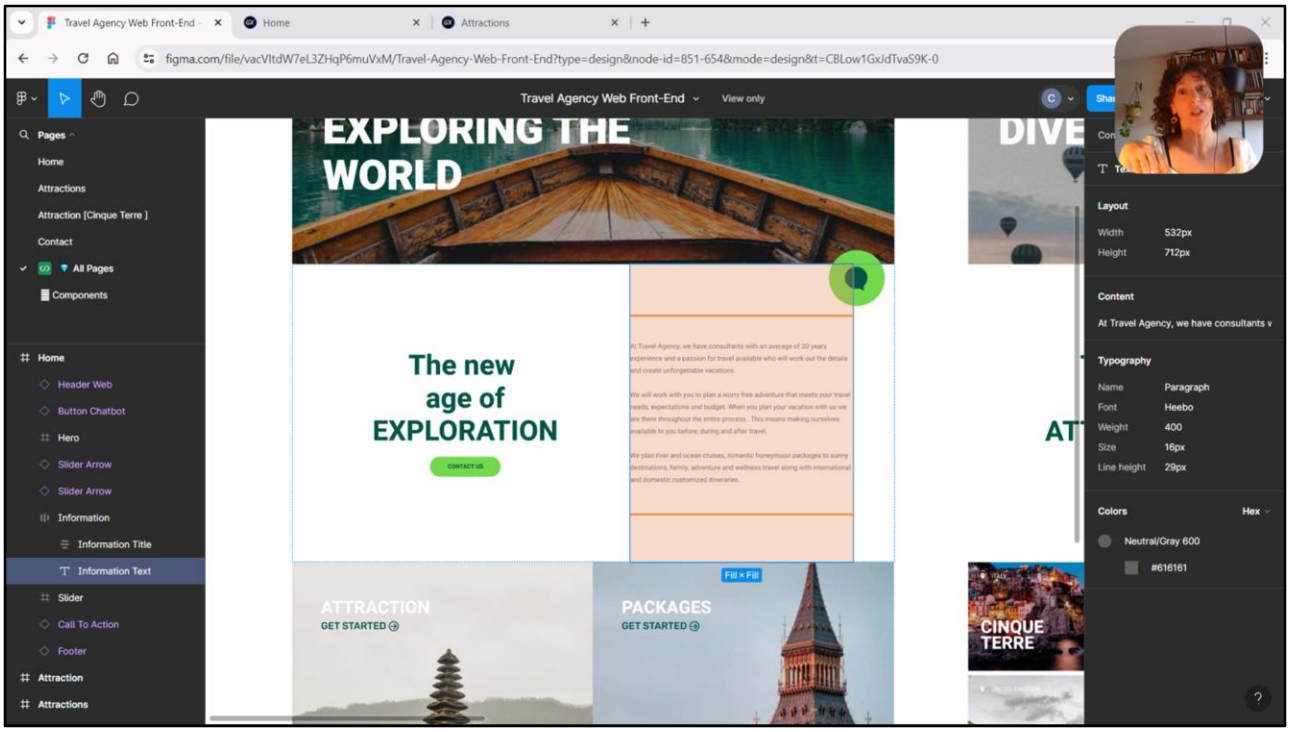
GeneXus by Globant

First Layout in GeneXus. Style (cont)

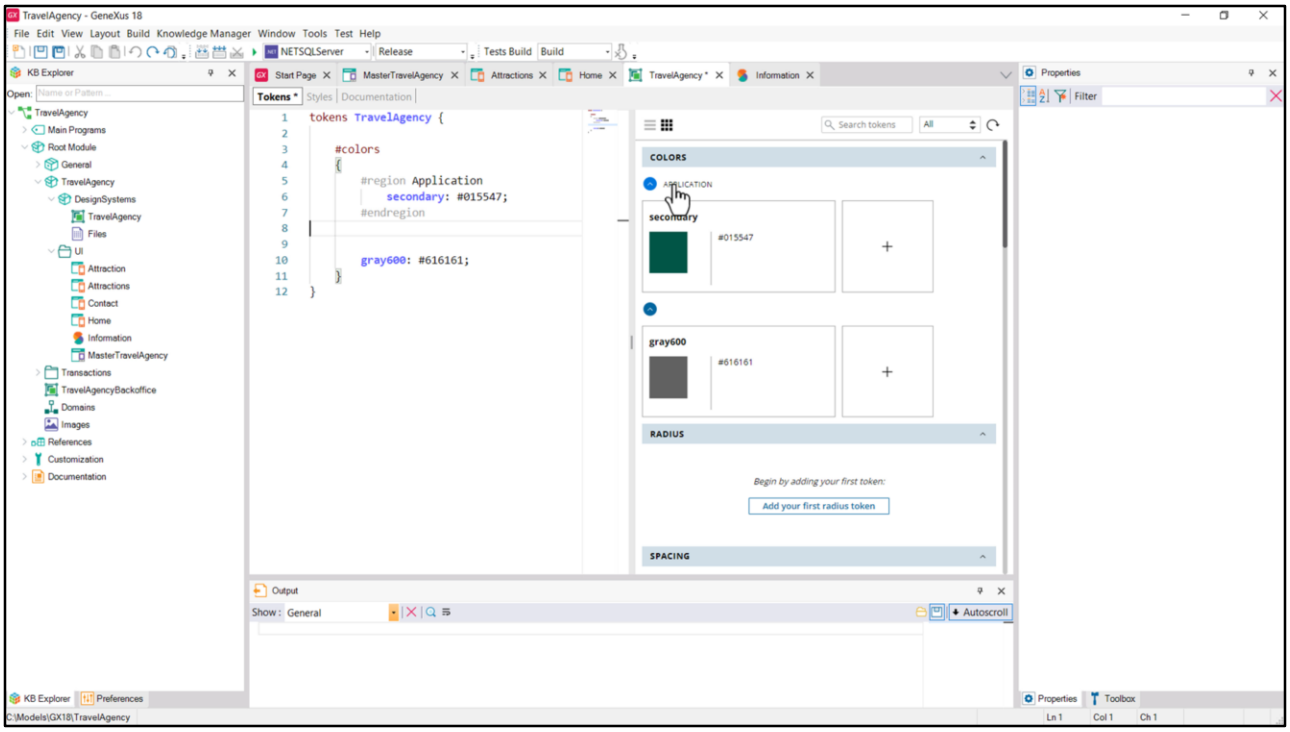
Spacing, logical properties, composite classes.

Cecilia Fernández



Neste vídeo daremos estilo ao parágrafo de nosso stencil e veremos algumas particularidades que não apareciam no caso do vídeo anterior, no do textblock. Por exemplo, veremos como definir uma margem superior e inferior para o elemento, veremos as diferenças entre propriedades físicas, ou seja, aquelas que têm a ver com as coordenadas acima, abaixo, esquerda, direita, versus as propriedades lógicas, isto é, aquelas que são estabelecidas em relação ao início e fim do elemento, entre outras coisas.

Deixaremos para o próximo vídeo a análise do estilo do botão.



A princípio, para dar estilo ao parágrafo, vamos começar fazendo o mesmo que fizemos para o texto da esquerda.

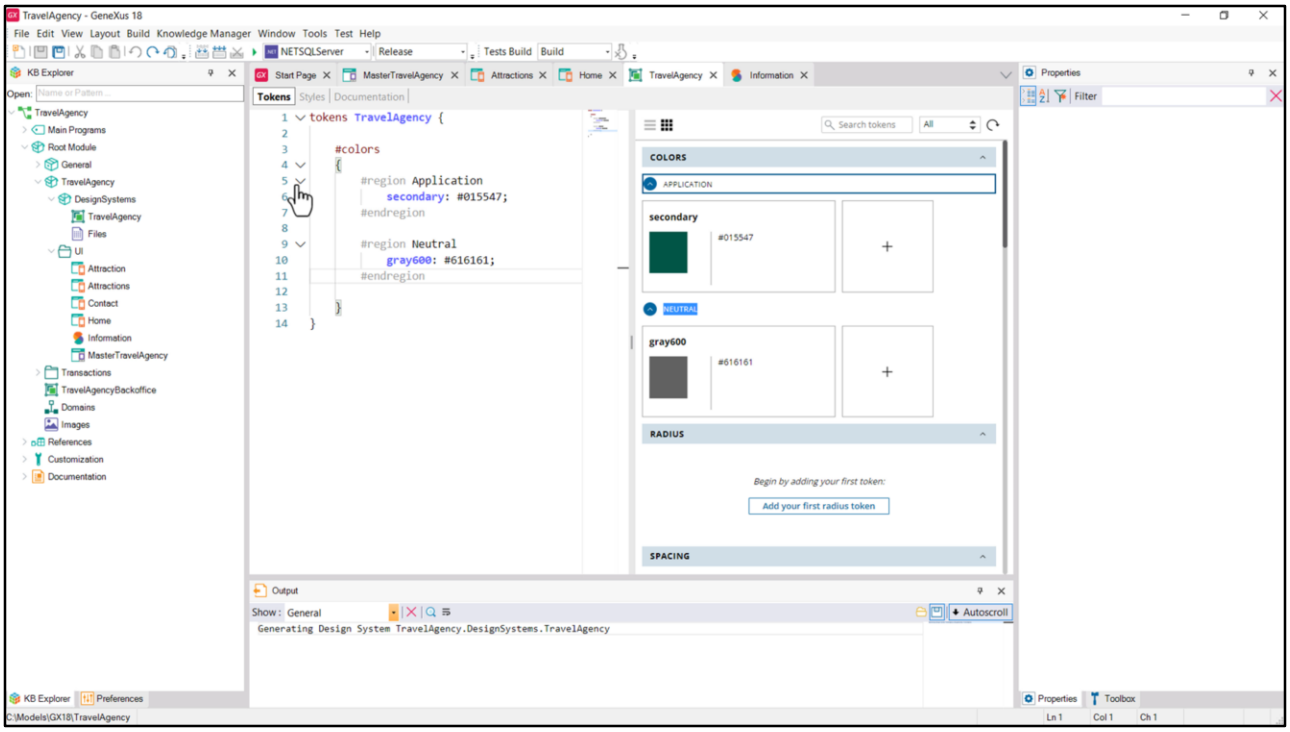
Entre suas propriedades, vemos nas tipográficas o estilo Paragraph. E a cor é esse tipo de cinza. Vou copiar... e se eu colar... vemos que aparece essa propriedade... o que vou copiar é o valor…

…porque vou começar criando esse token, gray600, na aba de tokens de meu DSO. E ali eu o copio. Vemos que está aparecendo no editor da direita... Poderia modificá-lo a partir daqui se quisesse, em vez de fazê-lo a partir daqui.

E vou começar a dar uma ordem dentro da informação, definindo aqui dentro o que chamamos de regiões, que são blocos no DSO que só servem para organizar internamente a informação. Não têm nenhuma outra função.

Então, podemos definir uma região... com cerquilha... aí damos um nome para a região, vamos chamá-la de Application, por exemplo, para a região que irá conter os tokens de cor da aplicação. Aqui está o Secundary, mas depois teremos também o Primary, e outros. Cuidado, que aqui me restou um sinal a mais... aí está.

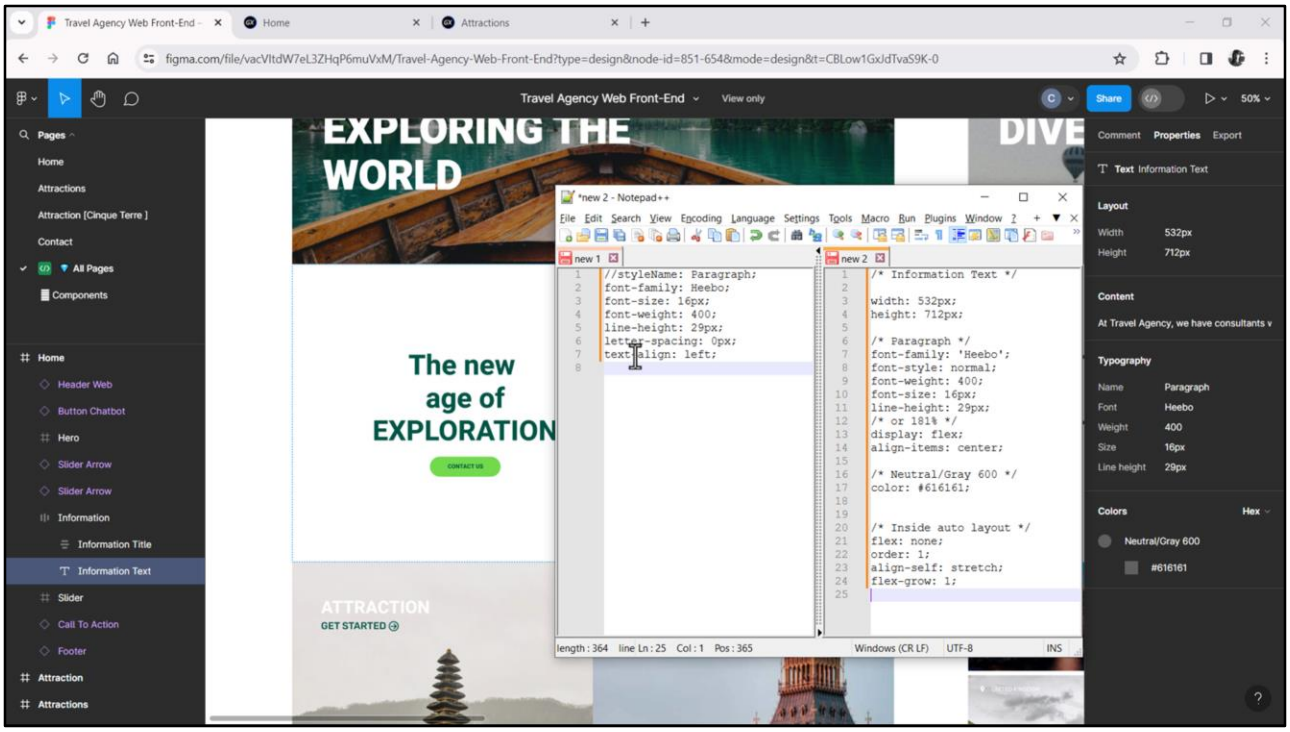
Vemos que tendo adicionado aquela região, já está aparecendo para mim, no editor de cores da direita, no gráfico, está aparecendo separado o que é esta região, desta outra que ainda não demos nenhum nome, porque é a que ficou de fora.



