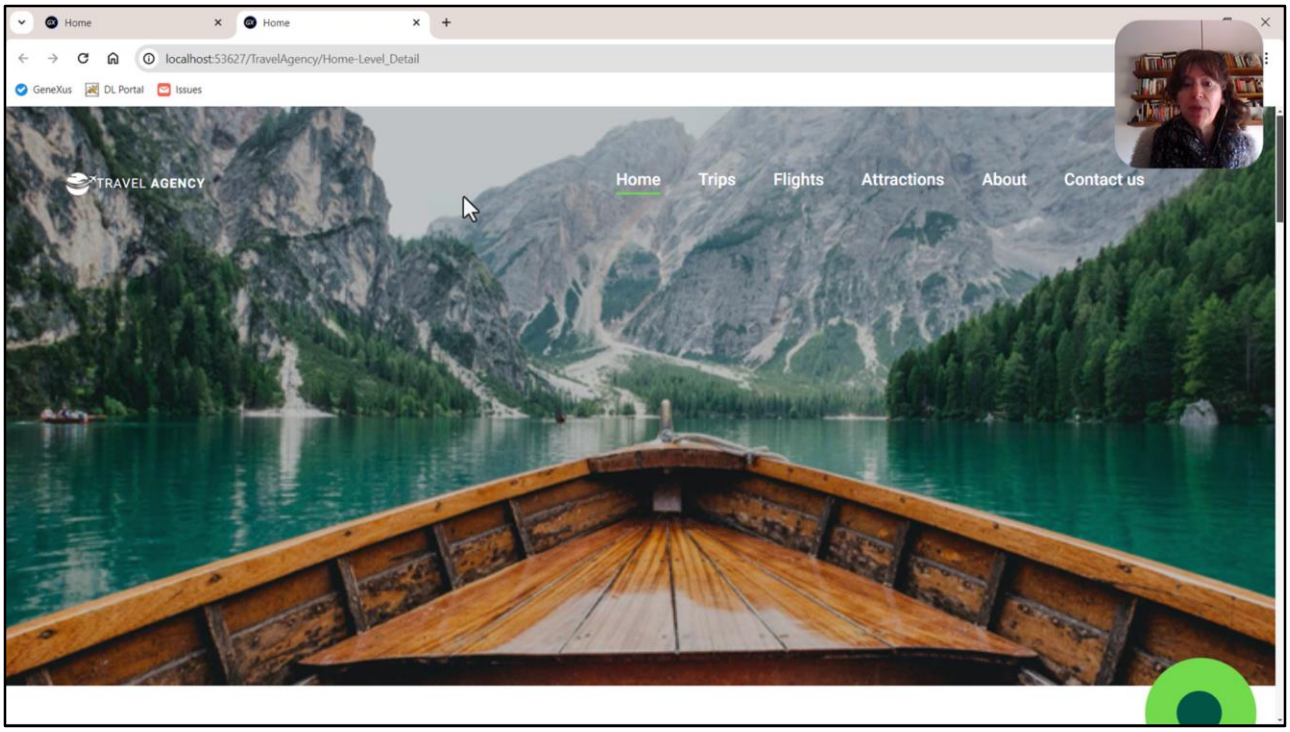
GeneXus by Globant

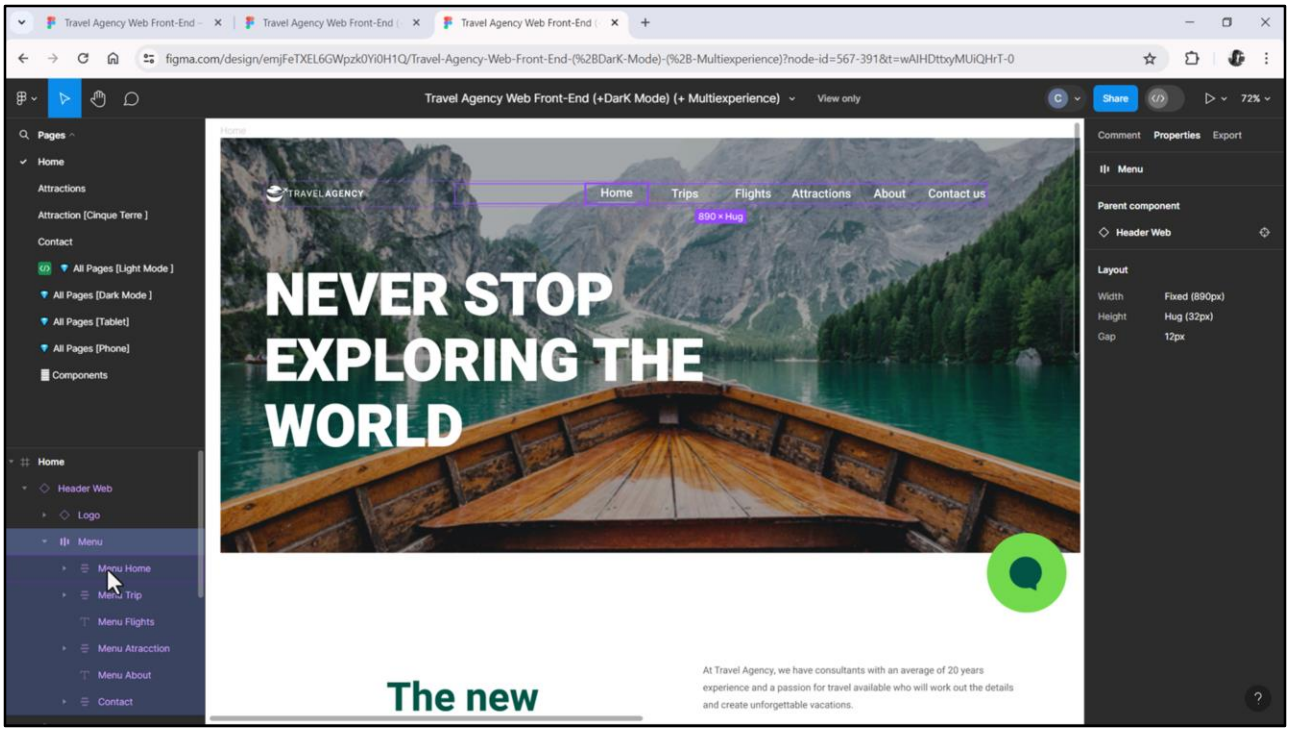
# Header in GeneXus: logo, menu and title

Cecilia Fernández



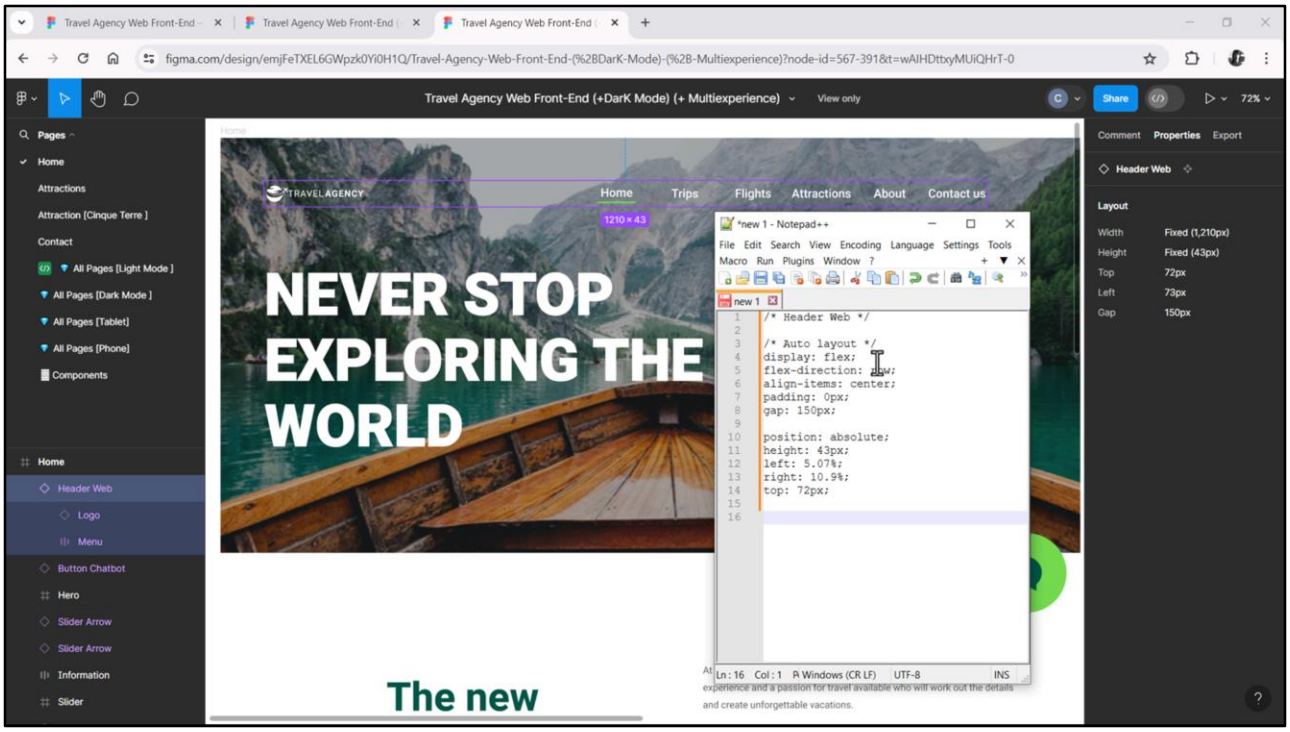

Bom, no vídeo anterior nós conseguimos isso, ou seja, conseguimos implementar a imagem do Header. O que faremos agora neste vídeo é implementar o logotipo e o menu. E vamos deixar para outro vídeo a implementação da mudança de tela nas navegações, ou seja, quando eu, por exemplo, clico nessa opção Attractions, não apenas carregue a página das atrações aqui, que será no contentplaceholder, como também mude tudo o que precisa, ou seja, a imagem Hero e também o texto que a sobrepõe.

Se tivermos tempo neste vídeo, também implementaremos o texto. Senão, deixaremos para o próximo.



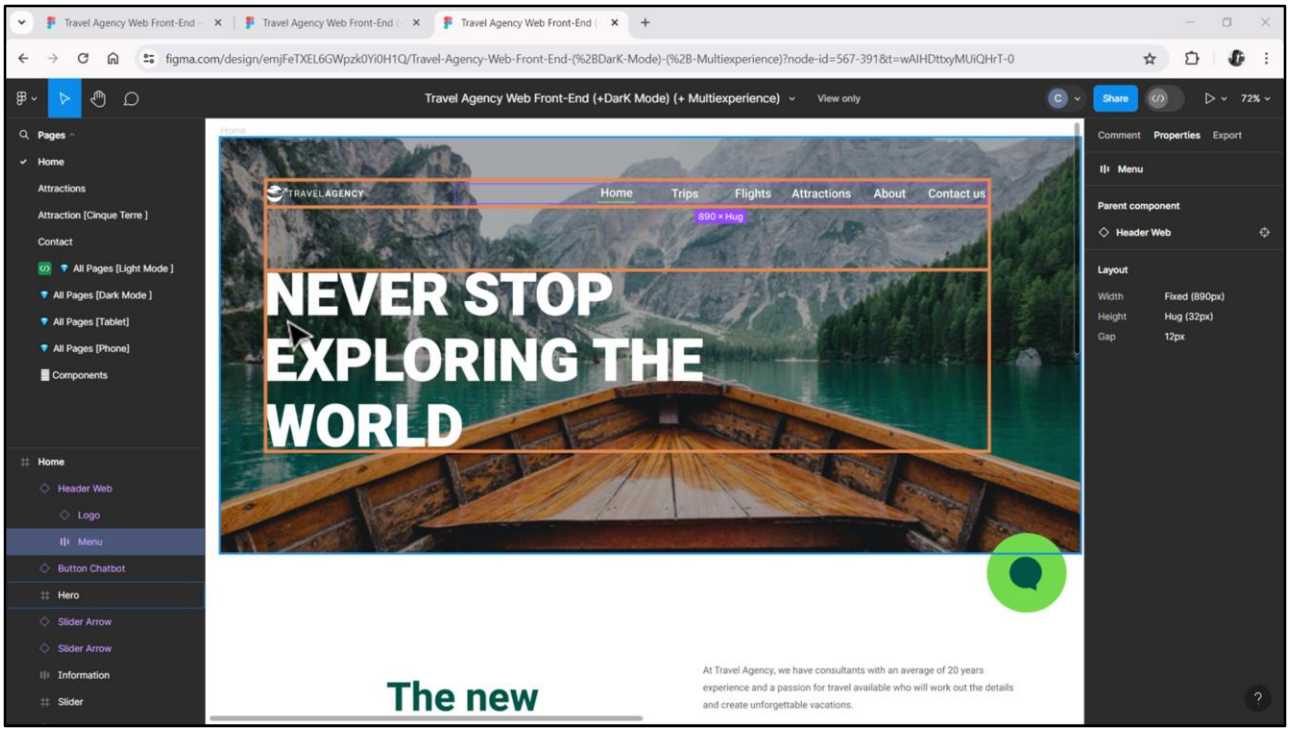

No Figma vemos que são implementados como um componente que contém:

- Outro componente com o logotipo: que por sua vez é composto por um ícone e dois textos.
- E um container com auto layout, ou seja, Flex, para o menu, ou seja, uma sequência de itens.



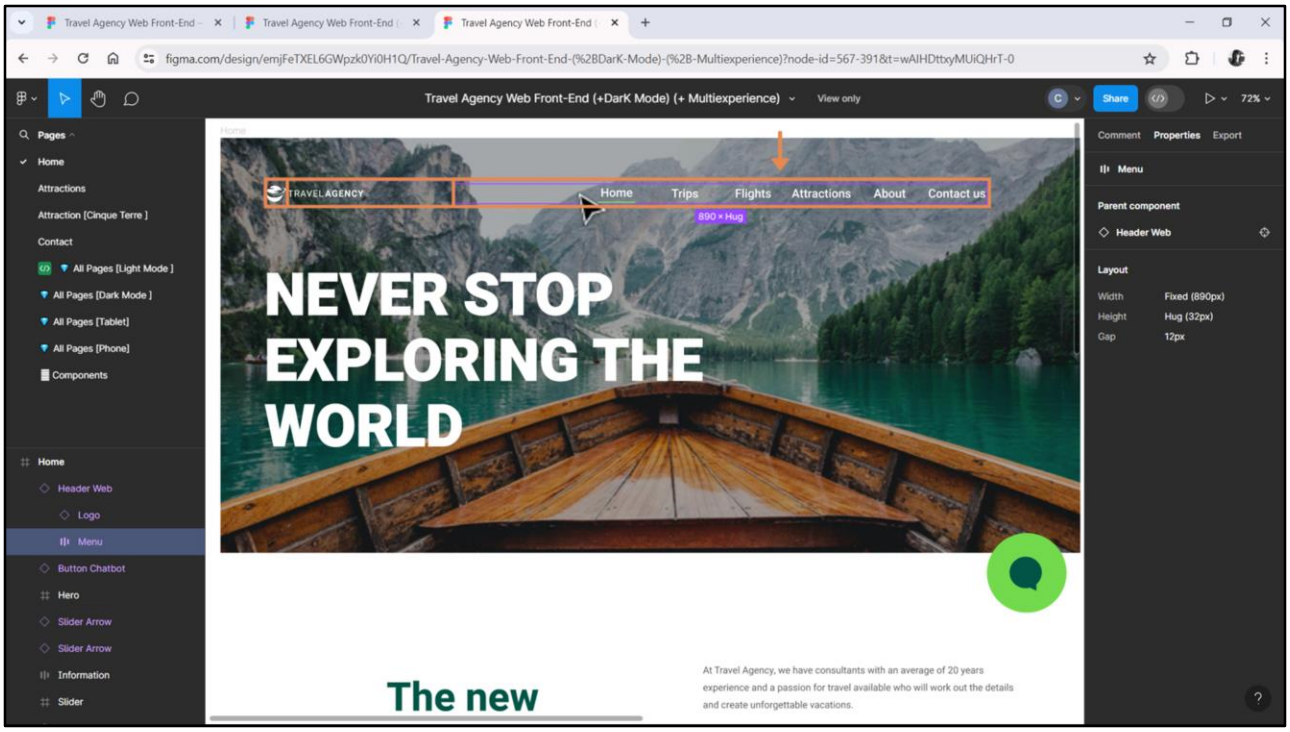

También podemos ver que Chechu agrupou ambos os componentes neste “Header Web” que também é um container com auto layout (vemos isso claramente neste gap que aparece aqui). E se extrairmos as propriedades CSS terminamos de verificar.

E aqui vemos, por exemplo, graficamente, esse gap entre os dois elementos: o menu e o logotipo.

