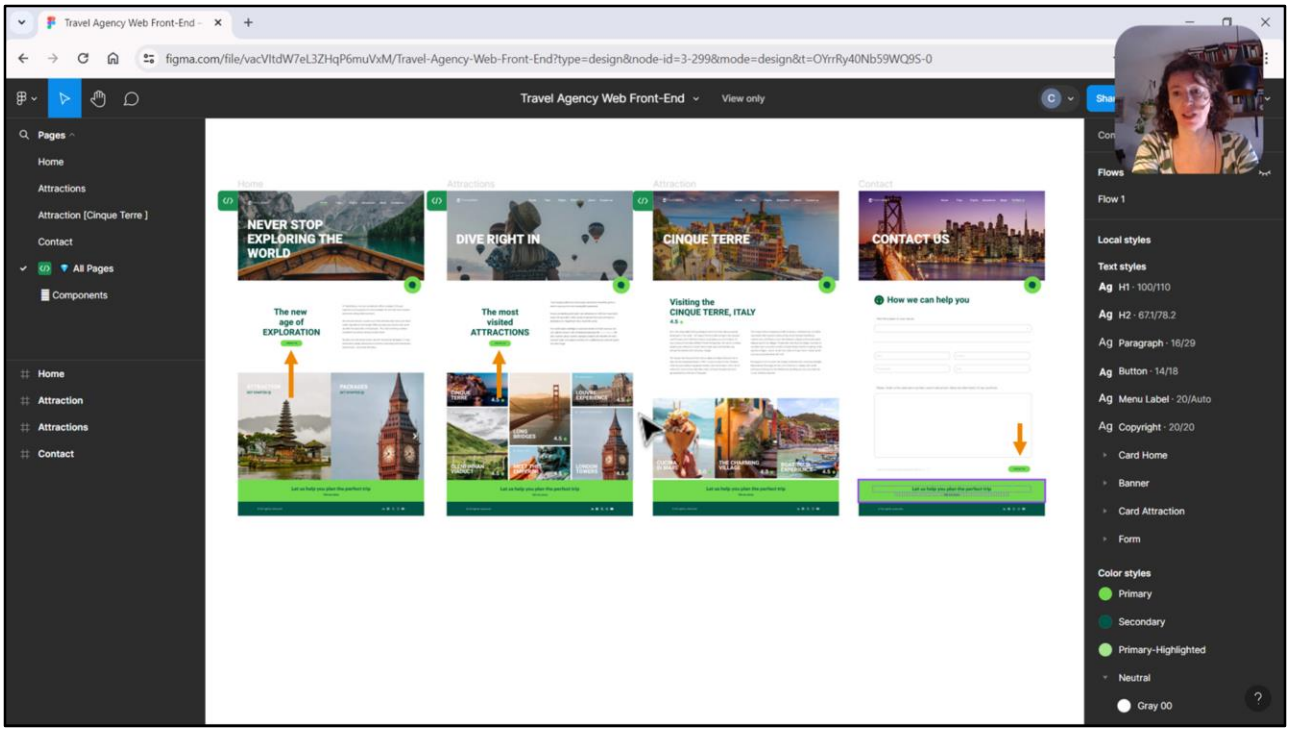
GeneXus by Globant

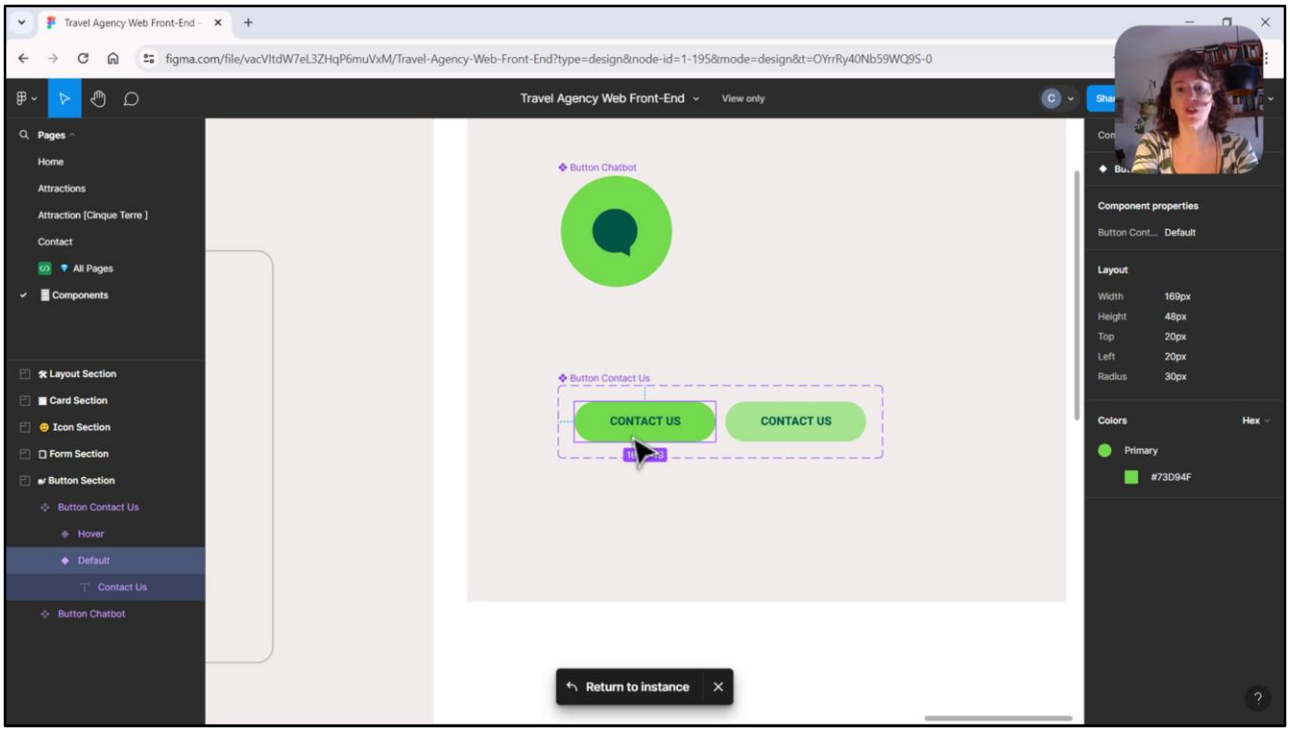
First Layout in GeneXus. Style (cont.)

Button style with behavior

Cecilia Fernández



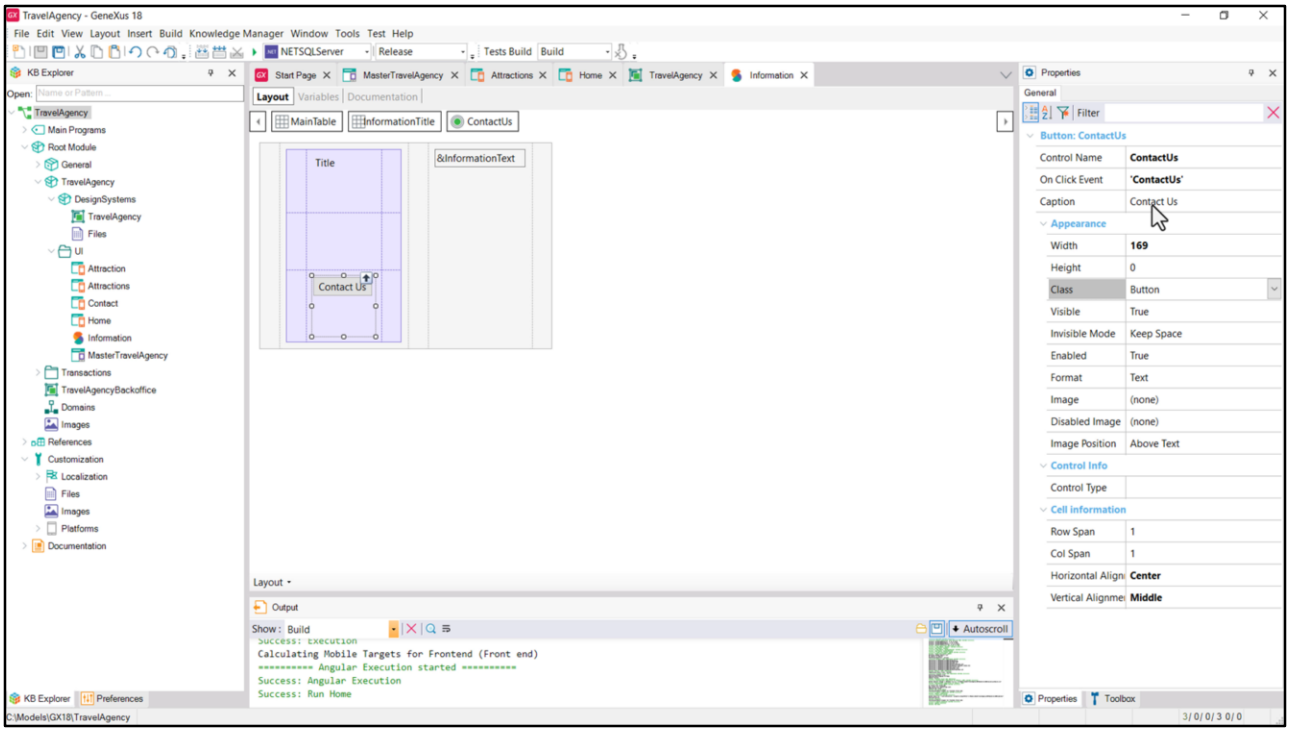
Bem, neste vídeo vamos nos concentrar no botão que aparece nestas telas, nestas três telas: na Home, na de atrações e na de contato.



Justamente devido a essas repetições é que Chechu escolheu modelar este elemento como um componente. E então aqui o vemos na página de componentes.

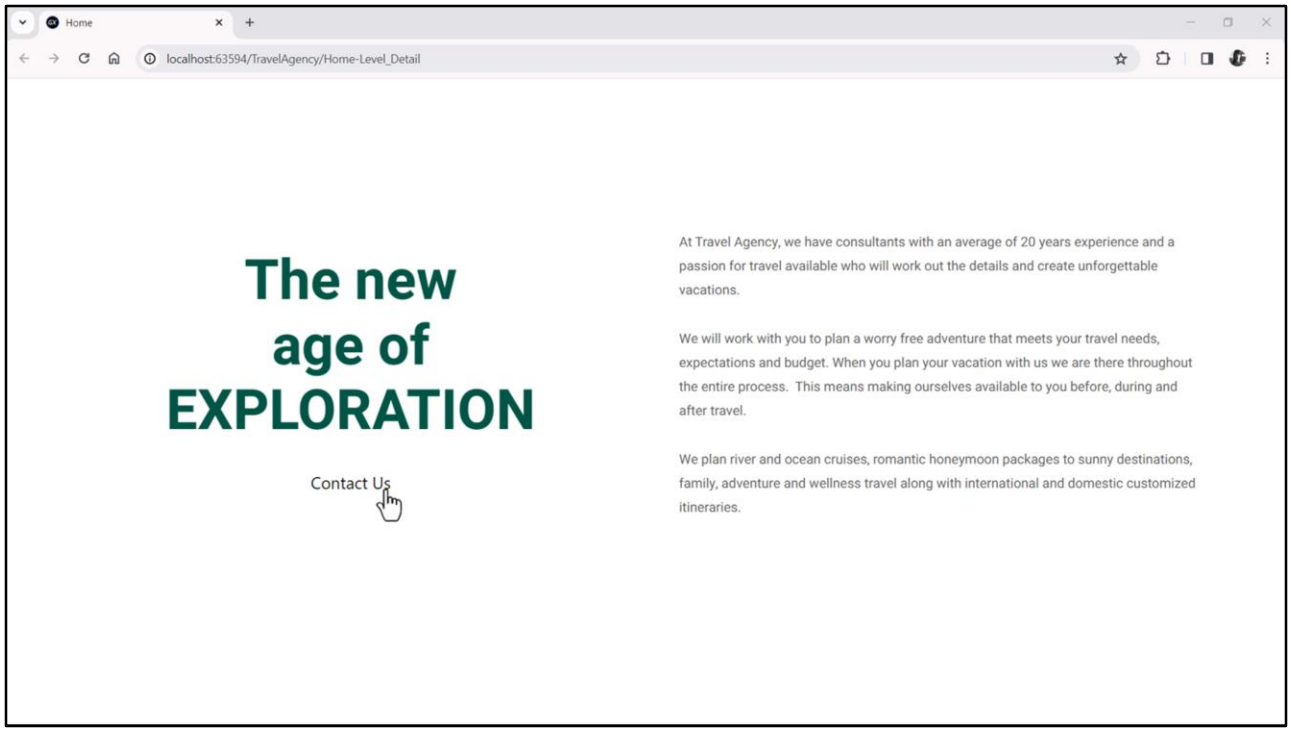
De fato, aqui estamos vendo uma variante do botão, que é a variante default, que é como o botão será visto por default em todos os nossos layouts. Mas Chechu também modelou a variante Hover, que é quando é feito hover sobre o botão.

Vamos começar pela variante Default.



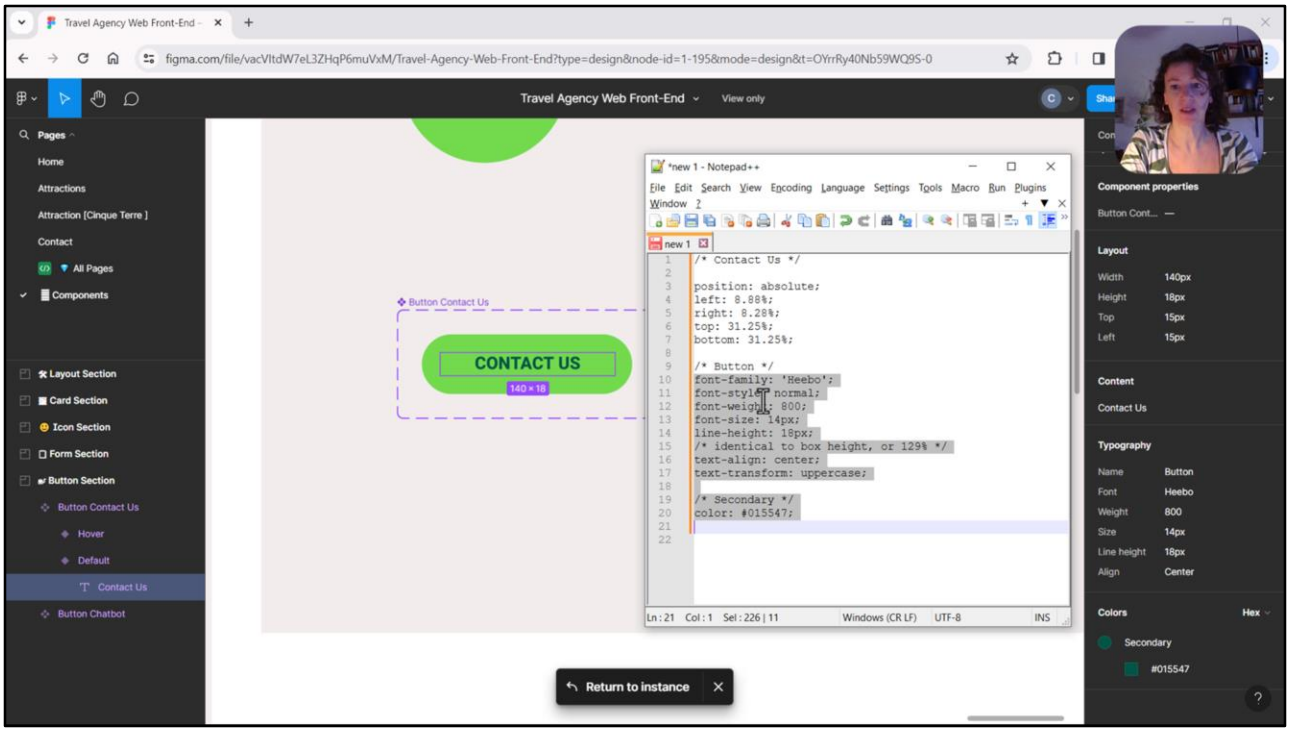
Em GeneXus, nós temos o controle do tipo botão, que foi o que inserimos no Stencil, e ao qual configuramos este Caption.

Se prestarmos atenção às propriedades, vemos que tem como classe associada a de nome Button... aqui vemos o botão no panel Home.