Poderíamos deixá-la assim, sem colocá-la dentro de nenhuma região, ou posso definir também uma região, que vou chamar de Neutral para cores neutras. E ali dentro vamos colocar esse cinza, porque evidentemente depois haverá outros cinzas.

Bom, e aí vemos que podemos colapsar... quando isso crescer fará mais sentido. Mas já o deixamos desta maneira para ir organizando então a aba de Tokens desta forma. Vamos gravar.

Aqui também... vemos que aparecem essas possibilidades de colapsar e expandir a região. E o mesmo vai valer para a aba de estilos. Logo veremos.



Agora vamos em busca das propriedades CSS do parágrafo... se as copiarmos daqui diretamente... pareceriam ser exatamente as propriedades CSS do estilo de Chechu.

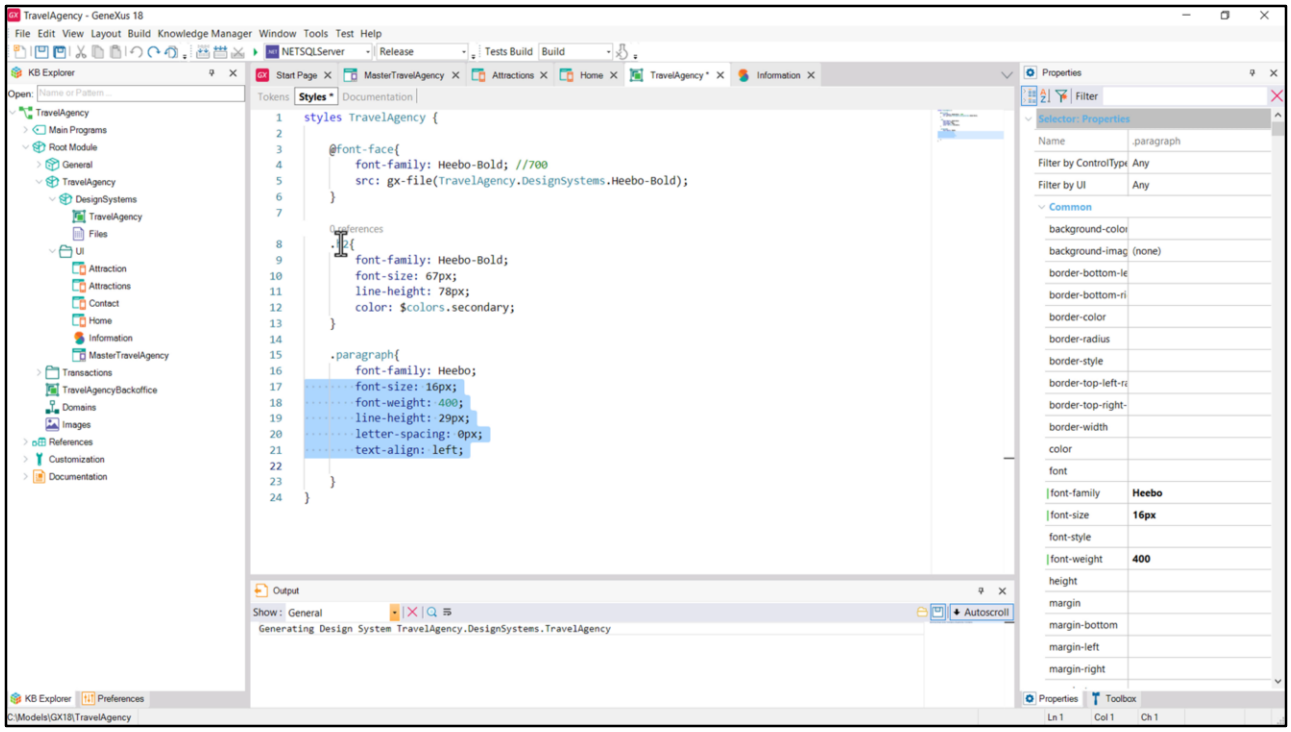
Vejamos a diferença em solicitar exatamente as propriedades como código CSS...

Vemos que há algumas que estão aparecendo aqui, como letter-spacing ou text-align e que não estão aparecendo entre as propriedades CSS listadas.

Provavelmente isso ocorra porque são default: se não forem escritas, irão assumir estes valores.

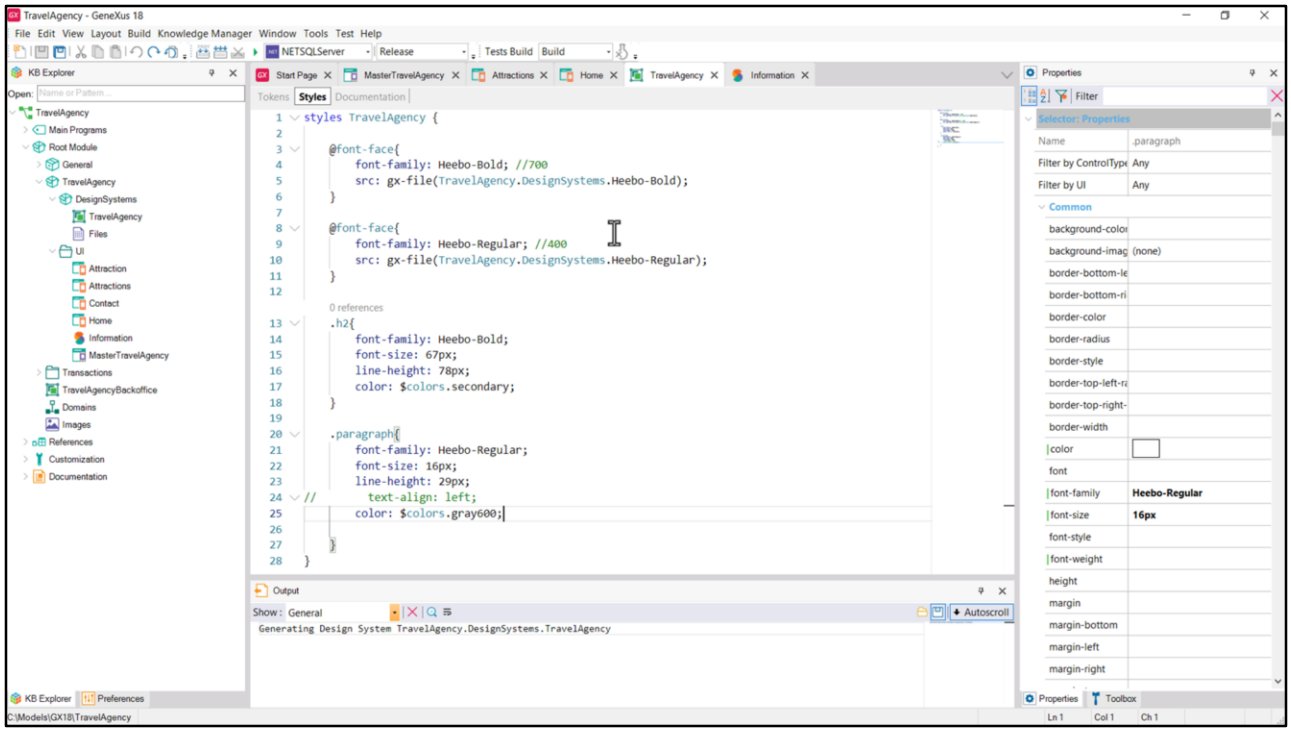
Por outro lado, aqui está a font-style normal que não está aqui, mas que não está evidentemente porque está contemplada dentro da família de fontes e o peso.

Copiaremos estas daqui, então, e vamos para o nosso DSO...



E na aba de estilos selecionaremos a classe que utilizaremos, a que chamaremos igual ao estilo de Chechu: “paragraph”. Copiemos as propriedades.

O uso de minúsculas é para respeitar a convenção de estilos de CSS BEM, que entre outras coisas nos diz que todas as classes deveriam estar em minúsculas. Devemos ter em mente que tudo o que escrevemos em um DSO (da mesma forma que acontece com html em conjunção com CSS) é case sensitive, portanto não será o mesmo uma classe paragraph que começa com minúscula e uma Paragraph que começa com maiúscula. Você tem que ter cuidado com isso.



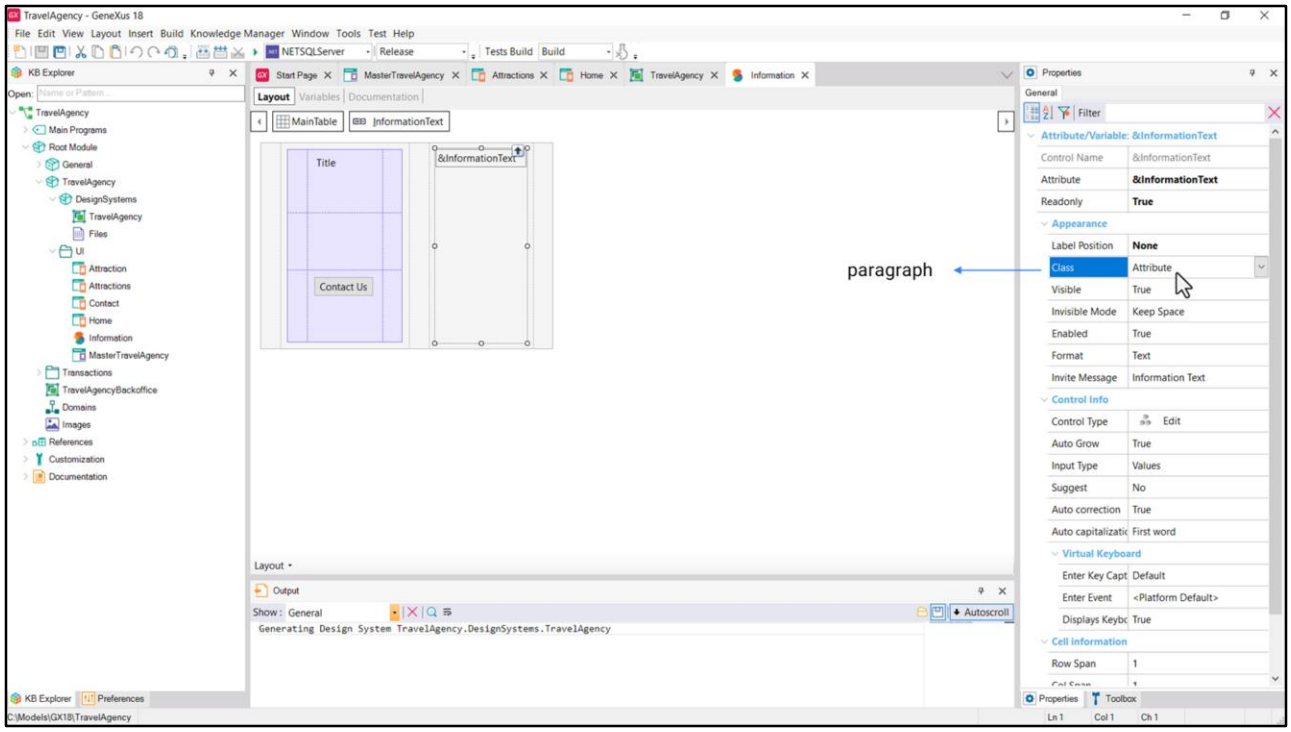
Temos que integrar a fonte em nosso caso, assim como fizemos para o caso da classe h2, no nosso DSO. Portanto, utilizaremos a mesma regra font-face.
Font-family... a chamaremos de Heebo... e qual das Heebos é a de peso 400? Era a regular, então lhe chamaremos desta maneira... vamos escrever como comentário os 400 pontos de peso.