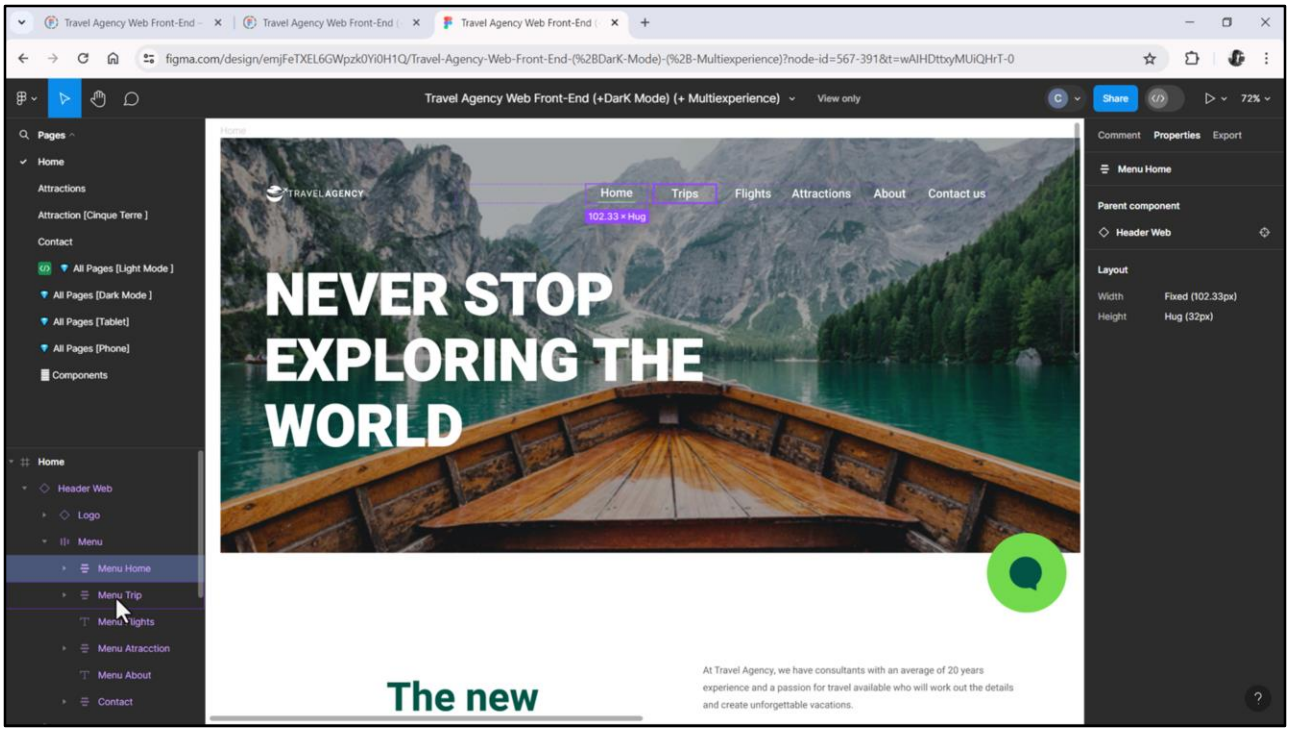



Mas não vamos implementá-lo como um flex, e sim como uma tabela, porque, como podemos ver, o logotipo está alinhado à esquerda com este título. Então será mais fácil utilizar uma tabela para alinhá-los.

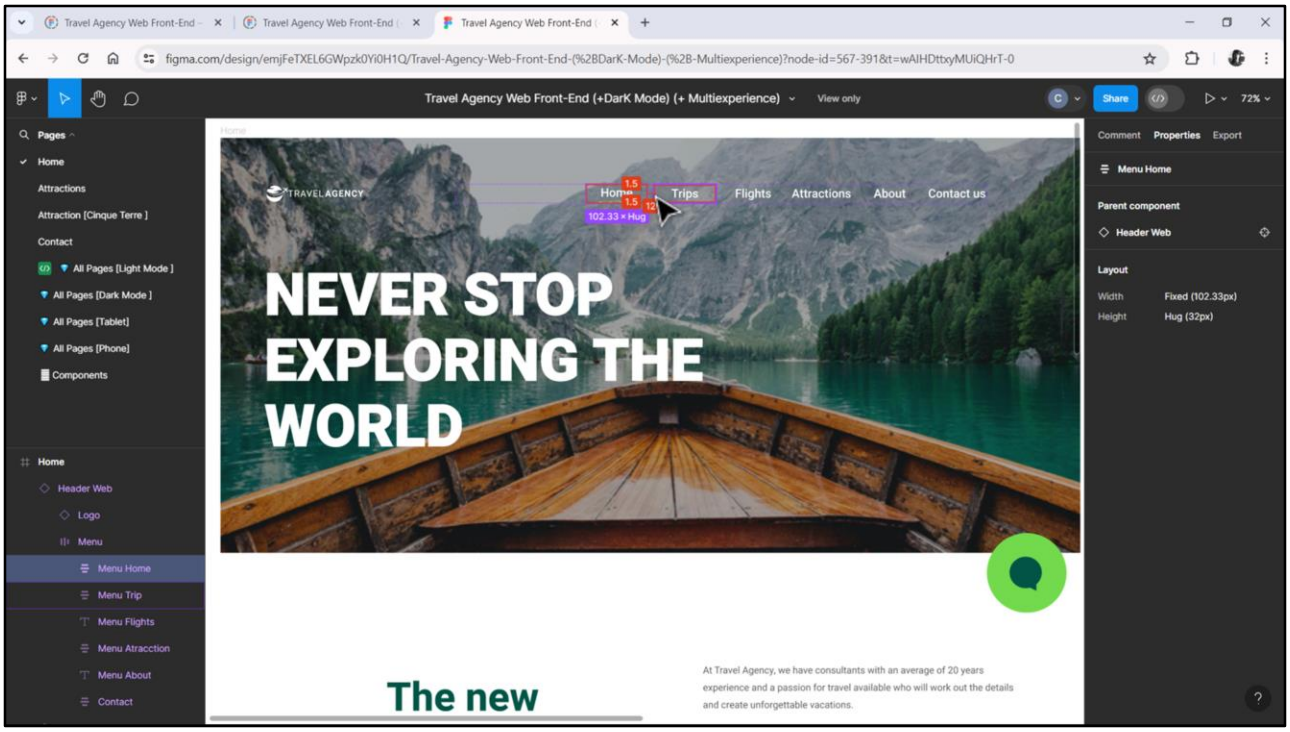
Tabela que terá 3 linhas, a do meio para espaçamento. Por enquanto, vamos criar uma única linha, pensando apenas no logotipo e no menu. Depois adicionaremos as outras duas.




Sempre podemos imaginar muitas maneiras de implementar um layout. Vou escolher colocar o ícone em uma coluna, as palavras “Travel Agency” do logotipo em outra coluna e uma terceira coluna para o menu propriamente dito, que escolherei implementar exatamente como Chechu fez, com um flex.

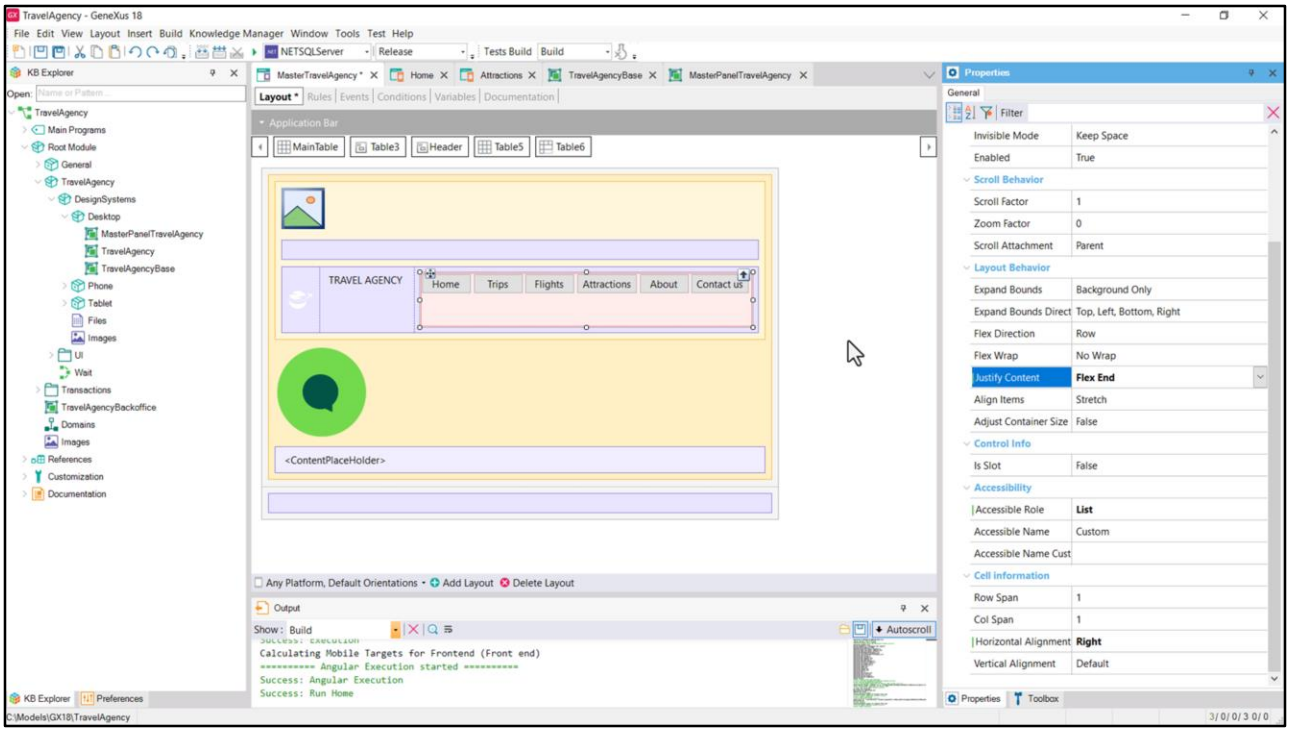



Será uma sucessão na direção horizontal de botões: um para cada ação. Serão botões e não textblocks por tudo o que dissemos quando falamos sobre acessibilidade. Prefiro um flex neste caso em vez de uma tabela, porque me dá mais flexibilidade: cada botão ocupará o espaço que precisa, de acordo com a quantidade de letras de seu texto (isso é especialmente importante se quisermos a aplicação para vários idiomas). E também posso distribuí-los igualmente através da propriedade gap aplicada à classe do flex.




Aqui Chechu não percebeu (e digo isso com convicção porque acabei de perguntar a ela) que o que fez foi fixar as larguras dos espaços de cada texto e foi isso que distribuiu igualmente, a 12 pixels um do outro, mas dessa forma, repare que o espaço entre o texto Home e Trips ficou muito maior do que entre Attractions e About, por exemplo. E isso é um erro.

Por outro lado, se eu estivesse interessado, caso não houvesse largura de tela suficiente para conter todos os textos, em vez de ter uma barra de scroll horizontal seja feito wrap, e alguns dos textos, das opções, digamos, do menu, aparecessem em uma segunda linha, então eu não conseguiria isso com uma tabela, mas consigo com o flex.

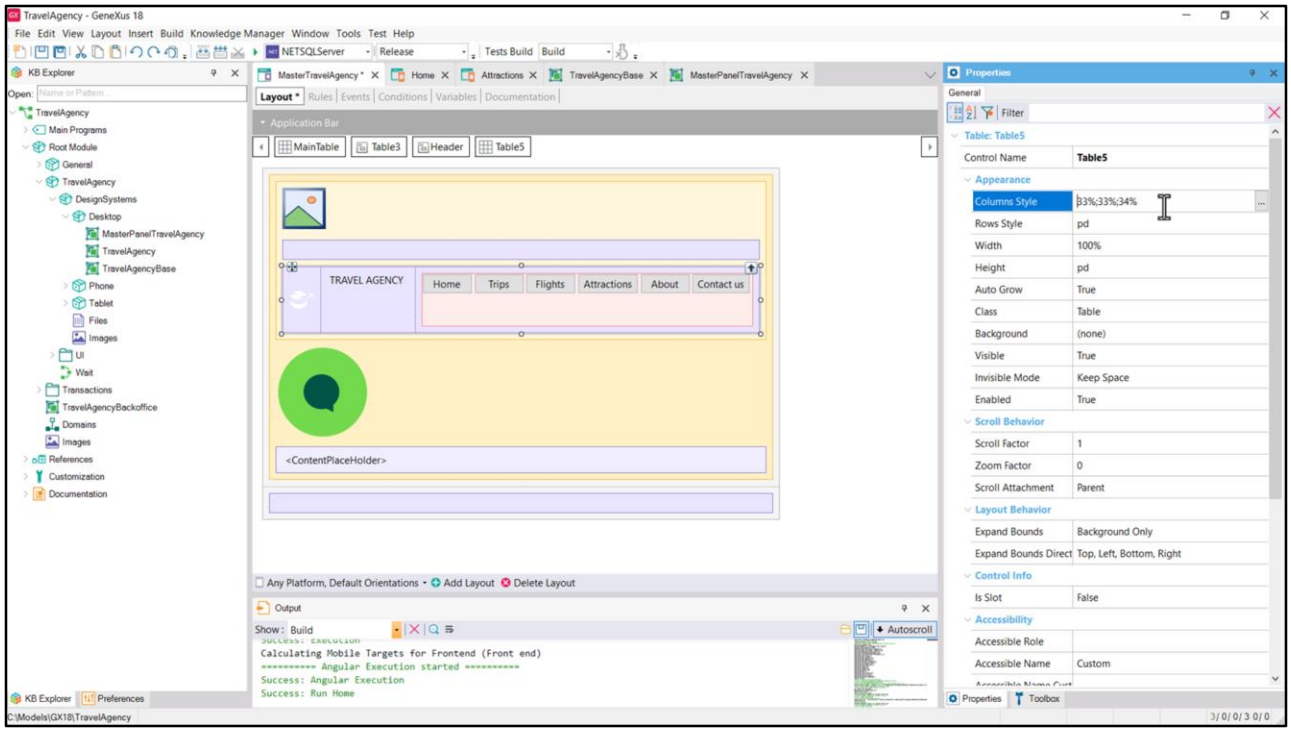



Então vamos começar inserindo uma tabela com 3 colunas.

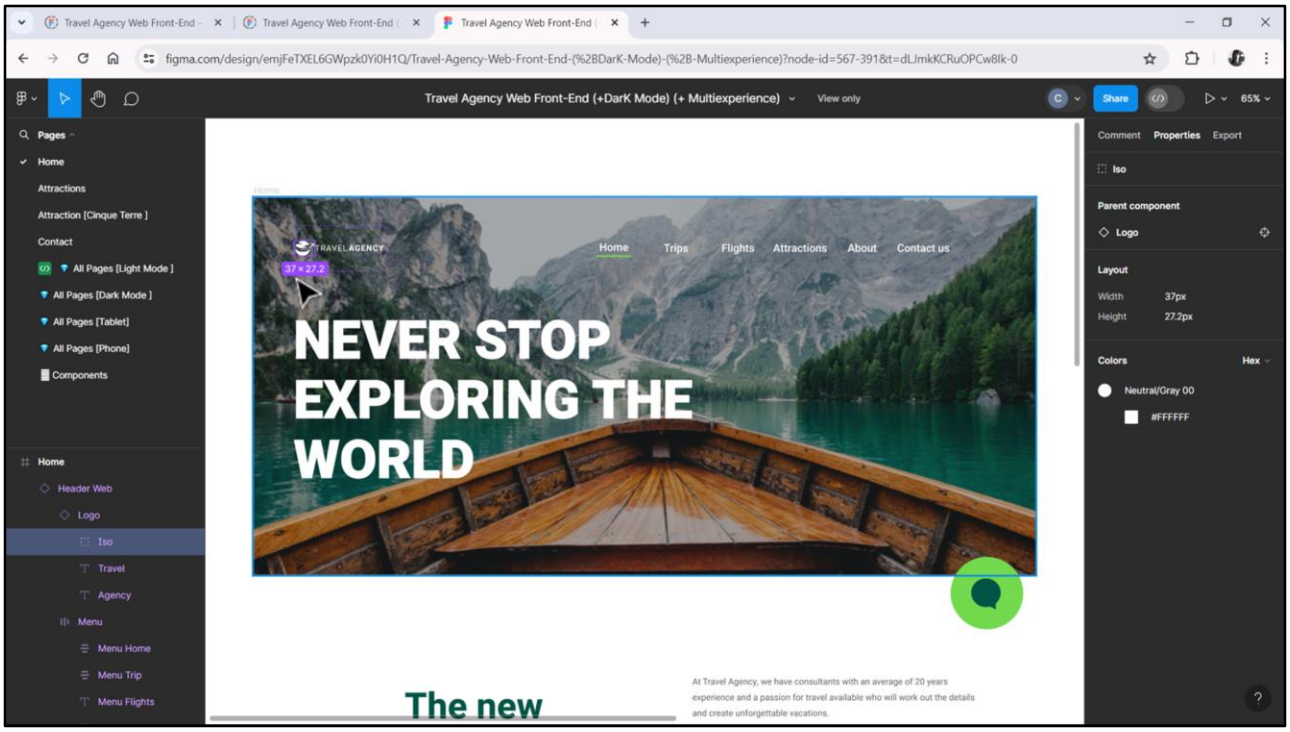
Na primeira colocaremos o ícone de Travel Agency. Na segunda colocaremos um textblock, de caption (por enquanto) TRAVEL AGENCY. E na terceira colocaremos um container Flex.

Nesse flex já inserimos os 6 botões: um para cada ação. A este container flex vamos atribuir a Accessible Role: List, para indicar justamente que será uma lista de itens. Já podemos aproveitar e dar alinhamento horizontal pela direita.

