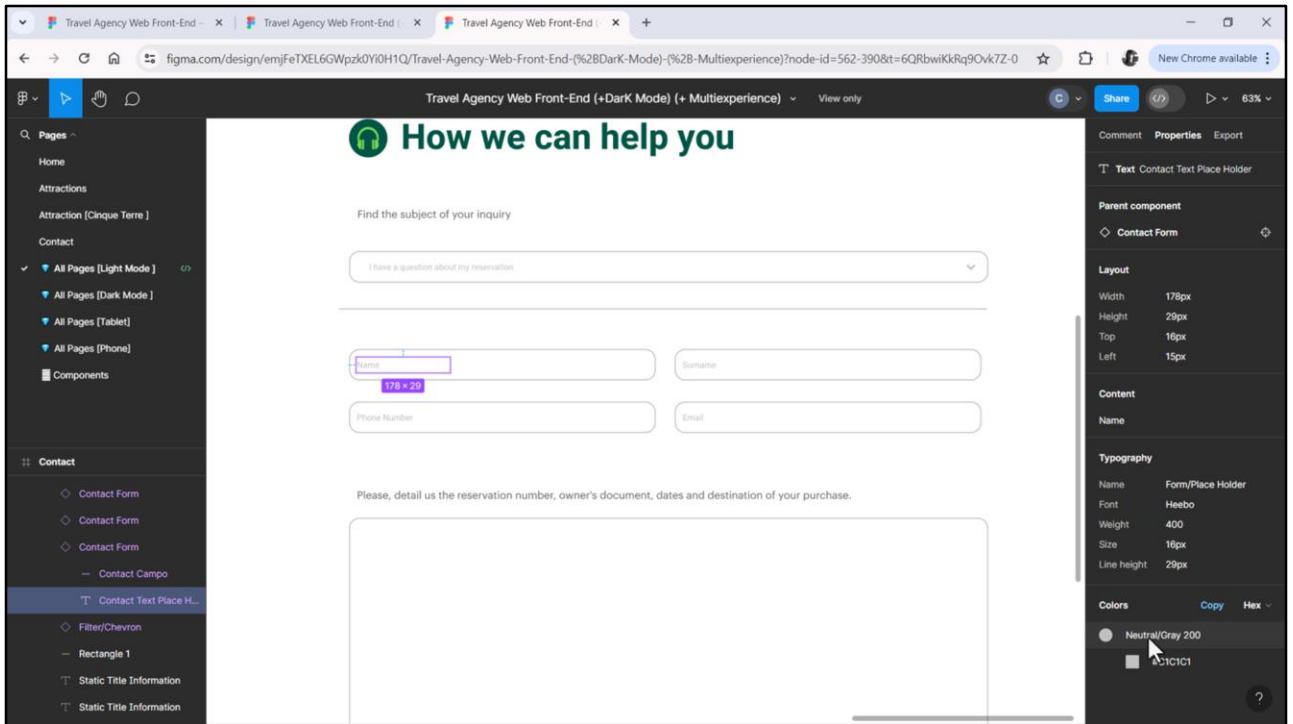
New Chrome available



# How we can help you

Find the subject of your inquiry

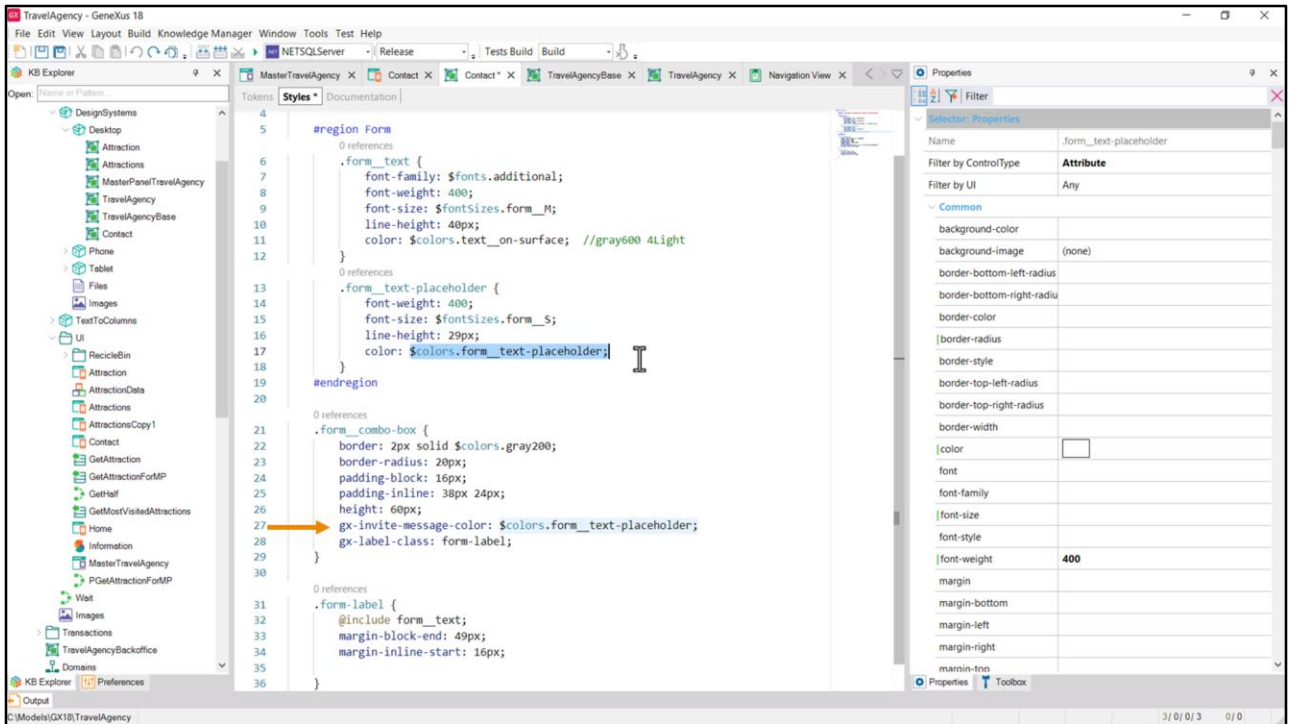
I have a question about my reservation



Não fui muito preciso antes: na verdade, no design de Chechu não vemos em lugar nenhum a tipografia correspondente aos valores que o usuário irá inserir nos campos.