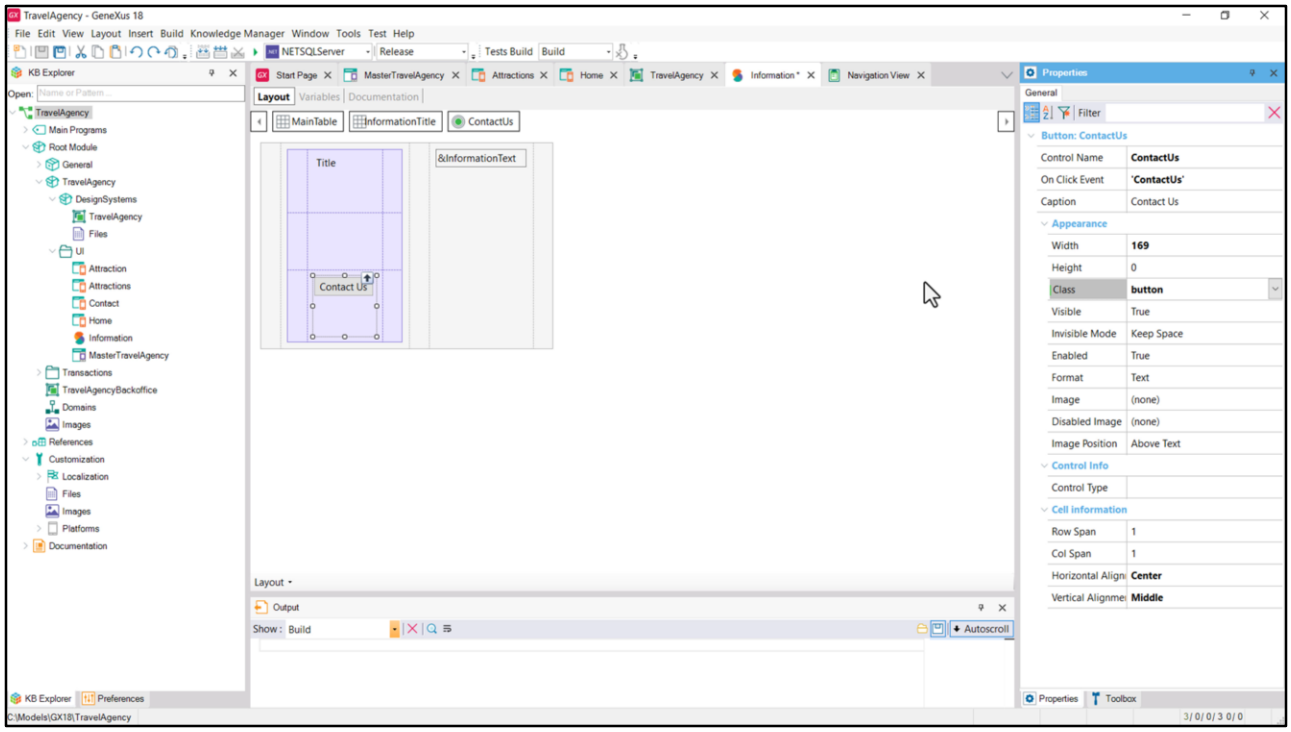
Essa classe não está definida em nosso DSO, e é por isso que em execução vemos o botão assim...

Podemos inspecioná-lo...

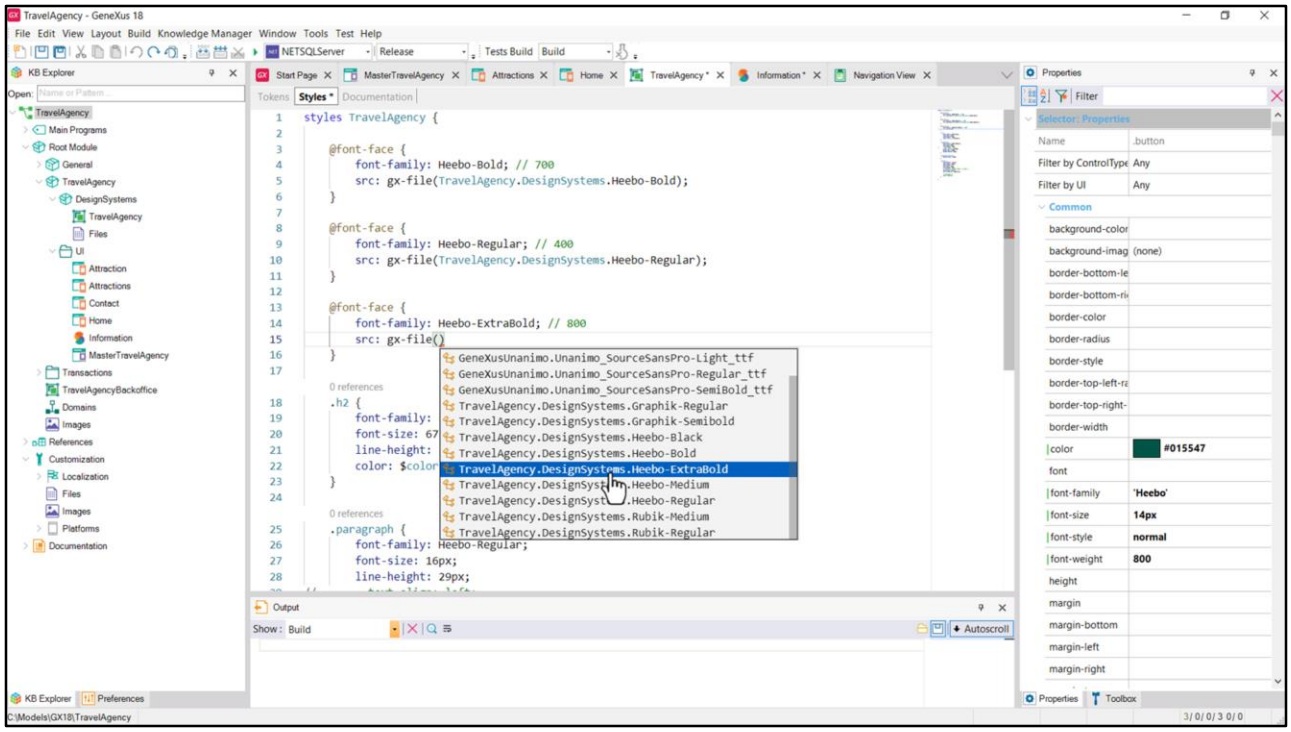


Agora, em Figma não temos um elemento do tipo botão. Para construir um botão, temos que fazê-lo agrupando dentro de um container duas camadas: a camada do fundo e uma camada sobreposta que será a que vai conter o texto, que seria o caption no caso de nosso botão GeneXus. De fato, se formos observar as propriedades deste texto, que corresponde à camada superior, vemos que Chechu chamou a tipografia, criou um estilo de texto tipográfico chamado Button, com essas características.

O que podemos fazer é copiar o código CSS como tínhamos feito antes, colá-lo em um Notepad, copiar as propriedades…

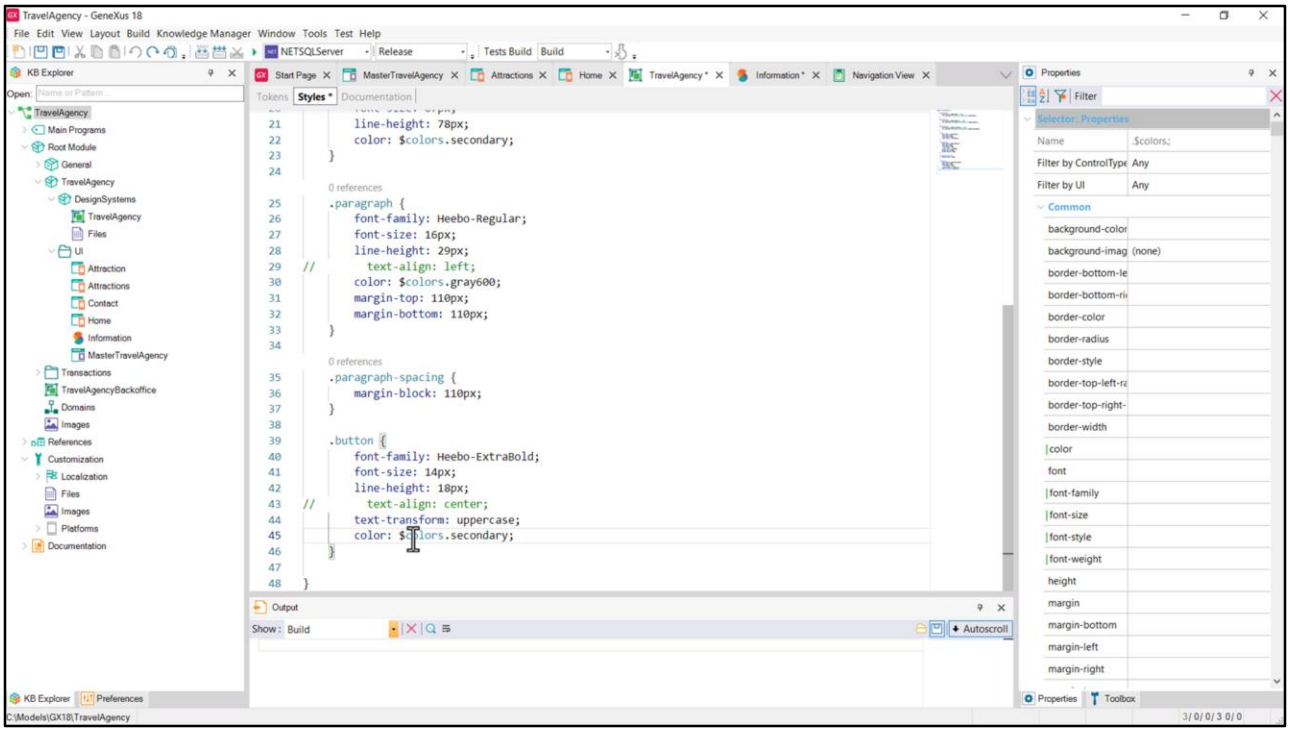


… e ir ao GeneXus e associá-las dentro do nosso DSO às da classe Button. Mas lembre-se que, como estamos utilizando a nomenclatura BEM, vamos usar o nome de classe em minúscula –como isto é case sensitive não será a mesma coisa a classe em minúscula que a classe em maiúscula- bem, vamos nomeá-la em minúscula…



e vamos ao nosso DSO e vamos escrever esta classe com minúscula.

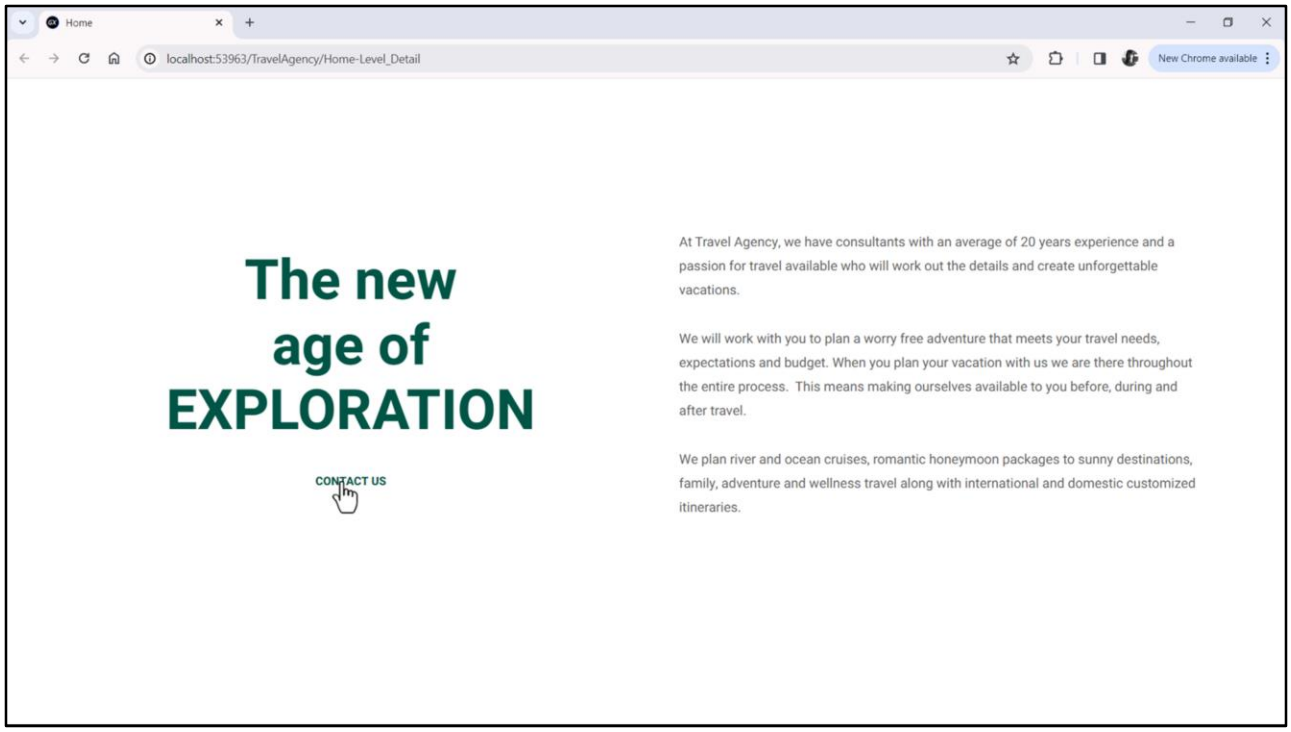
Copiamos aqui dentro as propriedades... e vamos fazer o mesmo que fizemos antes para os outros controles, dos vídeos anteriores, ou seja, vamos introduzir a família de fontes, esta Heebo que vemos que é de peso 800. Então, vimos aqui e especificamos a regra font face... vou chamar de ExtraBold... vai corresponder a este arquivo que inseri antes na KB.



E agora o que faço, como sabemos, é isto. Bem.

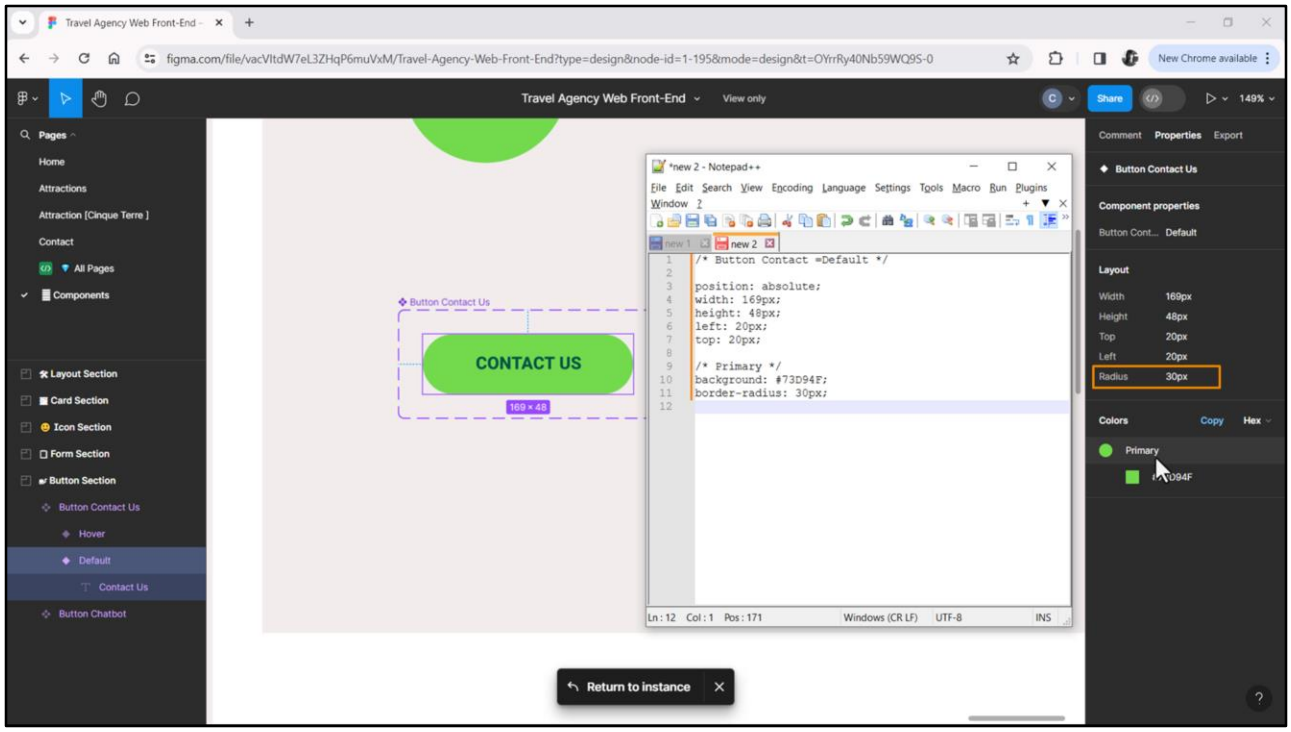
Text-align center não vou precisar, como vamos ver. E aqui temos a propriedade CSS text-transform que é para fazer com que, independentemente de como esteja escrito o texto, sempre fique em maiúsculas, tudo em maiúsculas. E então, por outro lado, à cor vamos associar, não o valor absoluto, mas o nosso token de cor, secondary... tendo este cuidado... que aqui ficou isso errado. Bem.

Vamos testar isso assim como está.



Bem, vemos a mudança, certo? Que passou tudo para maiúsculas, que tem a cor que queríamos, o tamanho, etc.

Agora está faltando o que tem a ver com o background.

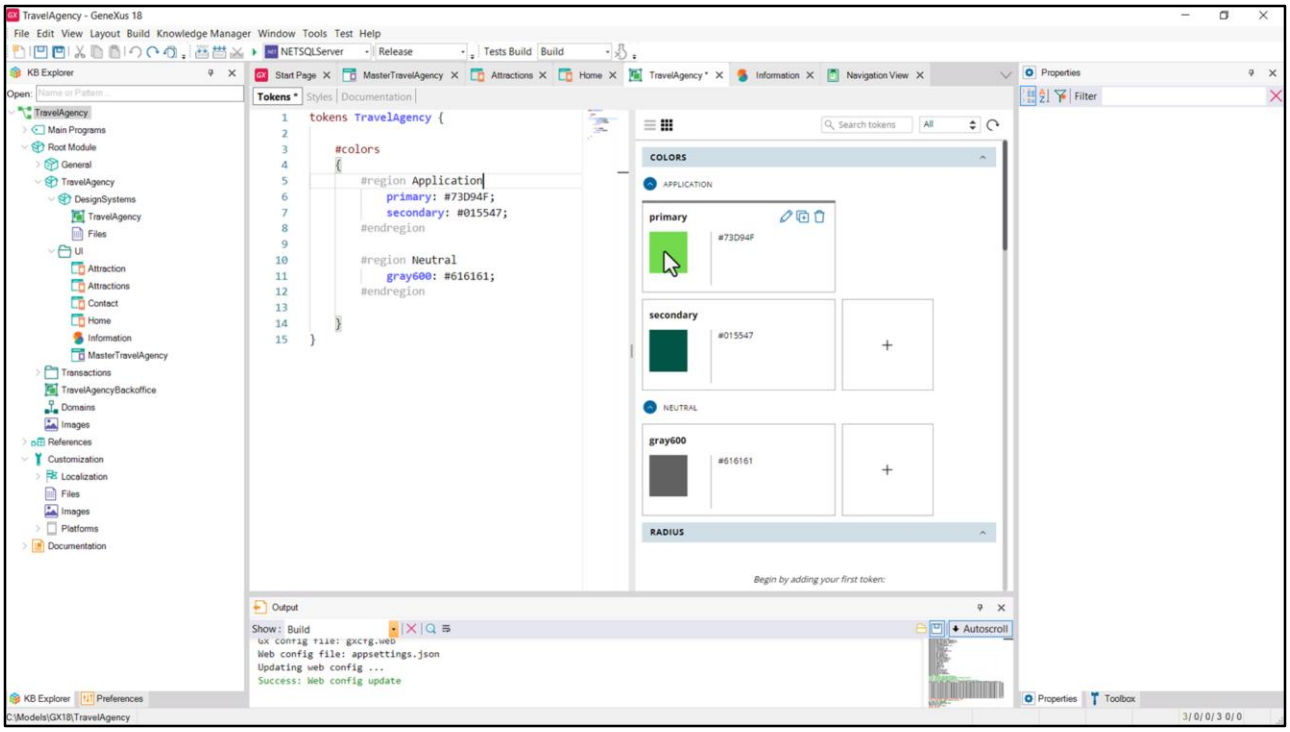


Então vamos ao Figma e nos posicionamos sobre esse background.