A direção do Flex, vemos que é a correta, é a de linha. Para Wrap, por enquanto vamos deixar a opção default que é não fazer wrap. A justificação do conteúdo vamos alterar para Flex End, para que todos os botões sejam justificados em relação ao final do flex e não ao início, para que sejam posicionados deixando o espaço livre na frente e não atrás.
O alinhamento dos itens em relação ao outro eixo, o eixo y, queremos que seja centralizado, mas por enquanto deixarei o default, veremos isso depois.

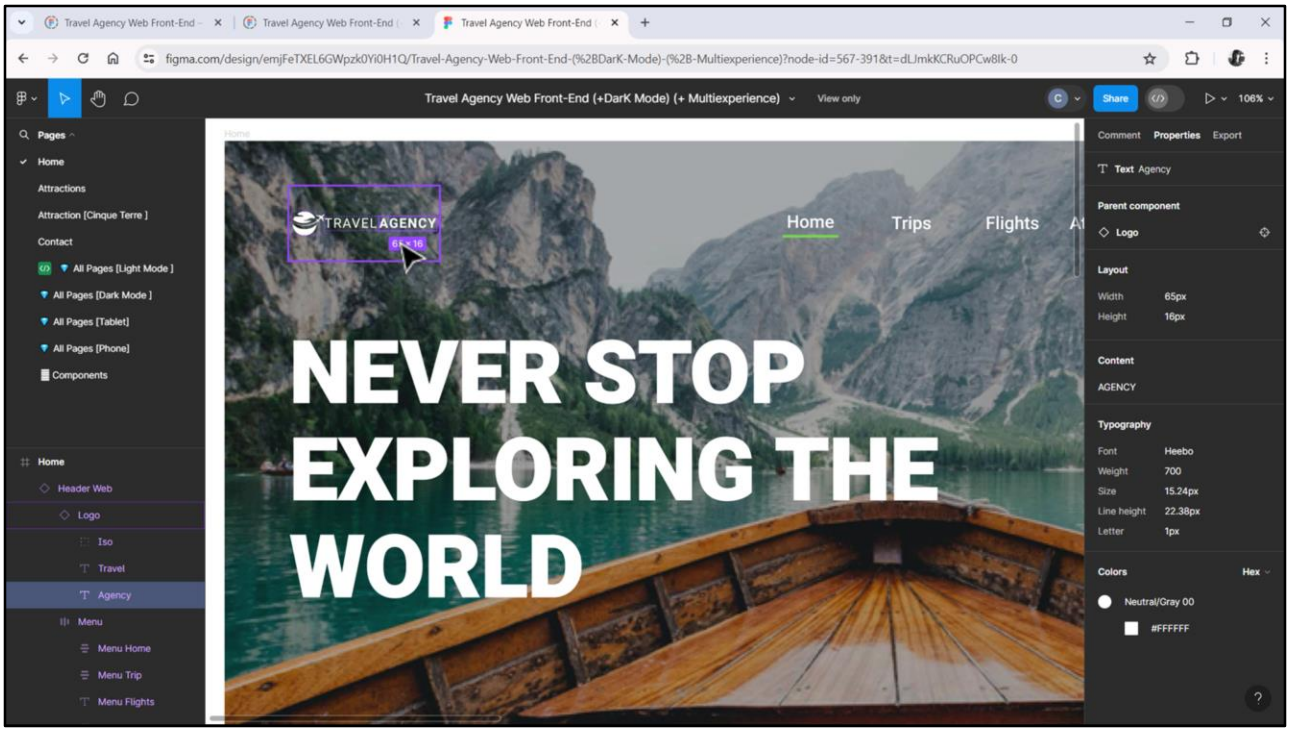



Agora vamos definir as dimensões da tabela: as columns style, rows style, Height.

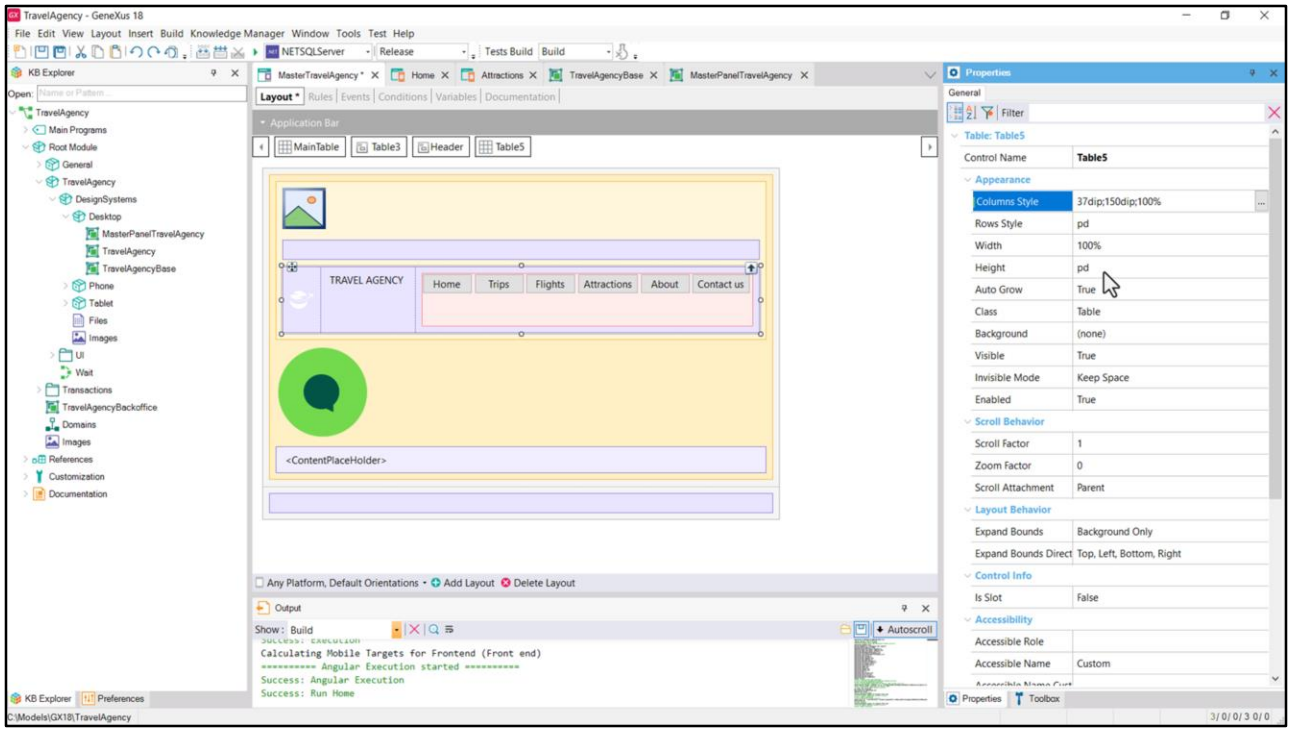



Poderíamos colocar de altura da tabela esses 43 dips.

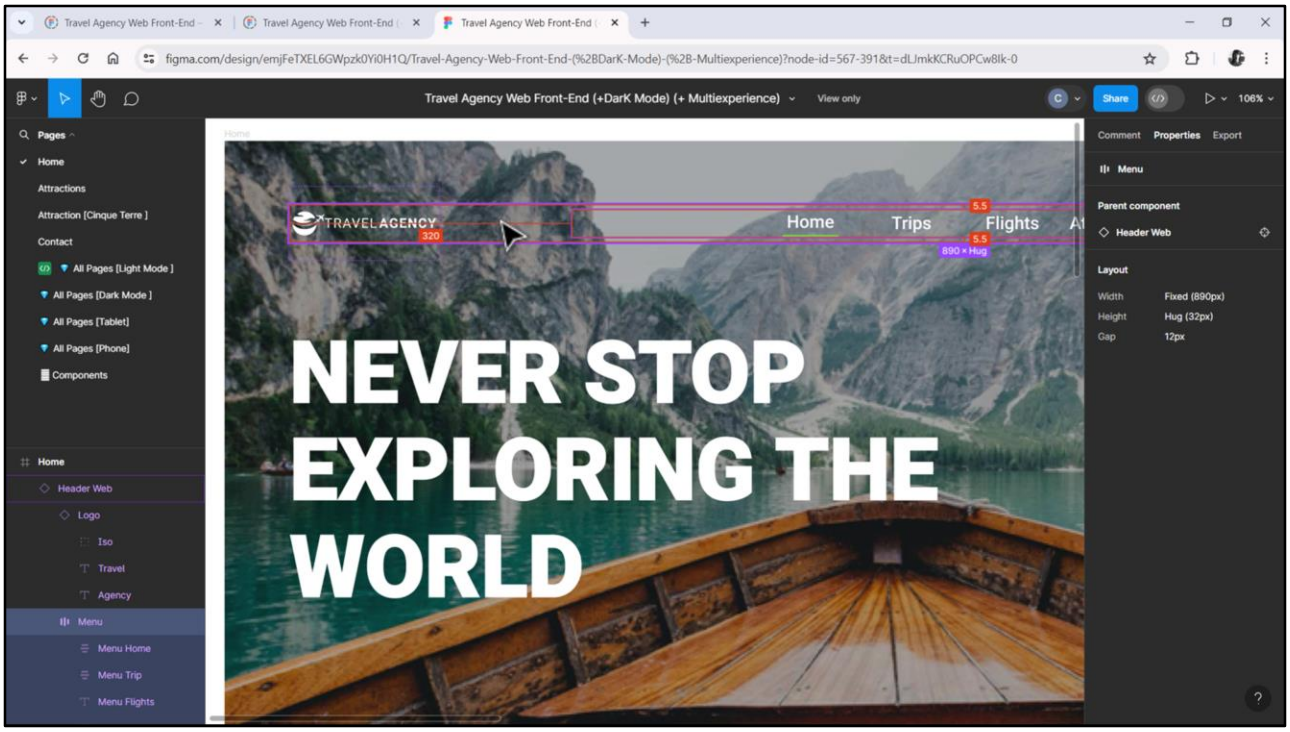
E então... o ícone terá 37 dips de largura, então daremos essa largura à coluna 1.



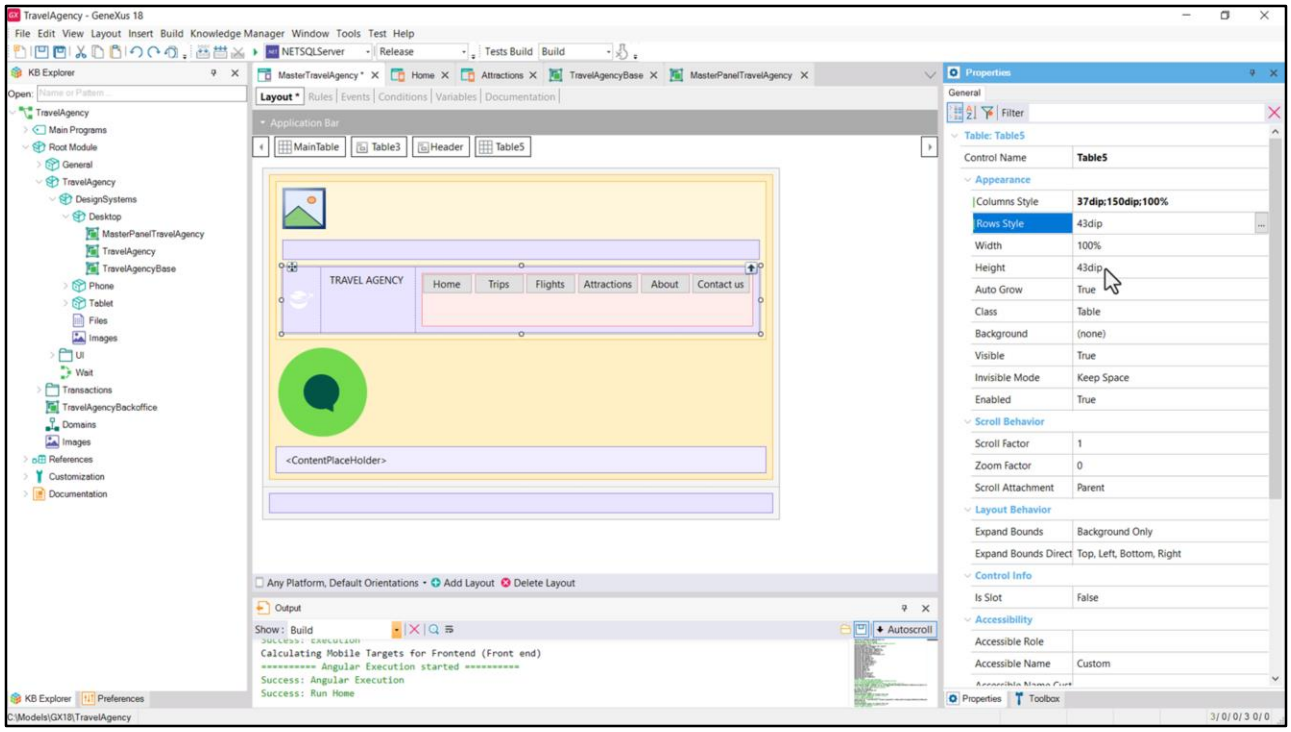

Depois, colado sem espaços, vem o texto Travel Agency, que soma 62 + 65, 127. Então poderíamos dar à coluna 2 uma largura de 150 dips, e à 3 100% da largura restante…



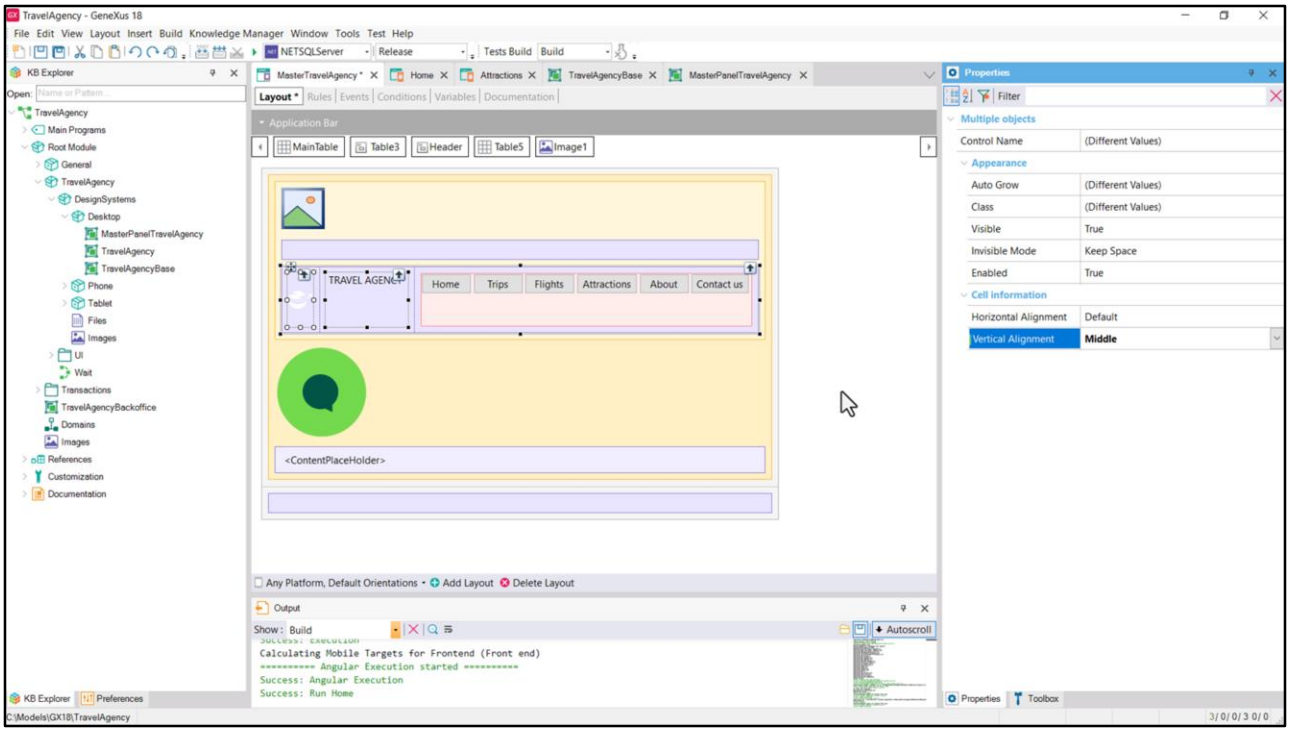

Então… 37 dips, 150 dips, 100%… E a altura da linha ou da tabela?




Seria este, de 43 dips. E vemos que todos os controles aparecem centralizados verticalmente em relação às suas bordas.