E outra vez, onde está localizada? No arquivo que havíamos integrado no vídeo anterior para a KB. Então, a partir de que fiz isso, posso chamá-la aqui desse modo.

Esta propriedade, vou eliminar, porque ela já está contemplada nesta outra... font-size 16, line-height...
E essas duas, a princípio, não precisaremos delas, estávamos dizendo, porque o espaço entre as letras, se não se especifica nada, já é de 0 pixels. Então, vou eliminá-la.
E por enquanto, essa aqui, vou deixar comentada.

Só nos resta adicionar a propriedade de cor, a cor que assumirá o texto que tenha esta como sua classe, que será o token de cor gray600.

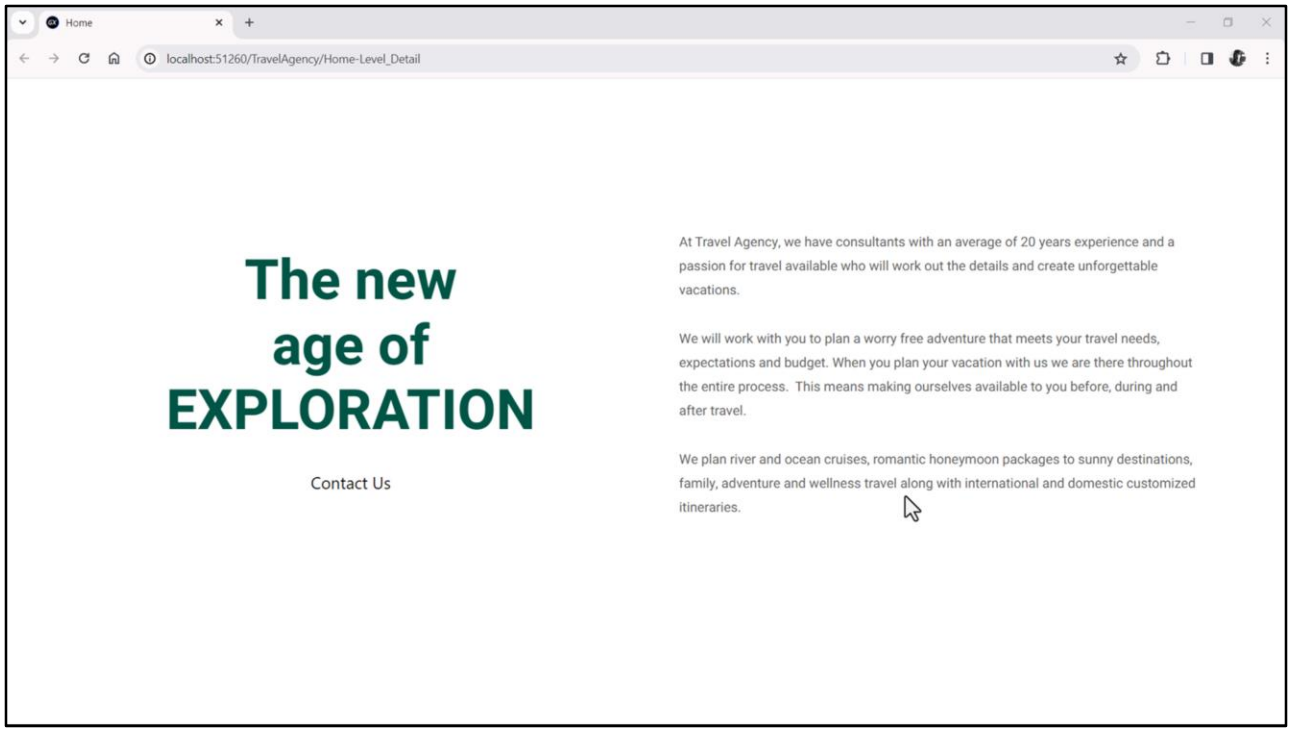
Bem, gravamos.



E o que nos resta é associá-la ao controle variável, neste caso, a classe que acabamos de criar. Não será esta classe Attribute, que era a que tinha por padrão, que, como não está selecionada em nosso DSO é como se estivesse, mas vazia, ou seja, sem associar valor às propriedades, pelo que assumirá os valores default no navegador. Dizíamos, então, mudar esta classe pela paragraph.

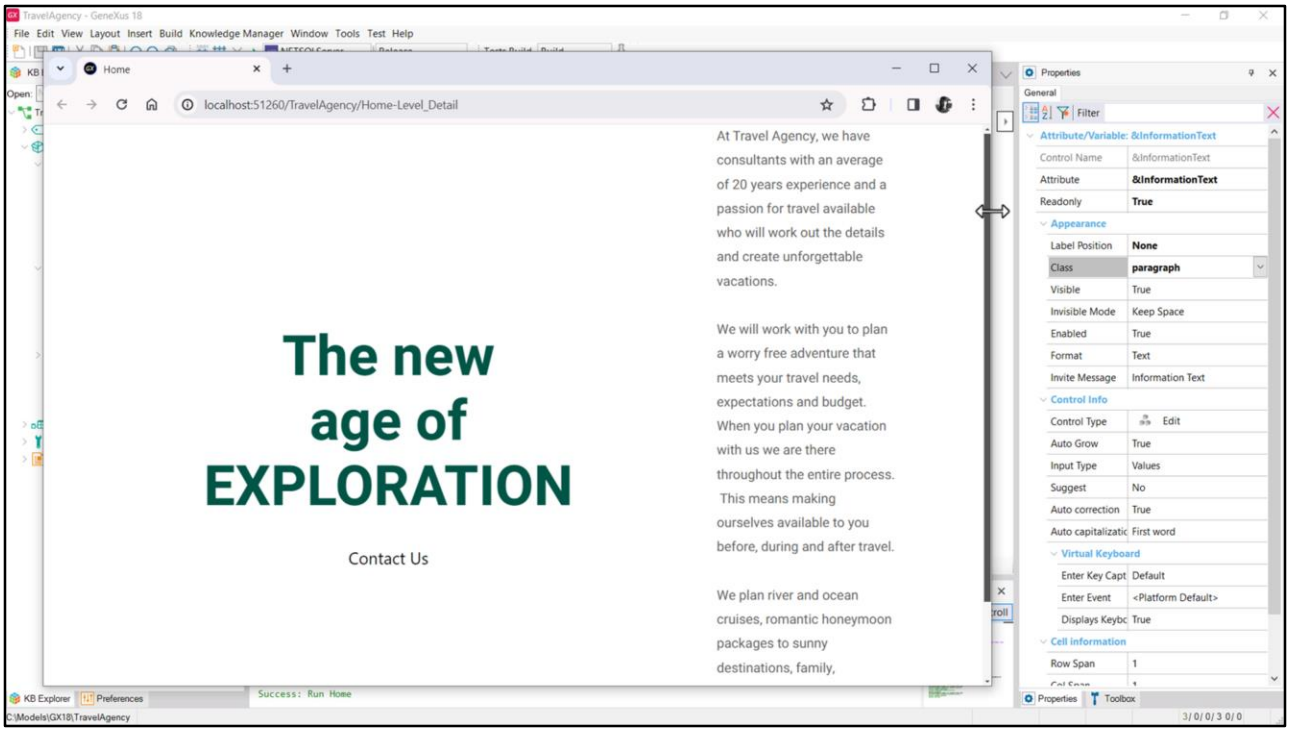
Feito isso, gravamos... e como vimos antes... fica alterado automaticamente para a variável instanciada no panel Home, e o mesmo será para a variável instanciada no de Attractions.

Bom, então vamos executar para testar...

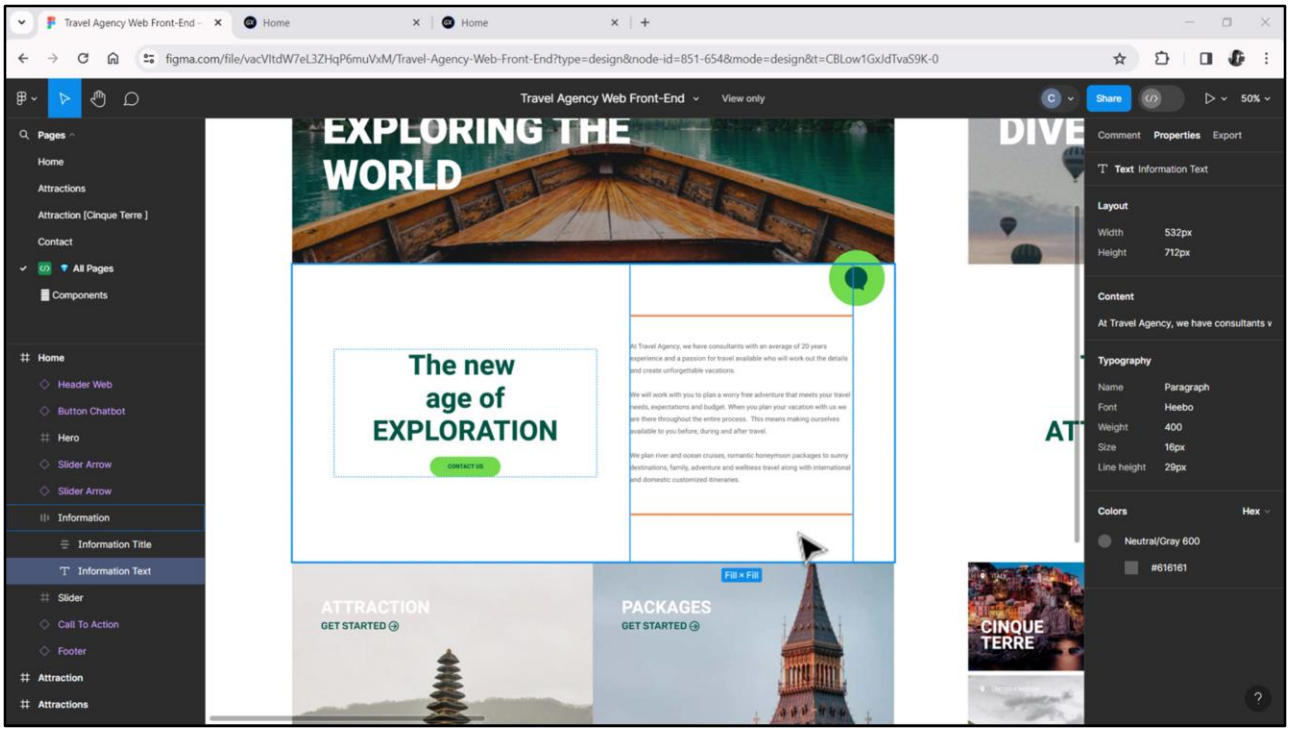


... perfeito.

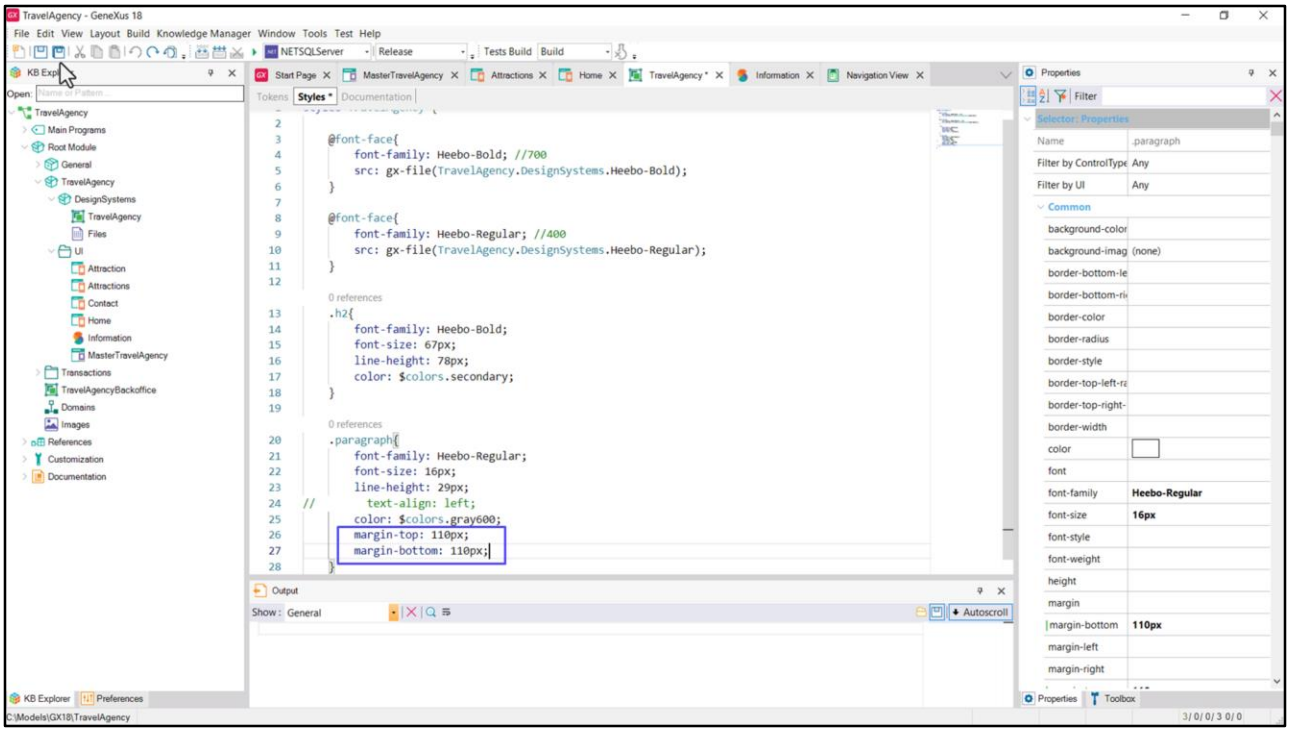
É claro que a propriedade text-align left não foi necessária, pois deixamos ela comentada e ainda assim está sendo alinhado pela esquerda.