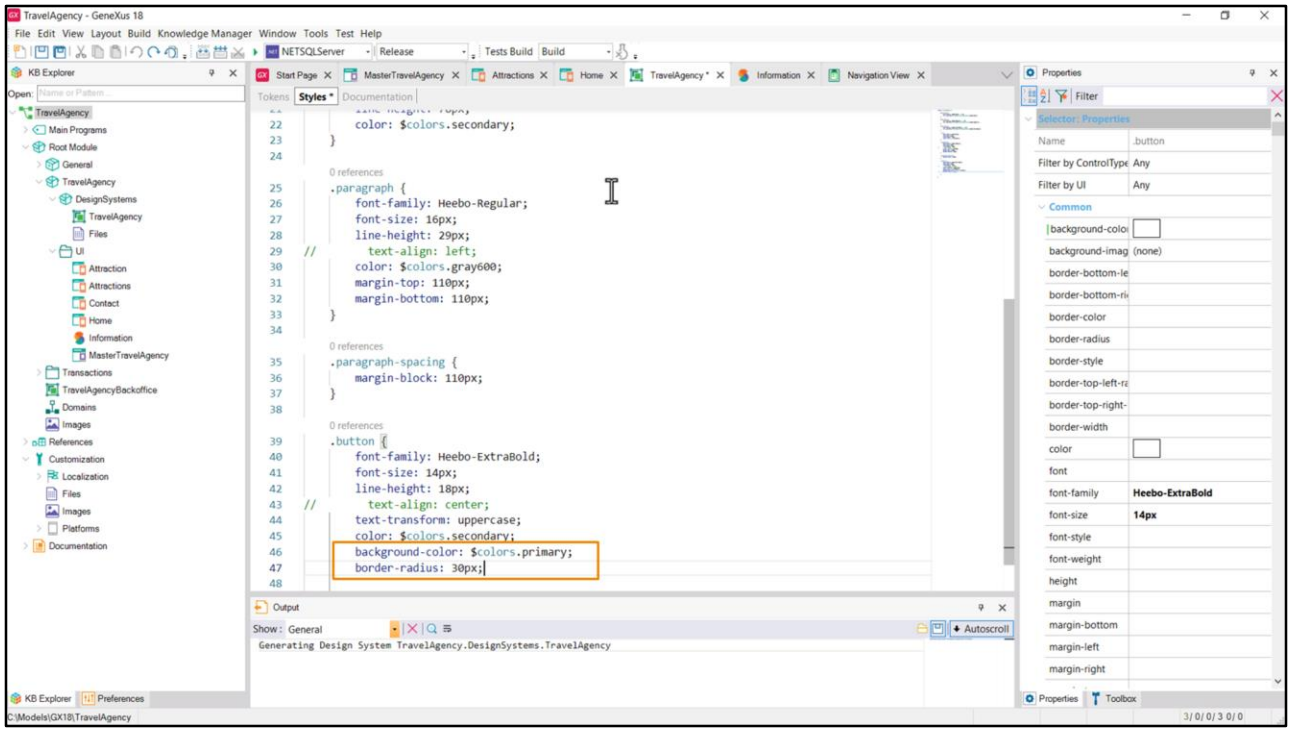
E aqui vemos as propriedades. Em particular, vamos prestar atenção a esta Radius, que é a que vai dar justamente essas bordas arredondadas.

Vamos, novamente, escolher as propriedades CSS, colá-las aqui... e então vemos a propriedade de cor, background, que é esta daqui, que está mostrando esta cor, verde, à qual Chechu deu um nome ao estilo de cor, chamou-o de Primary. Que é a primeira coisa que vamos fazer, converter isto em um token primary, com este valor. E então, além disso, temos que adicionar esta propriedade CSS à nossa classe button.

Então... vamos copiar primeiro isto...

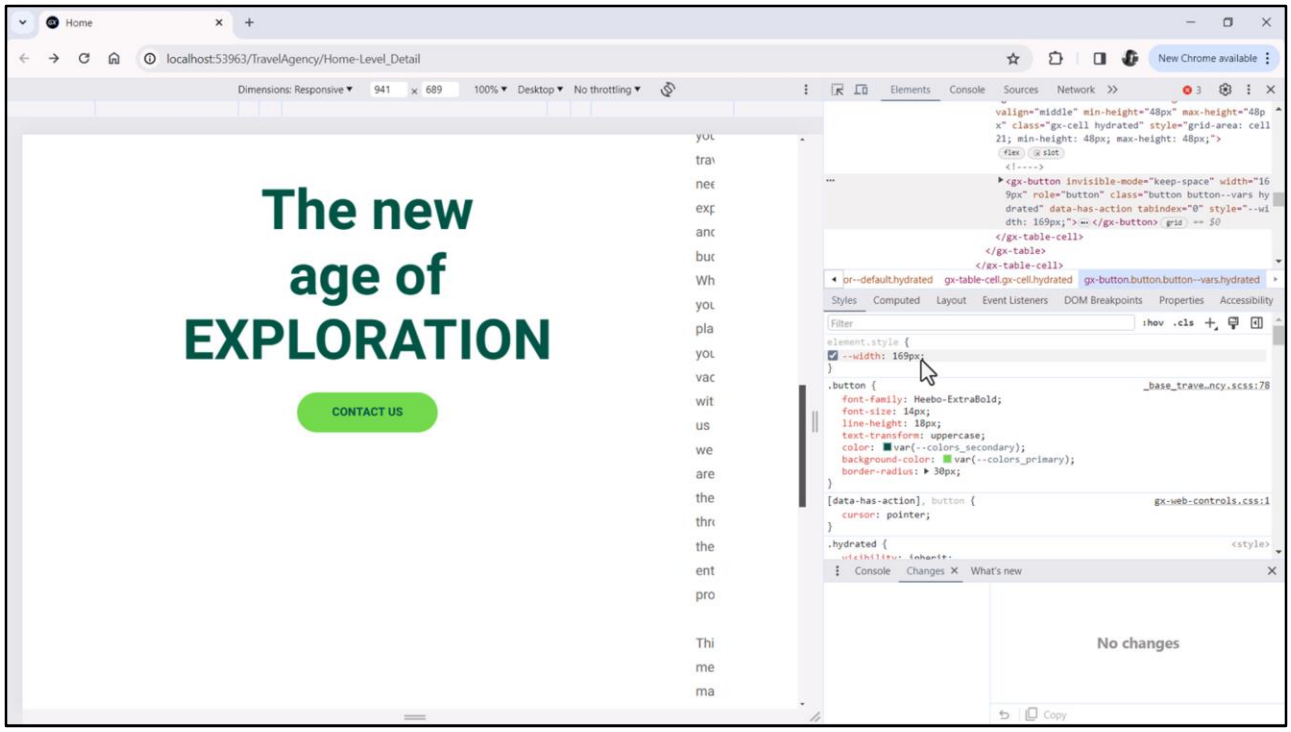


...vamos ao nosso DSO, à aba de tokens e adicionamos este token, o primary.



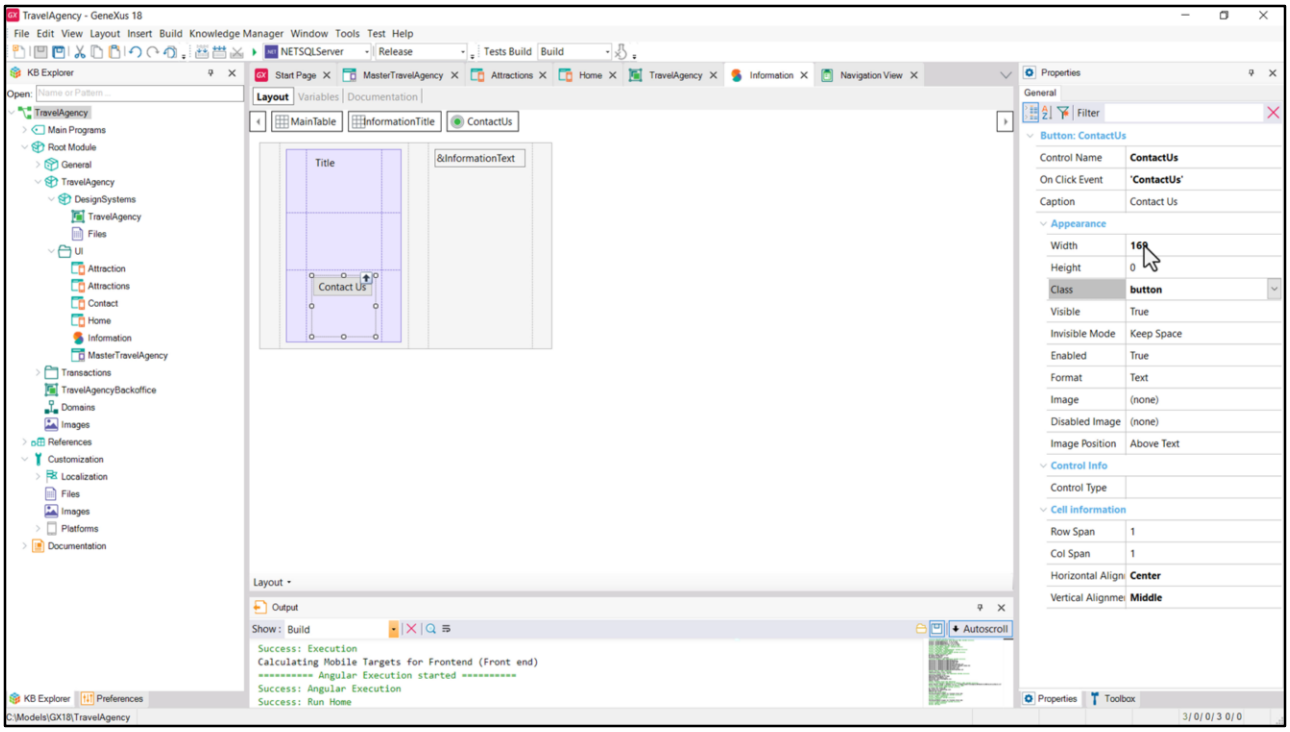
E depois vamos à classe, adicionamos a propriedade background-color... e por outro lado vamos copiar esta outra.

Vamos testar isso assim agora...

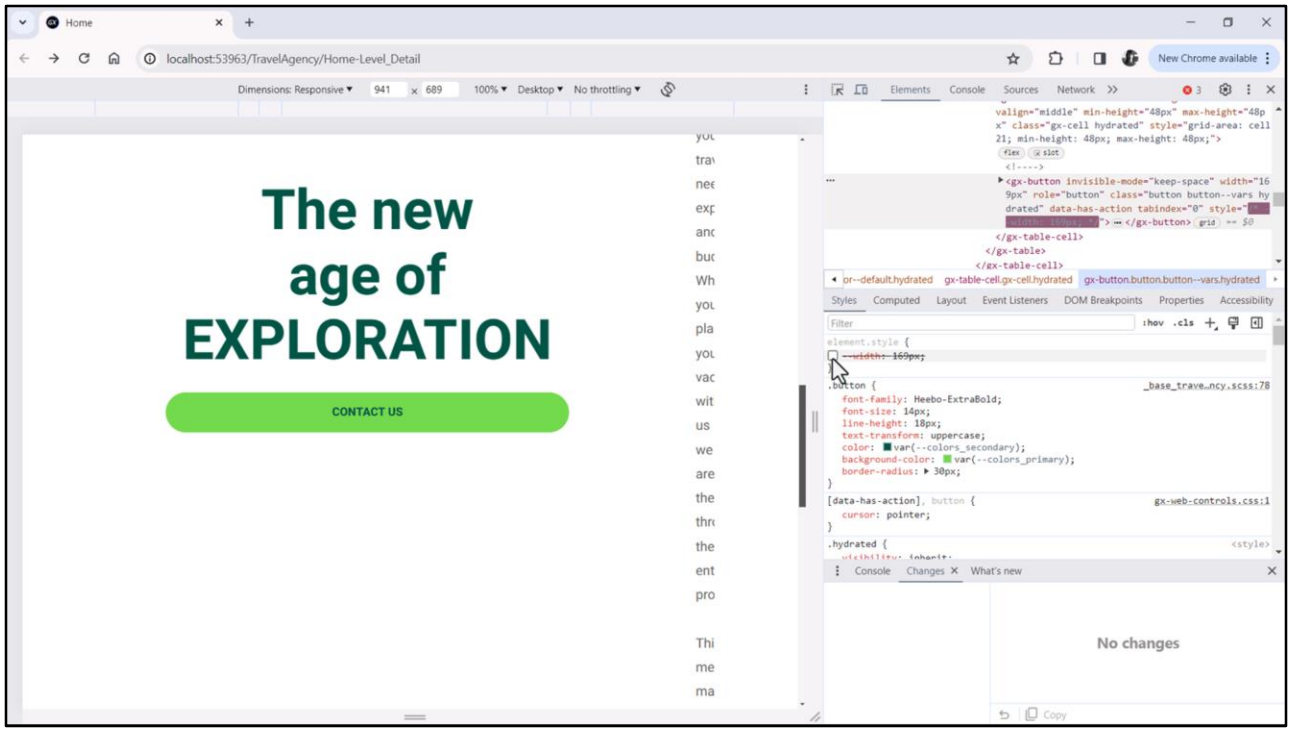


Bem, parece que isso seria tudo.

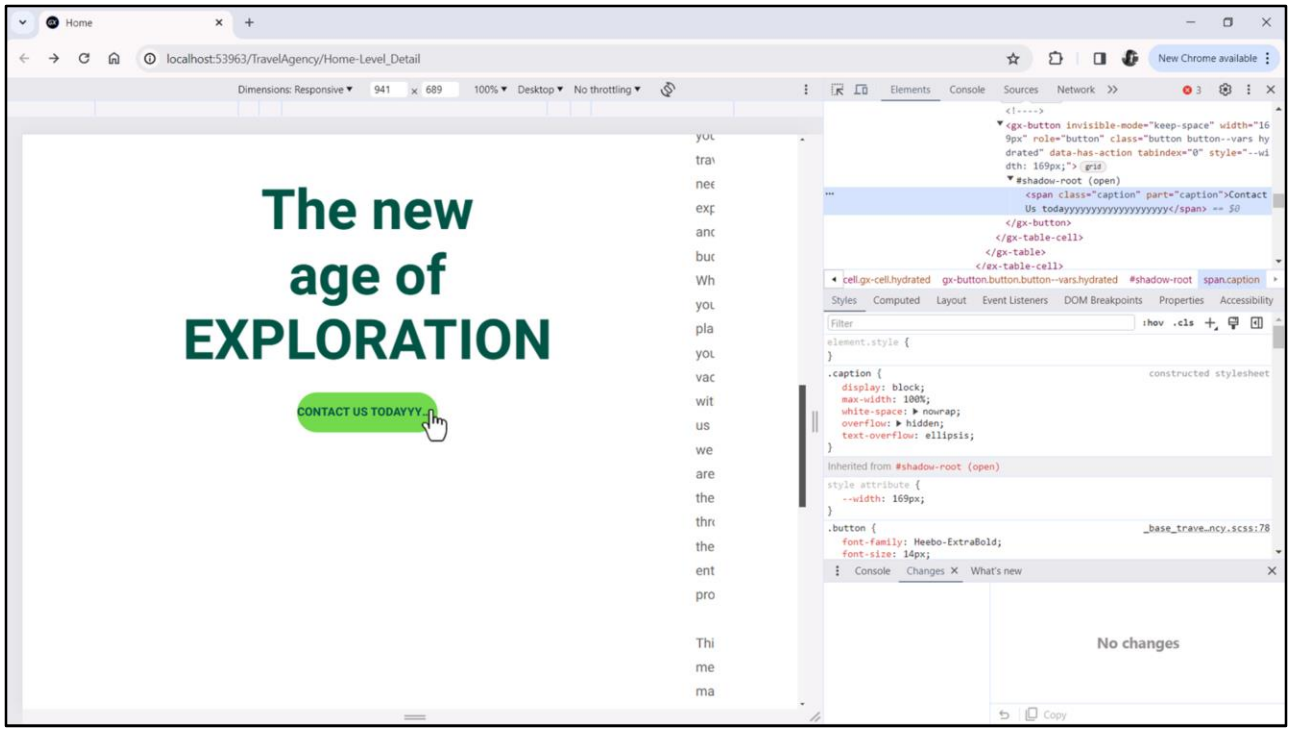
No entanto, pensemos no que aconteceria se traduzíssemos a aplicação para um idioma em que este texto fosse mais longo. O que aconteceria com a largura do botão? Se o inspecionarmos, vemos que está assumindo como largura este valor, 169 pixels, que, de onde vem?