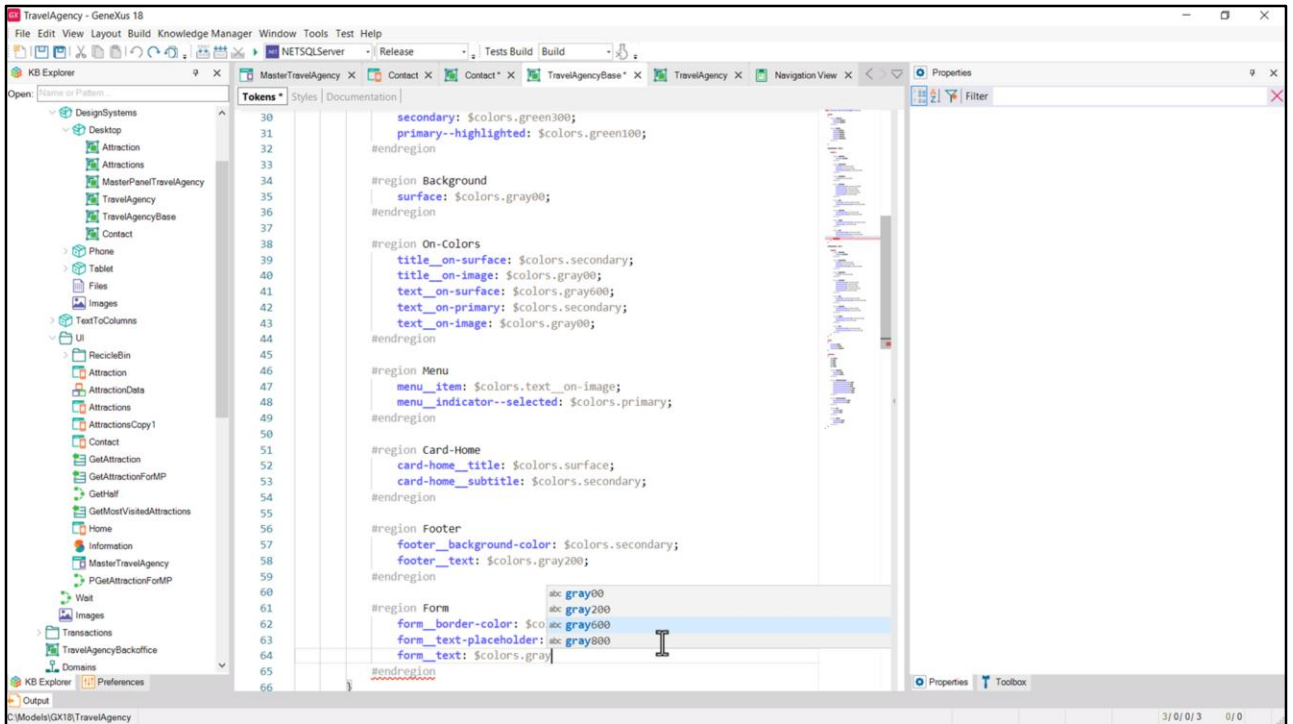
Nossa classe `form__text-placeholder` na verdade corresponde às mensagens de convite.

Em geral, a mensagem de convite tem a mesma tipografia do conteúdo do campo, exceto pela **cor**. Em geral, a mensagem do convite é muito mais clara, normalmente um cinza claro.

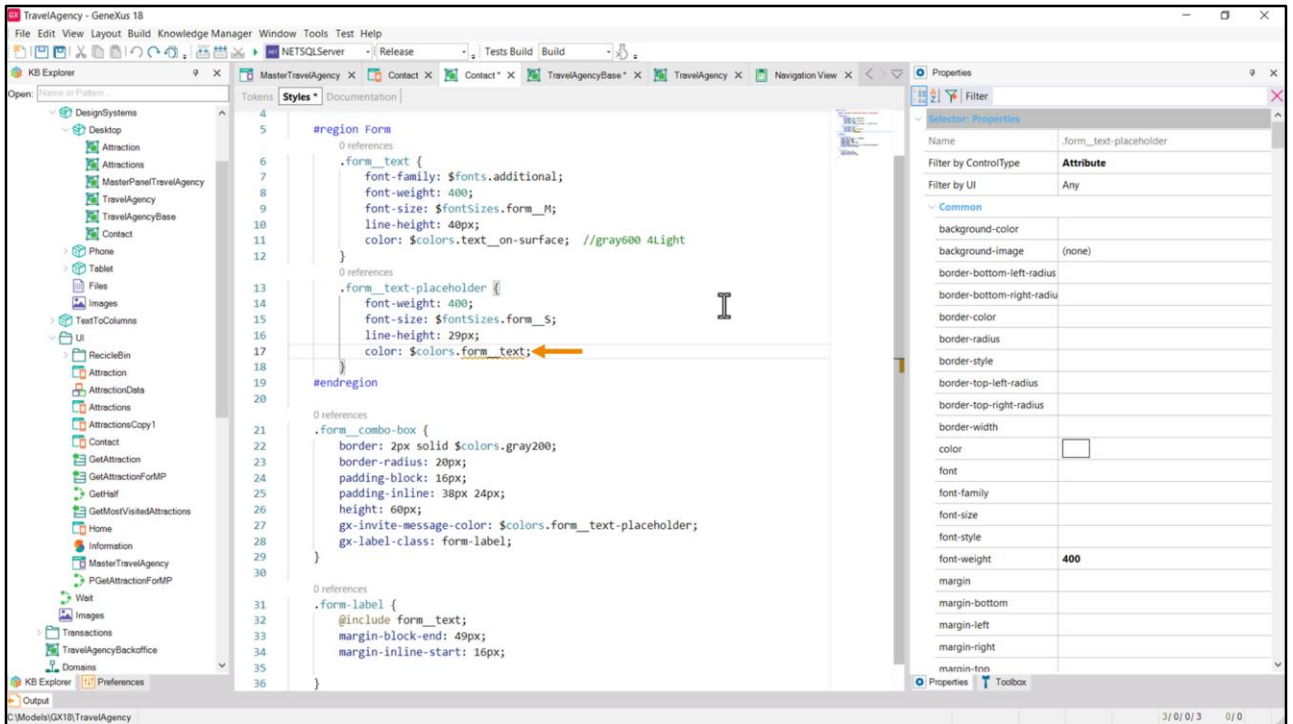


É por isso que existe a propriedade para poder indicar a cor.