É o momento de trazer à tona algo que havíamos deixado pendente em um dos vídeos anteriores... E era não ter definido um espaçamento vertical para esse texto... lembram que se diminuíssemos a largura da tela ficava grudado nas bordas superior e inferior da tabela?

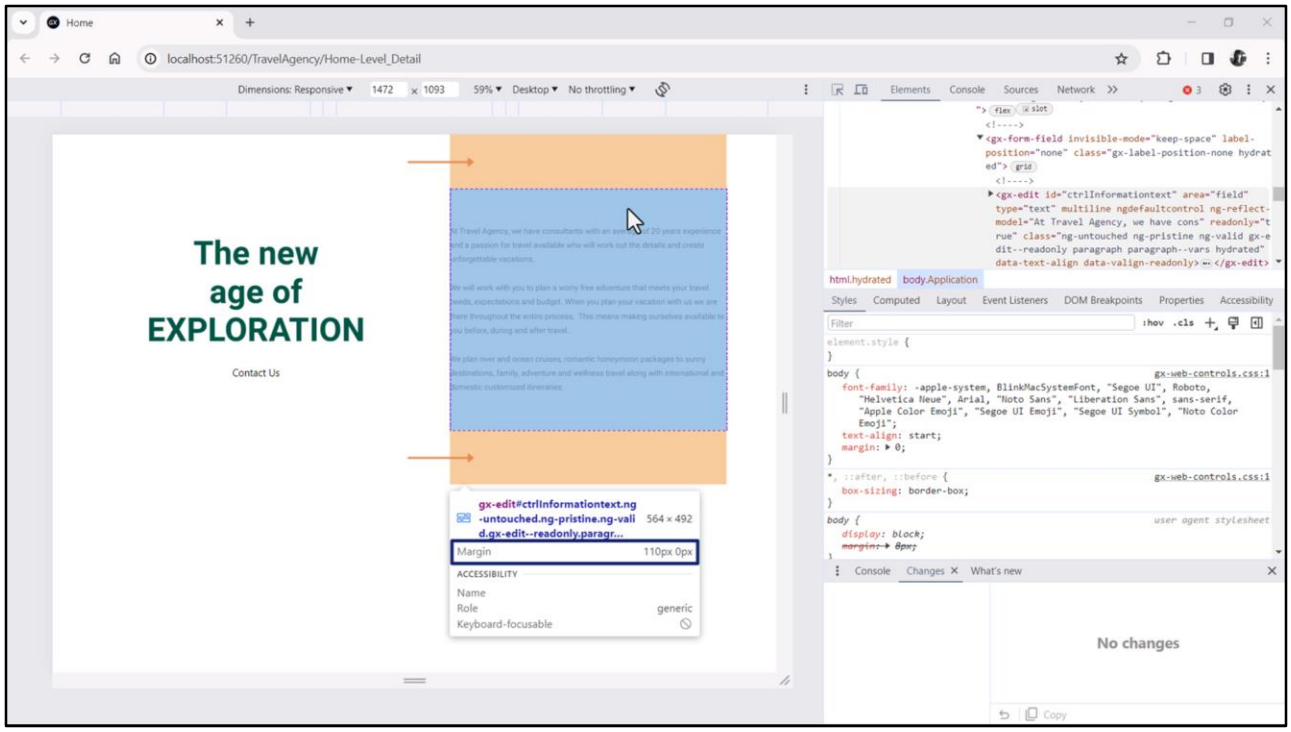


É que tinha Fill para a altura. Naquele vídeo tínhamos visto que um bom espaçamento de cima e de baixo poderia ser de 110 pixels e que poderíamos conseguir isso especificando uma margem superior e inferior para este elemento, desse valor. Como se Chechu pudesse ter adicionado aqui essas propriedades... não pode, claramente; o estilo tipográfico é apenas tipográfico. E também não há outra forma de fazer.



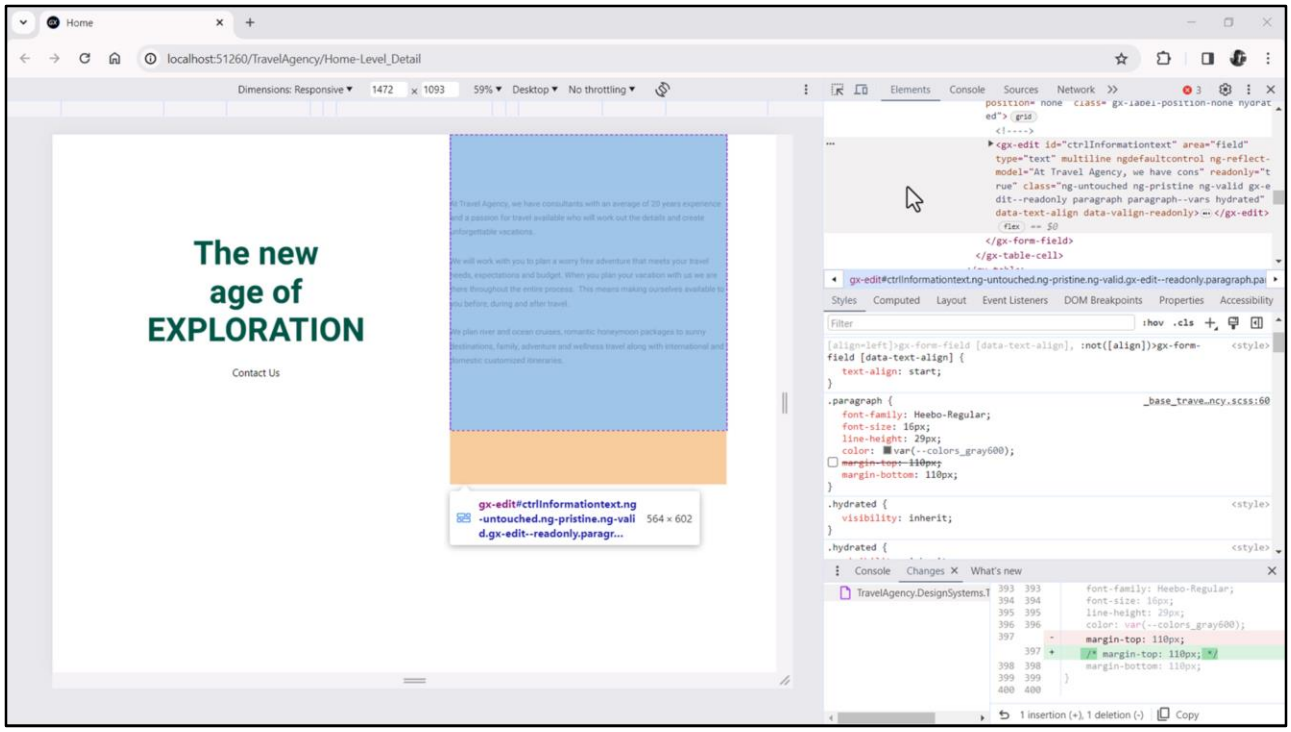
Mas nós, em nossa classe, não temos essa limitação. Então poderíamos escrever isto…

Vamos testar…

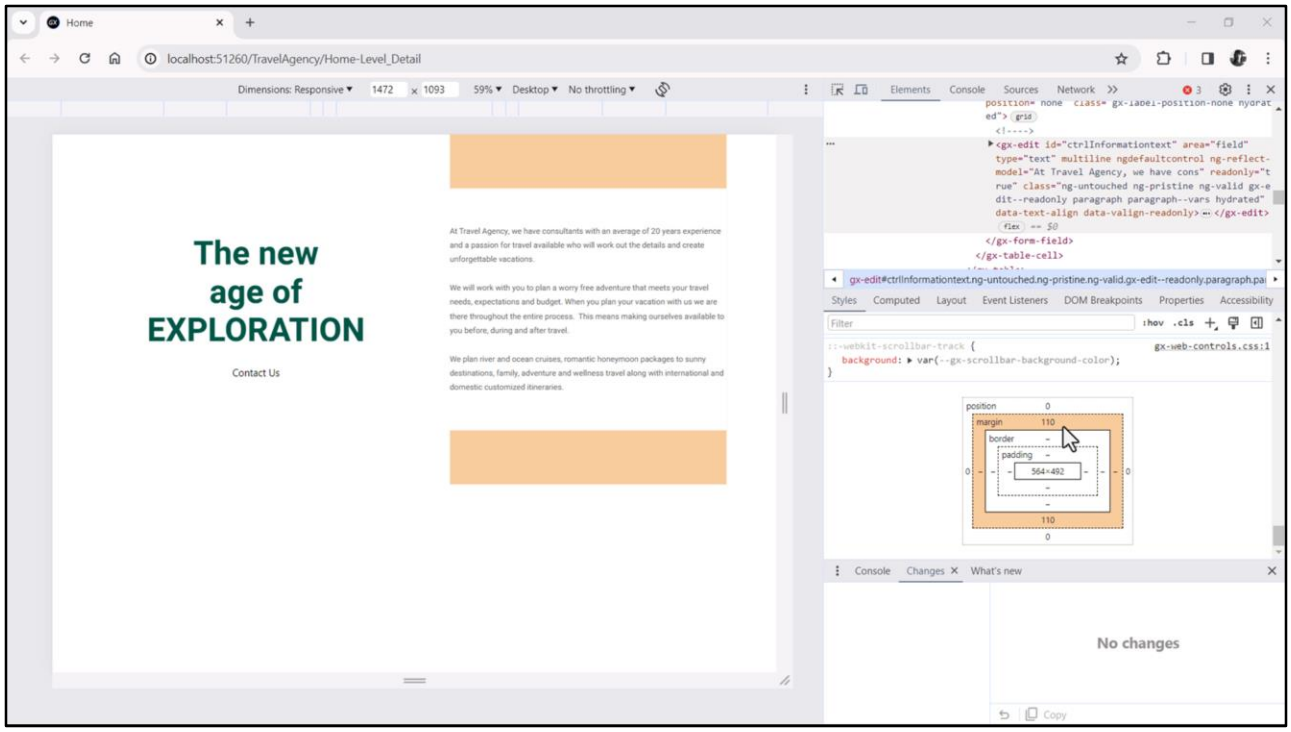


Vemos essa margem sendo respeitada, acima e abaixo.

Vamos inspecionar o html resultante. Aqui vemos claramente o elemento e as margens superior e inferior.