Do valor que tínhamos dado à propriedade Width no nível do botão dentro do stencil. Tínhamos predefinido a largura do botão para este valor. E tínhamos dito que se deixássemos o valor default que era um 0, o que ia acontecer era que o botão ia se expandir para ocupar a largura da célula em que se encontrava.

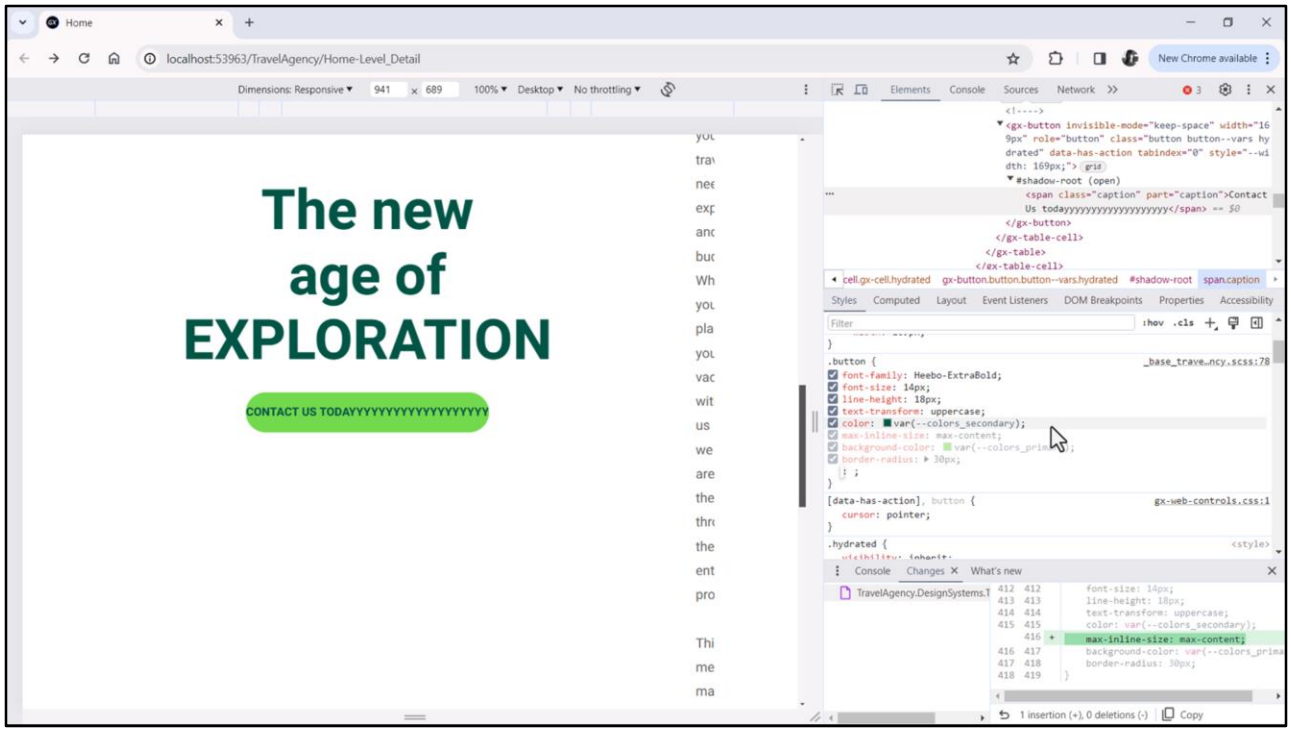


E podemos ver isso muito claramente aqui mesmo, se eu fizer isso.



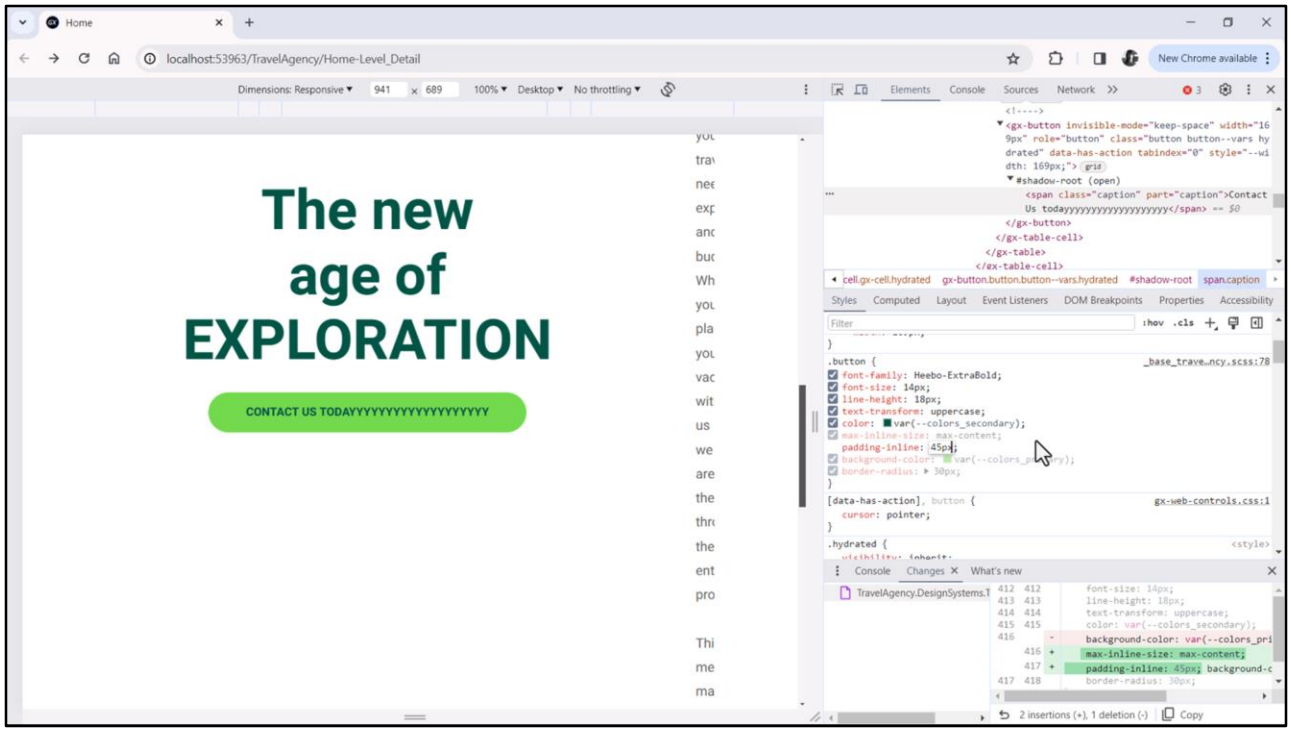
Bem, mas, perguntava, o que acontece então se eu adicionar a este Caption, se eu o tornar mais largo. Por exemplo…

Bom, o que estamos vendo é que o texto está se expandindo até o limite da largura do botão. E aí o que está fazendo é colocando reticências, ou seja, truncando, cortando o texto... e este não é o comportamento que vamos querer. O que vamos querer é que a borda do botão também se expanda, deixando um espaço nas laterais. Ou seja, não nos interessará a largura fixa de 169 dips (isso está muito bom para o texto "Contact us" exatamente, com essas dimensões, mas não se traduzirmos a aplicação).



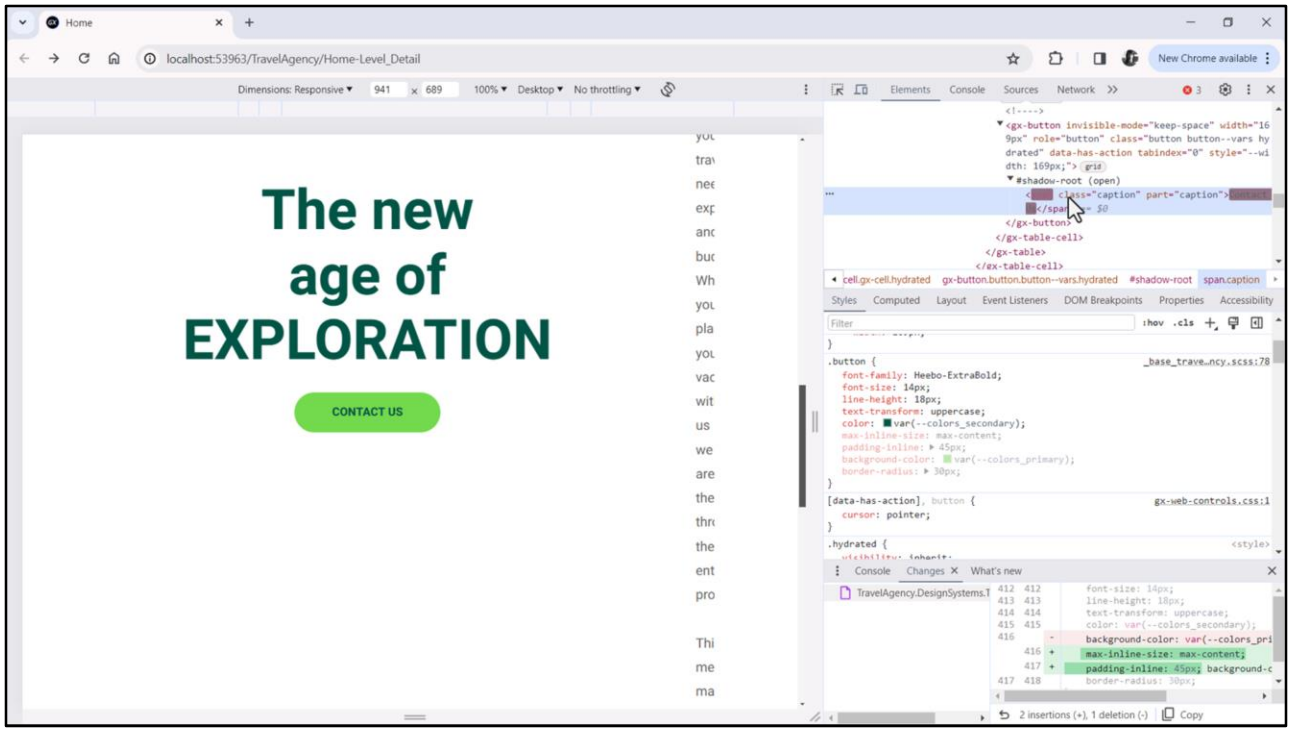
Portanto ... vamos testar o seguinte. Venho às propriedades da classe e vou adicionar uma propriedade max-inline-size, e vou escolher como valor dessa propriedade, max-content. Isto é, que o tamanho máximo inline, ou seja, na direção horizontal, corresponda ao conteúdo máximo.

E aqui estamos vendo: como o botão se expandiu para ocupar então todo o conteúdo.



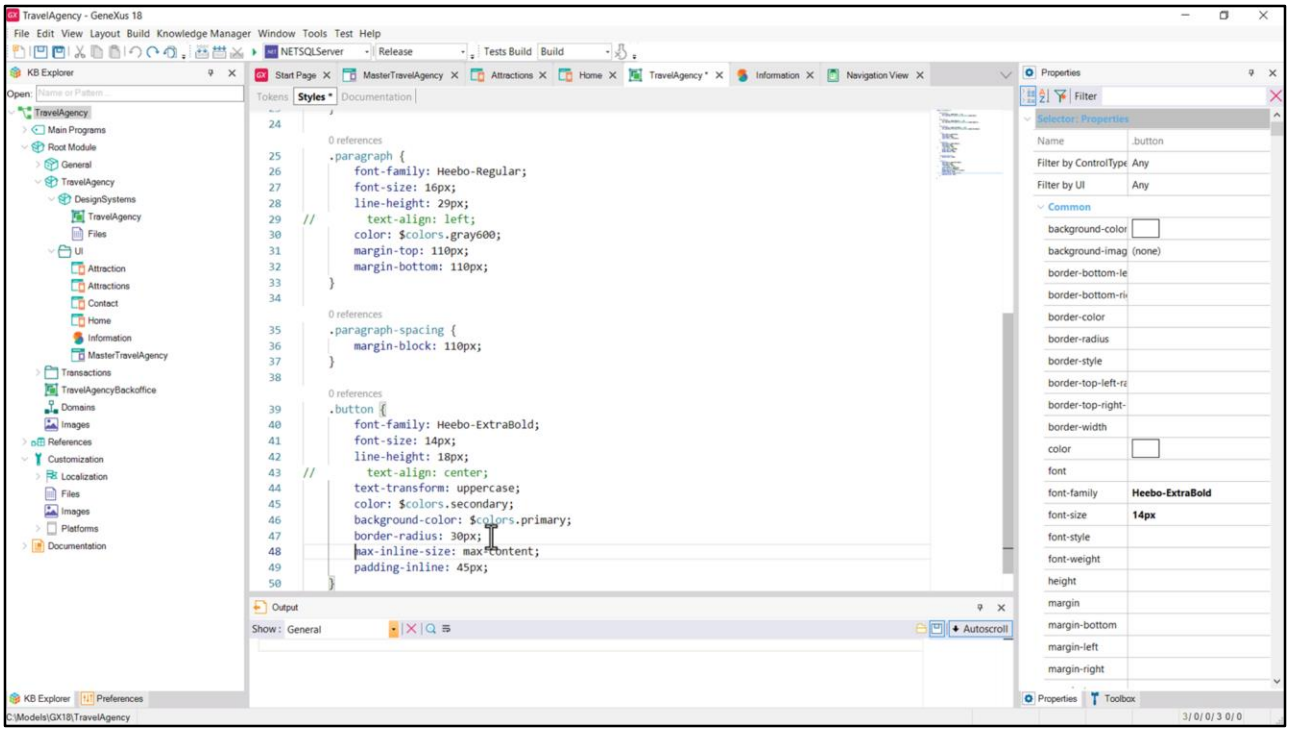
O que nos estaria faltando? Deixar um padding, que no conteúdo também esteja considerado um padding... padding de um lado e padding do outro. Então, por exemplo, vamos testar, padding-inline (para utilizar, novamente, como sugerimos antes, as propriedades lógicas em vez das físicas), e por exemplo colocamos 45 pixels. E aí o vemos.

Então, está tendo 45 pixels do início, 45 pixels em relação ao fim e a largura do botão está sendo a que necessita para que todo esse conteúdo esteja envolvido.

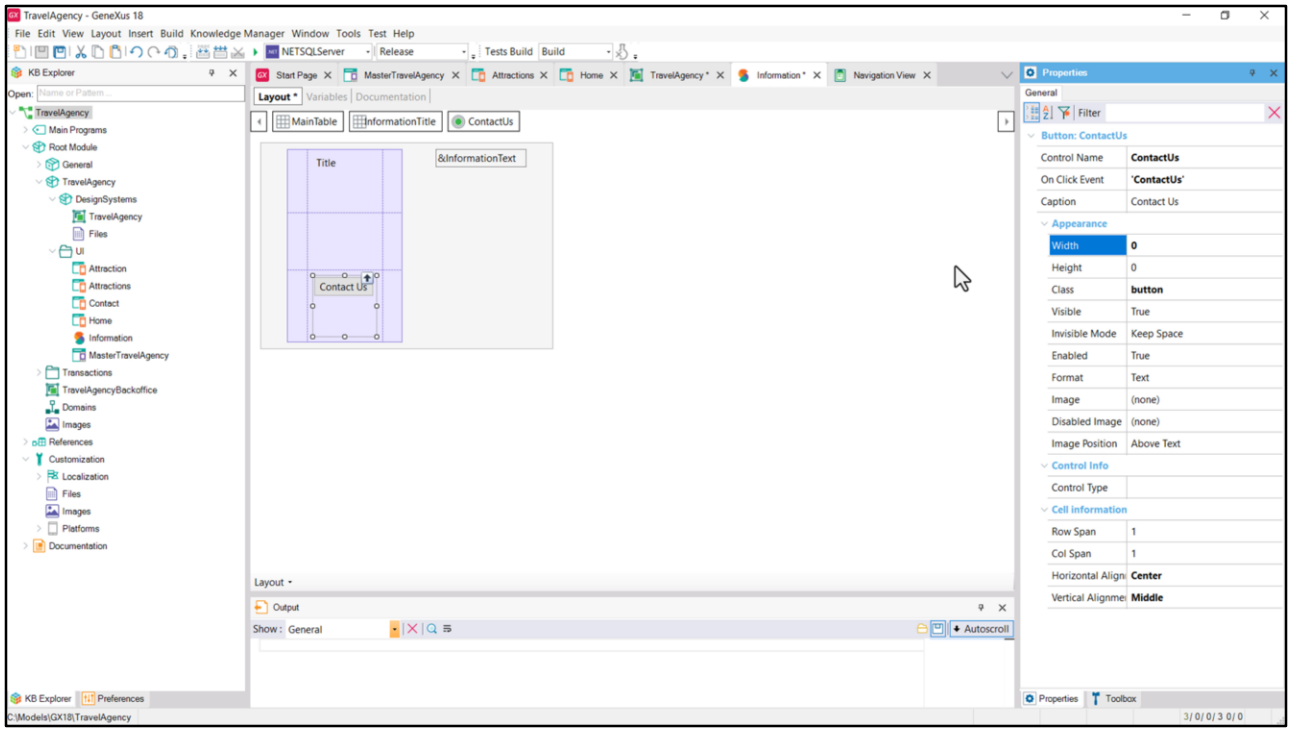


De fato, observemos que se agora voltarmos a deixar o caption como queríamos, Contact Us, é imperceptível a diferença com o que tínhamos antes.