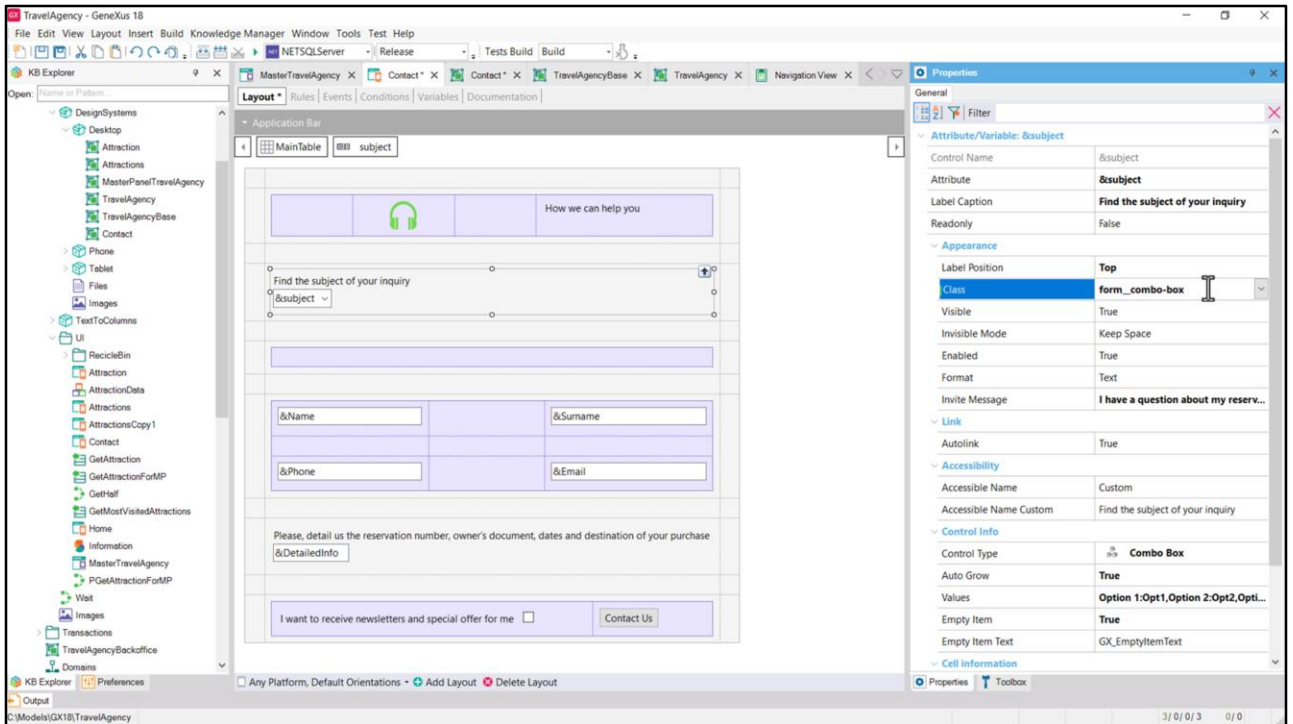
Então este valor está correto, mas o que não sabemos, é a cor que deveria ter o conteúdo do campo. Isso teria que ser consultado com a designer.



Isso possivelmente nos levaria a definir outro token de cor para o conteúdo... vamos colocar este valor, por exemplo...

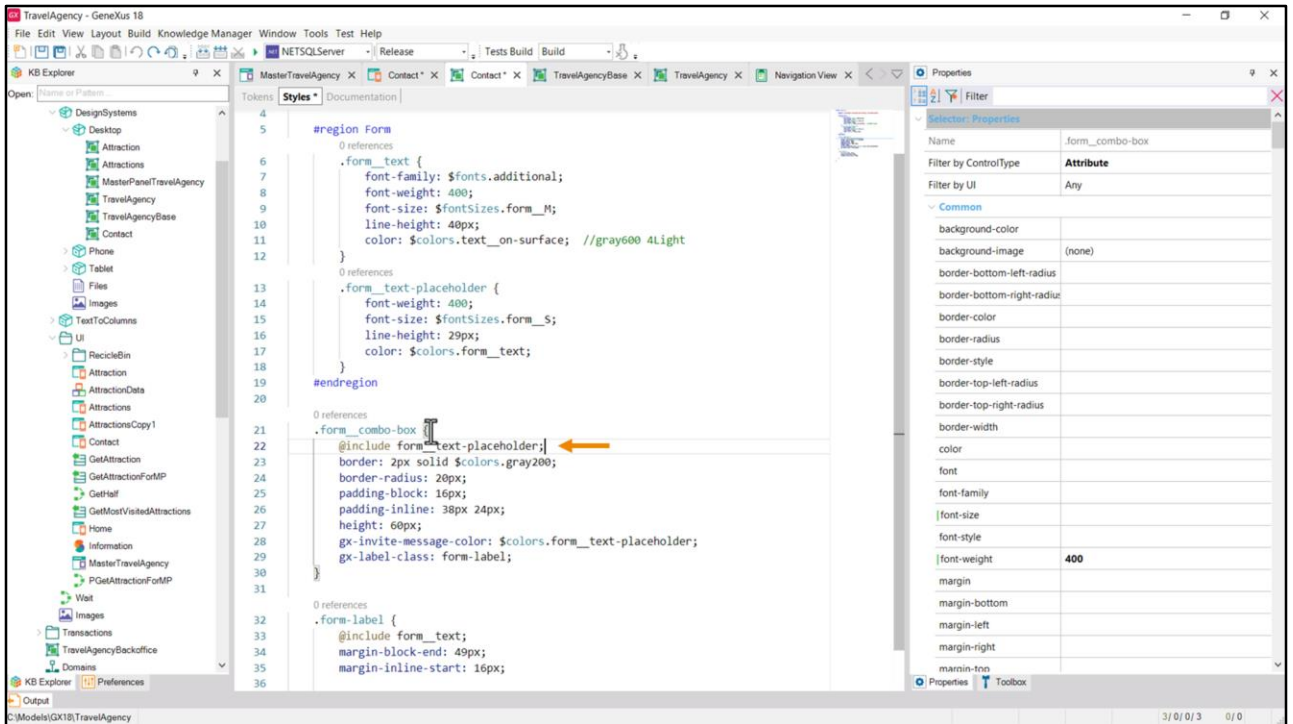


...E aqui faríamos referência a esse token.