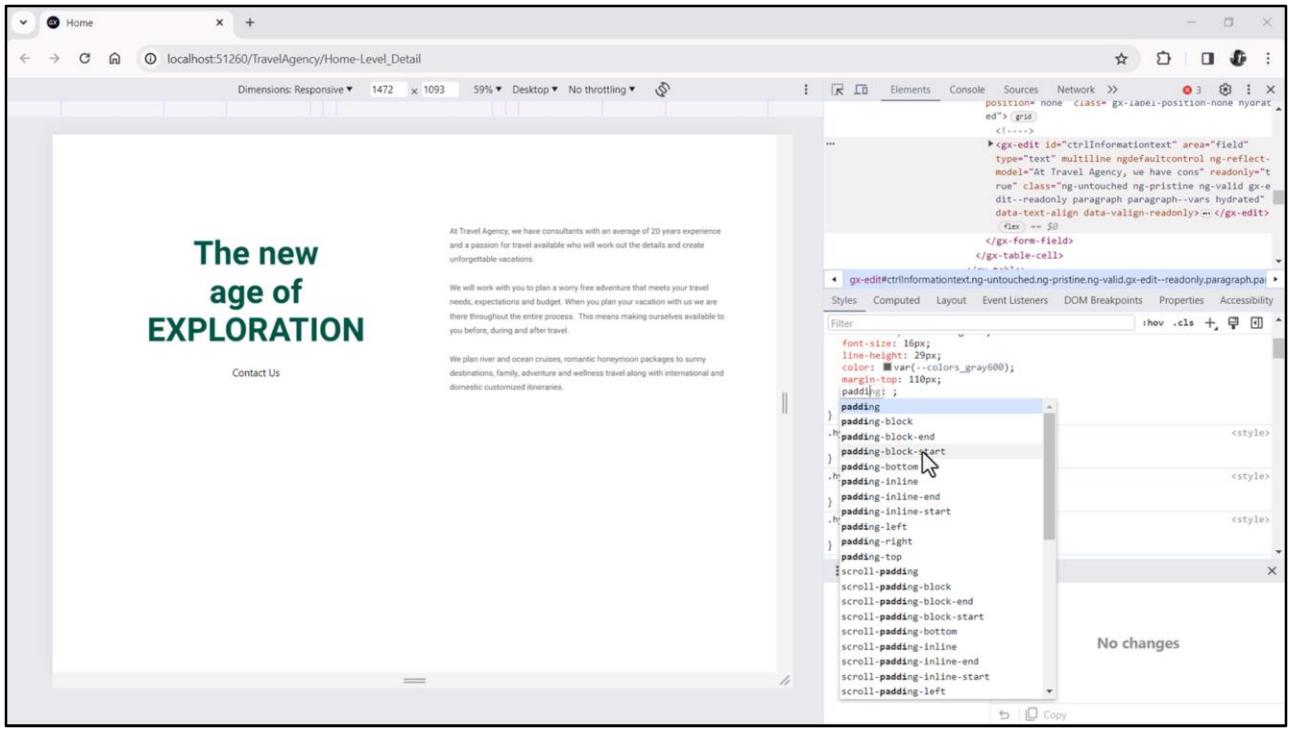
Se procurarmos aqui a classe que está aplicada, que é a paragraph, vemos as duas propriedades e, por exemplo, poderíamos apagar margin-top e ver como, ao fazer isso, efetivamente desapareceu essa margem superior.



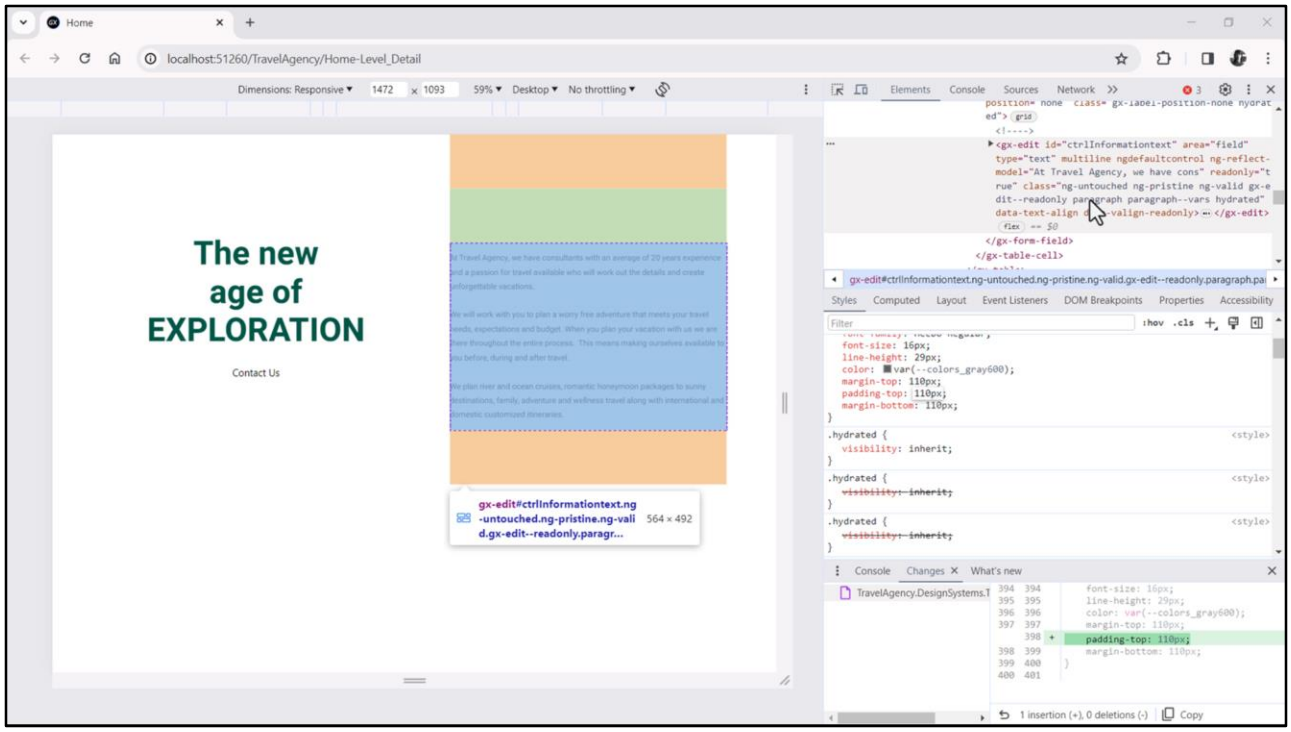
Podemos visualizar abaixo de tudo justamente esses 110 pixels de margem de cima e de baixo. Vemos que não tem definido borda, nem padding, e aqui temos o elemento interno.

Por que optamos por dar esse espaçamento através de margem superior e inferior e não o fizemos através de padding? Visualmente veríamos a mesma coisa, vamos testar mudar a margem de 110 para um padding de 110, para a margem e padding superiores. Vamos deixar a margem inferior como está, para ver a diferença.

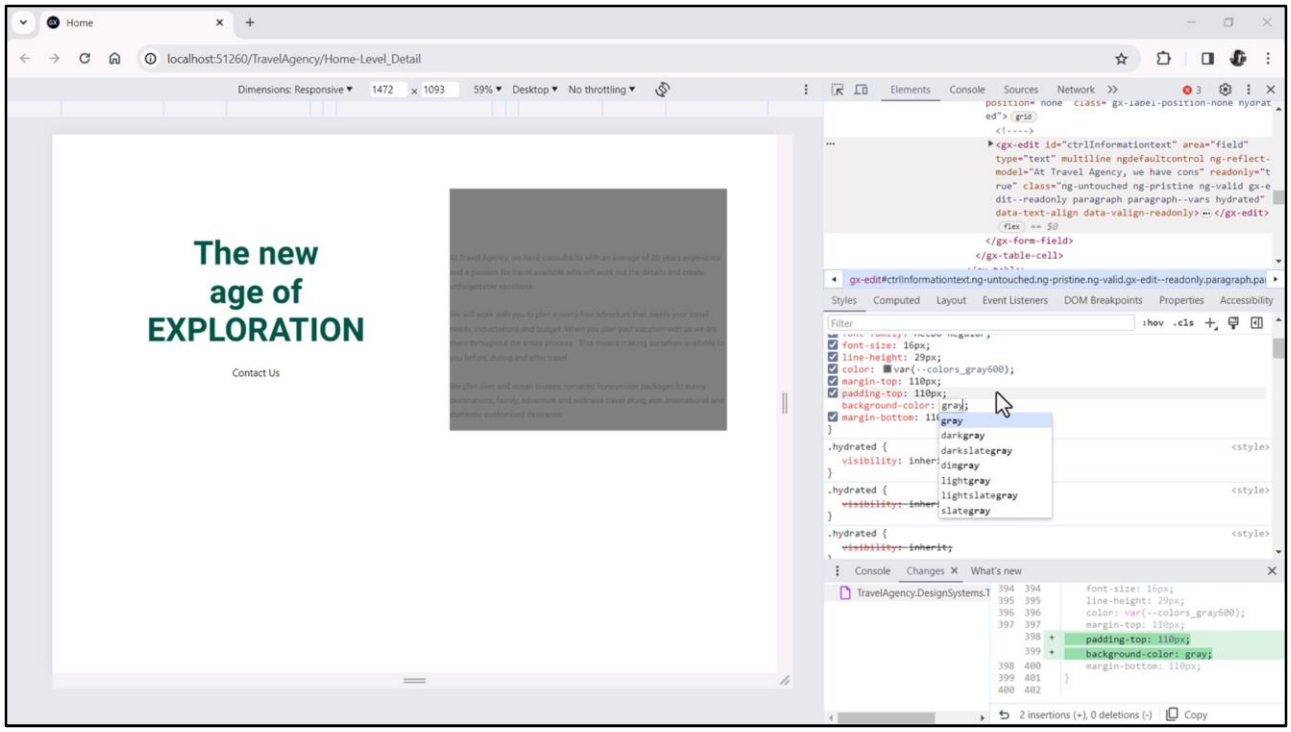
Mas aqui já vou adiantar: o elemento é o que existe dentro da borda, deste limite para dentro. A margem estará por fora do elemento e vamos ver quais implicações tem isso.



Então antes, vou voltar para classe, e vamos colocar a propriedade padding… padding-top vamos escolher… vou deixá-los mencionados porque daqui a pouco vamos entrar nisso, mas vemos esses padding block block block e pandding inline? Sim?

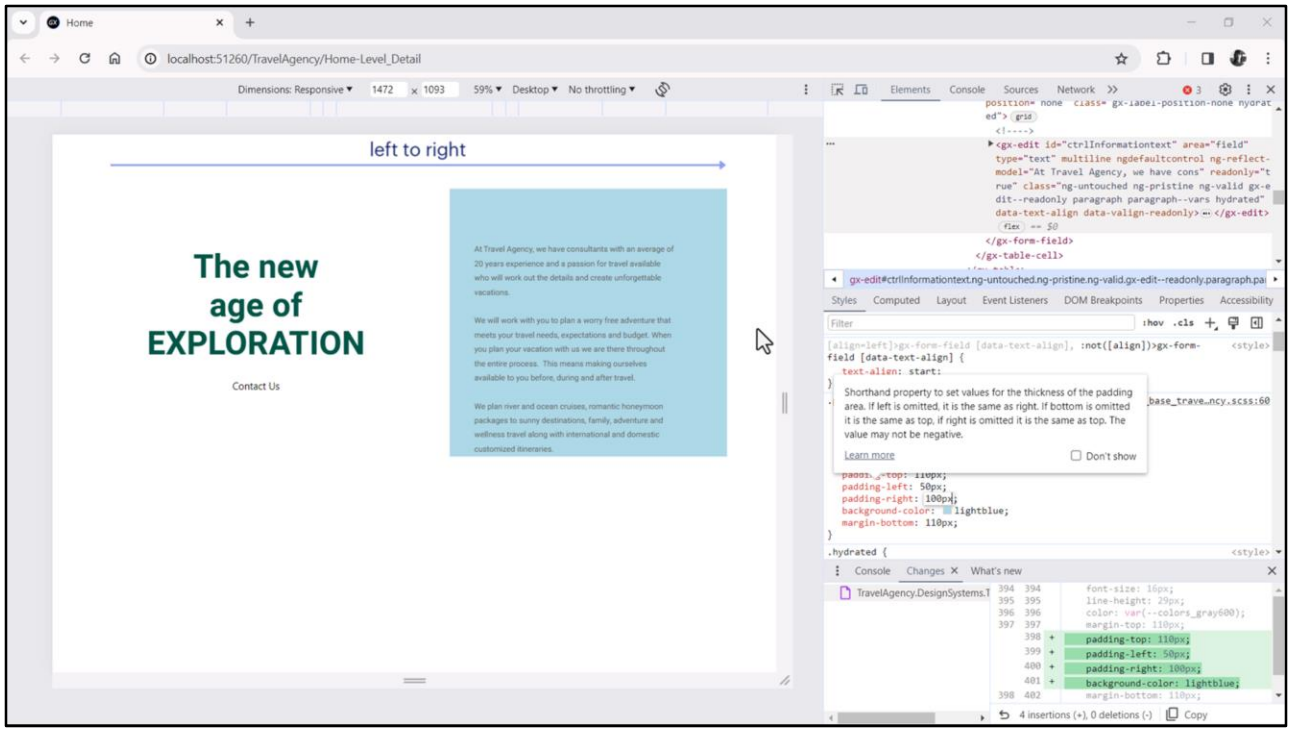


Bem, padding-top 110 pixels. E vemos isso claramente ali indicado: é essa área em verde.



Vamos ver o que acontece –vou deixar as duas: a margem superior e o padding superior- e vamos ver o que acontece se dou uma cor de fundo a esse elemento. Background color… e por exemplo vamos colocar um cinza.