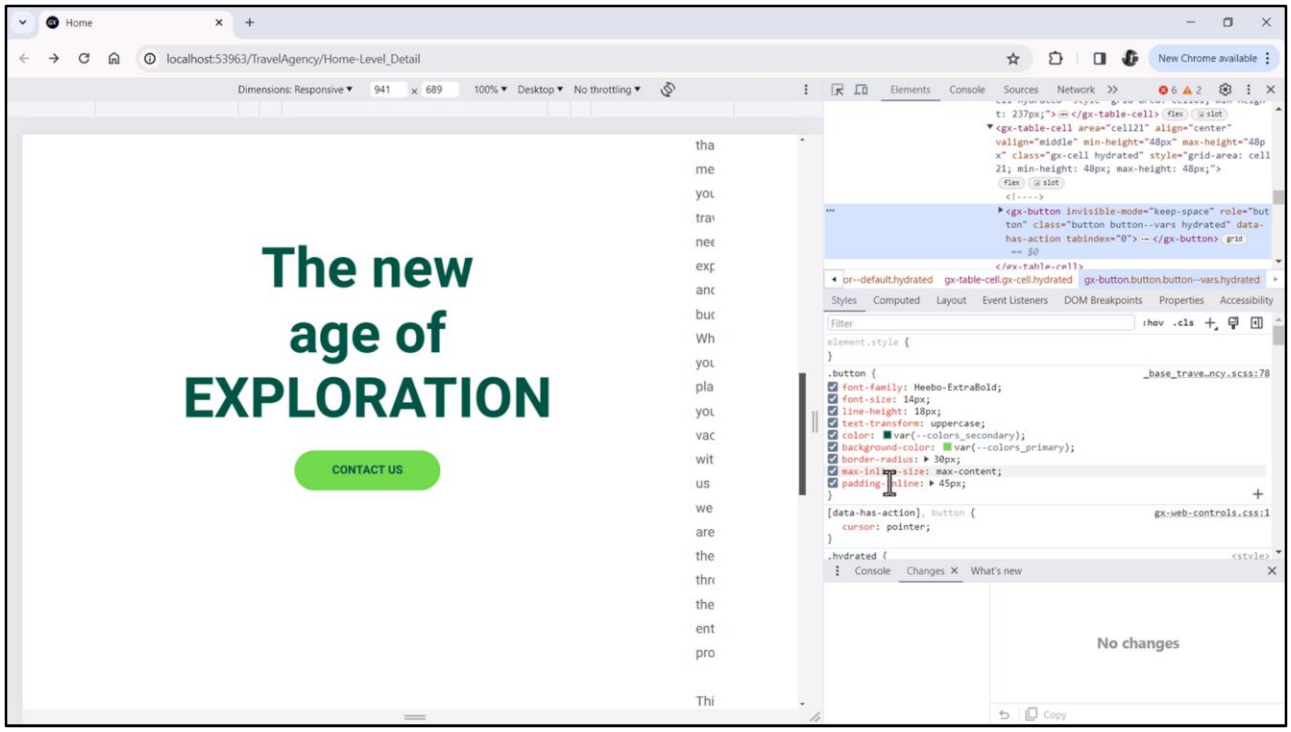
No entanto, esta forma vai permitir que se traduzirmos a aplicação e mudarmos a largura do texto, o botão será redimensionado para que sempre fique bom.



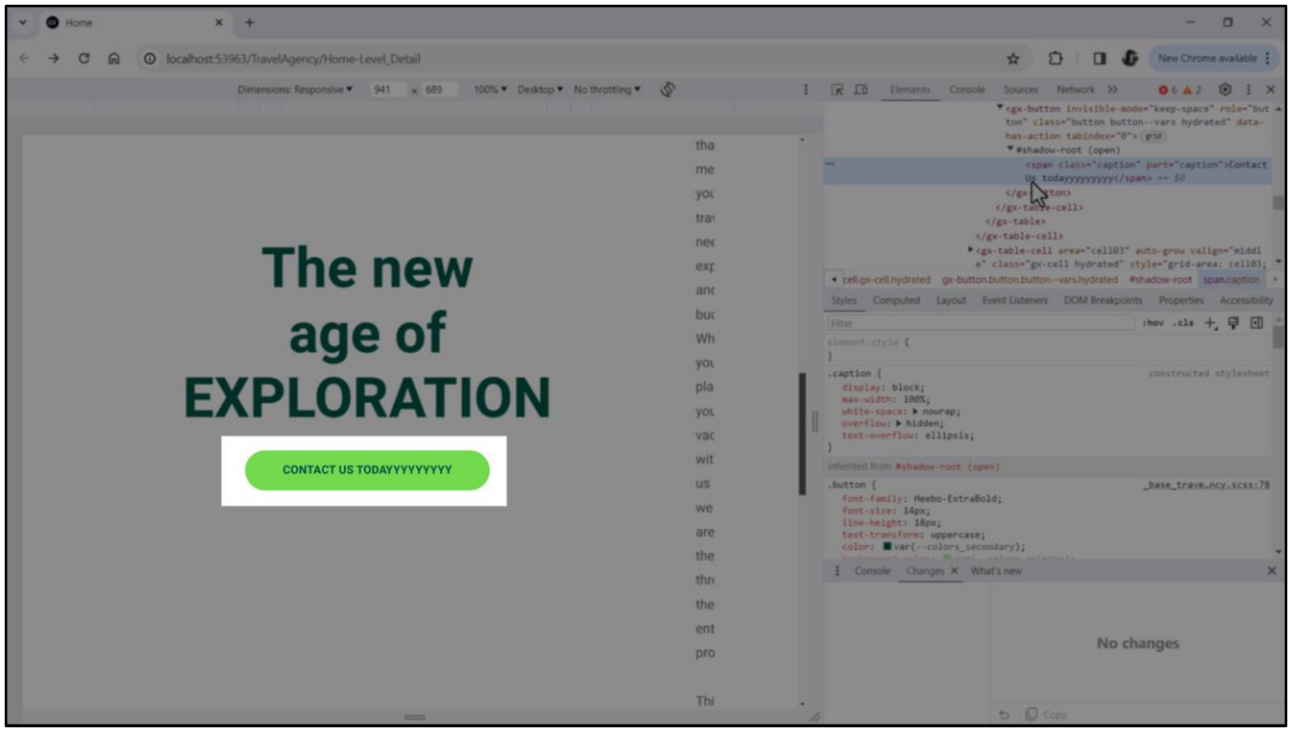
Bem, então vamos levar estas duas propriedades para nosso DSO.



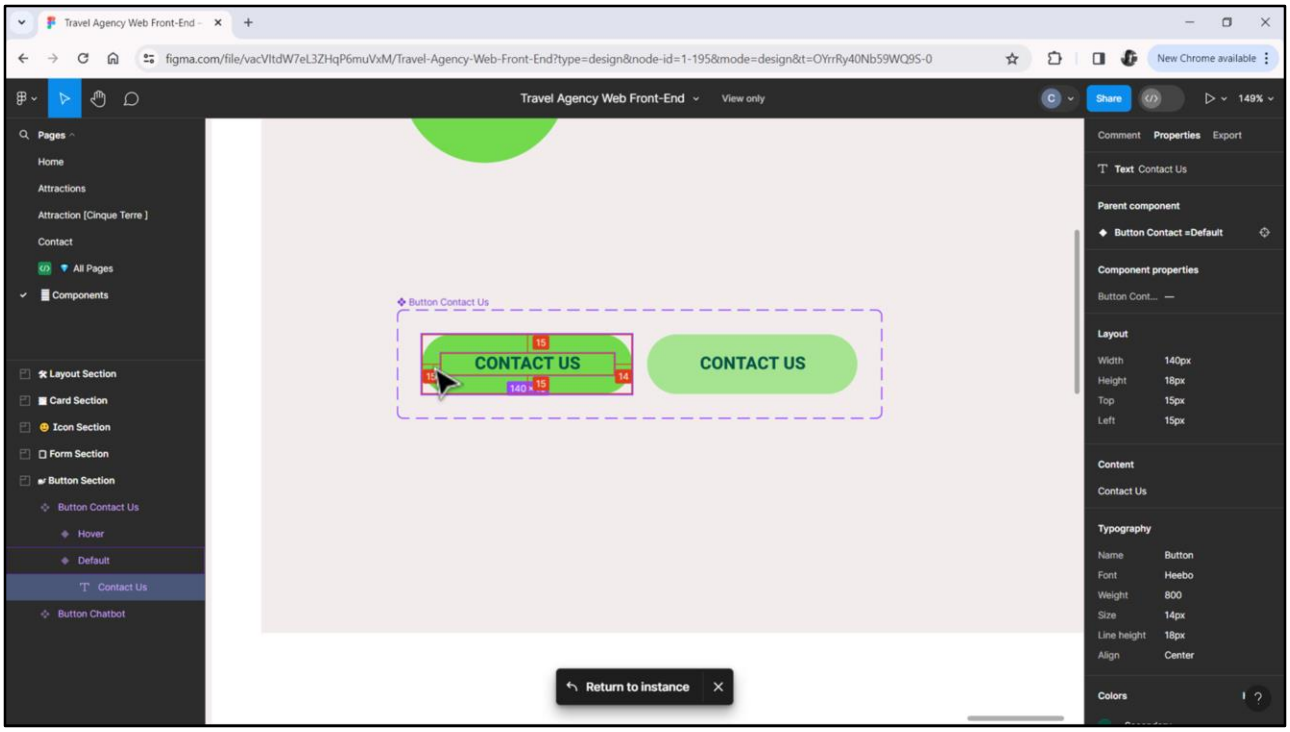
E vamos remover a propriedade Width, deixemos o default no nível do controle do botão no stencil.



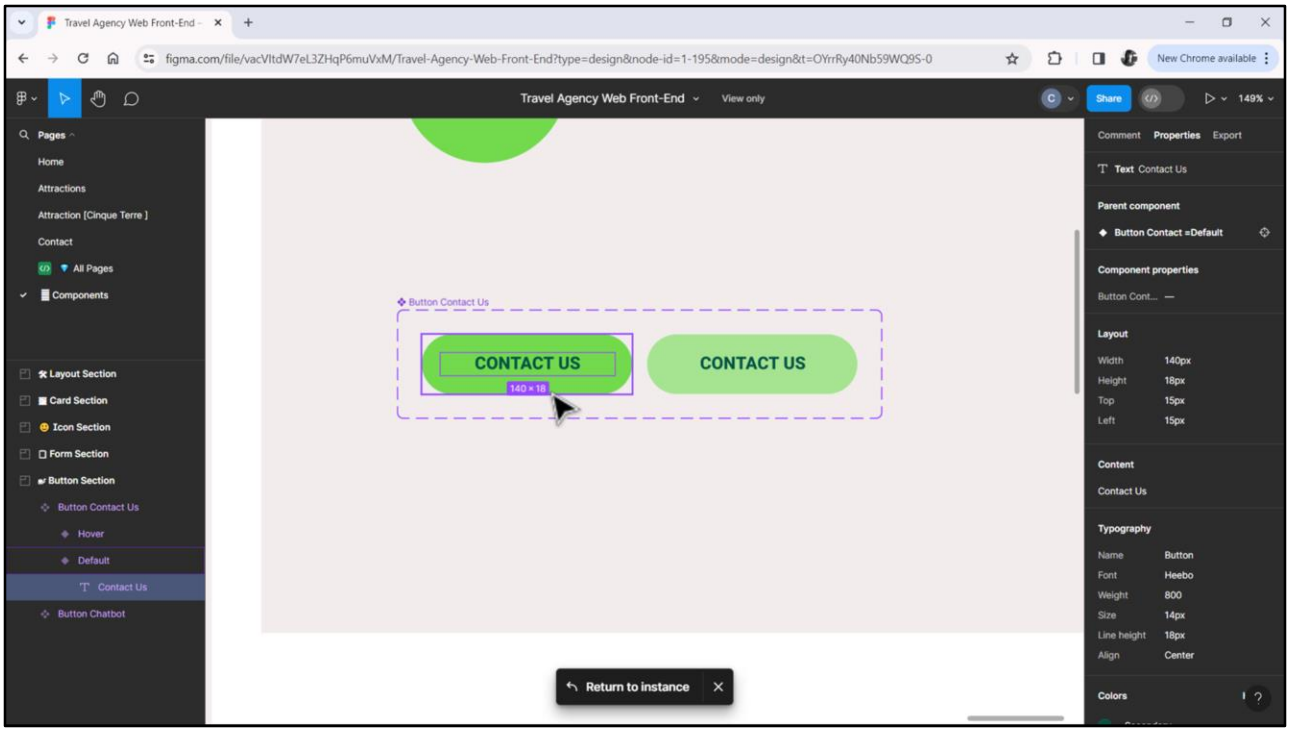
E agora vamos testar. Bem, como dissemos, é imperceptível... mas agora vemos incorporadas as propriedades na classe, não vemos a width, e se modificarmos o caption…



...vemos como automaticamente se redimensiona, agora sim, como desejávamos.

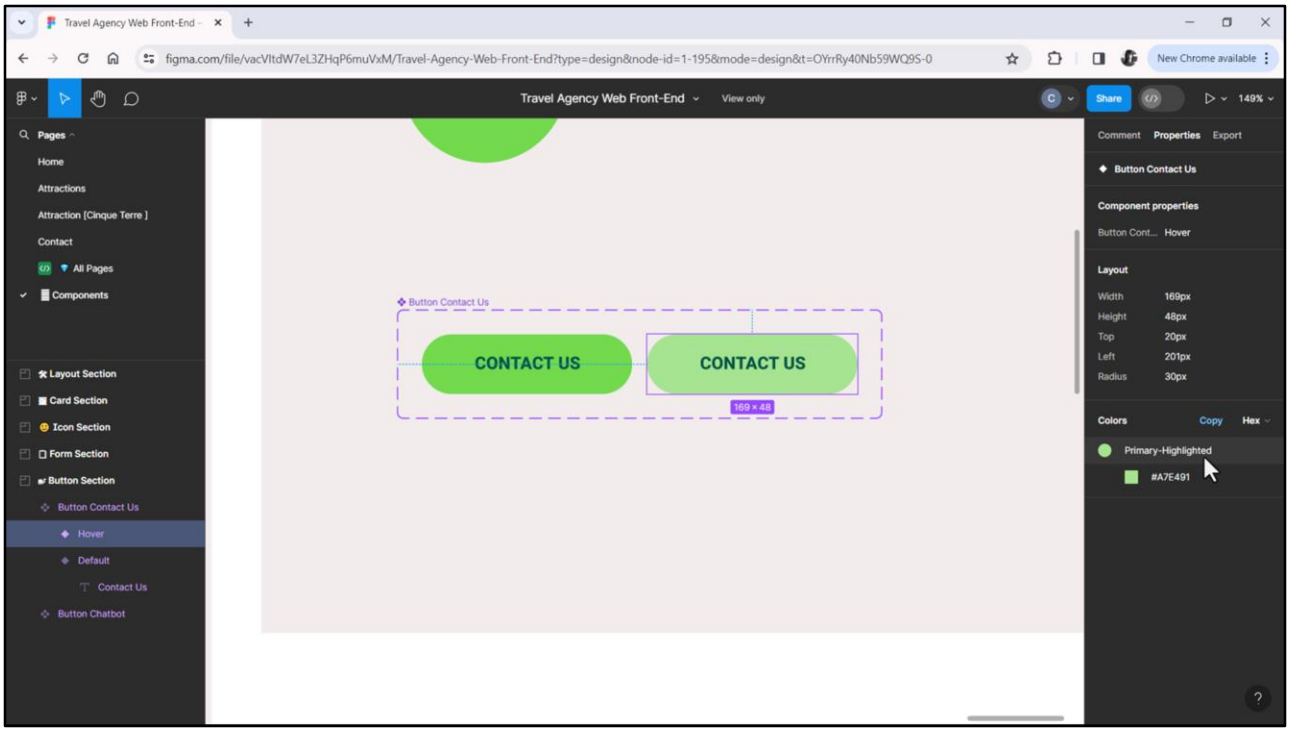


Poderíamos ter percebido algo disto no arquivo em Figma? Bem, assim como foi expresso, não. De fato, se observarmos, Chechu definiu a caixa de texto como ocupando 140 pixels de largura, ou seja, uma largura fixa, deixando para as laterais 15 pixels de um lado e 14 pixels do outro. Também podemos ver 15 pixels acima e 15 pixels abaixo, que nós não consideramos. Por quê? Porque demos ao botão a altura de 48 pixels, que corresponde à soma destes 15 mais estes 15 mais…



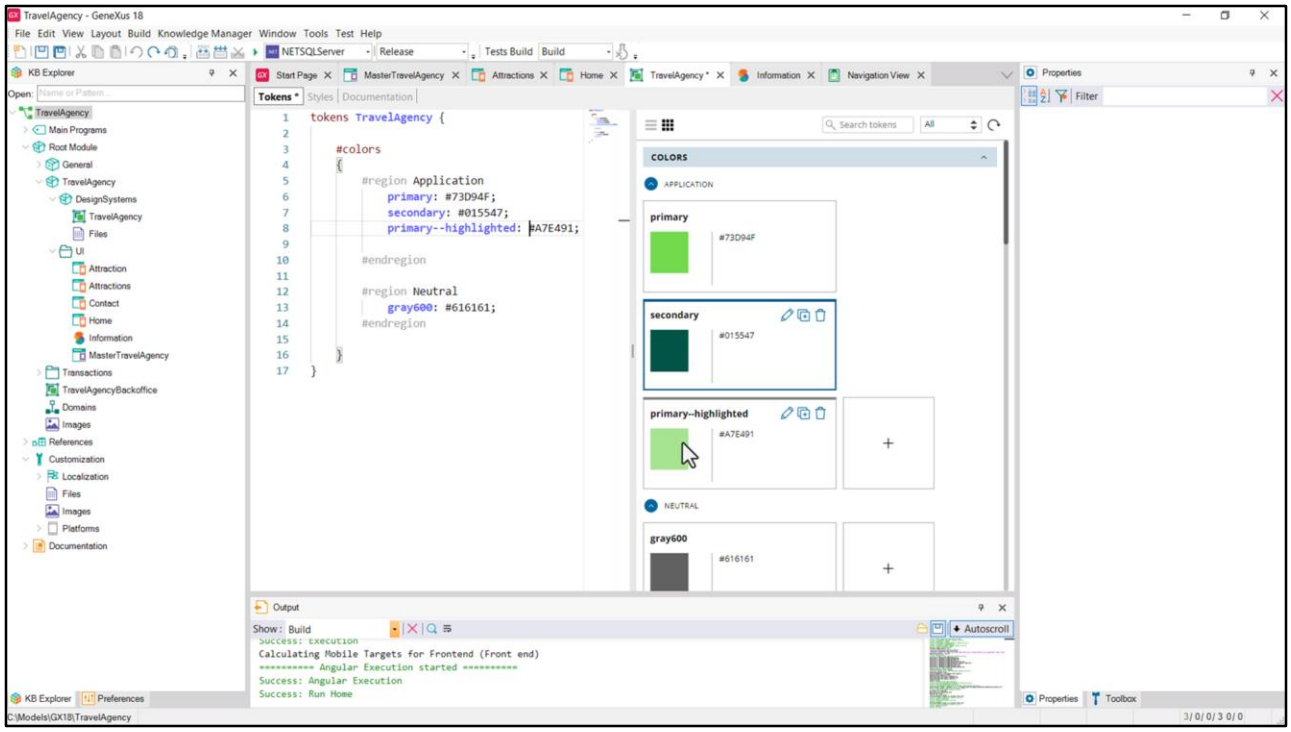
... estes 18 daqui do texto. Estes, não consideramos porque o botão não vai crescer para cima ou para baixo de acordo com a tradução. O que a tradução fará é, em todo caso, expandir para a direita e esquerda, mas não no outro sentido.