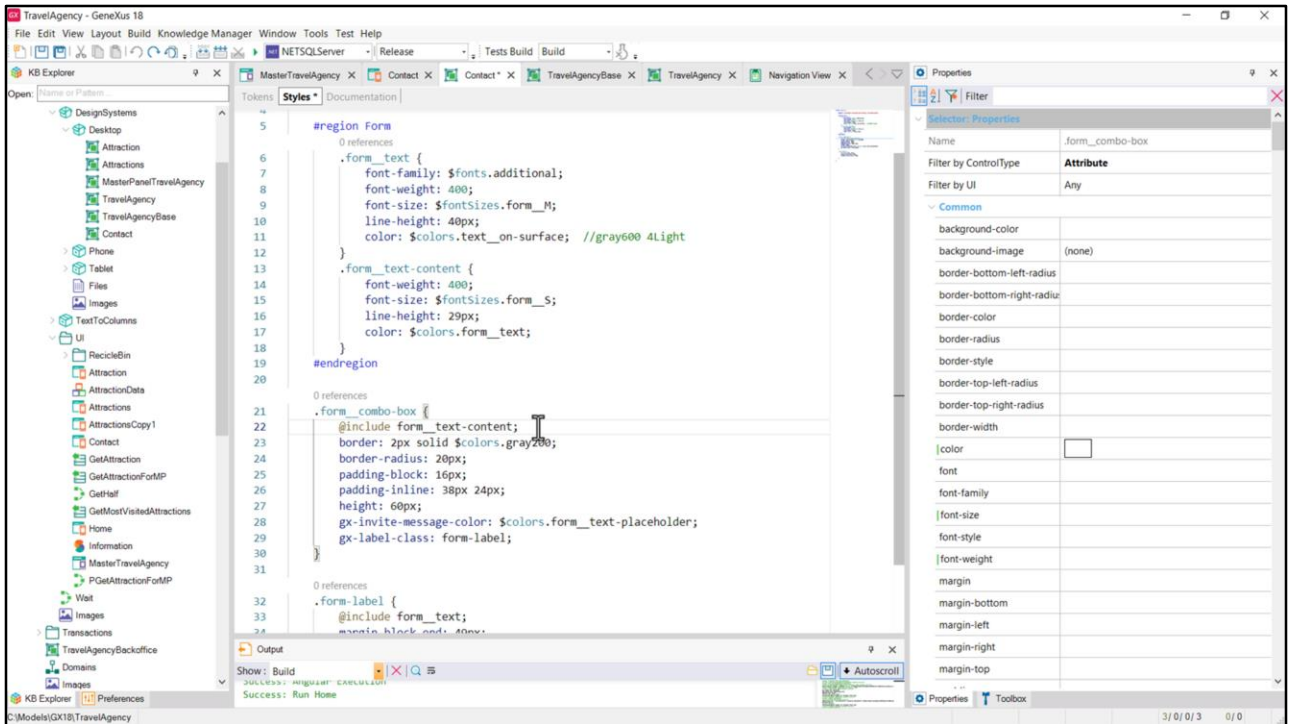


Para organizar um pouco melhor tudo isso, eu deixaria associada ao combo-box uma única classe que contenha tudo... e não duas...



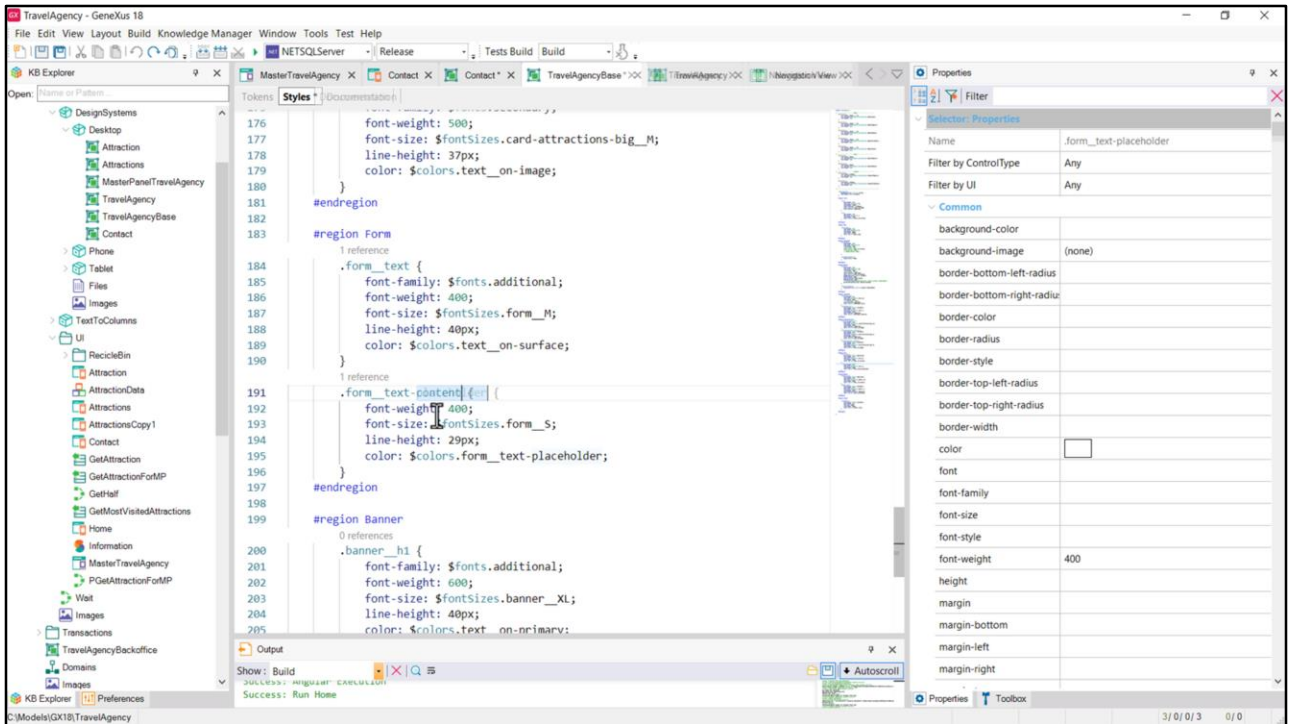


E então, nesta classe, incluo a tipografia.

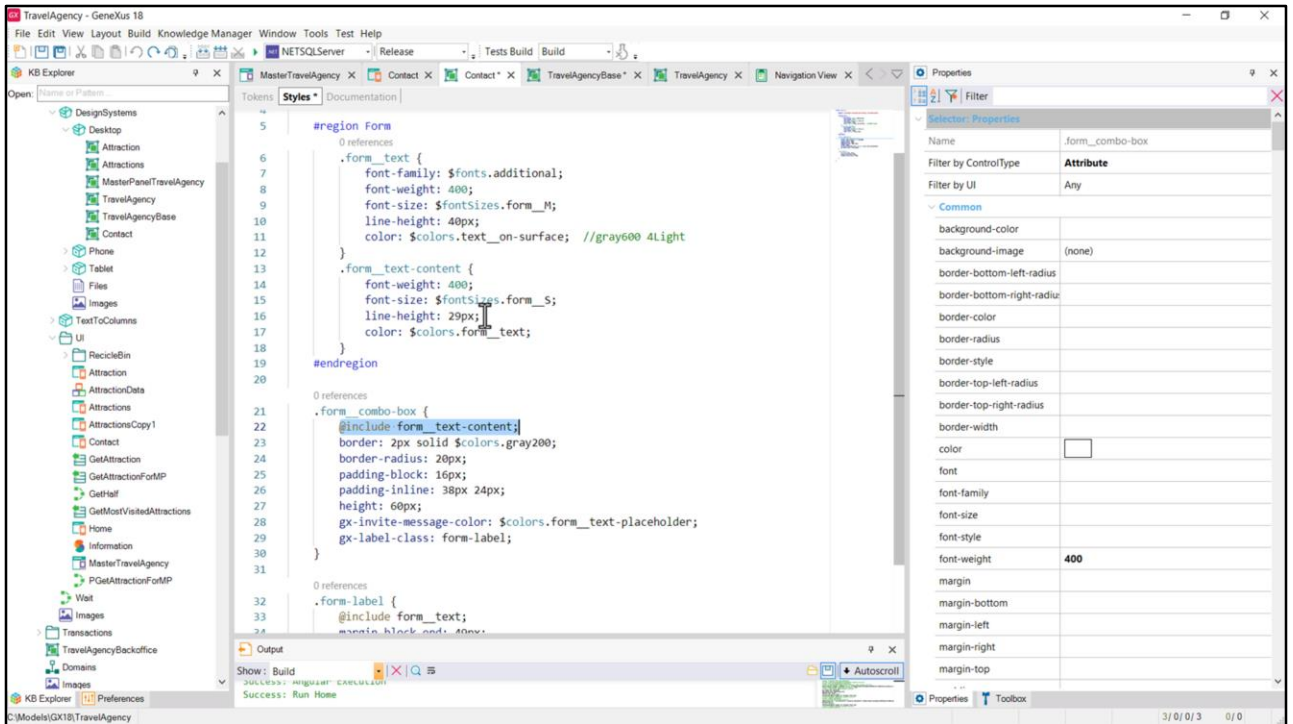


Neste ponto, percebemos que a análise inicial que fizemos das classes tipográficas que precisaríamos para os campos de entrada não foi muito completa e, na verdade, essa classe deveria ser chamada de form\_\_text-content...