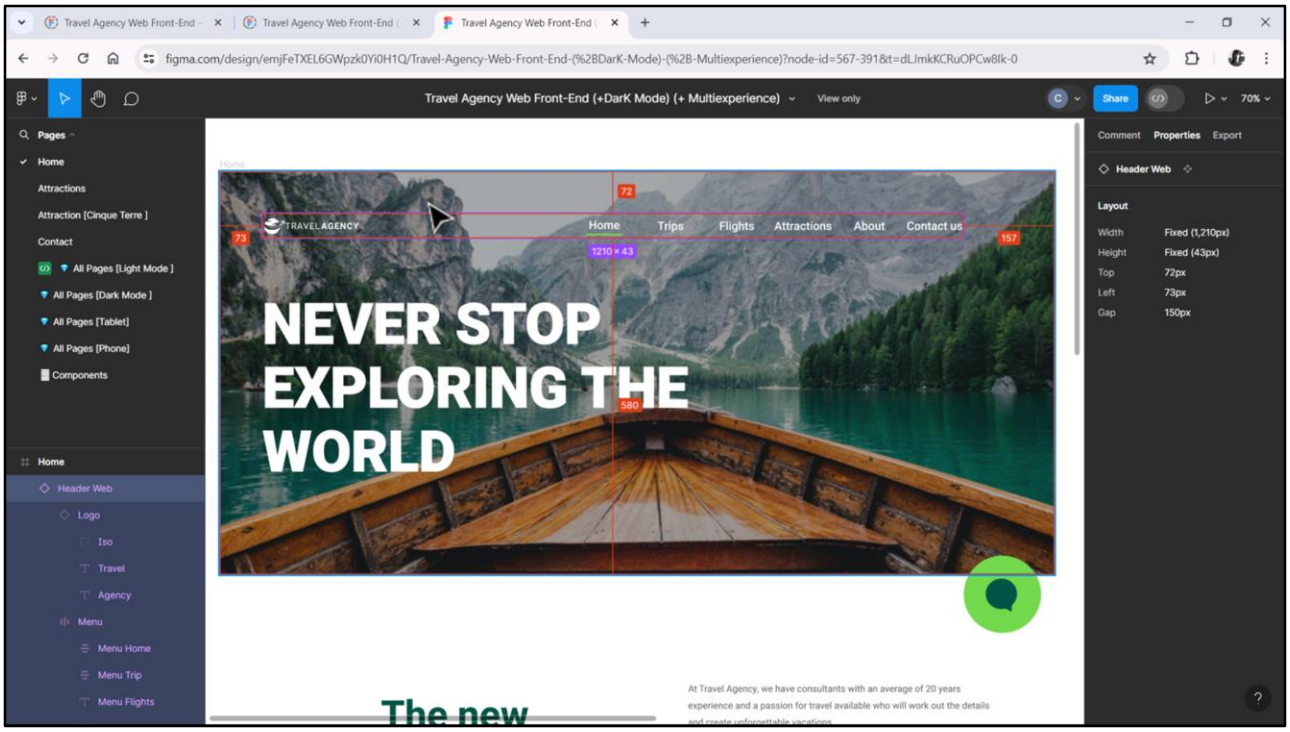



Bem, então coloco os 43 dips aqui. Automaticamente a Height ficará do mesmo tamanho…

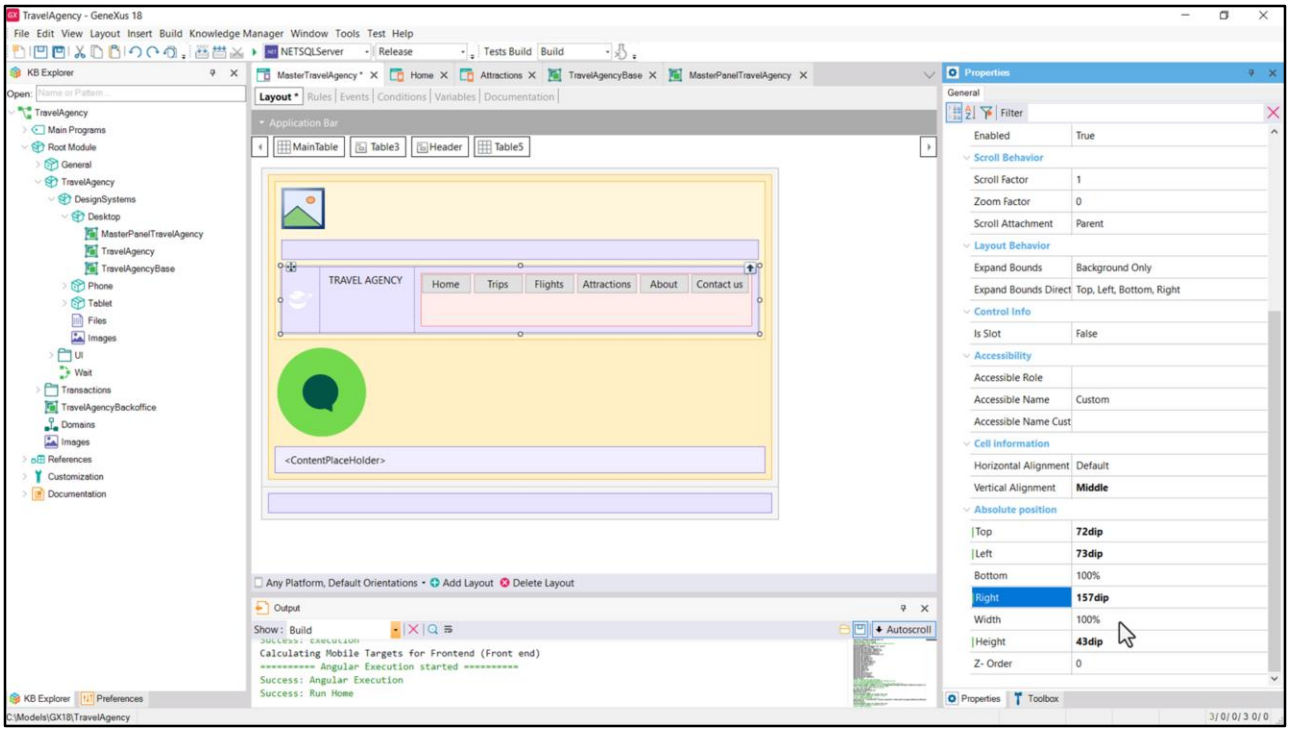



E então eu seleciono os três elementos… aqui o flex… e defino a propriedade Vertical Alignment para os três como Middle.

Bem, e agora, onde fica localizada essa tabela dentro do canvas que a contém? Temos que fornecer sua posição absoluta.

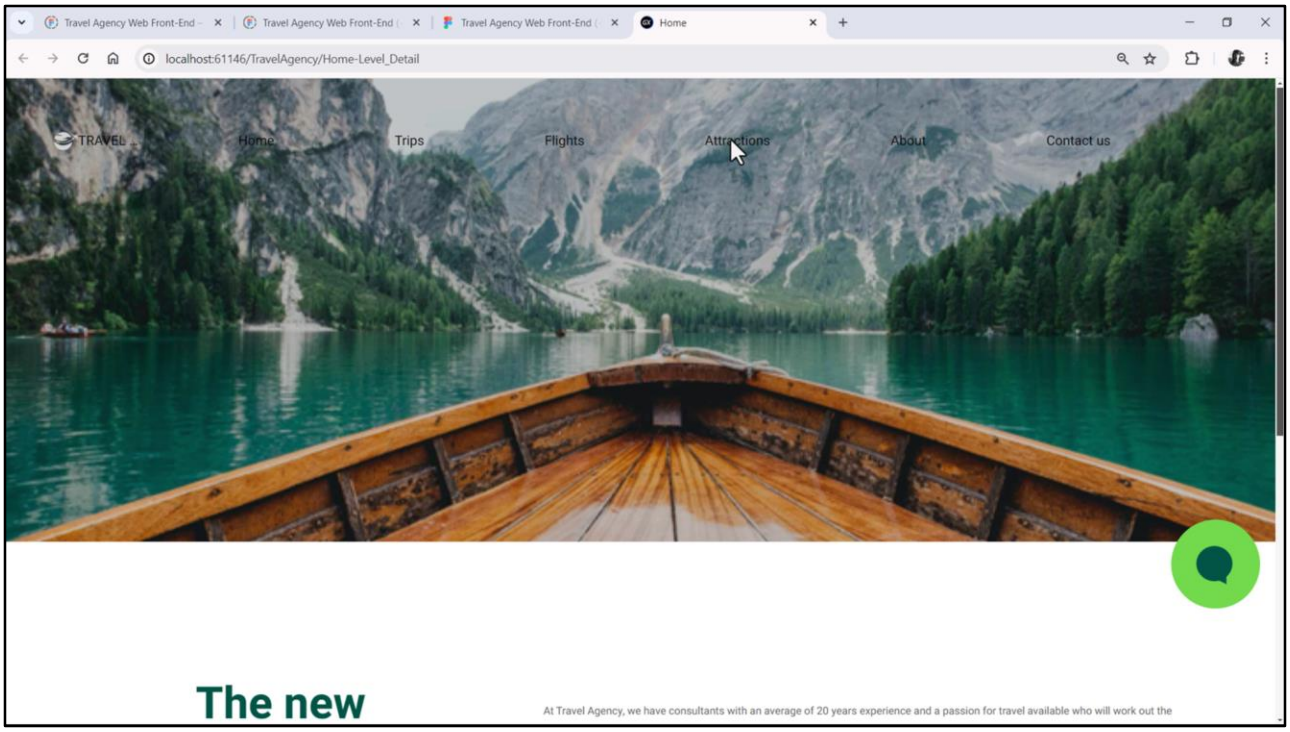



E é: de cima 72 dips, da esquerda 73 e da direita 157. Isso é o suficiente para nós. De bottom será os 100% restantes da altura do canvas.

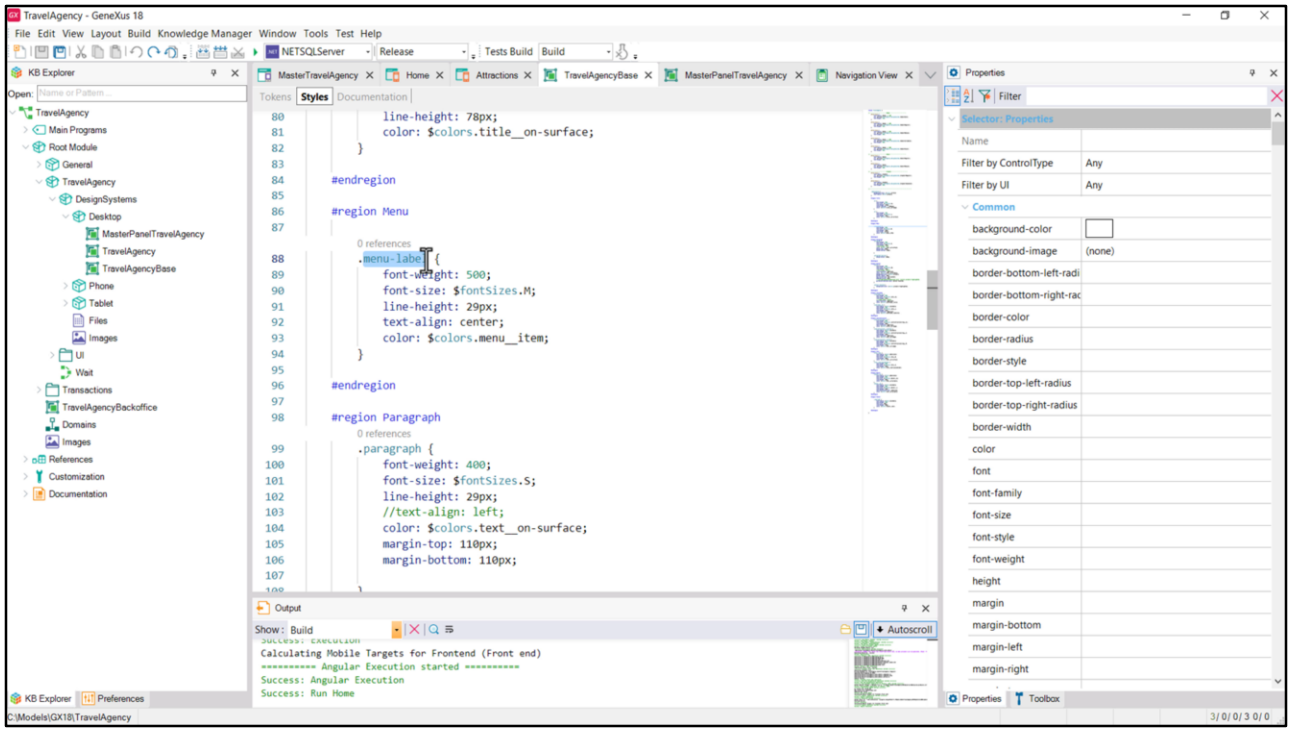



Então... top 72 dips, a altura da tabela era de 43 dips, e isso me deixa bottom dos 100% restantes.
Left 73 e Right 157.

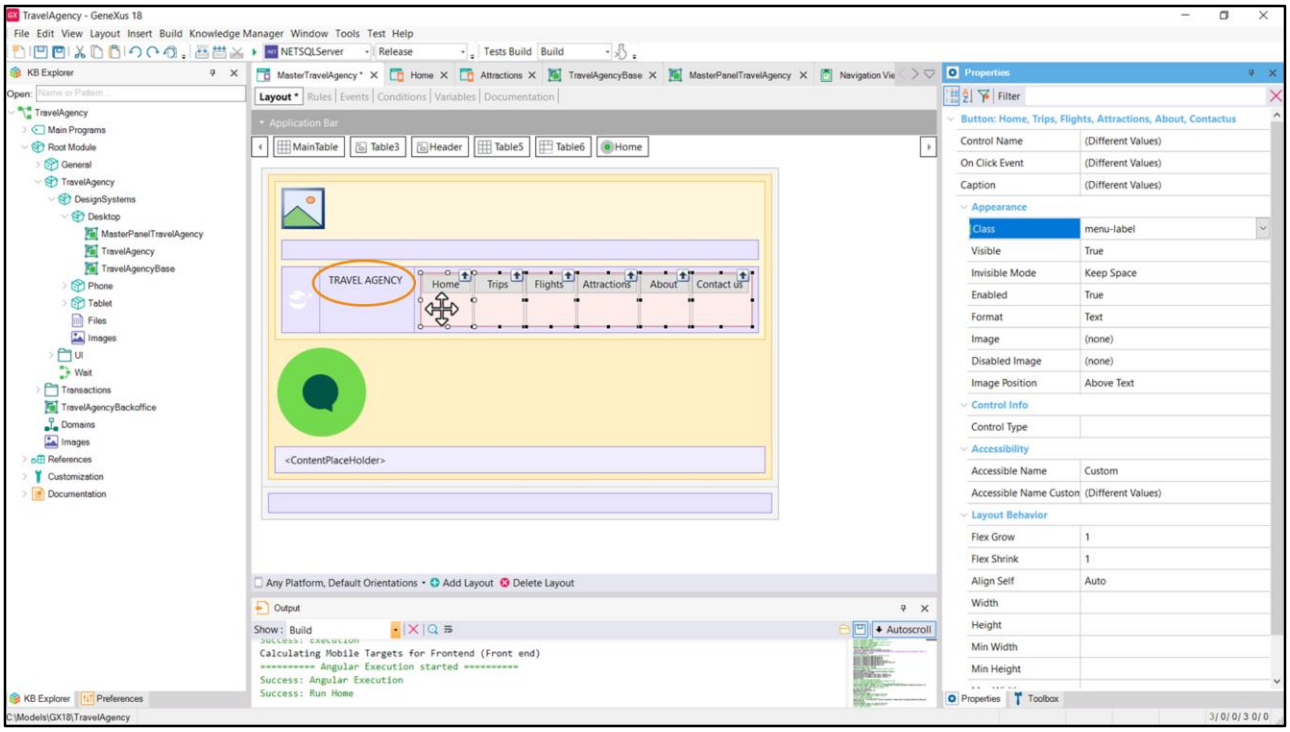
Vamos testar o que fizemos.




Bom, temos algumas coisas para trabalhar, certo? Para começar, os estilos tipográficos dos textos…

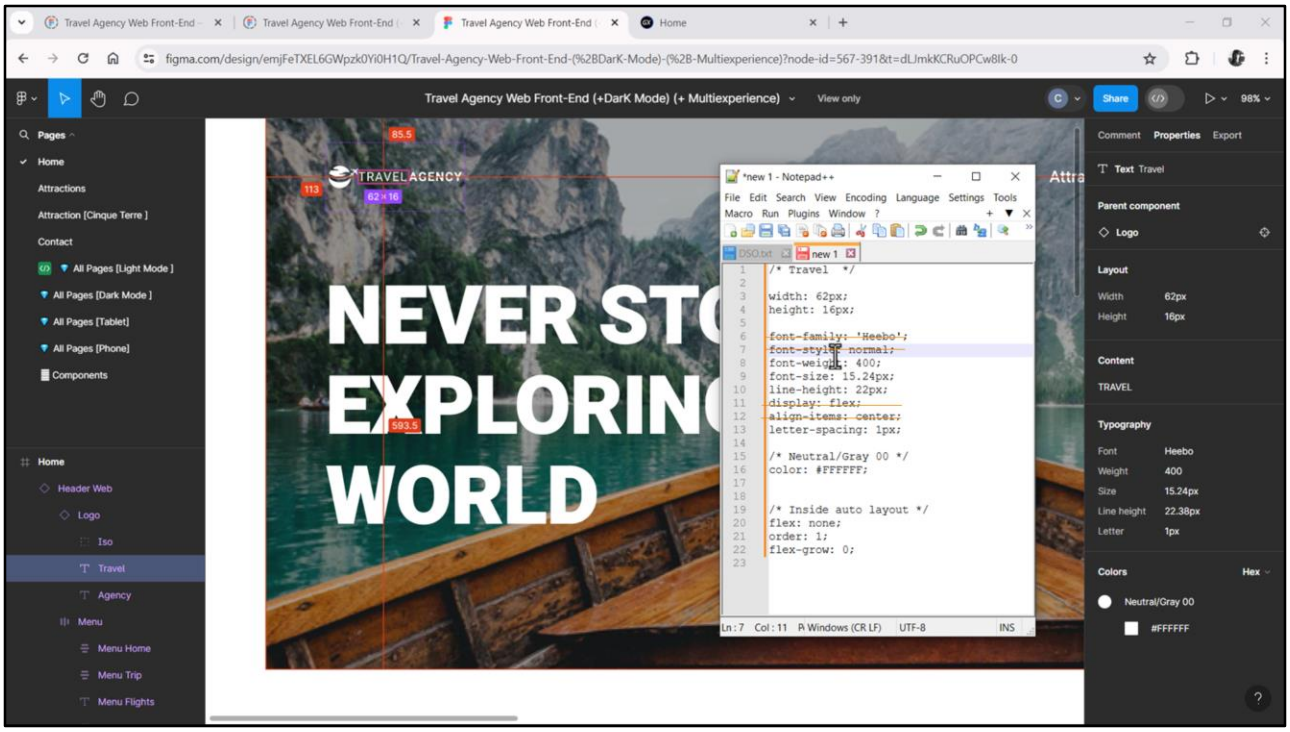



No DSO base tínhamos o estilo dos rótulos do menu…




Então vamos começar atribuindo esta classe a todos os botões de menu. Selecionamos todos eles e mudamos a classe para esta… vemos que efetivamente foi alterada em todos.