Em resumo, no nosso caso, nossa implementação teria correspondido a um design em Figma no qual esta largura fosse hug, ou seja, envolvesse o conteúdo, e também tivesse um padding-left e um padding-right, se vou dizer em termos físicos, desses 45 pixels que fornecemos.

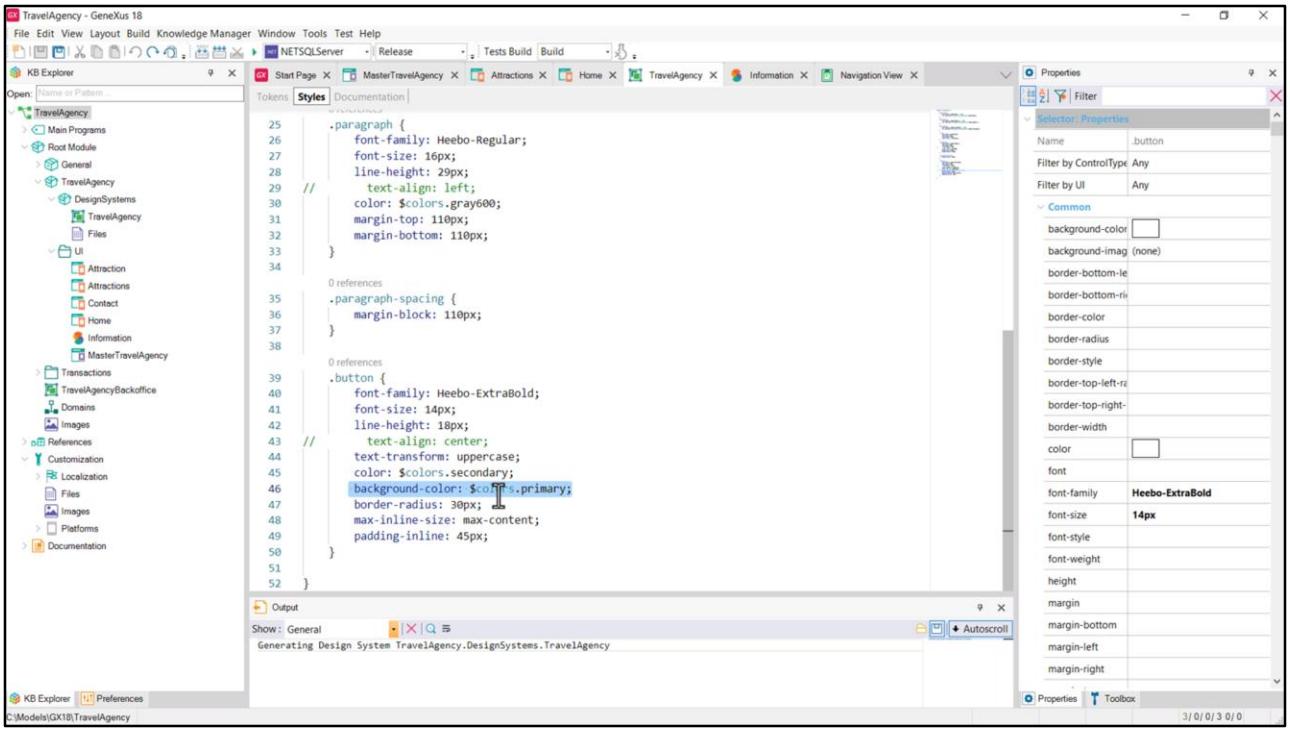


Bem, então agora estamos em condições de estudar como fazemos para que o botão mude sua aparência quando é feito hover sobre ele, também poderia ser quando clicamos nele, quando o ativamos, para que assuma este estilo, que, pelo que vemos, a única coisa em que difere do estilo default é na background-color. Que assume esta cor que Chechu deu este nome, Primary-Highlighted.

Então a copio...

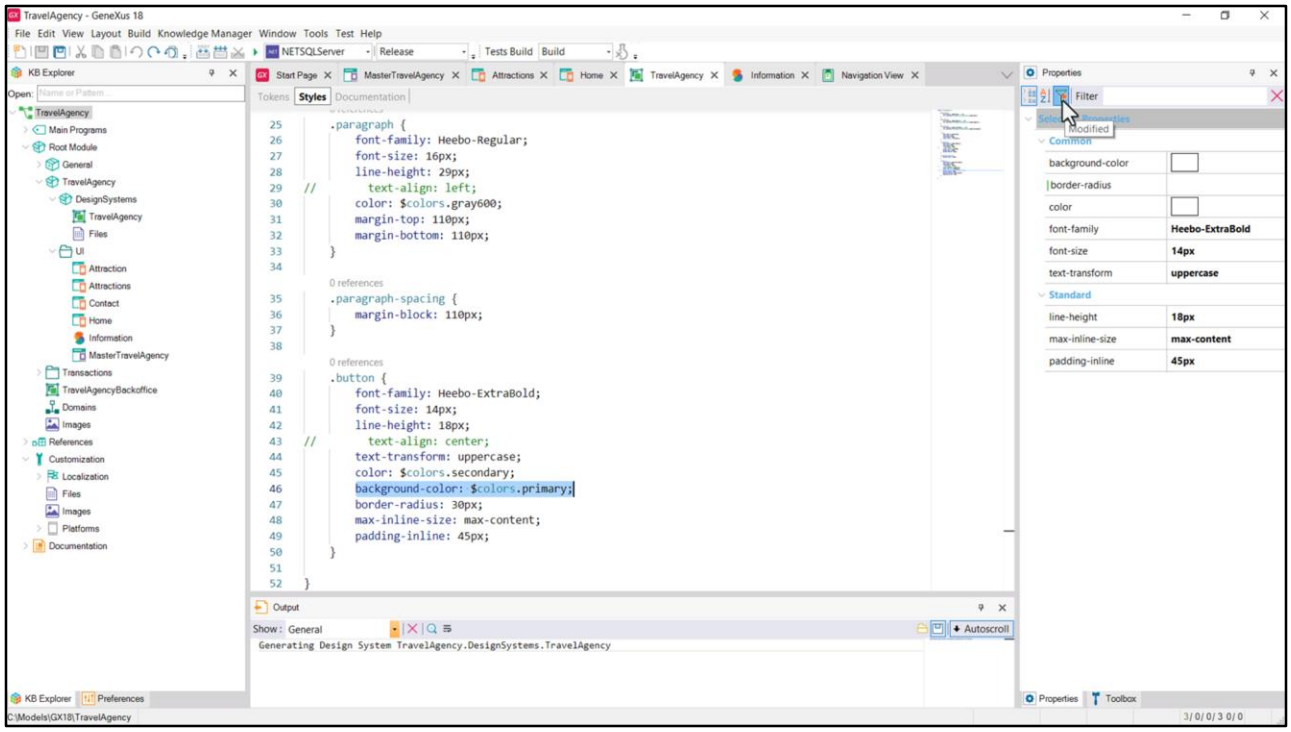


…e na seção de tokens eu adiciono…



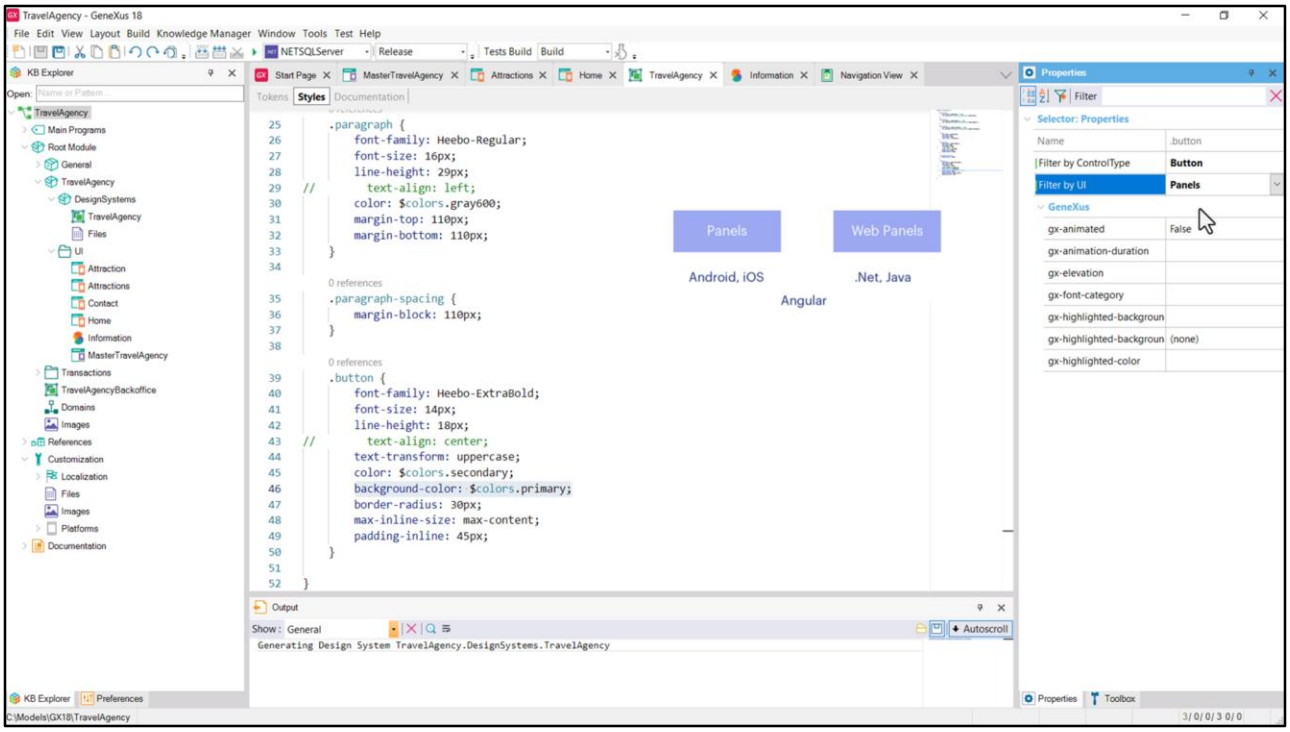
E agora o que tenho que ver é como faço para que a classe associada ao botão modifique sua propriedade background-color para assumir o token primary—highlighted quando é ativado o botão.

Notemos que, de certa forma, estamos falando de comportamento, certo? Quando o botão é ativado, precisamos mudar uma das propriedades da classe que comanda seu estilo.



Pois bem, acontece que as classes não apenas suportam a utilização de propriedades CSS, mas também propriedades específicas de GeneXus.

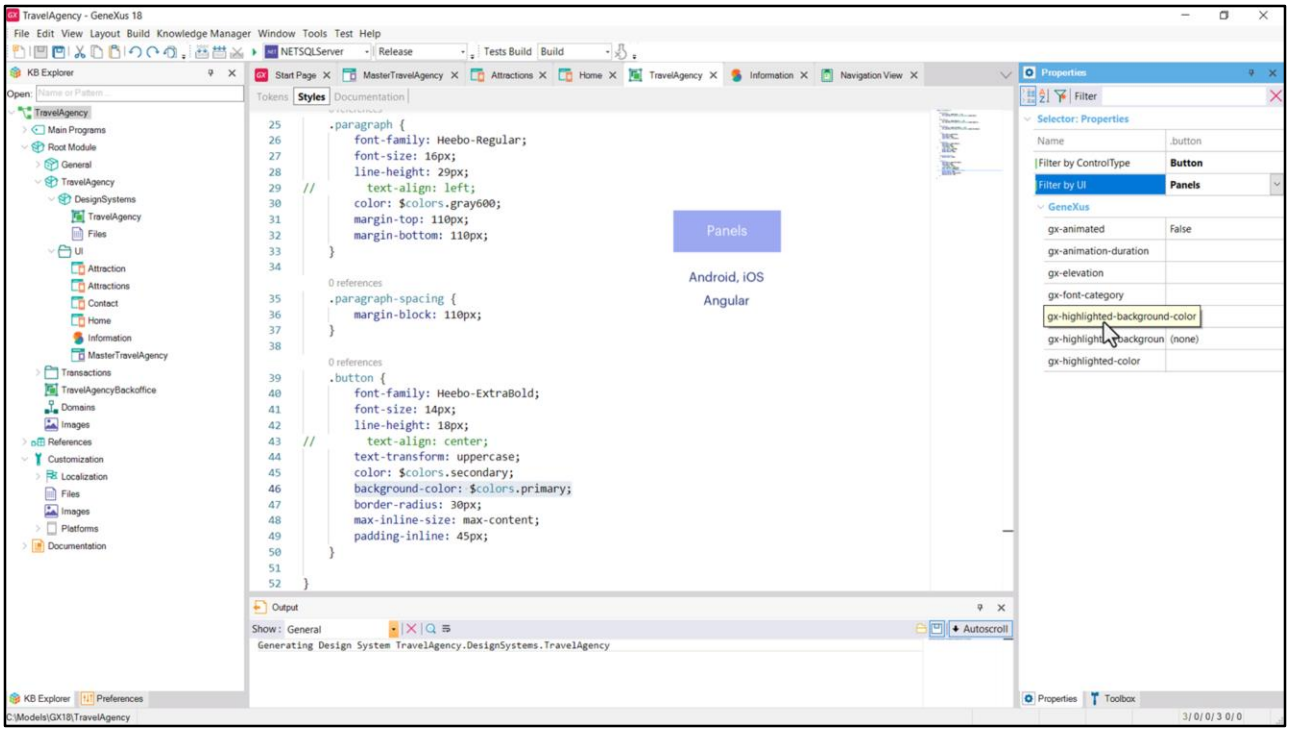
Assim, por exemplo, se formos para a janela de propriedades... bem, aqui podemos filtrar as propriedades que estamos utilizando no nível desta classe…



...mas também vemos que temos estes dois filtros para poder visualizar não todas as propriedades, mas, por exemplo, aquelas que correspondem a controles do tipo botão. Ao fazer isso, vai mostrar-me apenas as propriedades válidas para esse tipo de controle.

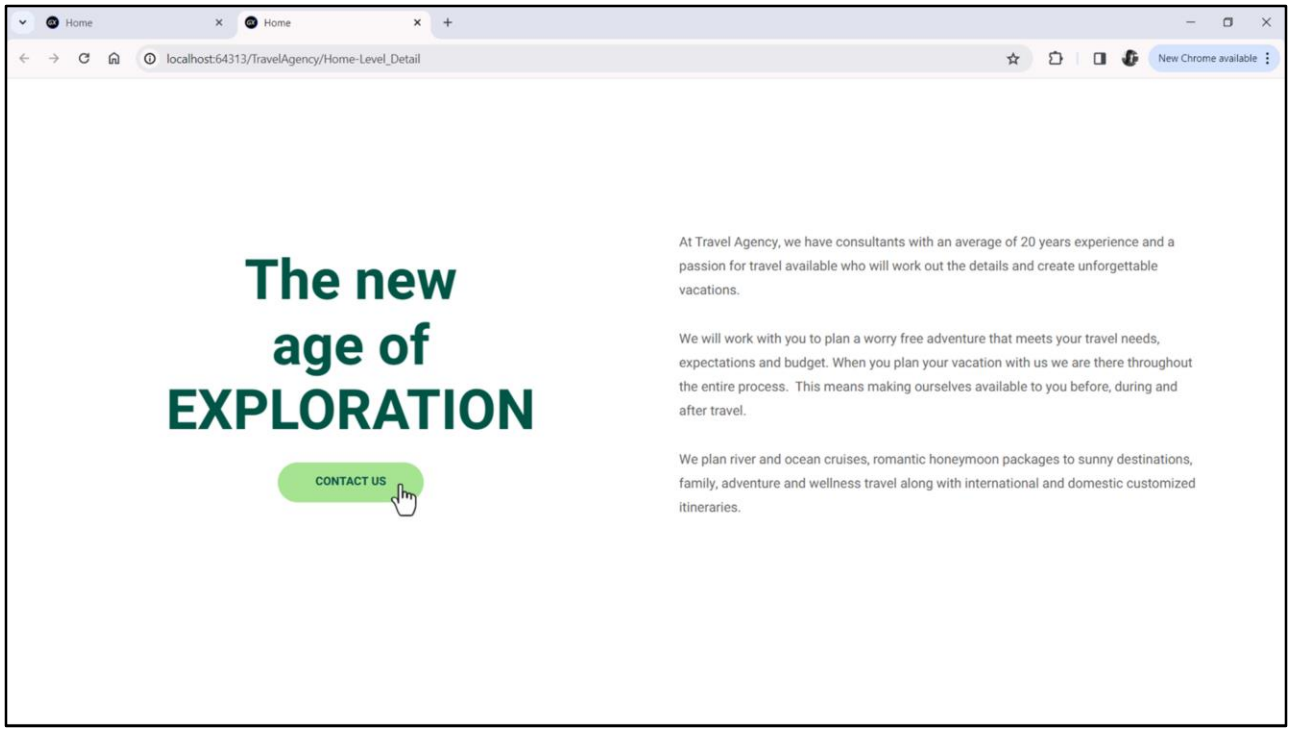
E, por outro lado, também posso filtrar pelo tipo de interface de usuário... ou seja, ou todas, que é o que estou filtrando agora... ou poderia escolher o mundo Panels ou o mundo Web Panels…