Como é a primeira vez que a usamos, é o momento de mudar o nome.

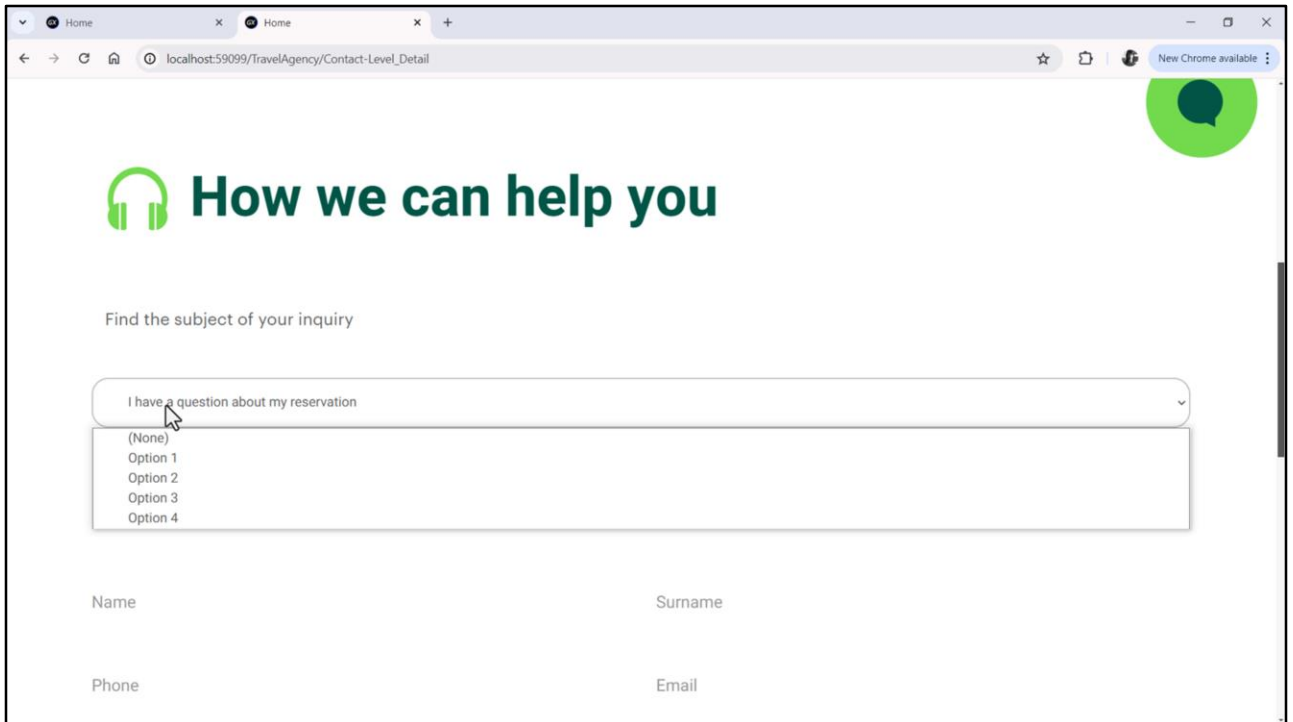


Se não a apagamos do DSO base, então tenha cuidado porque a teremos repetida.

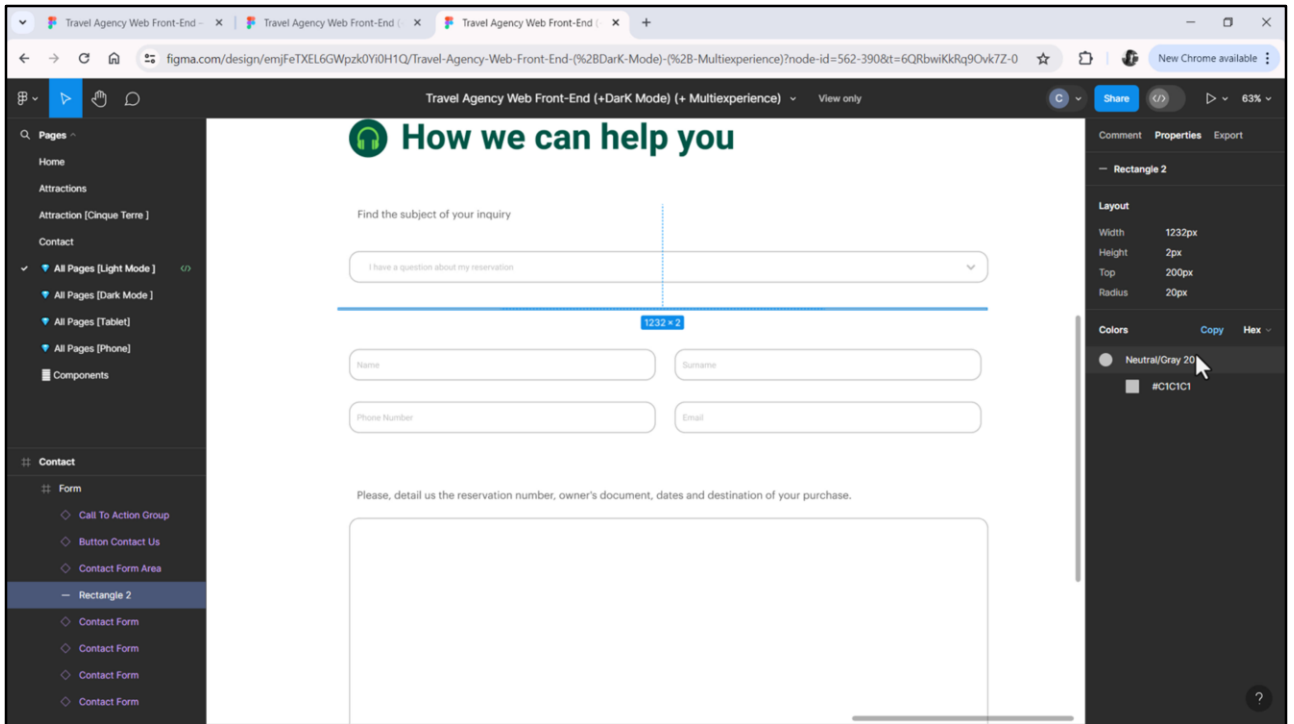


A copieei para este porque, lembrem que se eu quiser utilizar a regra include, por enquanto é necessário que a classe a ser incluída esteja no mesmo DSO.