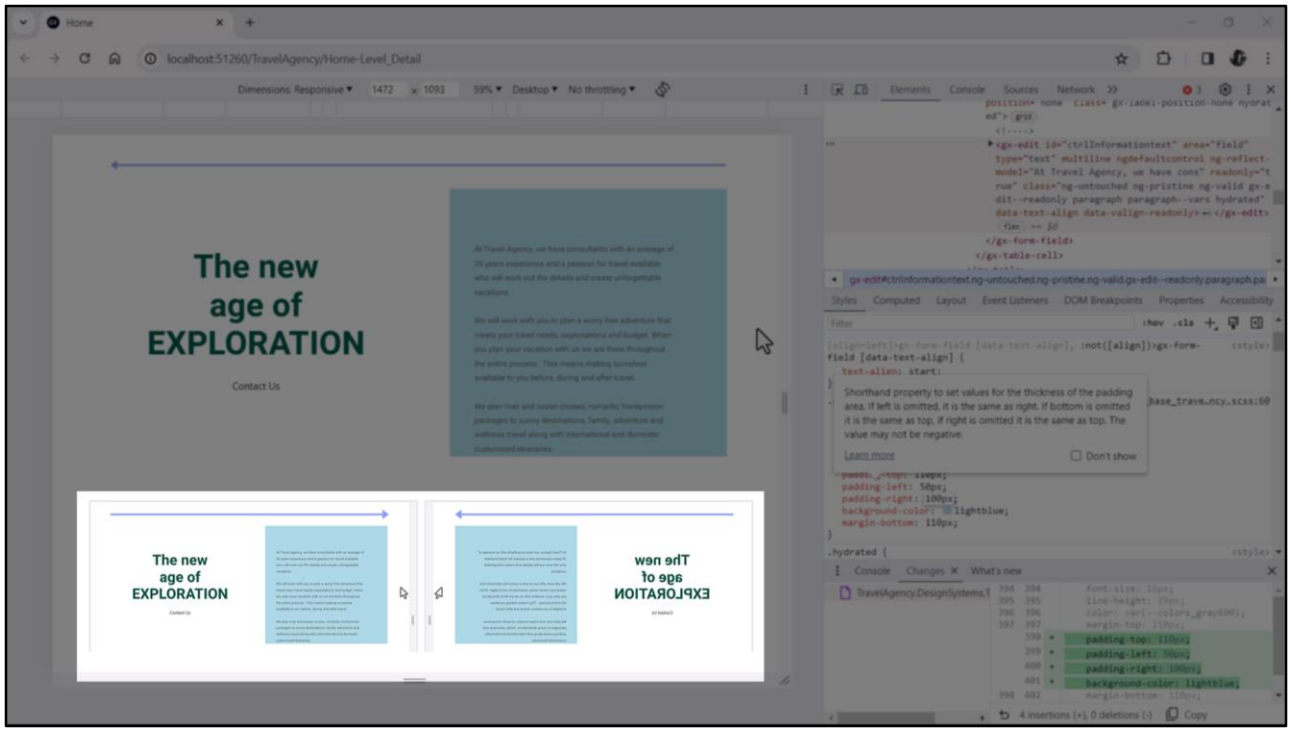
O que está acontecendo? Vemos que está ficando em cinza tanto a área interna do controle, como a área do padding, essa área verde. Enquanto à área da margem não está sendo aplicada essa cor de fundo. A mesma coisa, se implementássemos um evento, clicável no nível deste conteúdo, esta área também seria clicável, enquanto a área de margem não seria.



Bom, agora sim, vamos entrar neste assunto da diferença entre as propriedades físicas (padding top, left, right, bottom) versus as propriedades lógicas, e agora vamos ver do que se trata isso.

Antes, vamos mudar este gray para um mais claro. Light… bom, lightblue vou deixar. E o que vou fazer para que se note bem a diferença das questões é escrever um padding… left… de 50 pixels… e um padding right de 100, o dobro. Ou seja, da borda esquerda temos um padding que é a metade do padding da direita.

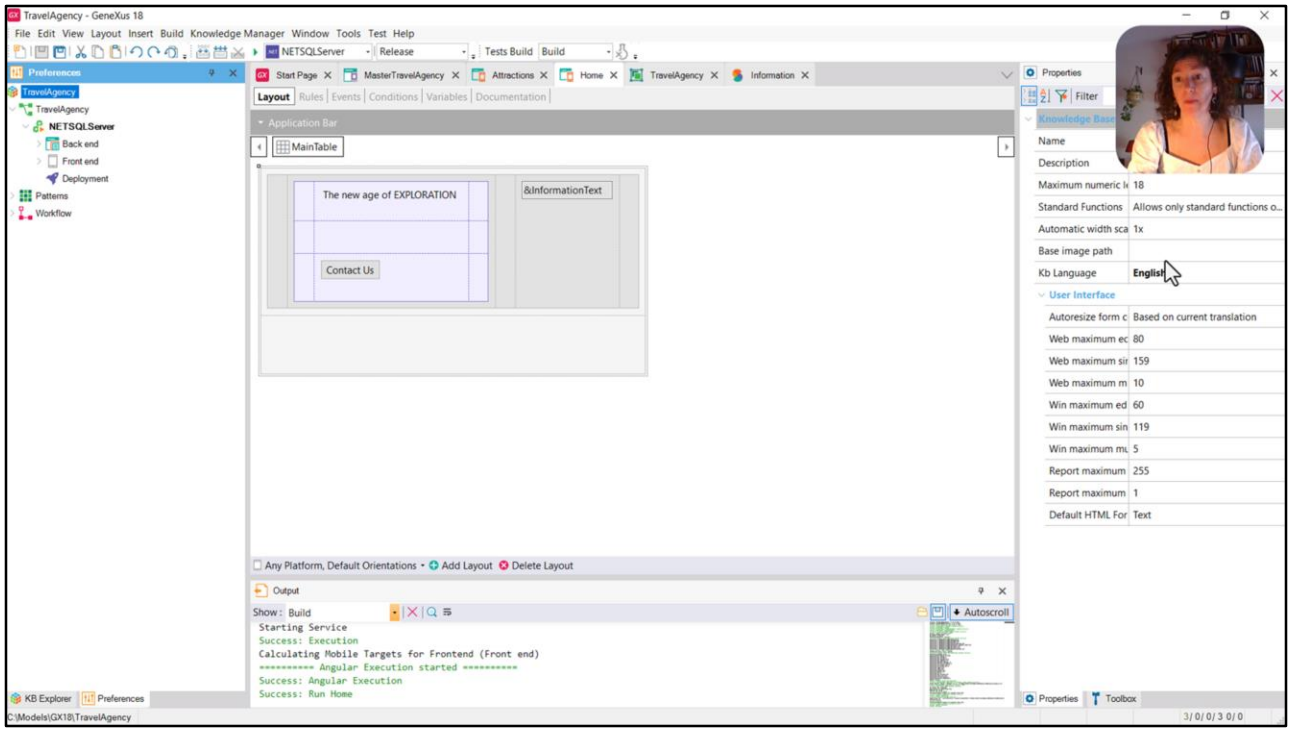
Agora bem, a pergunta que me surge… estamos executando esta aplicação com uma direção do texto que vai da esquerda para direita…



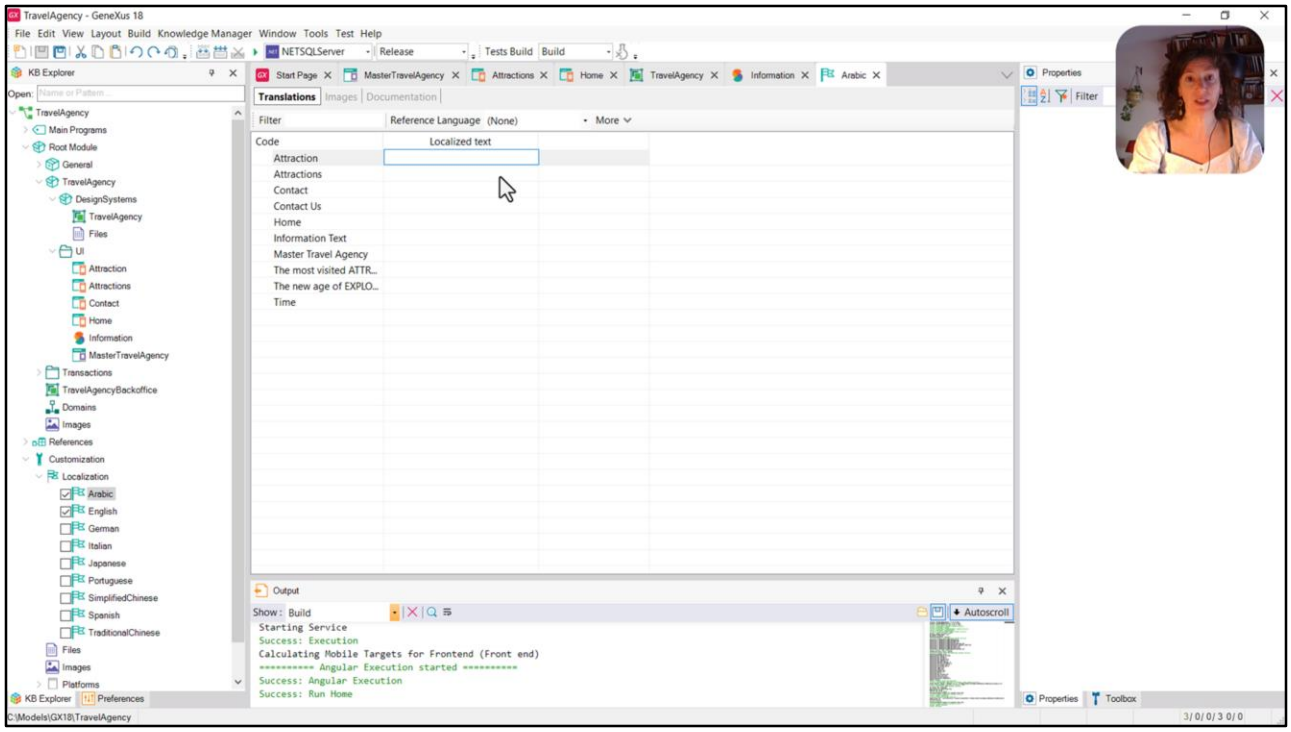
Agora, se estivéssemos executando a aplicação para hebraico ou para árabe, teríamos que visualizar tudo isto o que estamos mostrando aqui, como se tivéssemos um espelho que invertesse completamente a direção. Portanto seria em uma direção right to left.

O que é que queríamos que acontecesse nesse caso, ou seja, ao mudar a direção de fluxo da tela?

Que a tabela fosse invertida em espelho, onde a primeira coluna da esquerda se transforme na primeira da direita, a segunda na segunda, a terceira na terceira, a quarta na quarta e a quinta na quinta, nessa ordem. No nível da localização dos controles em nossos layouts, isto será feito automaticamente.

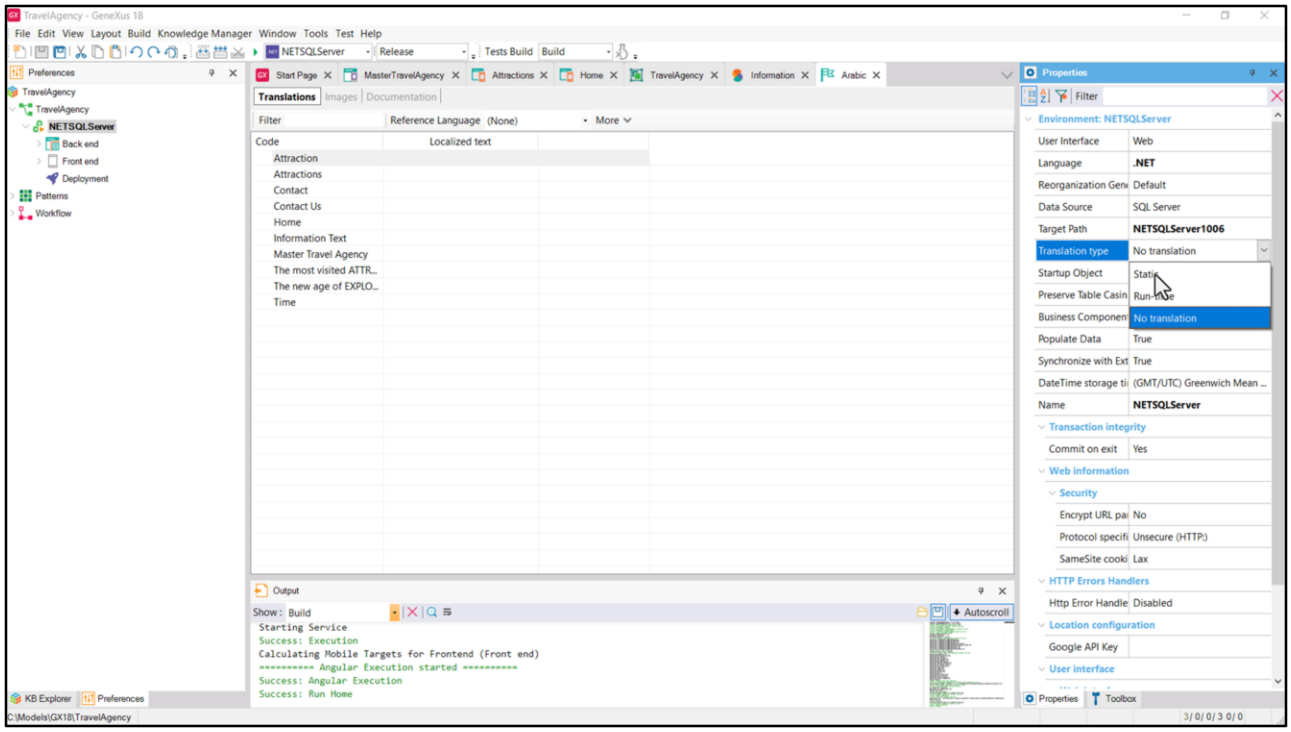


Claro, estou me referindo ao caso em que queremos traduzir a aplicação. Se viermos para a aba de Preferences, podemos ver que no nível da KB temos a propriedade Kb Language com o idioma English.