Se filtro pelo mundo Panels, que é no qual estamos agora... (lembre-se que o mundo Panels surgiu quando os dispositivos inteligentes, ou seja, quando as aplicações nativas para telefones e tablets; depois Angular foi incorporado a este mundo... Mas Angular é um mundo intermediário, é um híbrido entre o mundo Panels e o mundo Web Panels e isso vamos ver agora num instante)…



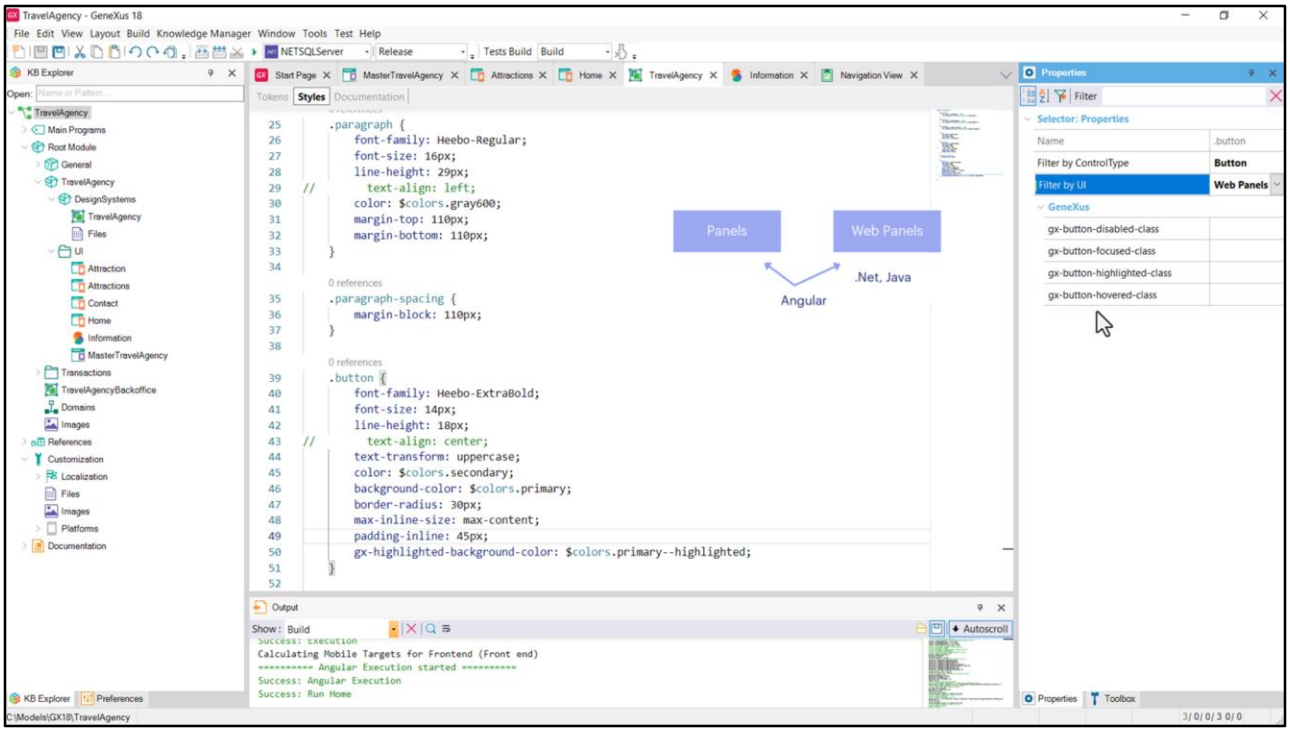
Mas bem, agora estou filtrando por interface de usuário Panels. E aqui vemos que estão aparecendo estas propriedades GeneXus, (gx, hífen e um nome de propriedade)... e bem, estas são as que eu poderia eventualmente utilizar dentro da classe, de uma classe que vou associar a um controle do tipo botão em um panel... e bem, são classes, que cada uma tem o seu comportamento.

Vemos entre as classes esta: gx-highlighted-background-color, que é coincidentemente uma propriedade que nos serve justamente para isto que estamos buscando... vou colocar como cor a do token de cor primary--highlighted. Vamos testar...



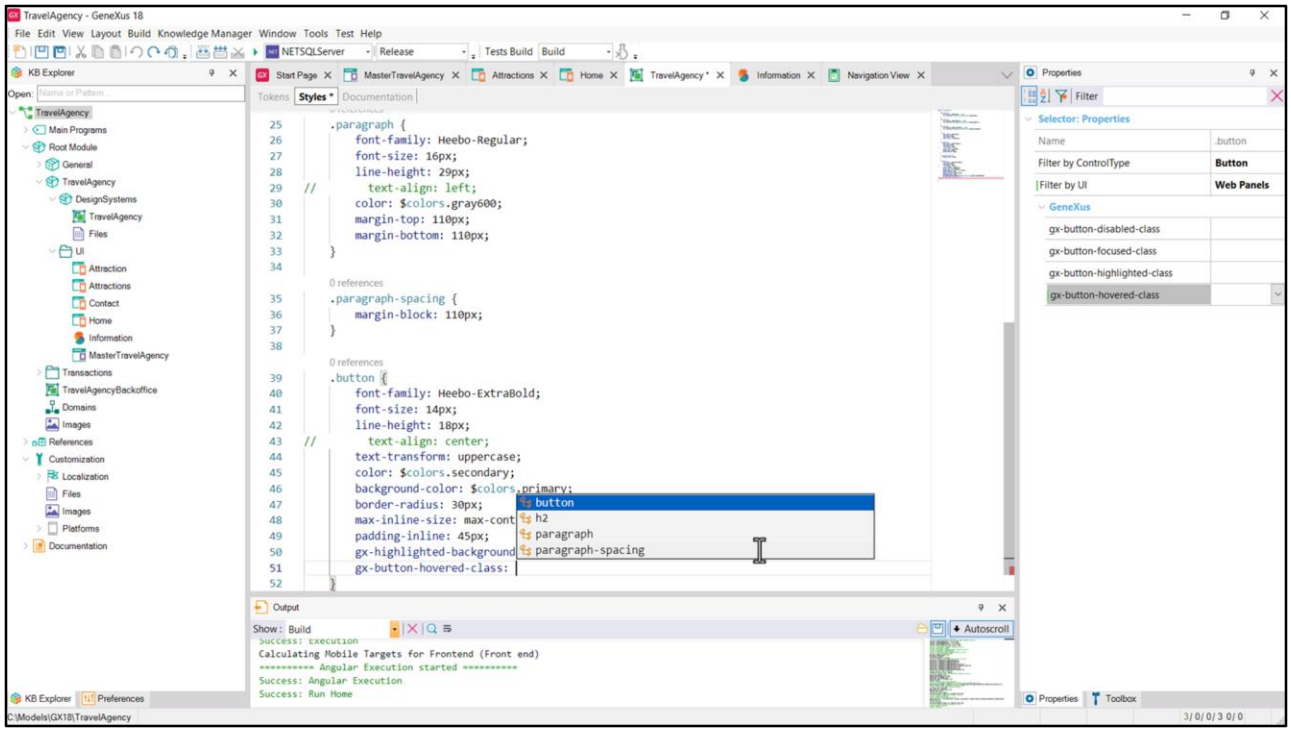
E aqui vemos que quando pressionamos, aparece efetivamente a background color em outra cor. Estou pressionando com o mouse, não soltei porque quando soltar vai chamar o panel Contact que não tem nada programado por enquanto, por isso vemos isso.

E se quiséssemos que isso acontecesse ao fazer hover e não apenas ao ativar o botão?

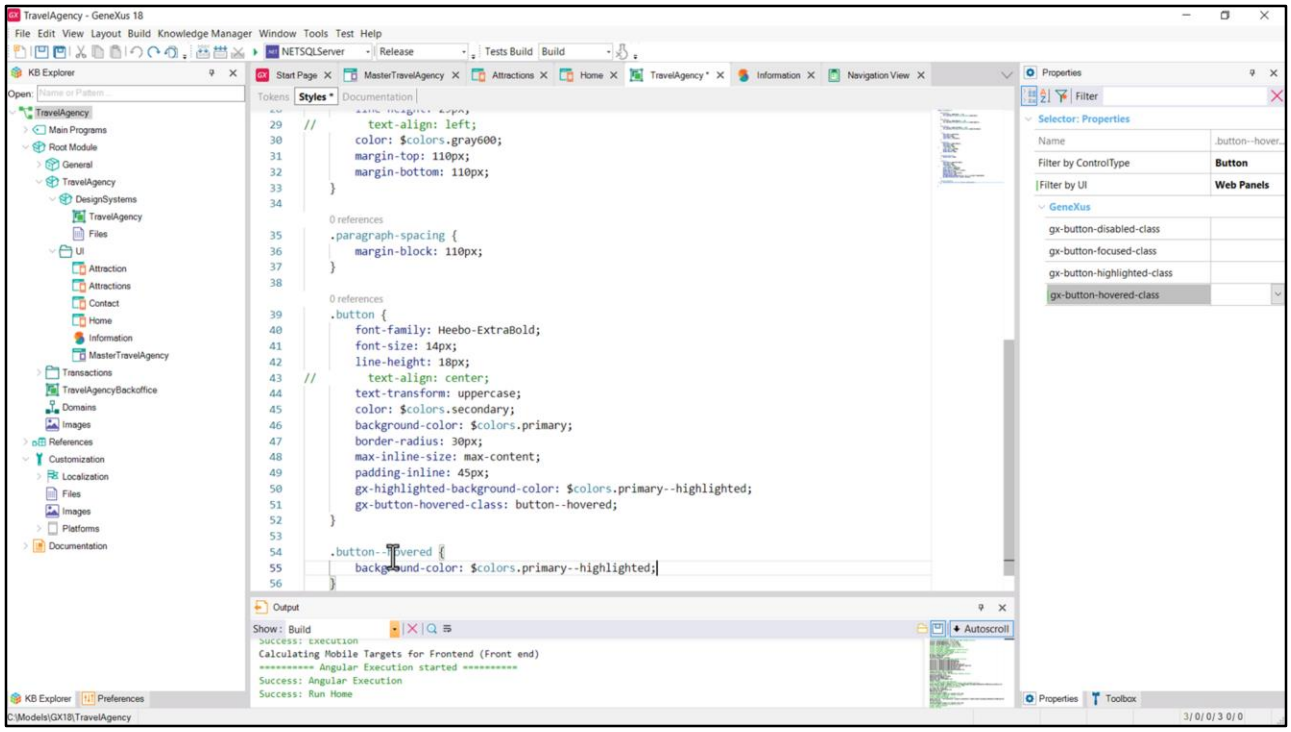


Se viermos procurar uma propriedade gx-hover-background-color, não a encontramos. Ou seja, as propriedades aplicáveis a um botão dentro de um panel são estas e não incluem essa. É que, insisto, estas propriedades surgiram com o mundo dos panels para aplicações nativas: Android, iOS.

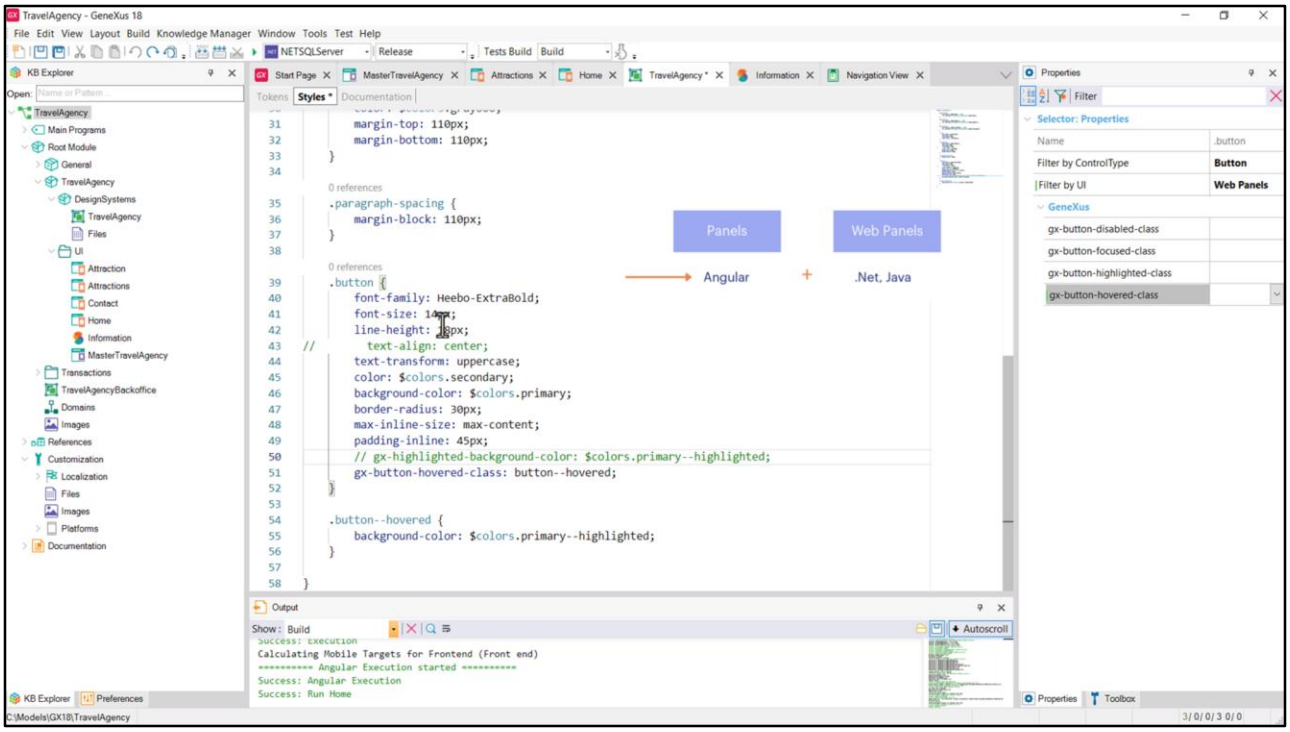
No entanto, observemos o que acontece quando mudo o filtro para o de Web Panels. Vemos que aparecem estas outras propriedades associadas aos botões, que incluem esta hovered. Estas propriedades, o que estão indicando - vejam que terminam com class - o que estamos indicando se escolhermos esta propriedade é que…



… vamos fazer isso… estamos indicando qual é a classe que vai comandar o estilo do botão que tem essa classe associada quando for feito hover sobre ele. Observemos que estão sendo oferecidas as 4 classes que este DSO tem especificadas até o momento.



Vamos criar uma nova classe que vamos chamar de button--hovered, e nela... é onde vamos especificar que a background-color seja a primary--highlighted.



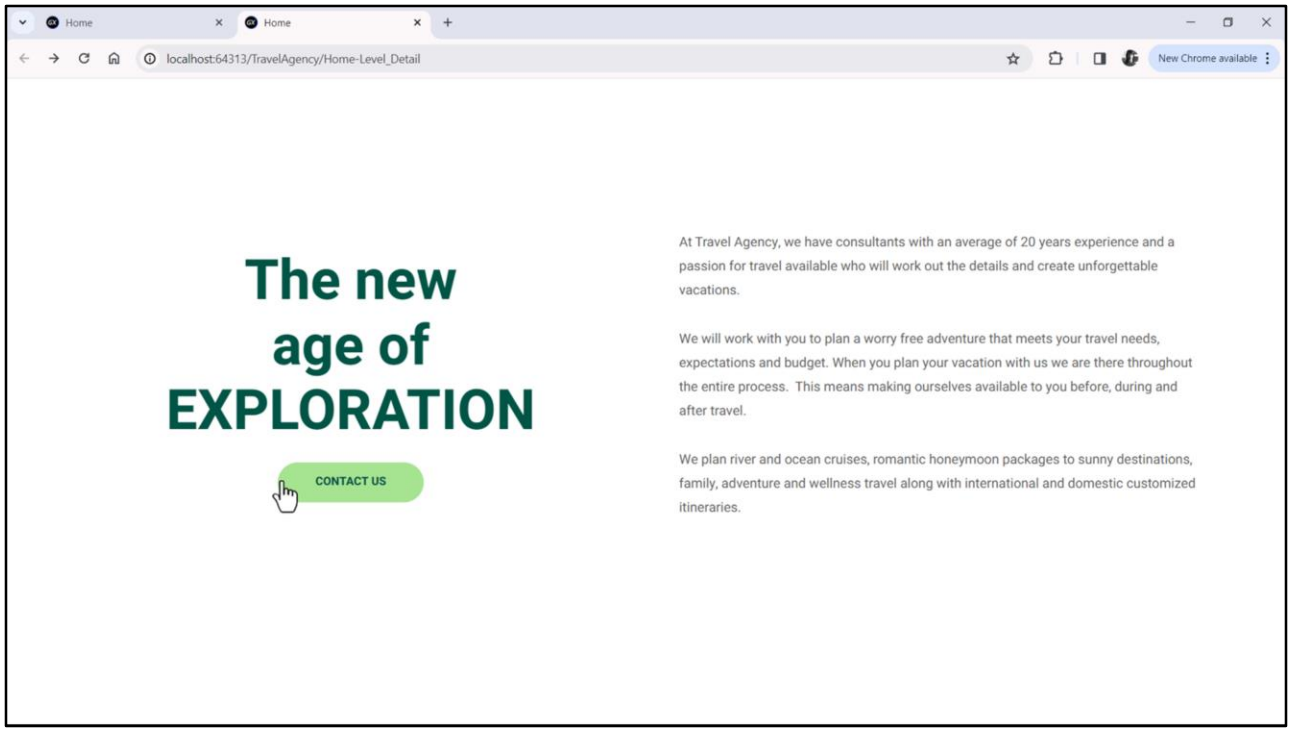
Ou seja...