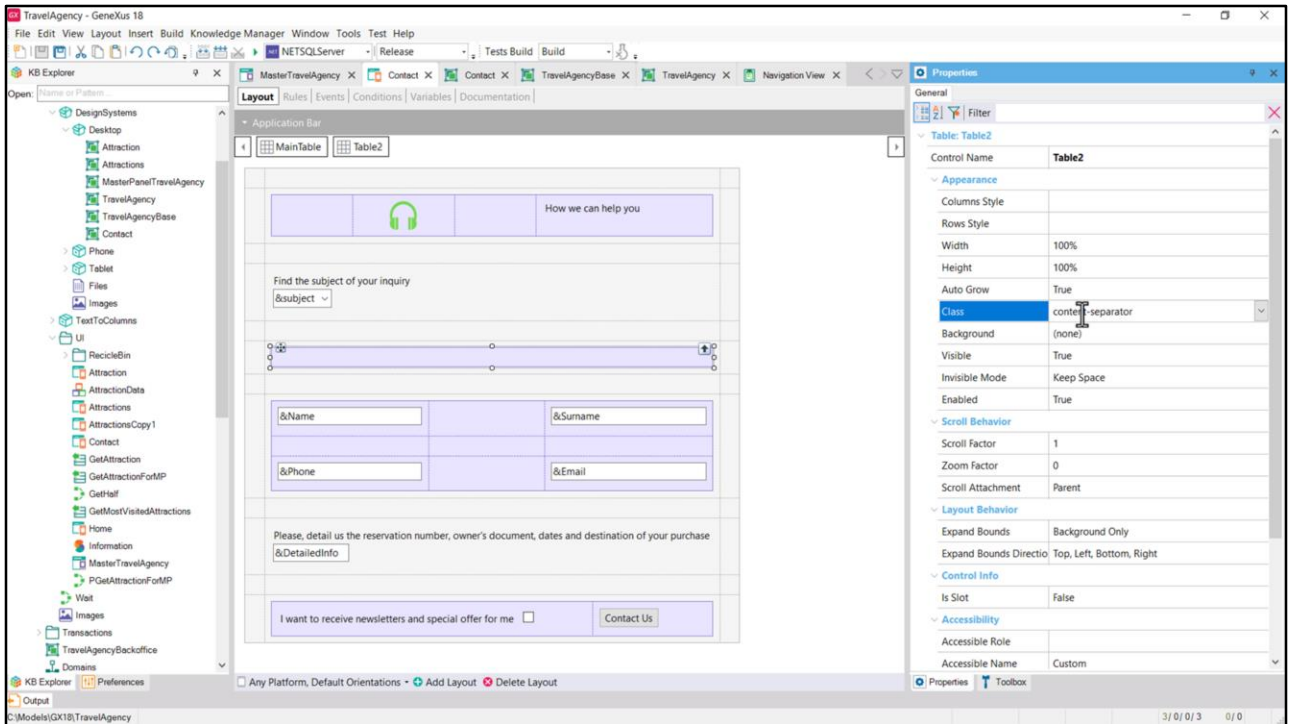
Com nossa solução anterior (com duas classes separadas para o controle) não era necessário tê-la aqui.



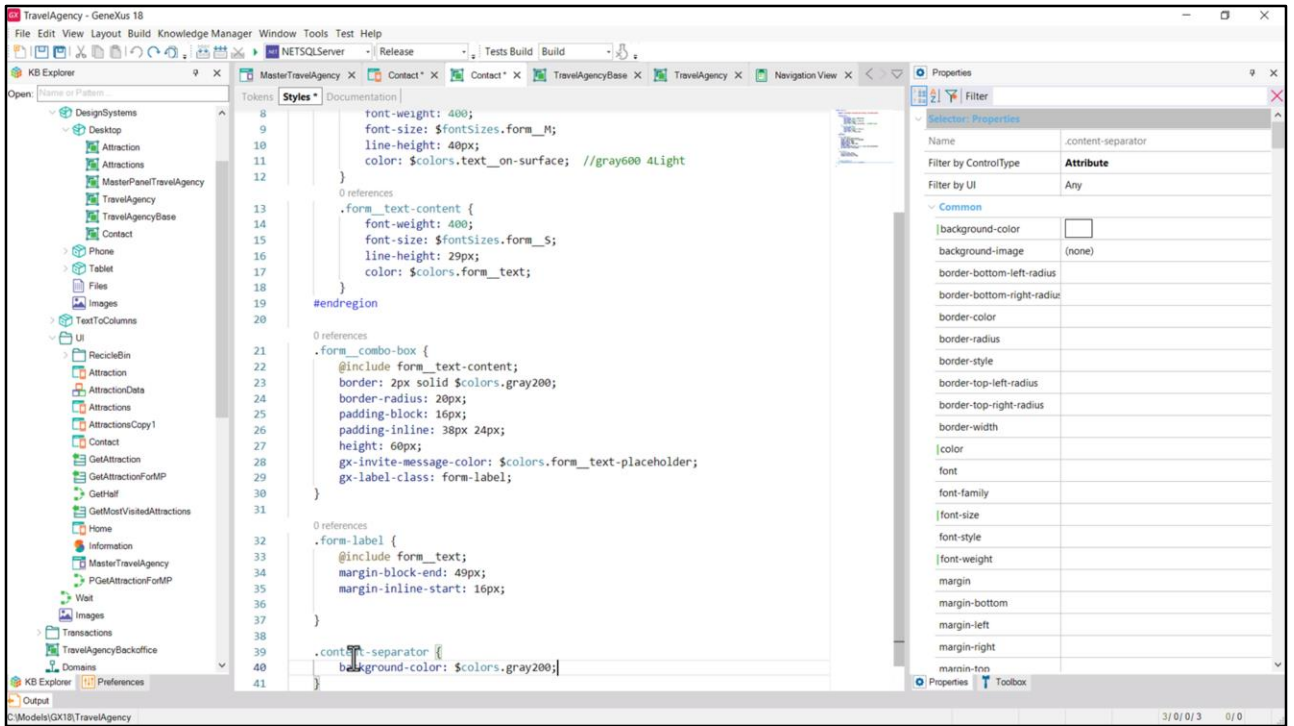
Se executarmos agora... temos um problema, que é que não estamos vendo a Invite Message com a cor gray200, o cinza claro, mas com a mesma da classe tipográfica. Isso ocorre porque, por enquanto, apenas é válida a propriedade `gx-invite-message-color` para os campos Edit. Ou seja, funcionará para todos esses outros campos, para os quais ainda não personalizamos o estilo, tudo está por padrão, mas é o que nos resta a fazer.



Para implementar a linha de separação que vemos no Figma, de 2 pixels e esta cor...