Se viermos ao KB Explorer, sob o nó Localization, vemos ali que temos este objeto English que está coincidindo com o idioma da KB. E é o único que temos no momento incorporado na KB. O que é que será feito? Serão recolhidos de todos os objetos de nossa KB os textos para serem traduzidos.

Então, desta forma, suponhamos que queremos traduzir a aplicação para o árabe. Então, conseguirá ligando essa propriedade, irá incorporar esse objeto dentro da KB, extrairá... vou abri-lo... todas as constantes, todos os textos que encontrou nos diferentes objetos, para nos permitir justamente escrever as traduções desses textos.



E então, dessa maneira, em seguida vamos para a janela de Preferences e podemos, no nível do Environment, estabelecer o tipo de tradução que queremos realizar. Se estática, onde vamos indicar então qual é o idioma para o qual queremos traduzir a aplicação, daqueles que temos incorporados na KB; ou se a queremos, pelo contrário, realizar dinamicamente. E ao fazer isso dinamicamente, podemos dizer no nível de nossos objetos com qual idioma queremos que seja renderizada a aplicação cada vez. Claro, vamos ter que adicionar alguma programação.

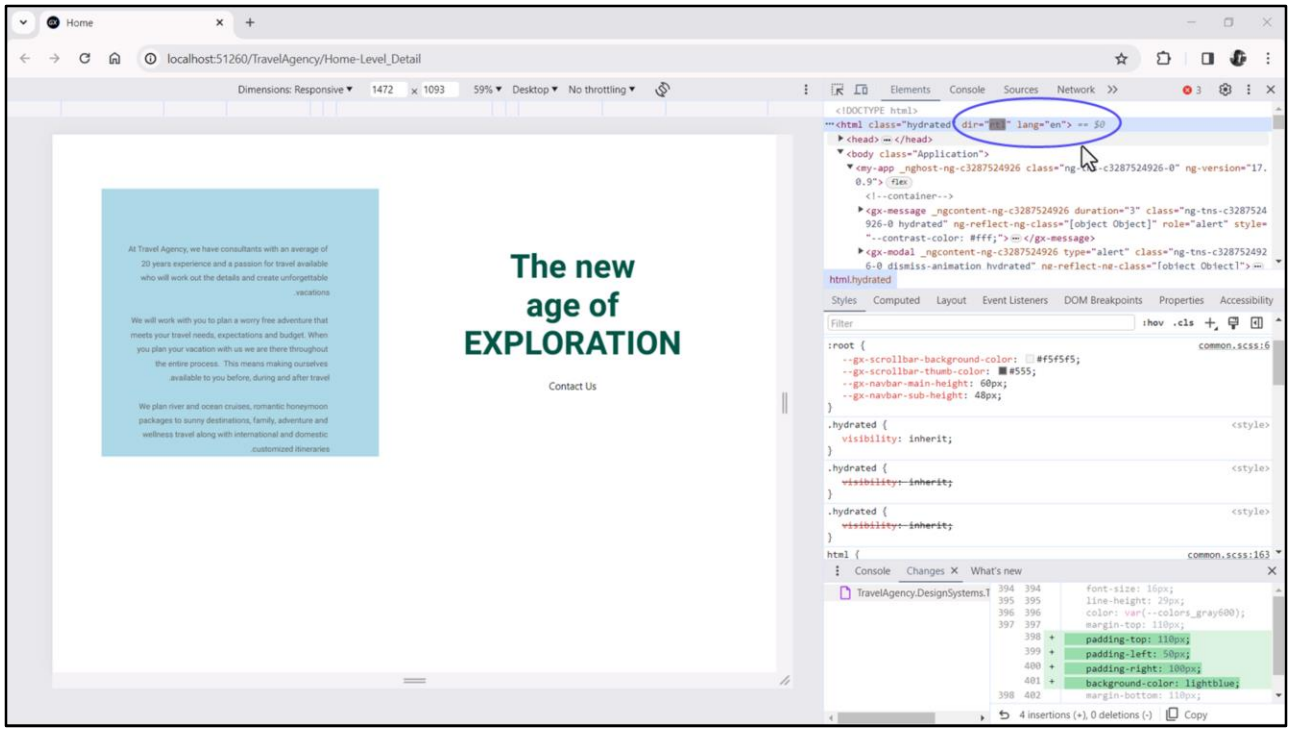
Bem, então este é o contexto em que estamos, certo? O que eu lhes dizia agora há pouco, é que uma vez que eu traduza a aplicação para árabe, automaticamente as colunas das tabelas vão se inverter em espelho. E assim também os textos. Vão aparecer no idioma que tenha escolhido, com as traduções que tenha inserido.

Agora, concentremo-nos no que nos interessava, no padding. Quando utilizamos referências esquerda e direita para proporcionar o padding, não nos demos conta de que na verdade as referências corretas seriam início e fim, em relação à direção de fluxo e não esquerda e direita… porque no fluxo right to left não queremos que o padding da esquerda seja de 50, e o da direita de 100, mas o inverso.

Necessitamos utilizar referências lógicas, ou seja, o padding na direção do fluxo (inline) em relação ao início, que seja de 50; e o padding na direção do fluxo (inline), em relação ao final do elemento de 100. Esta forma lógica valerá tanto para o fluxo left to right, como para o right to left. Ou seja, estamos nos tornando independentes das coordenadas esquerda e direita, que nos complicam justamente quando queremos fazer essa inversão.

Mas da mesma maneira e por consistência, se mudamos as propriedades em torno da direção horizontal, também teremos que mudar as propriedades que tem a ver com a localização acima, abaixo... Então é que aparecem, as propriedades análogas, mas lógicas: block-start e block-end.

Portanto, inline representa a direção horizontal, a de escrita e block, a vertical.



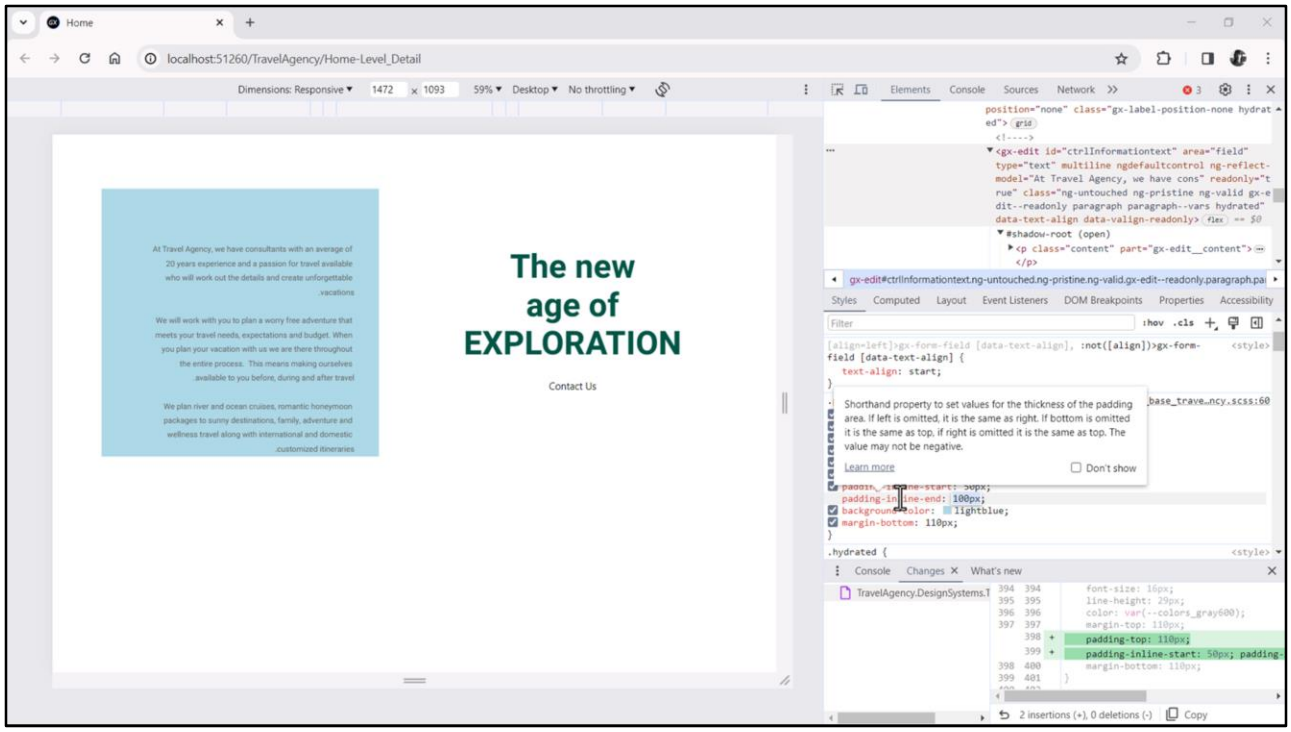
Para que nos fique claro tudo isso, vamos ver aqui. Temos, então, o padding-left e o padding-right. E o que vou fazer é vir para a tag html para mudar a direção de left to right para right to left, sabendo que o idioma é inglês, como dissemos.

Então agora quando pressionar enter… vemos que efetivamente se inverteu tudo o que queríamos,

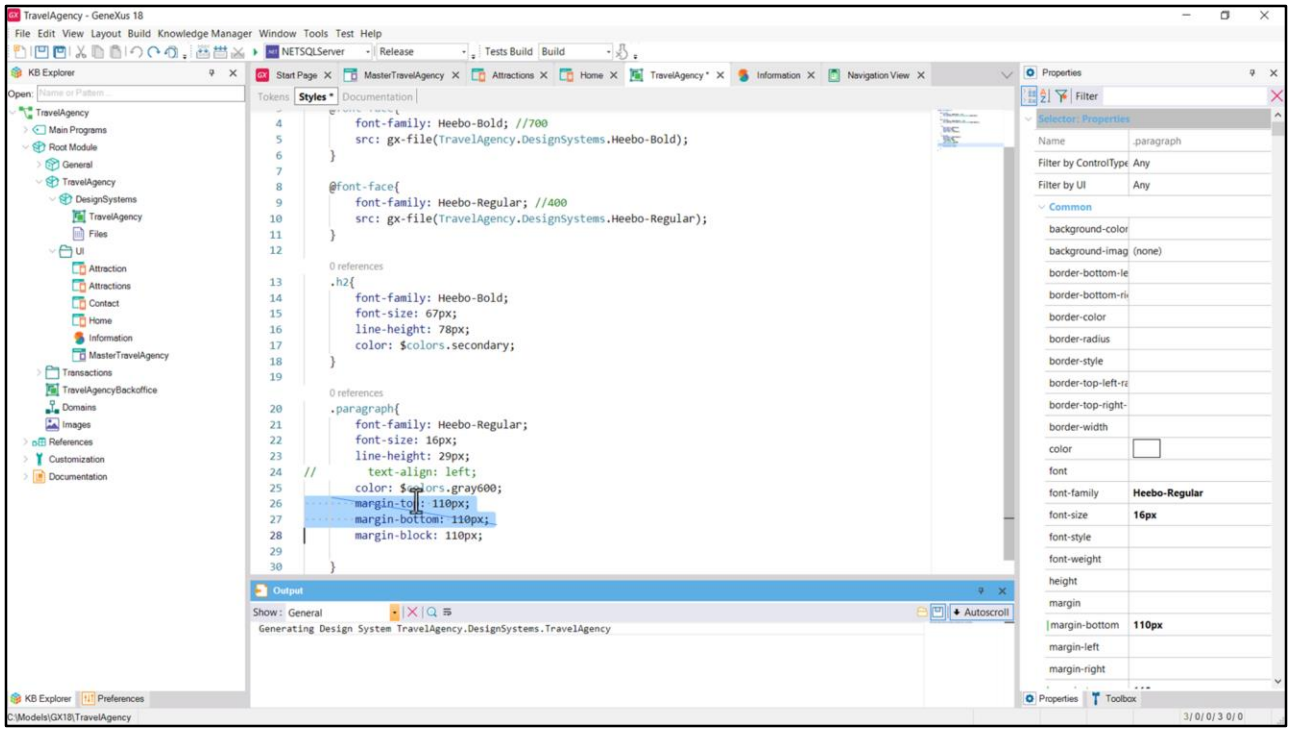
(os textos não porque o idioma é inglês, mas não nos importa, quando o idioma for árabe ou hebraico também se inverterão)