Vou deixar primeiro comentada esta, para que seja bem observada a diferença. Estou utilizando uma propriedade GeneXus que poderíamos, à primeira vista, pensar que não terá efeito sobre nosso panel porque estamos no mundo dos Panels e não dos Web Panels. No entanto, como trata-se de propriedades que têm a ver com o mundo web, serão levadas em consideração.

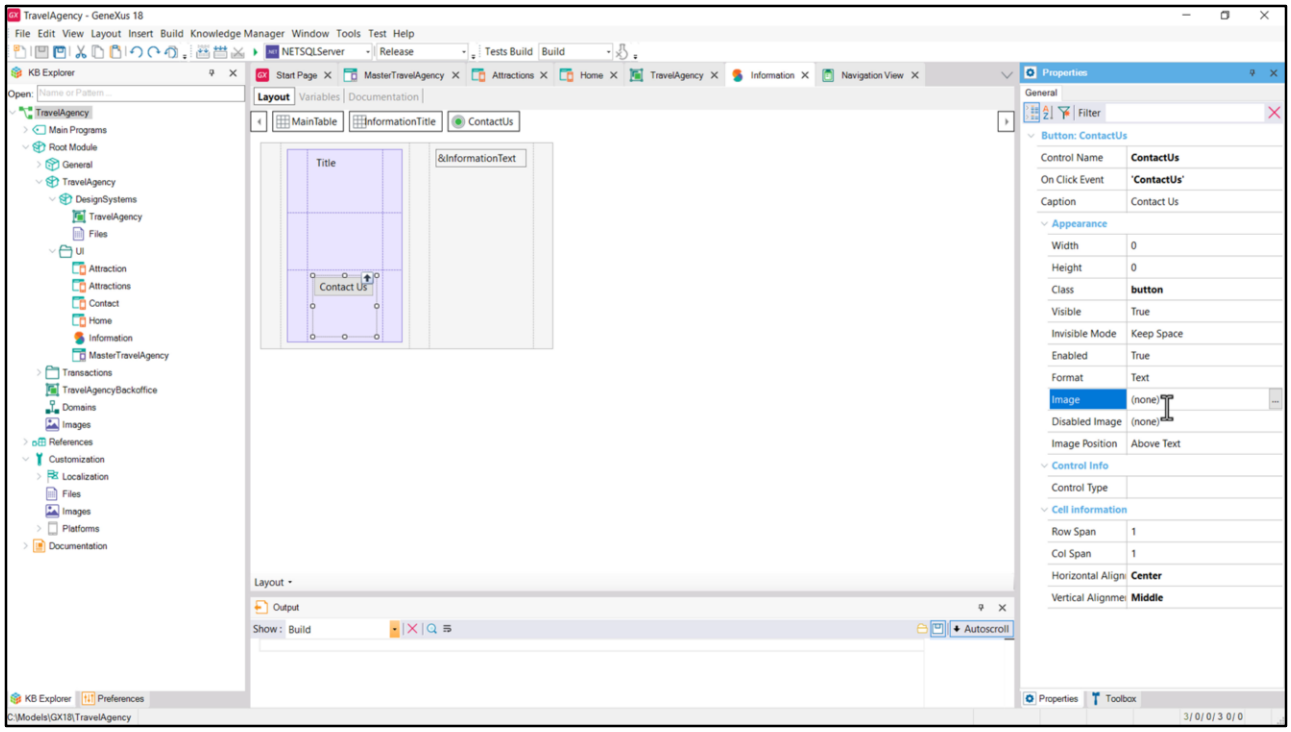
Ou seja, vou poder utilizar para um panel executado para Angular, não para este panel executado para Android ou para iOS, para Angular vou poder utilizar estas propriedades também.

Então, o que estou dizendo aqui, vou resumir é: quando ao botão que tem essa classe associada for feito hover, então, essas propriedades que estão aqui, correspondentes à classe do botão vão ser adicionadas ou sobrescrever aquelas que estão dentro desta outra classe, que temos aqui selecionada. E o que estamos indicando dentro da classe é que sobrescreva esta propriedade, a background-color. Poderíamos adicionar novas propriedades que não sobrescrevam estas, mas que simplesmente adicionem novos comportamentos de estilo e bom, aí seriam adicionados. Neste caso, como se chama igual a uma que já está definida aqui, bom, será sobrescrita.

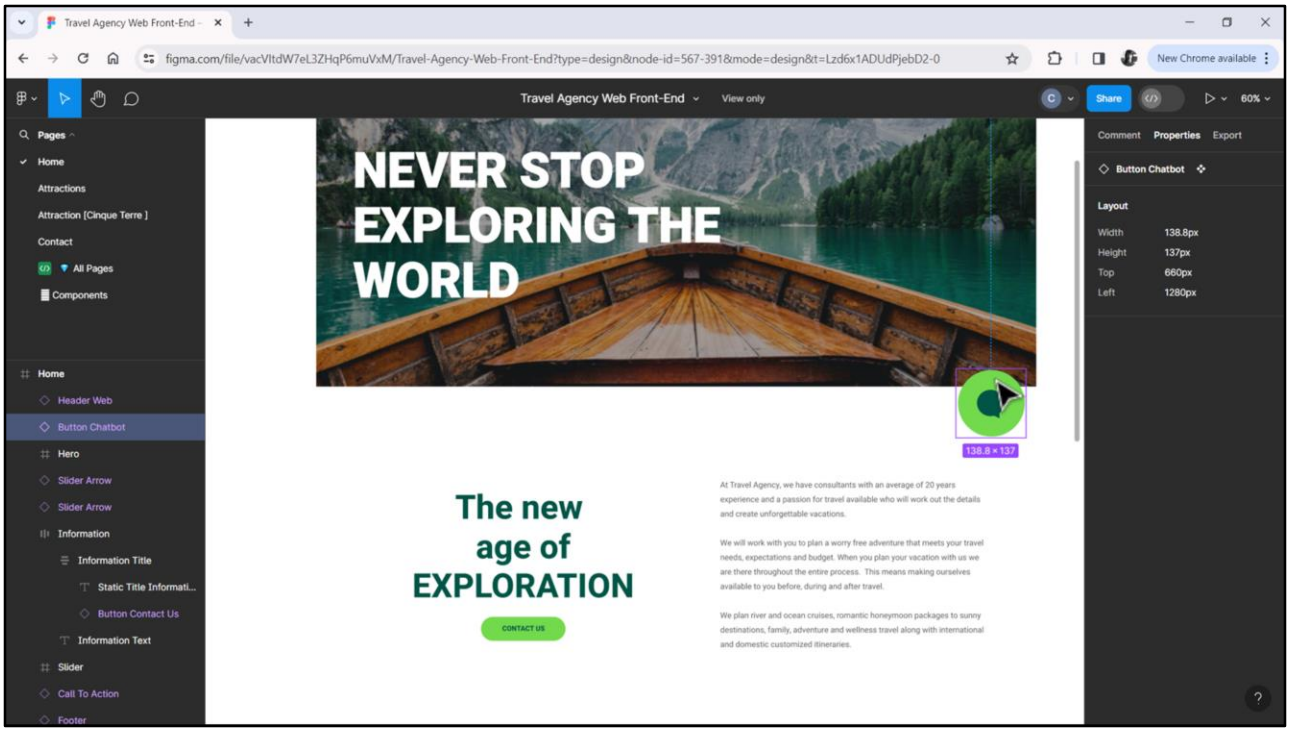
Vamos testar.



Bem, agora vamos ver o que acontece quando passo o mouse sobre o botão.

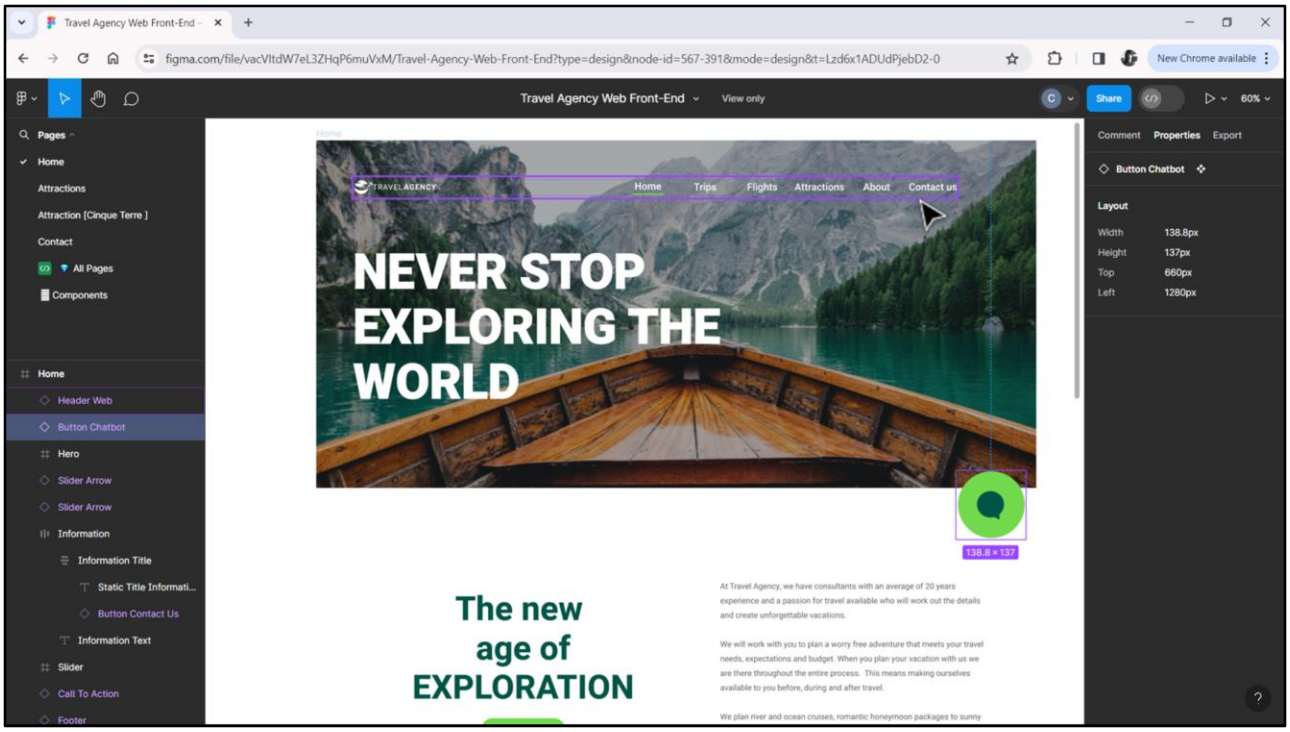


Bem, para terminar, comentamos que um botão pode ter um texto, uma imagem, ou ambos.



Um caso em que vamos ter um botão com imagem e sem texto será o do chatbot, que vamos implementar quando implementarmos o Master Panel.

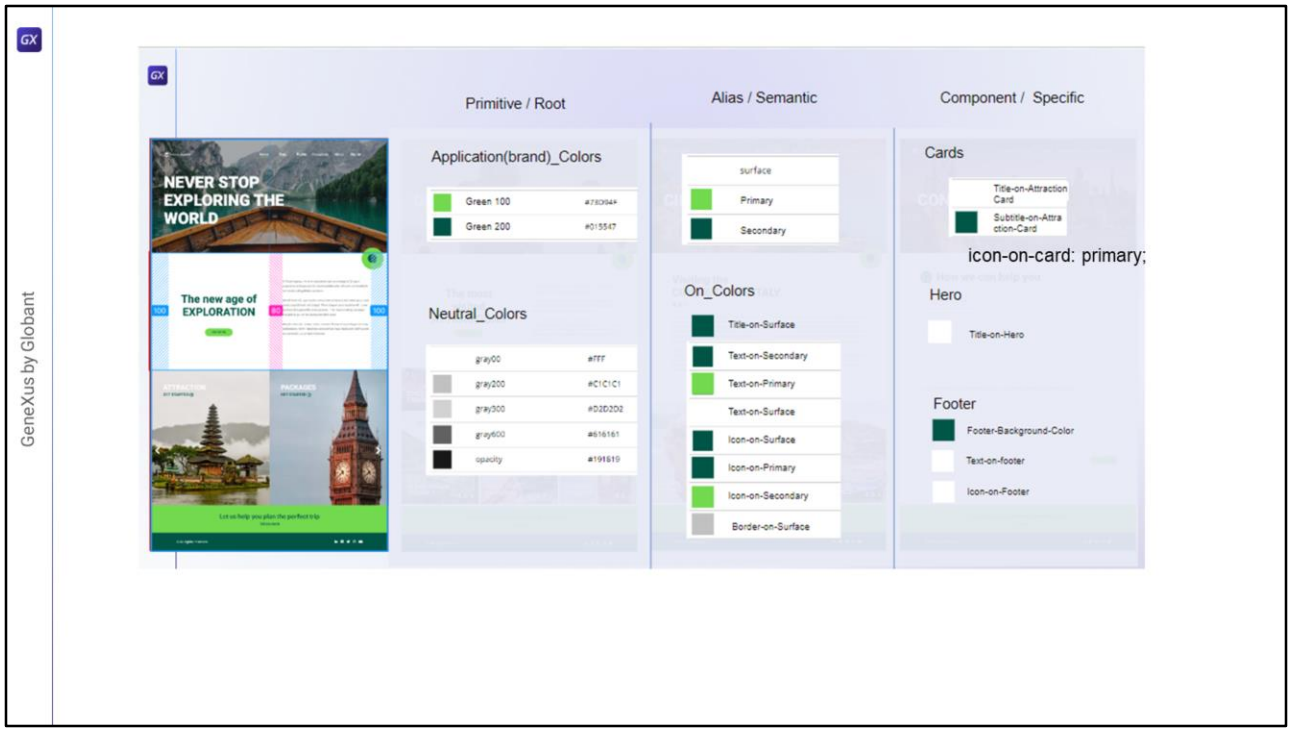
Mas aqui vemos isso que parece uma imagem, na verdade será um botão com uma imagem. Por que será um botão? Já adianto. Porque todo elemento clicável, ou seja, que vai responder a uma ação, deve ser implementado como um botão.



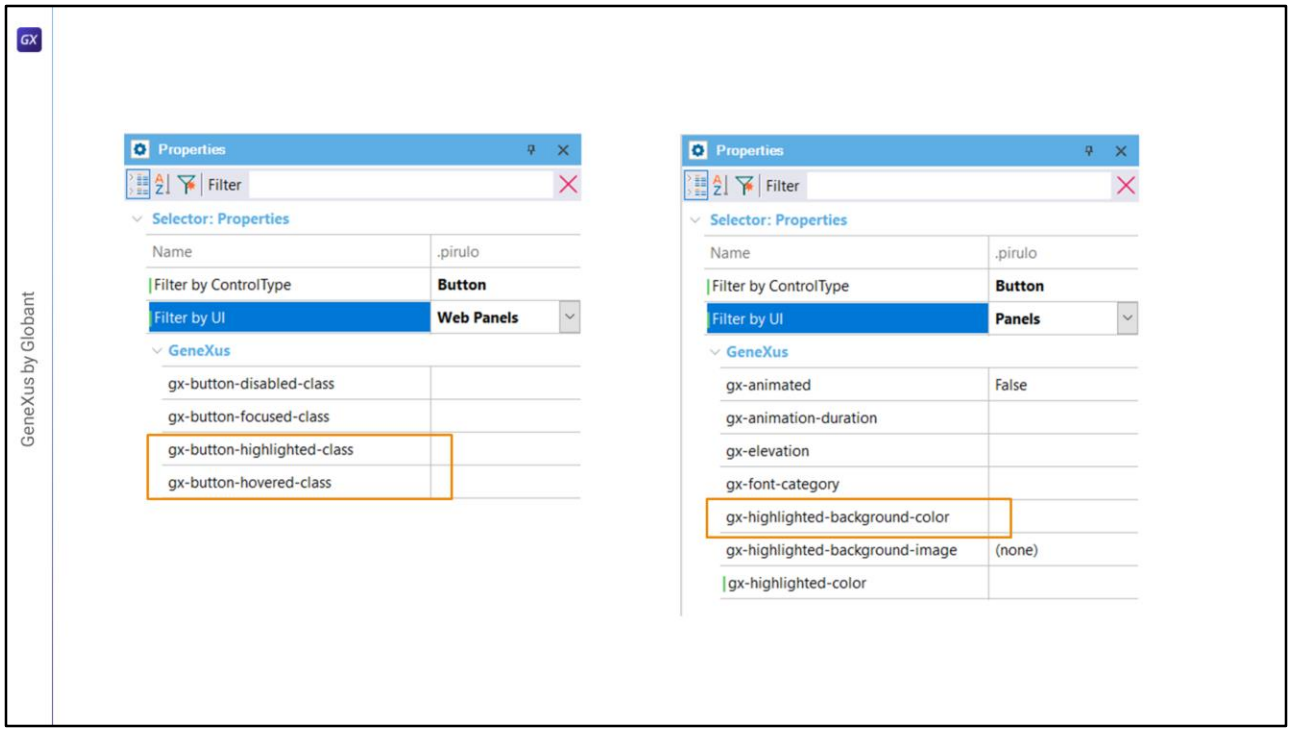
Assim, por exemplo, os elementos do menu, que irá tratar-se de botões que terão texto e não terão background, ou seja, o background será transparente. E por que isso? Por questões de acessibilidade, como veremos oportunamente.

Bem, espero vocês agora no próximo vídeo.


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



Agora sim, vamos ver como fazer com que o botão tenha a aparência que queremos.



Agora sim, vamos ver como fazer com que o botão tenha a aparência que queremos.