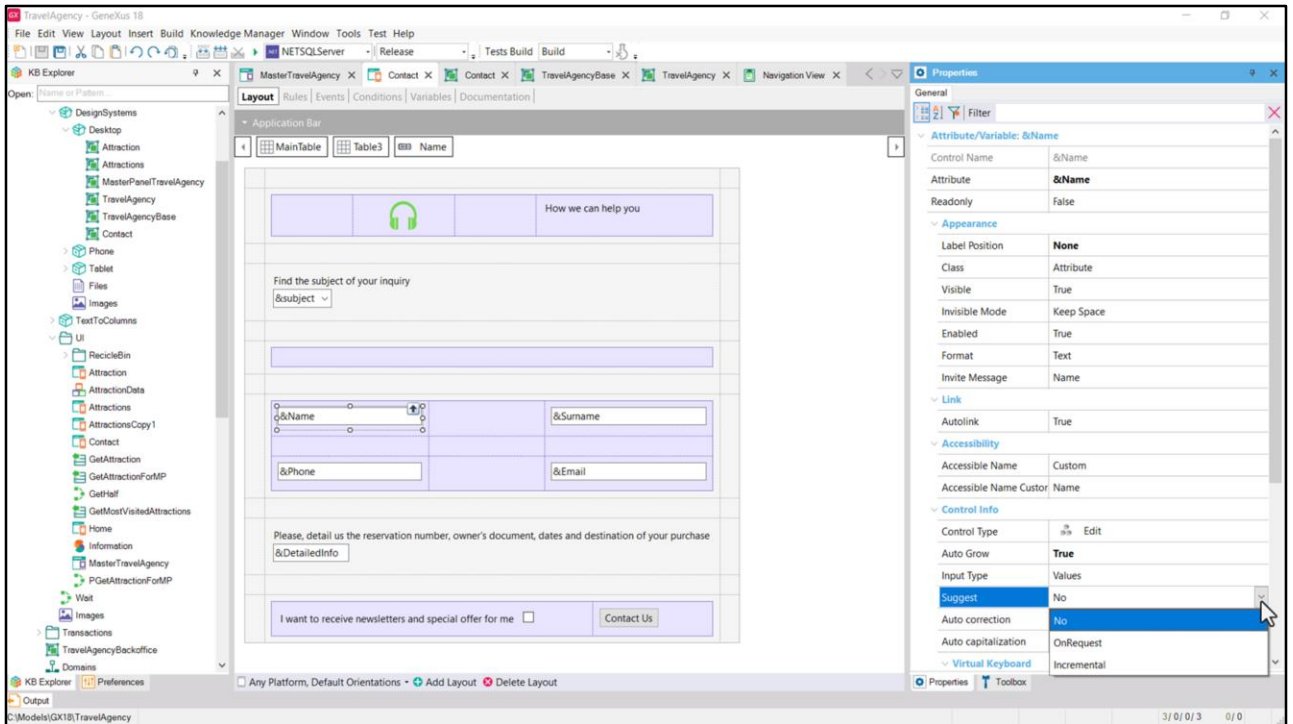


Foi que coloquei esta tabela de 2 dips de altura, à qual vou atribuir esta classe content-separator...



...onde à background-color darei esse valor.

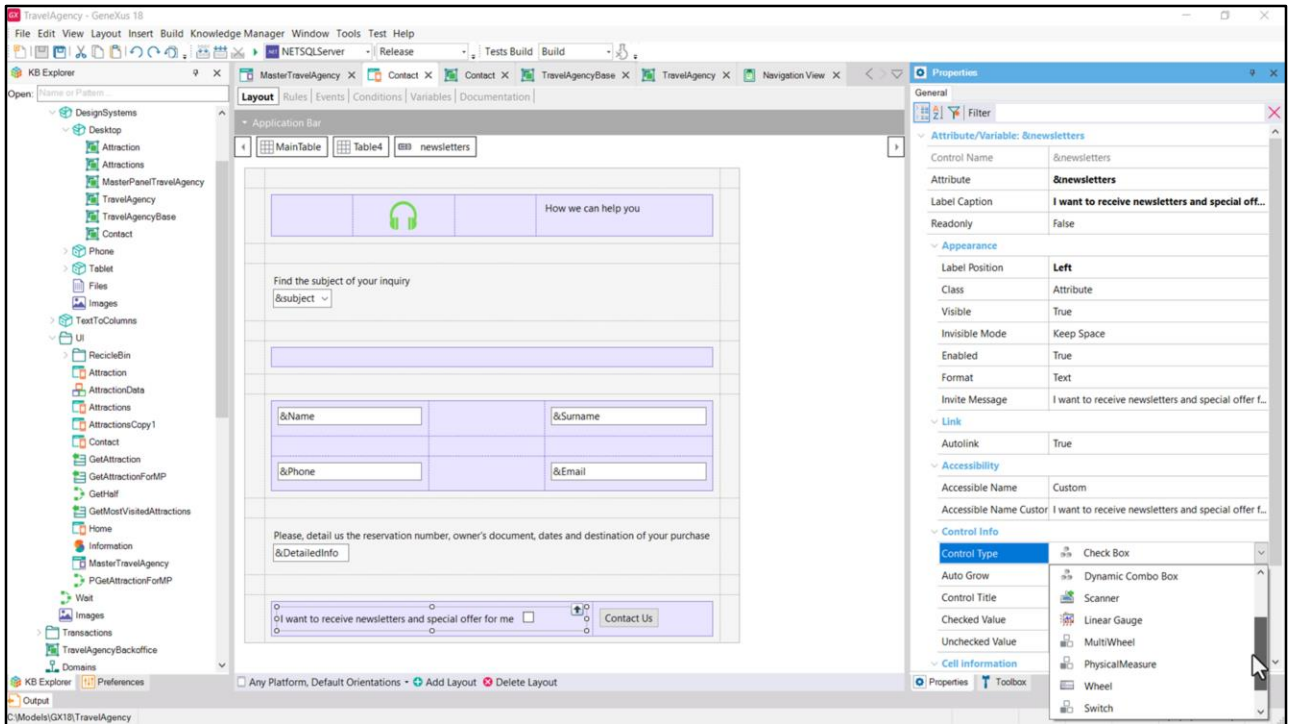




Agora deveria continuar fazendo o que fizemos para este campo combo box, para os outros...

Vou deixar isso como tarefa para vocês.