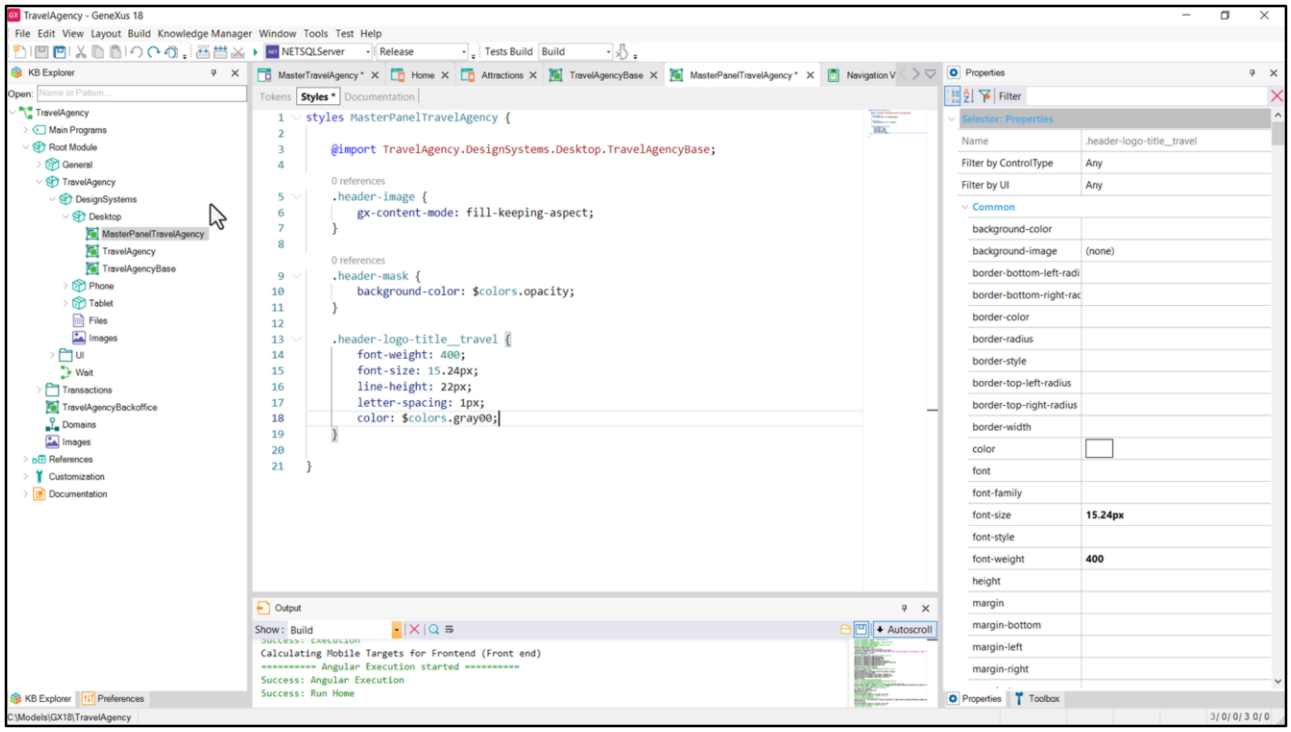
Não havíamos definido uma classe para o estilo tipográfico desses textos…



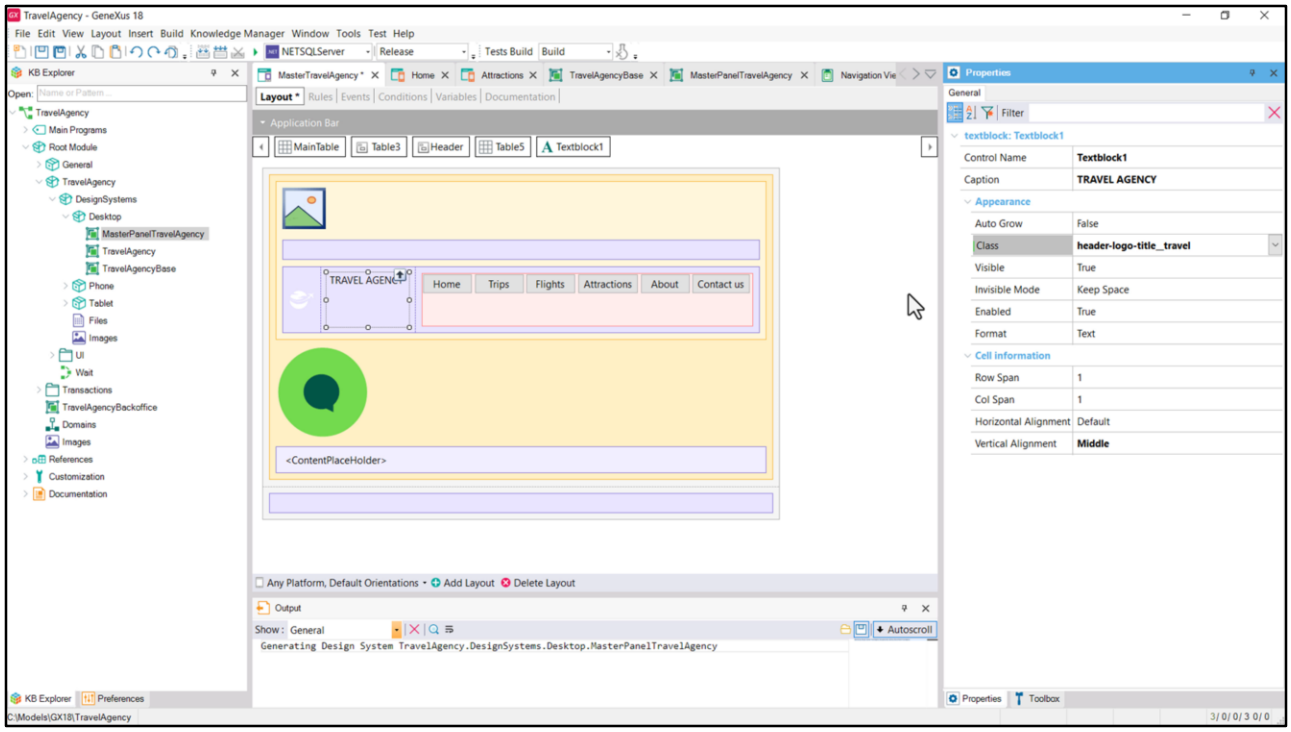

Vemos então que Chechu não criou estilo… e que a única diferença entre um texto e outro é o peso da fonte. Um é de 400, o outro é de 700.

Bom, vamos copiar o CSS porque precisaremos definir uma classe para esses textos. Removo estas propriedades que serão aquelas da fonte default, então não vou precisar delas... removo essas duas que são do elemento no flex de Chechu, e a que eu preciso adicionar além dessas 4 será a cor de nosso token gray00.

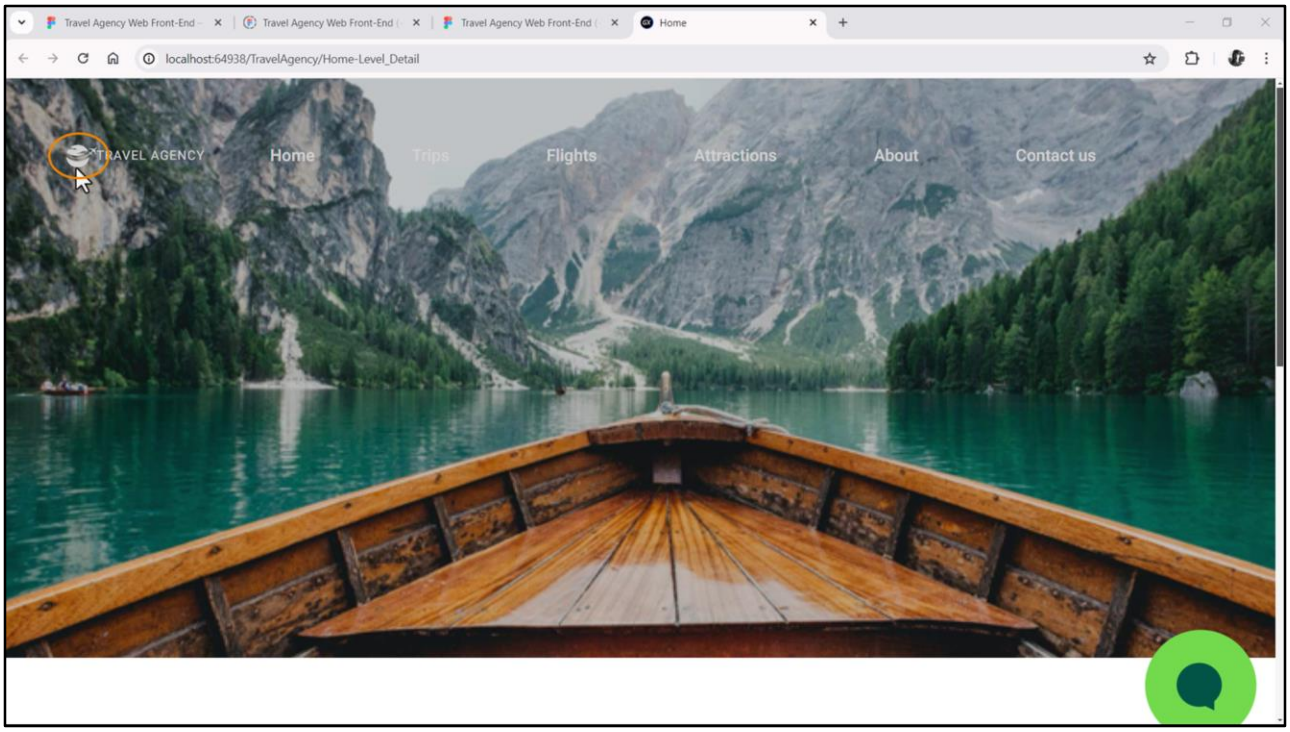
Copio estas 4…




E no DSO do Master Panel eu adiciono uma classe que eu vou chamar… assim. Depois vamos ver o porquê desse “travel” no nome…
Colo as propriedades… Adiciono a de cor…



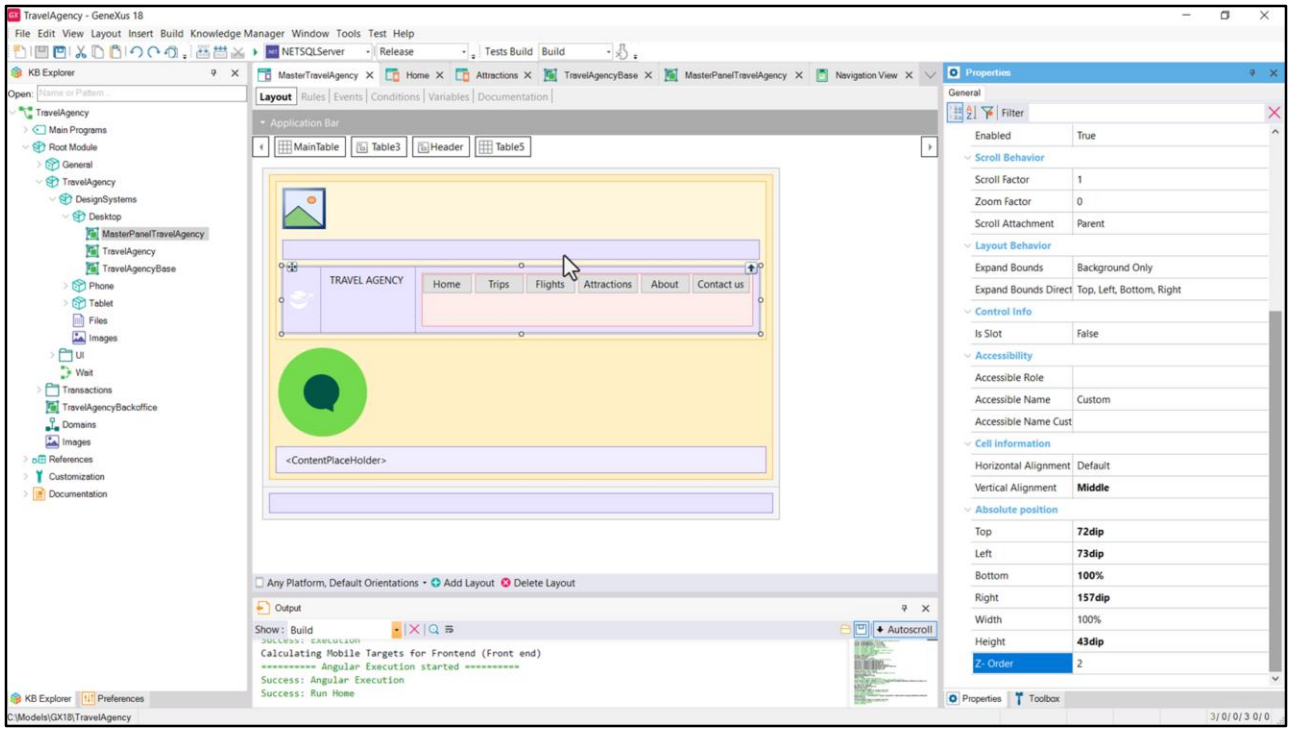

…E associo a classe ao textblock.



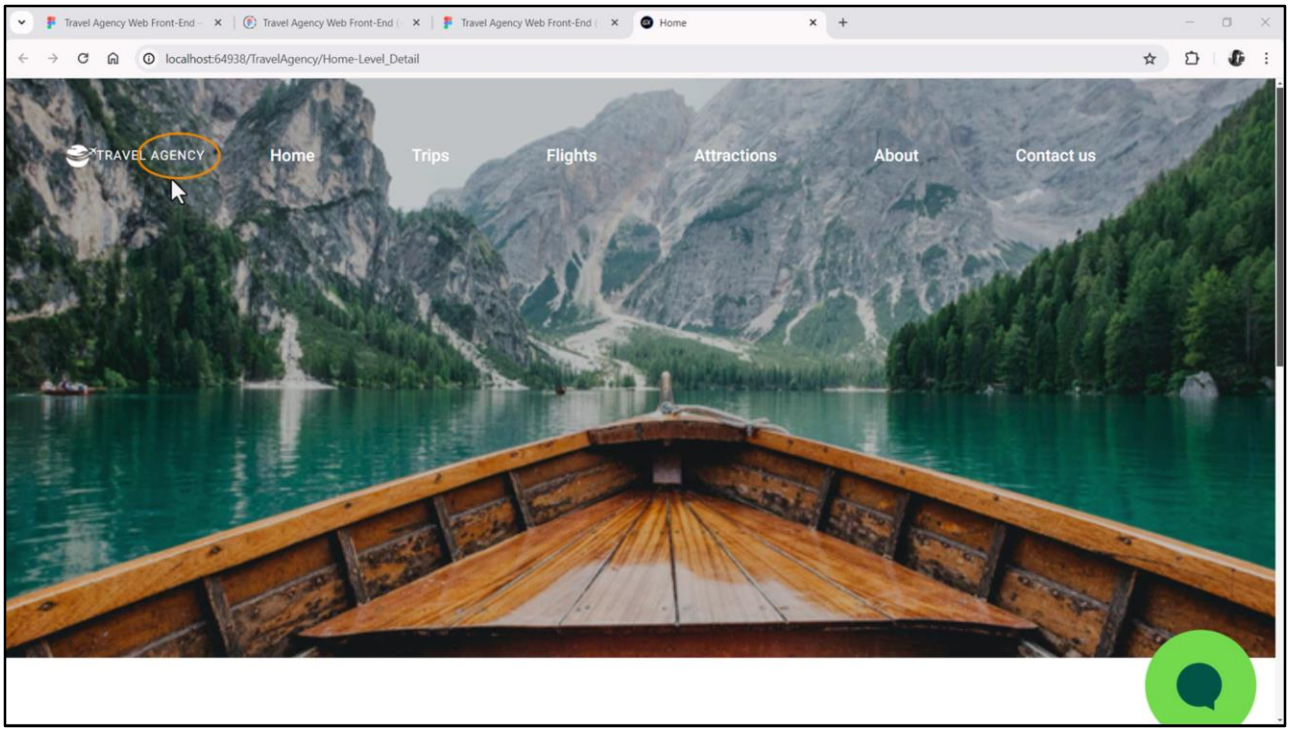

Vamos testar…

Se este ícone não aparecer, tentem fazer “Rebuild all” e tentem novamente. Ou insiram novamente o ícone na KB. Quando movemos o módulo dos ícones no vídeo de Assets, não sei se lembram, podem ter ficado internamente mal localizados ali. Assim como eu digo é arrumado.