Então, repito tudo o que queríamos será invertido,

salvo o padding… porque ao ter utilizado as propriedades físicas irá respeitar que o padding da **esquerda** seja de 50 e o da **direita** de 100. Nós não queríamos isso, queríamos que fosse invertido, portanto…

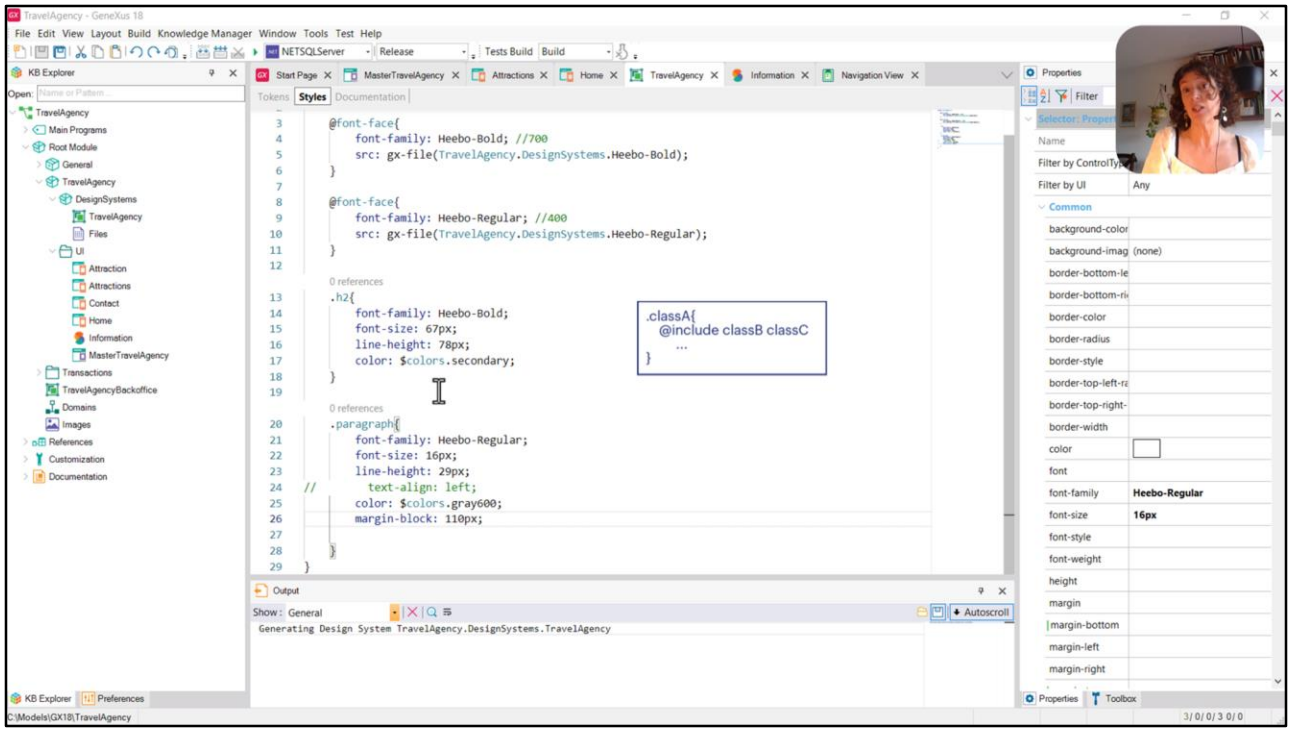


…vamos modificar as propriedades físicas trocando-as pelas lógicas… assim no lugar do padding-left vamos usar padding-inline-start de 50… e para o padding right vamos usar padding-inline-end. E agora sim.



Bom, tudo isso que vimos é para recomendar fortemente não utilizar as propriedades físicas, mas para utilizar as lógicas. No caso anterior, vimos o exemplo com padding, mas é exatamente o mesmo com margem.

Aqui estamos usando as físicas, então a sugestão é utilizar no lugar destas… a margin-block seria neste caso… start e end… Ou, se eu quiser utilizar a notação abreviada, apenas block, que neste caso me vai ser conveniente porque além disso, são exatamente o mesmo valor, então posso colocá-lo uma única vez. E remover estas duas.

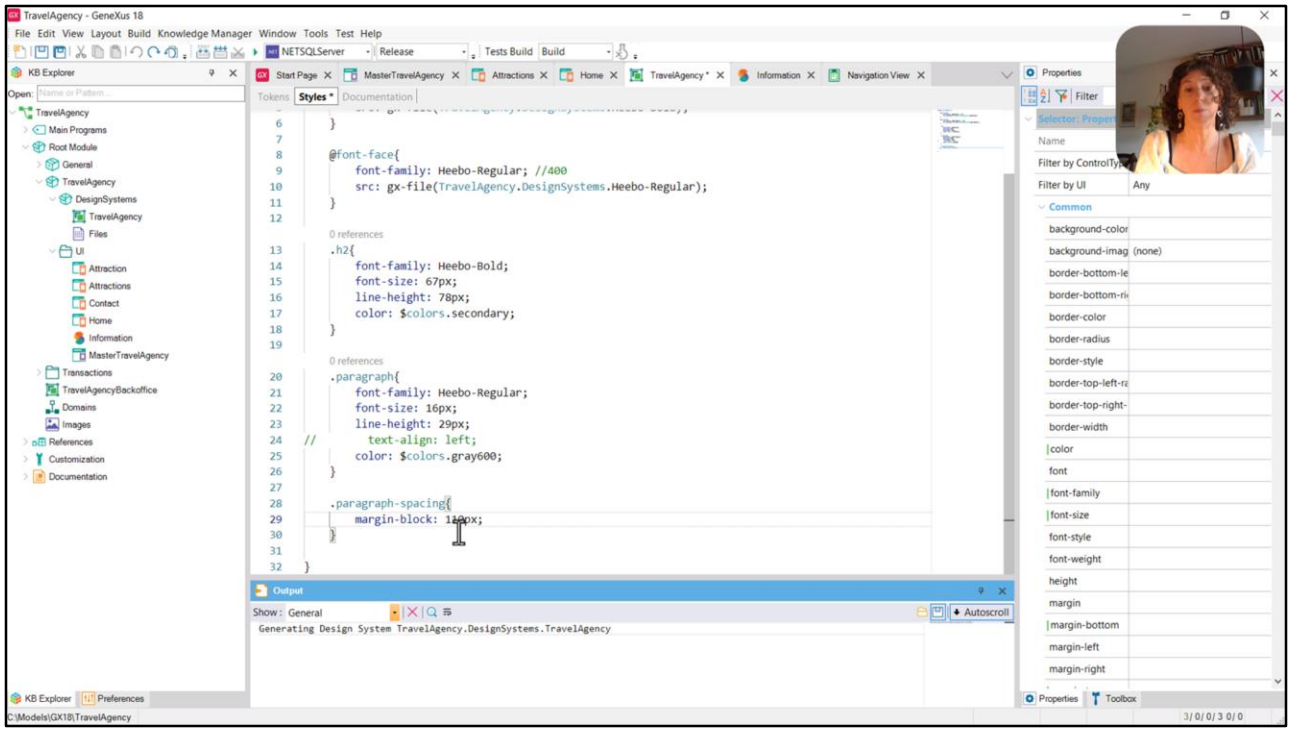


Algumas observações antes de terminar…

As classes são independentes do tipo de controle ao qual acabam sendo aplicadas. Por independentes, me refiro a que eu não tenha que dizer em nenhum momento que esta classe seja aplicada a tal ou qual tipo de controle. Dependerá do tipo de controle ao qual eu associe esta classe, se todas as propriedades vão ter efeito ou não, ou algumas não. Terão efeito aquelas que tenha sentido que tomem efeito, e as demais serão ignoradas.

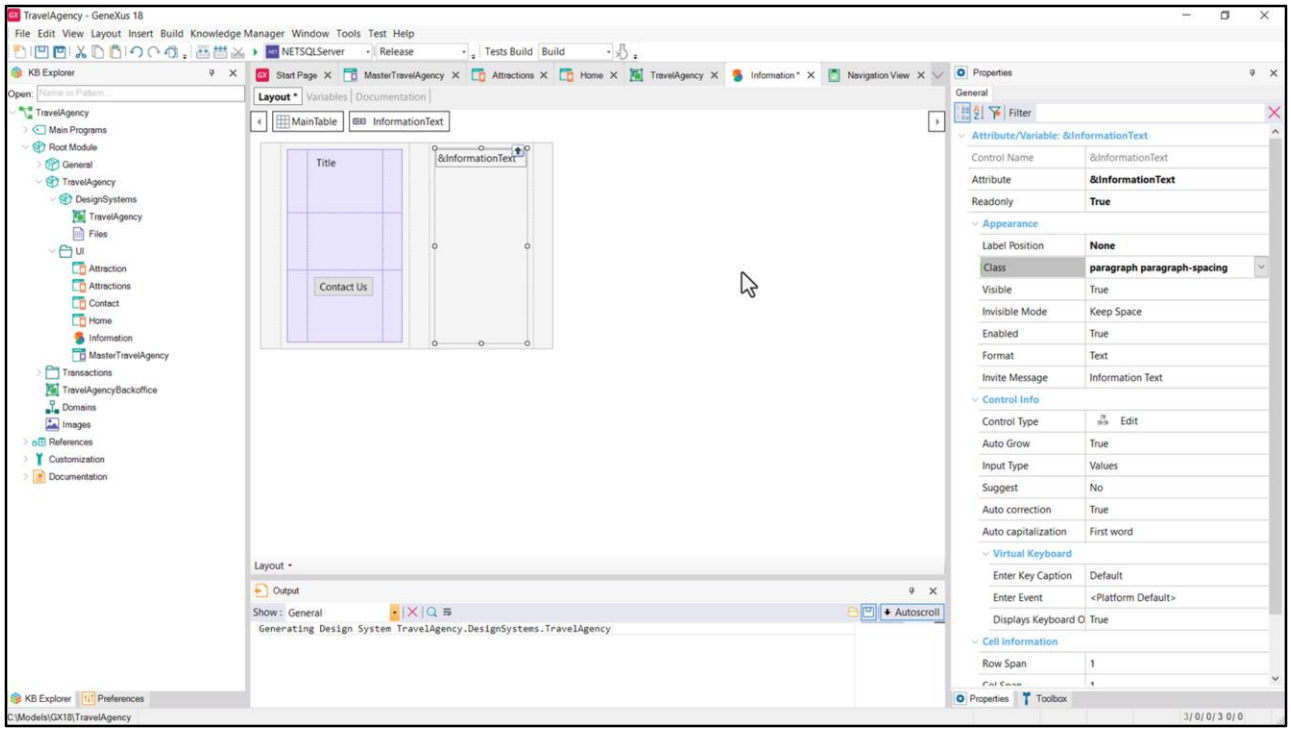
Neste caso, por exemplo, nós aplicamos a classe h2 a um controle do tipo textblock e a classe paragraph a um controle do tipo atributo/variável, que claro, neste caso são tipos de controle muito parecidos. Mas isso não é obrigatório.

Por outro lado, para dar estilo a um controle, não é necessário colocar todas as características de estilo dentro da mesma classe. As classes podem ser compostas, seja definido dentro da aba de styles classes compostas por outras classes; e isso será feito com a regra include; ou atribuindo ao controle mais de uma classe.



Por exemplo, suponhamos que queremos deixar separado o que são as características tipográficas de nosso parágrafo, do que são as características que tem a ver com o espaçamento.

Então, o que podemos fazer é remover esta propriedade daqui, a da margem, deixar então a classe paragraph limpa em termos tipográficos… e selecionar outra classe, a que por exemplo eu posso chamar assim… e onde, ali, nessa classe, é que defino a característica da margem.



E depois, o que faço (vou gravar) é vir ao meu stencil, e em vez do controle variável ter somente a classe paragraph, vou adicionar também esta outra. Então, a partir de agora, nosso controle terá duas classes.

Bom, e este é o tipo de decisão que vamos ter que ir tomando: se queremos isolar, separar, então, algumas propriedades em classes diferentes, para poder compô-las entre si.

Bom, visto tudo isso, já estamos em condições de começar a ver como damos estilo ao botão. Nos vemos no próximo vídeo.


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