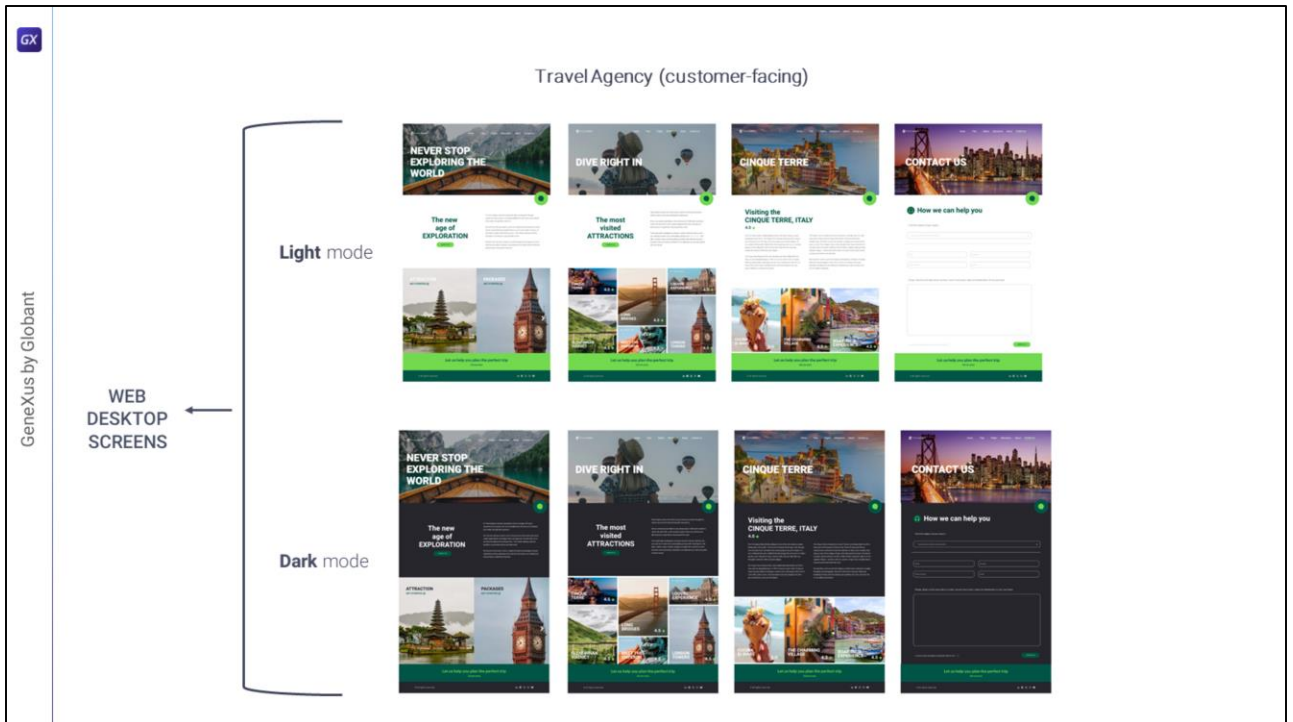
Vejam que estes são campos do tipo Edit sem Label e com Invite Message... Para os controles do tipo Edit, podem ser definidas uma série de coisas: por exemplo, a possibilidade de sugerir conforme o usuário digita... pesquisem na wiki de GeneXus e investiguem todas as opções.



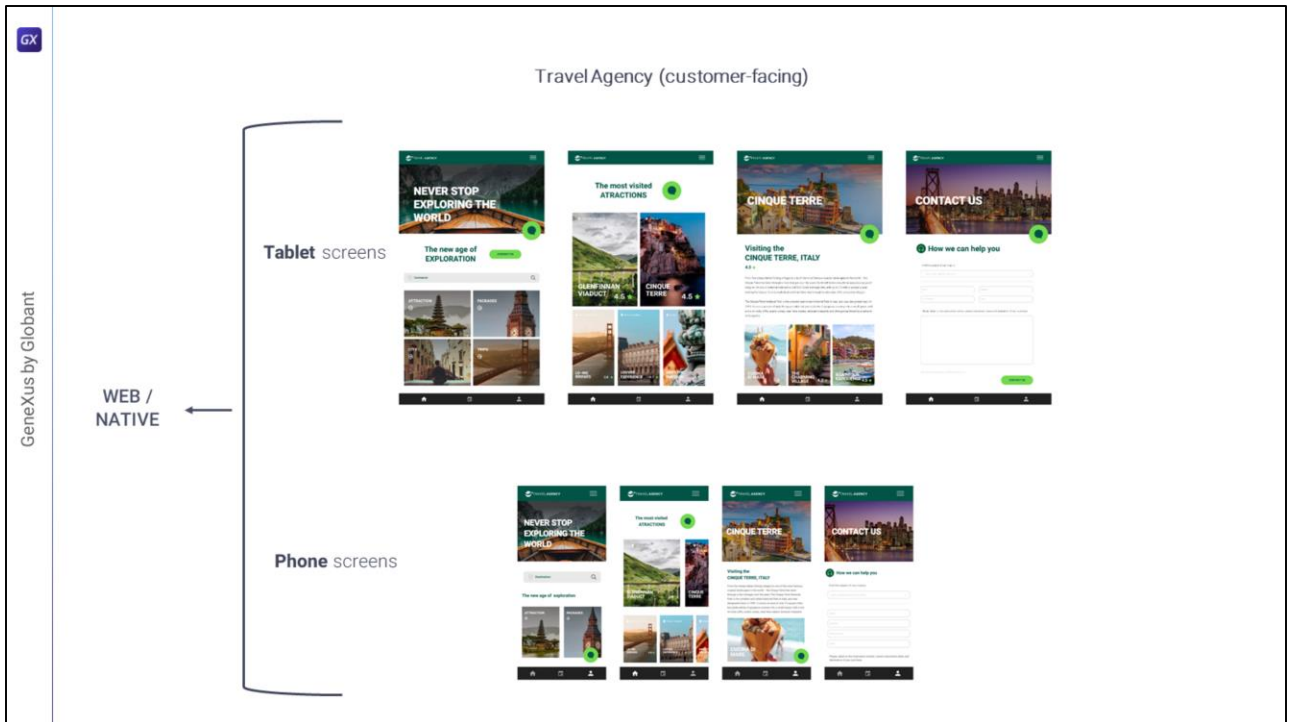
Esta outra variável também é Edit, mas tem label acima e não tem Invite Message...

E por último essa outra é do tipo Checkbox, com o rótulo à esquerda.

Observem que entre os tipos de controle para as variáveis temos muitas outras alternativas, que têm a ver com a User Interface, e que também convido vocês a pesquisar.



Com isso, considero encerrado o mais relevante que queria apresentar a vocês em relação ao desenvolvimento da aplicação Angular para tamanho Desktop. Dissemos desde o início que para tamanho Desktop só teremos que implementar esta aplicação, a Angular.



Então, quando entramos nos outros tamanhos de tela, tanto Tablet quanto Phone, o mundo se divide em dois paradigmas: o de Angular (Web) e o nativo. Ou seja, teremos que implementar ali as duas vertentes. E no mundo nativo, a vertente para Android e a vertente para Apple.

Falaremos sobre isso no próximo módulo, como uma introdução e análise geral de como deveríamos pensar, então, como seria conveniente pensar nas telas e na aplicação, enfim, para ser o mais eficiente possível e reutilizar o máximo possível. Agora, não apenas entre as telas da mesma plataforma, mas também para ser possível compartilhar o mesmo desenvolvimento e design para todas as plataformas, o que é um desafio. Mas falaremos sobre isso no que virá a seguir. Espero por vocês.

GX

GeneXus by Globant

**GeneXus**<sup>™</sup>  
by Globant

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