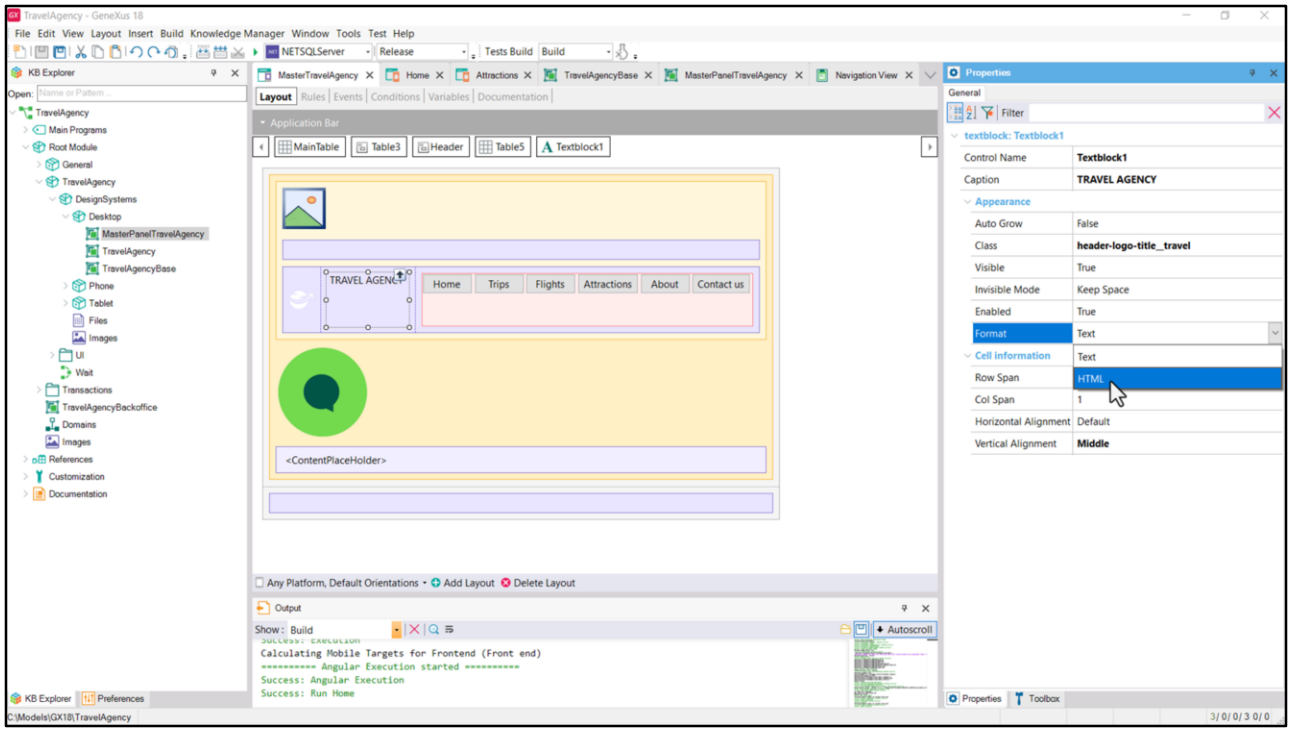
Algo estranho está acontecendo... Estão aparecendo em cinza esses textos em vez de branco... Eles estão ficando sob a máscara.



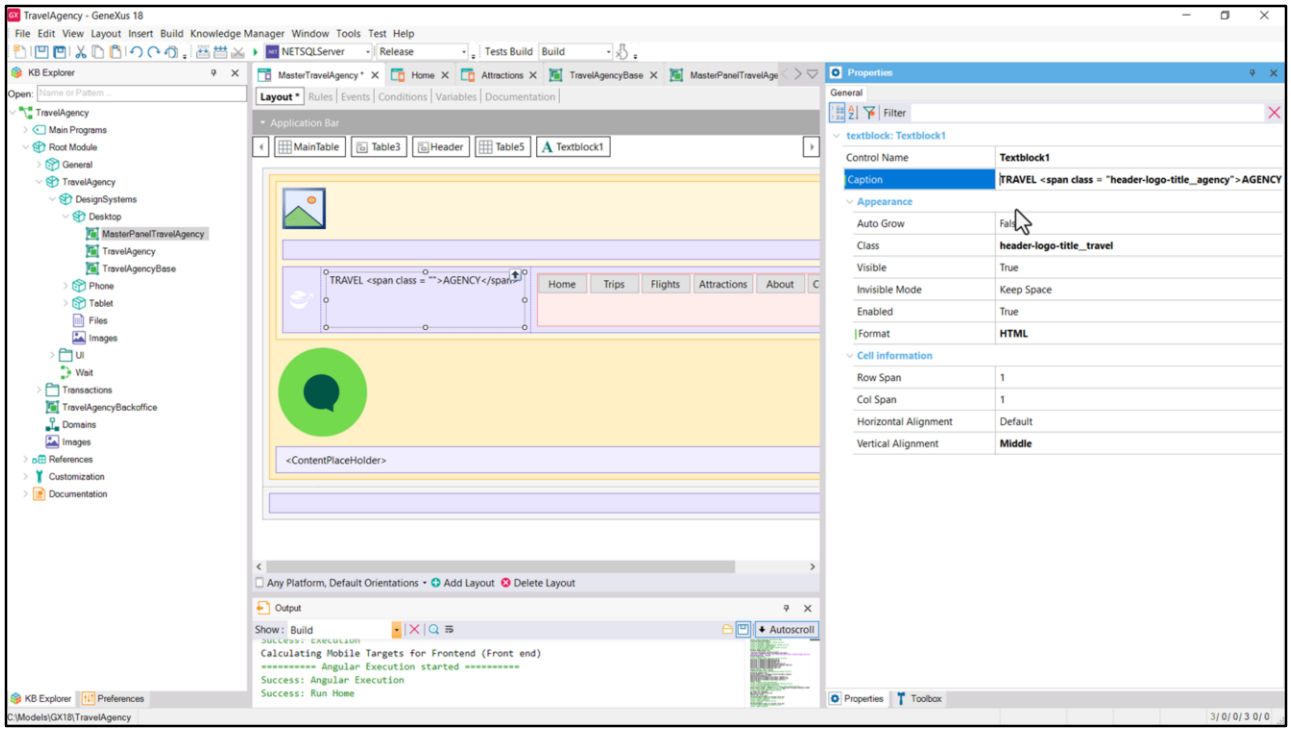

Deixei a Z-order default para esta tabela, o 0, e a máscara tinha 1. Vou atribuir o 2 para ela, para que fique por cima. Vamos testar agora…




Bom. Agora precisamos trabalhar no menu, que não está certo, mas também precisamos mudar para este texto o peso da fonte para que apareça de 700 e não de 400. Vamos começar por aí, para que possamos depois nos concentrar no restante e com mais calma ao menu propriamente dito.

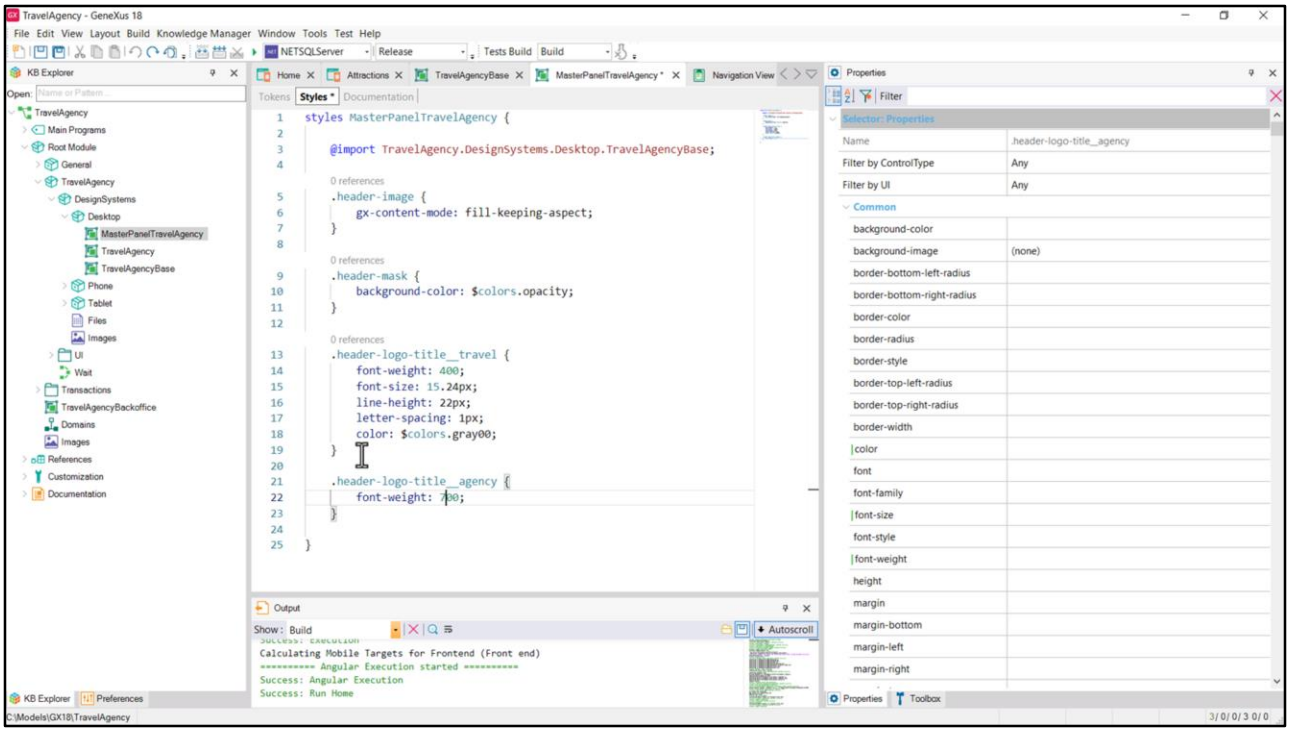



Como estamos na aplicação Web, podemos alterar o formato do texto para HTML. Isso é para que o Caption possa ser interpretado como html. Isso nos permitirá atribuir outra classe ao texto AGENCY através da tag span de HTML.

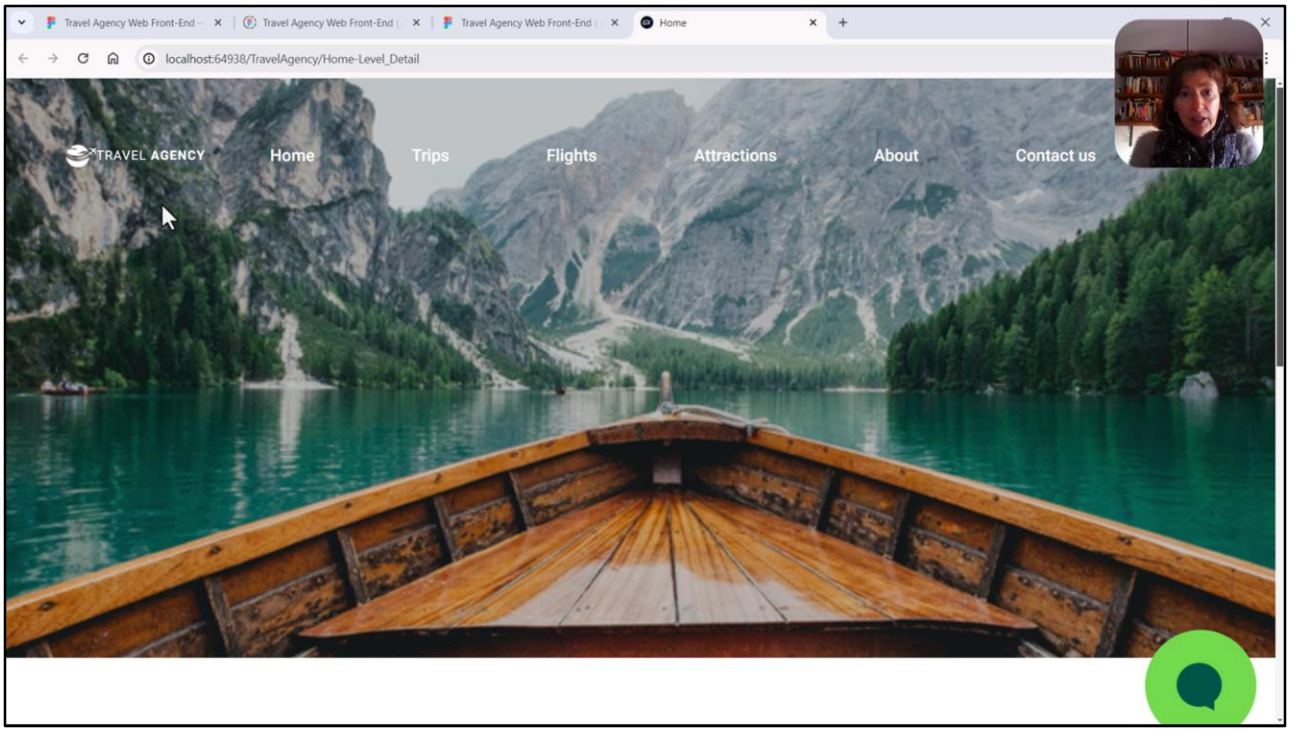



Temos esta classe do textblock, “header-logo-title__travel”, e o que faremos será atribuir para AGENCY a classe que criaremos, “header-logo-title__agency”, para que possamos dessa maneira, para a palavra AGENCY, alterar o peso da fonte. É tudo o que queremos fazer. Todo o restante queremos que tome as propriedades de “header-logo-title-travel”.

Então copiamos este texto…




…Viemos ao DSO e vamos especificar essa classe. Para a qual a única coisa que faremos é alterar o peso da fonte: de 400 para 700.



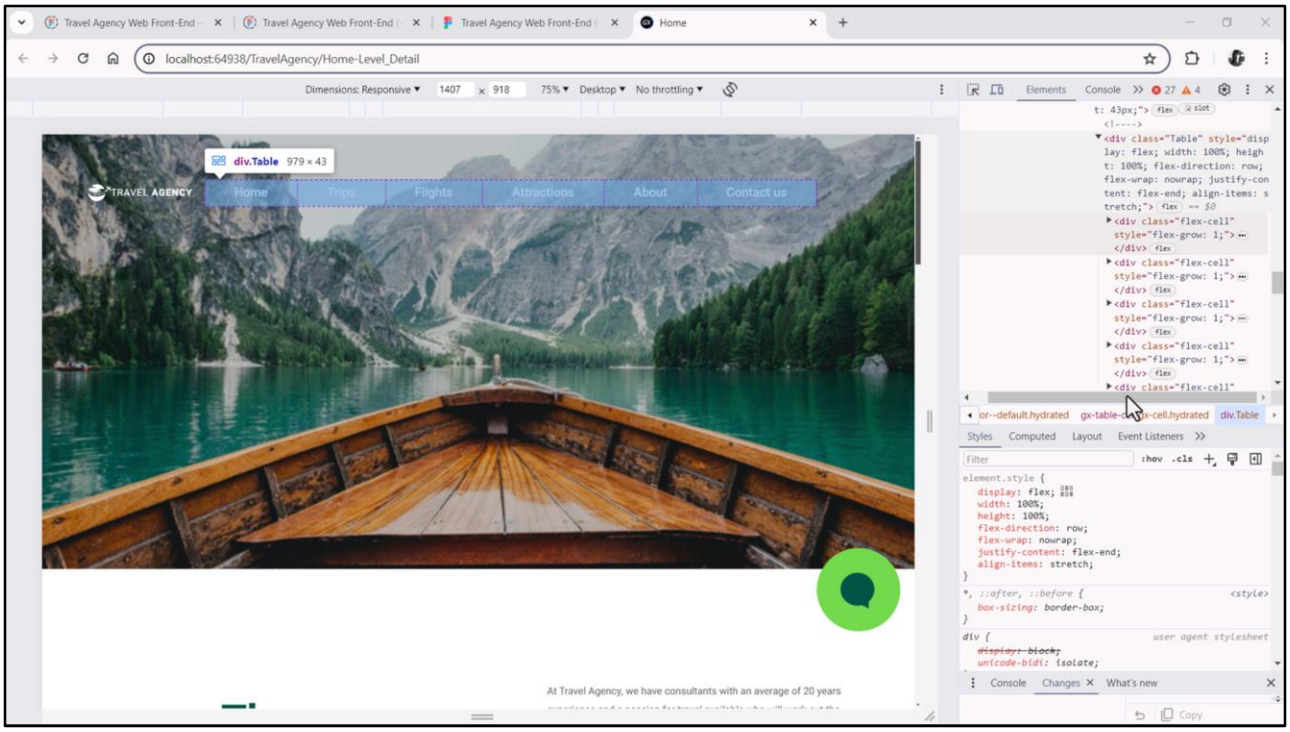

E aqui podemos ver isso.

O que acabamos de ver não significa que Angular permite colocar qualquer html nos textblocks e nos campos editáveis com formato html. Na verdade, removerá qualquer definição de um atributo style colocada ali. Mas o que permite é o que fizemos, usar classes do DSO dentro desse html.

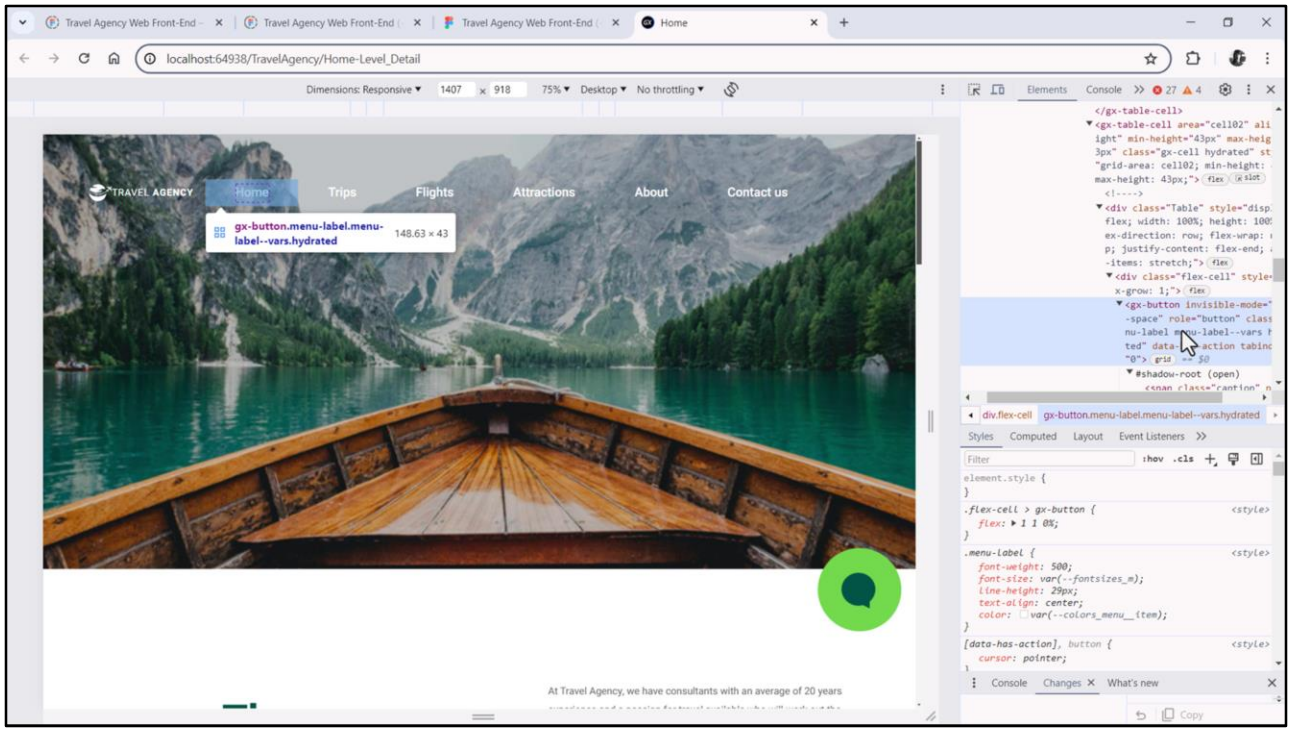
O oposto do que acontece com Android ou Apple: eles não permitem o uso de classes do DSO dentro do html e não removem o atributo style.

Então, inicialmente, para que a solução funcionasse para ambos os casos, ou seja, para Angular e também para Android e Apple, o que nos faria falta seria adicionar um atributo style onde colocamos ali as propriedades da classe que, no nosso caso, a temos no DSO. Então, dessa forma, se executar Angular vai remover tudo que tem a ver com o atributo style, vai remover, não vai levar isso em conta… mas ele vai levar em conta a classe. E caso contrário, se executarmos para Apple ou para Android, vai excluir, não levará em consideração a classe e levará em consideração o atributo style.

Como para tamanho desktop teremos apenas a aplicação Angular, neste caso não precisamos fazer nada.



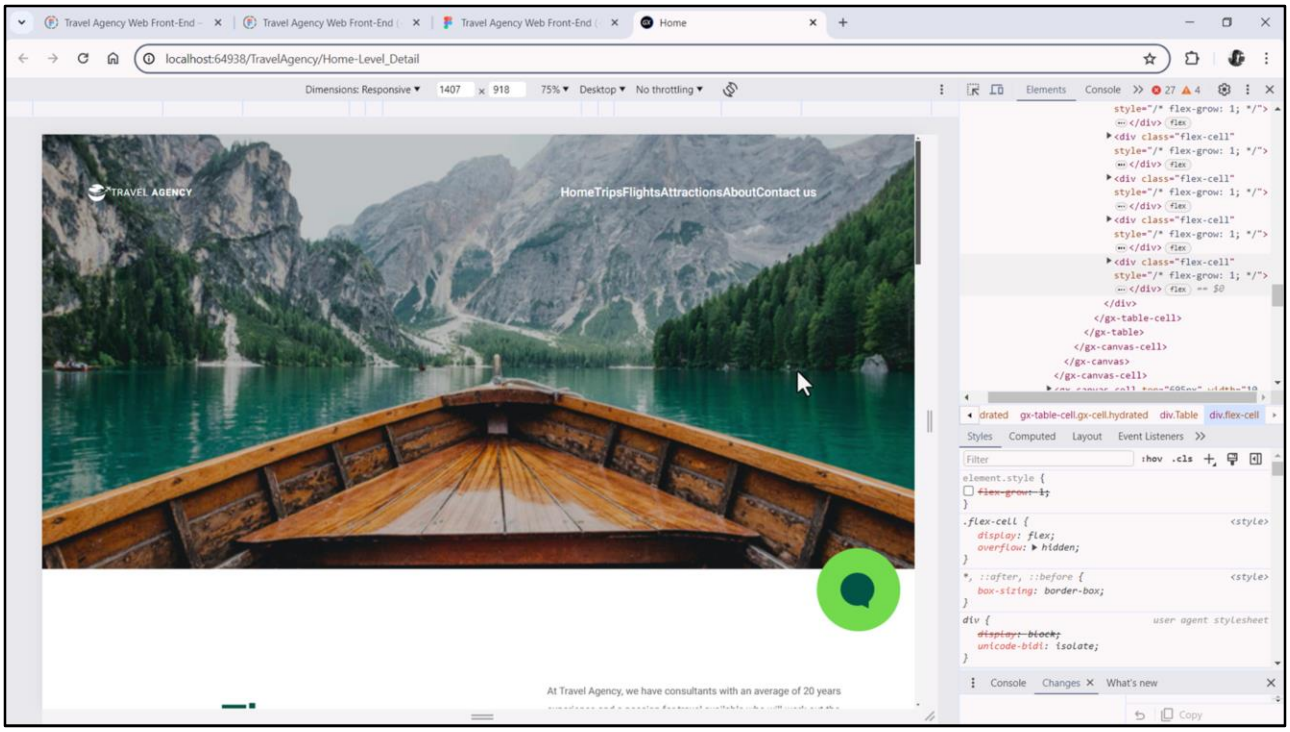

Agora sim, ao menu! Se o inspecionarmos como fizemos até agora, vemos aqui a tabela que corresponde ao Flex, e vemos que tem uma célula para cada botão.



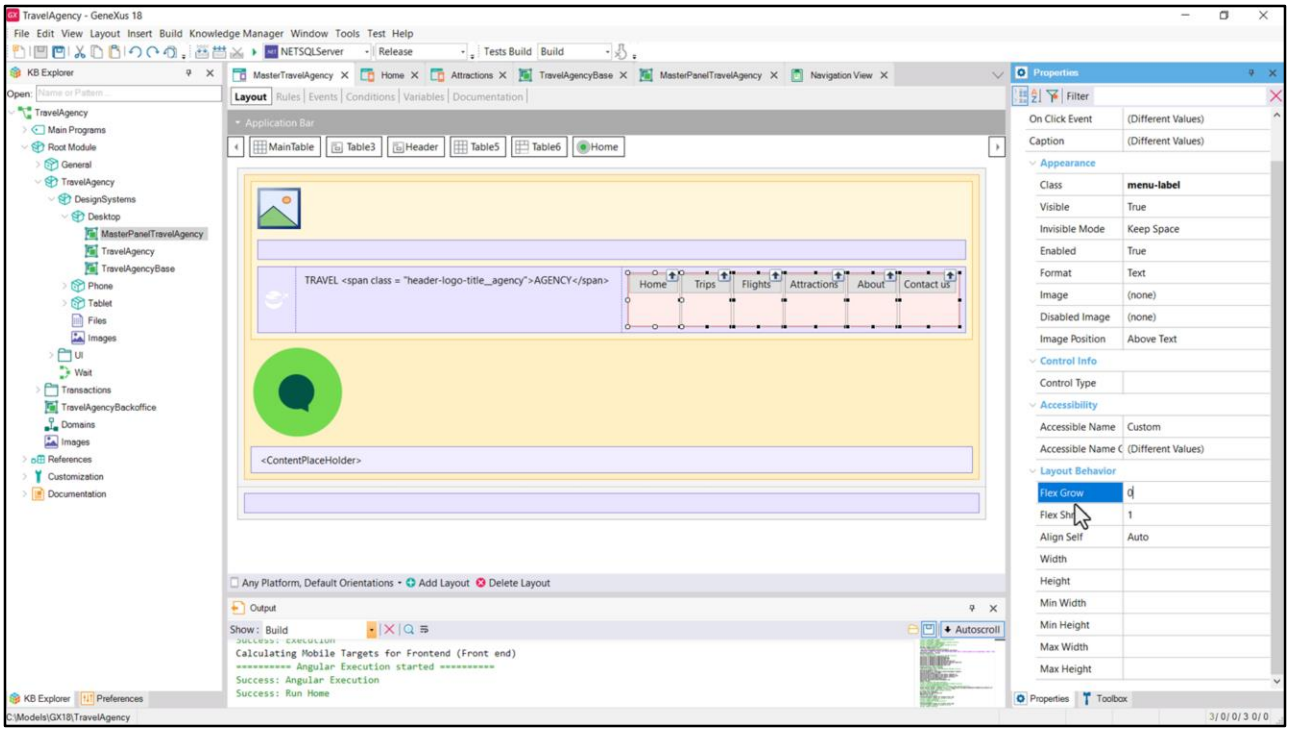

Então, aqui vemos a célula do primeiro botão, o botão e seu caption.

E o mesmo para os outros botões.

Observemos que está permitindo que cada botão cresça para ocupar uniformemente todo o espaço do flex livre.



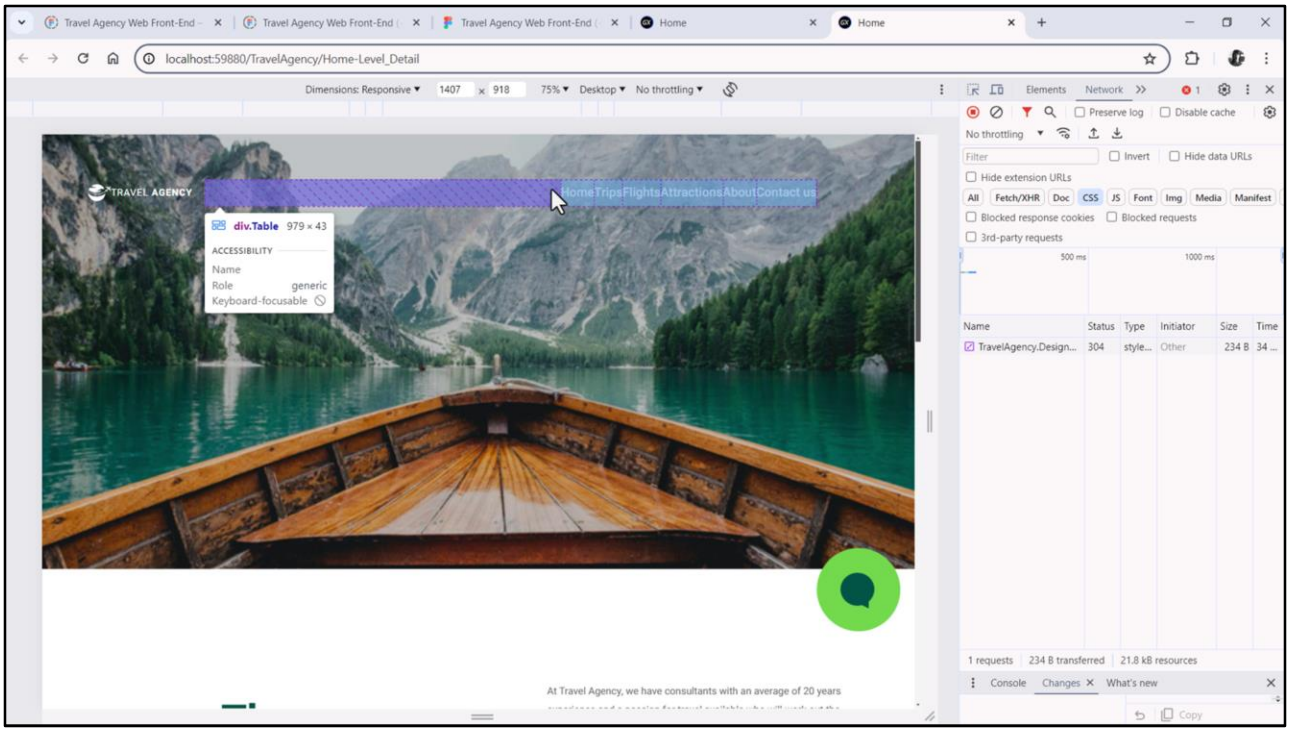

Isso se deve à propriedade Flex Grow em 1, que cada um dos botões possui. Vamos ver o que acontece se as removermos...



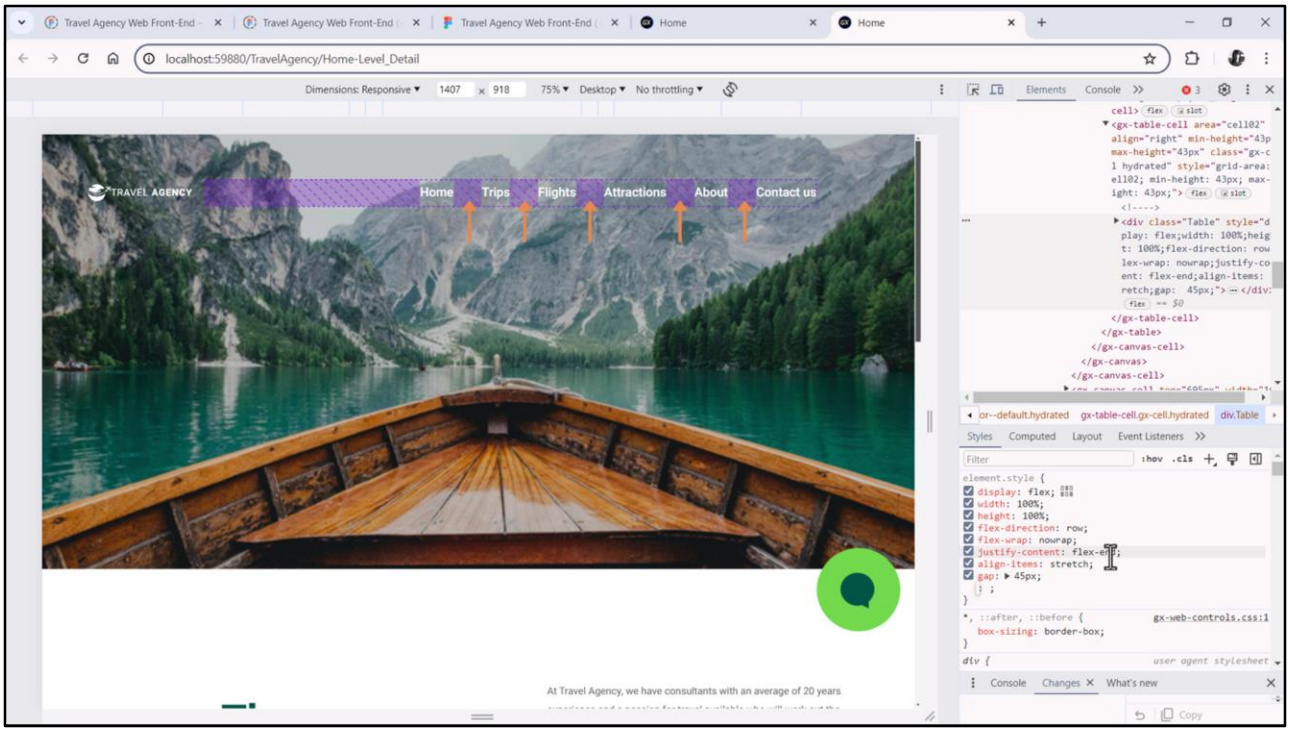

Em GeneXus vemos que embora o conteúdo esteja justificado de acordo com o final do flex, cada botão terá por default a propriedade Flex Grow em 1. Vamos ver a descrição da propriedade…

Diz que determina o quanto este filho vai crescer se houver espaço livre para distribuir entre os outros itens da linha. Se o item tiver um valor positivo para esta propriedade, entre todos aqueles com valor positivo será distribuído proporcionalmente o espaço livre.

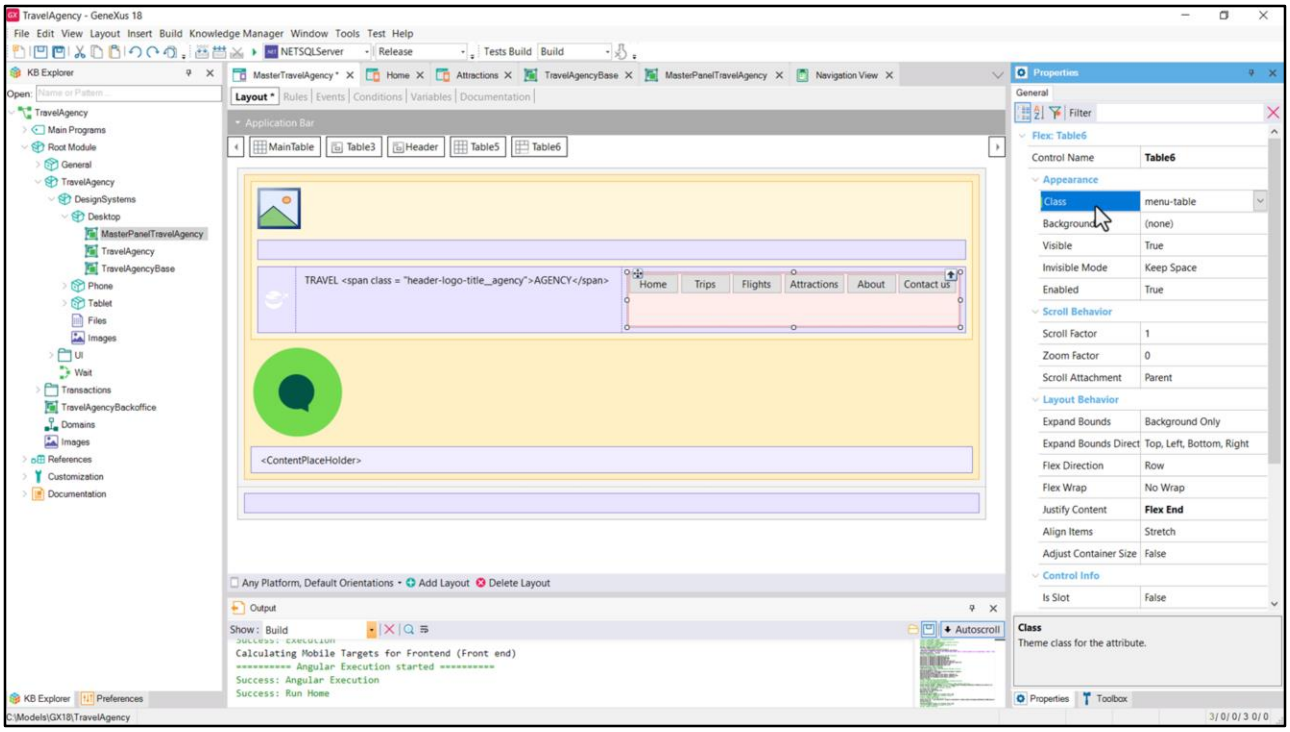
Todos eles têm por padrão o mesmo valor, 1. Então, vamos selecionar todos os botões e alterar seu valor para 0, para que ocupem exatamente seu espaço e nada mais. Vamos executar…



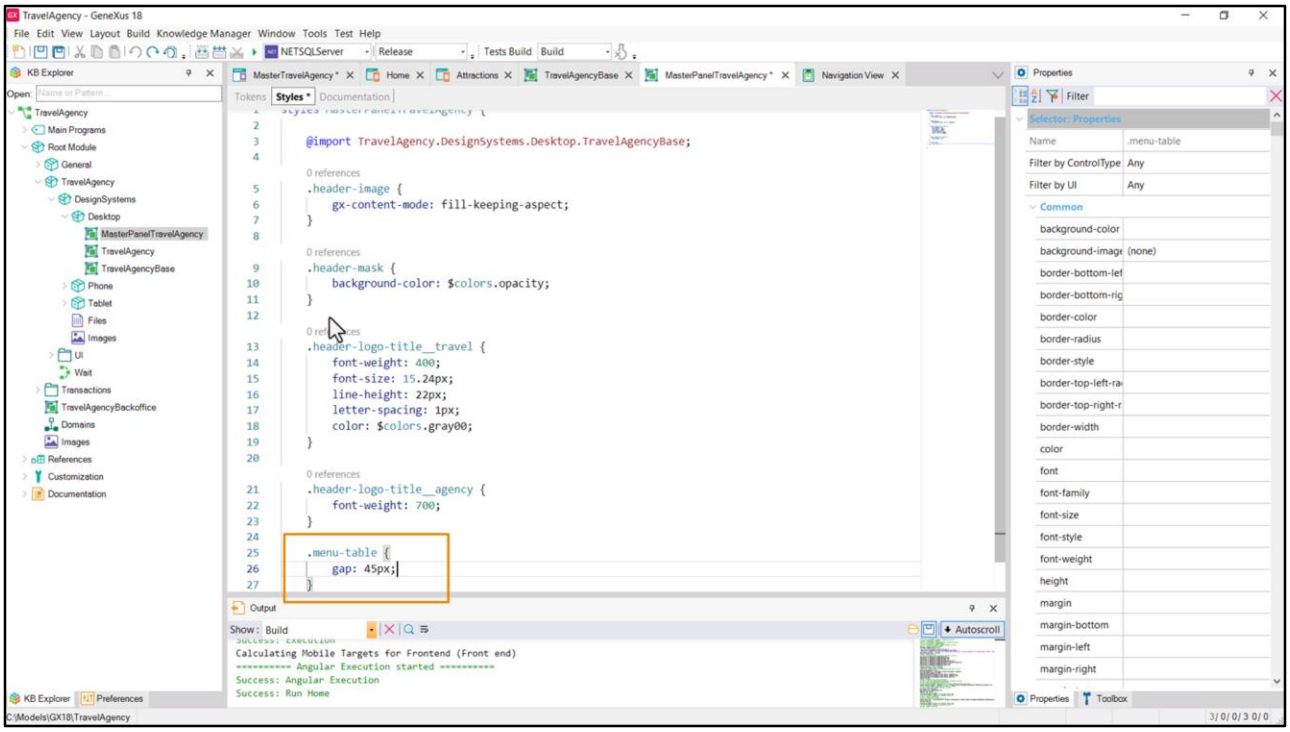

Se agora inspecionarmos… vemos todos os botões encostados no final do flex, e sem espaços entre eles.



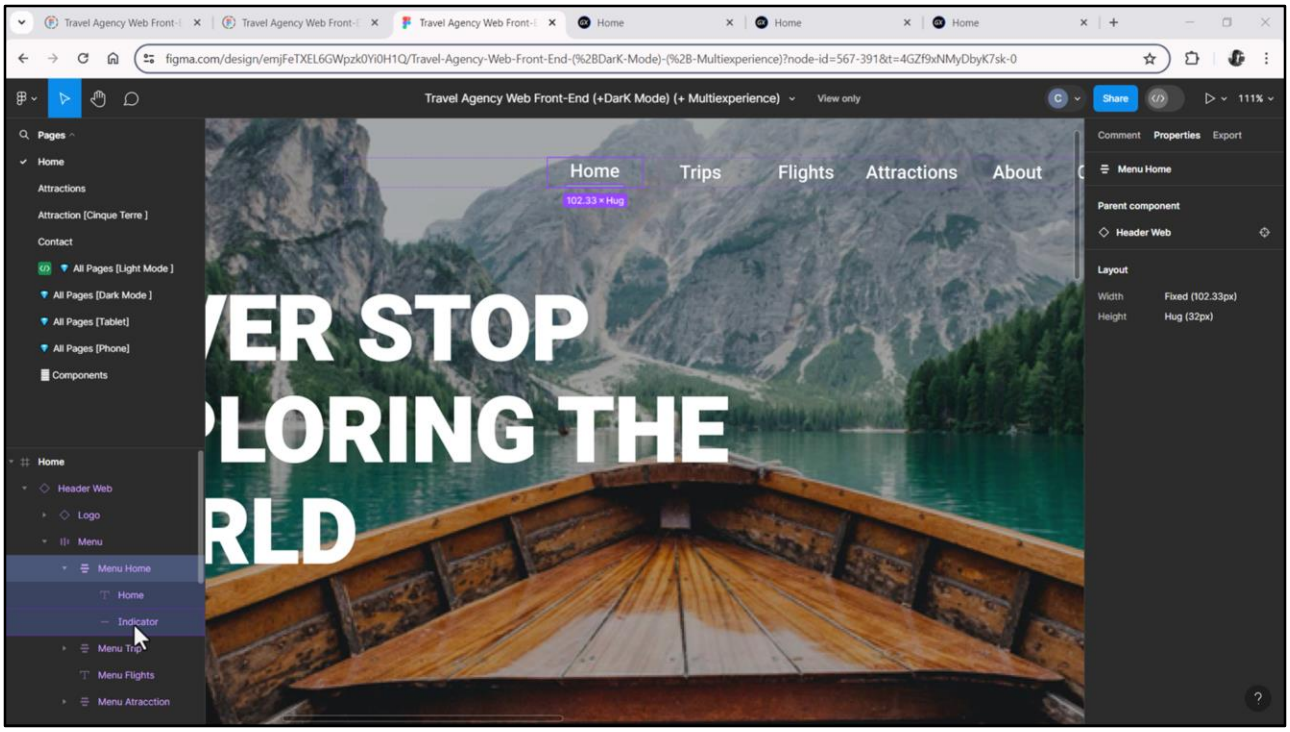

Para dar a eles um espaçamento uniforme, de modo que a distância entre o final de um caption e o início do outro seja a mesma, vamos testar a propriedade gap, com 45 pixels, por exemplo. Bem, parece bom. Ali vemos claramente como estão justificados em relação ao final do flex, e é por isso que fica todo esse espaço livre na frente, e como entre eles está funcionando esse gap de 45.




Esta é uma propriedade no nível do flex… e não a temos como propriedade estática do controle, então teremos que especificá-la em uma classe, que chamarei de menu-table.

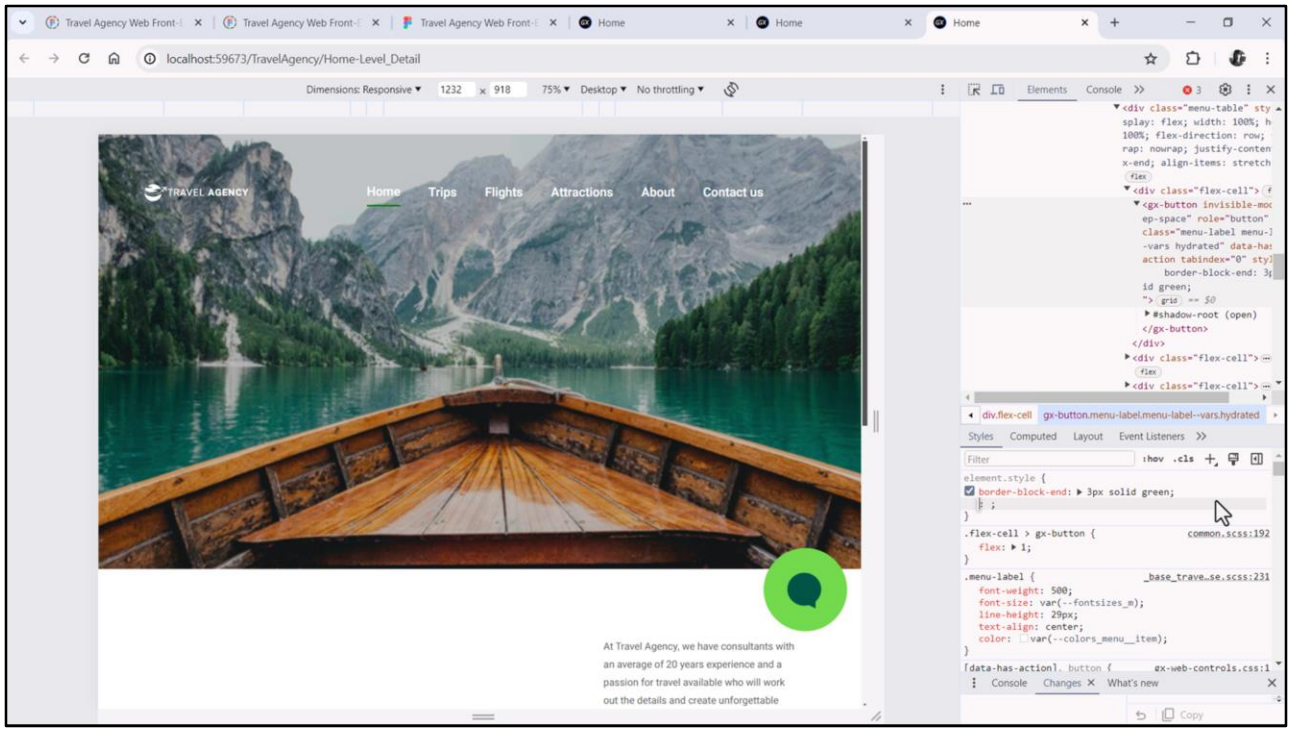



E ali coloco a propriedade gap.




Agora teríamos que pensar em como implementar o indicador da opção selecionada.

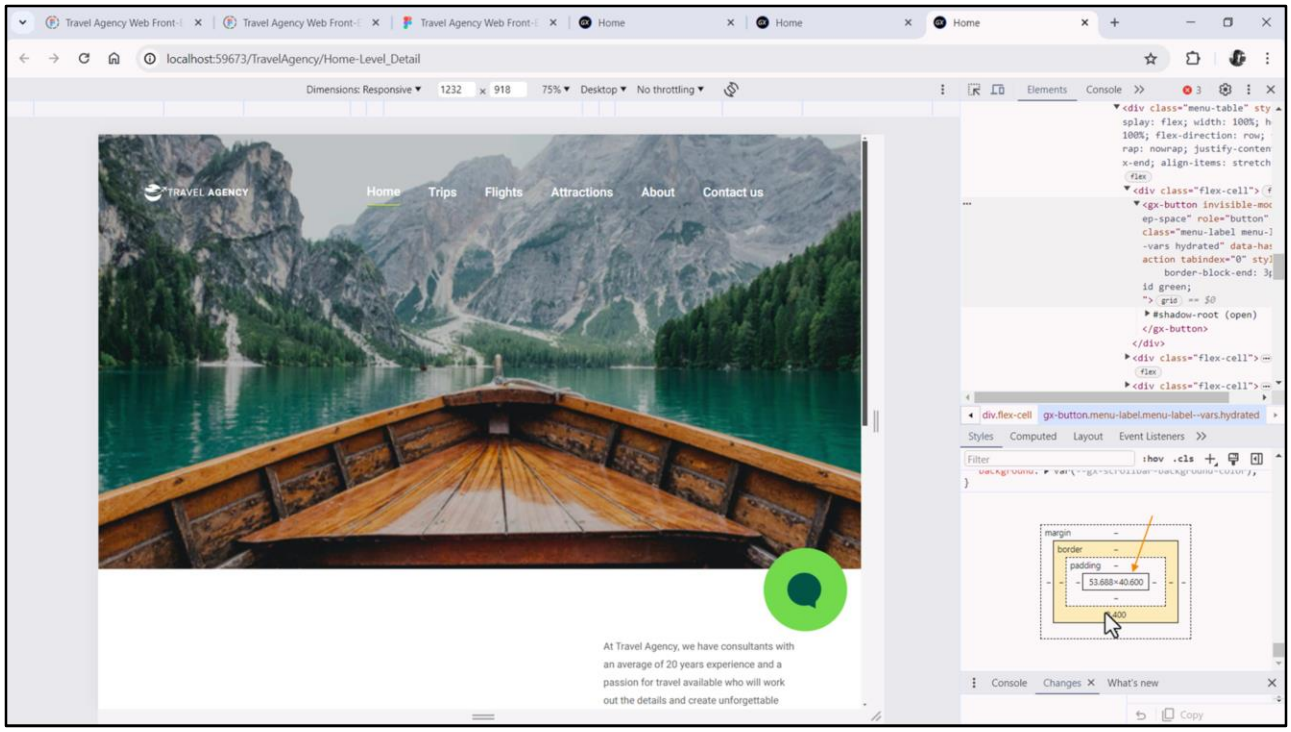
No Figma, vemos como Chechu implementou cada opção como um flex vertical, com o texto em cima e o indicador como uma linha abaixo.




Como se trata de uma ação, utilizamos botão em GeneXus, e não texto, e como qualquer controle, tem 4 bordas, uma de cada lado.

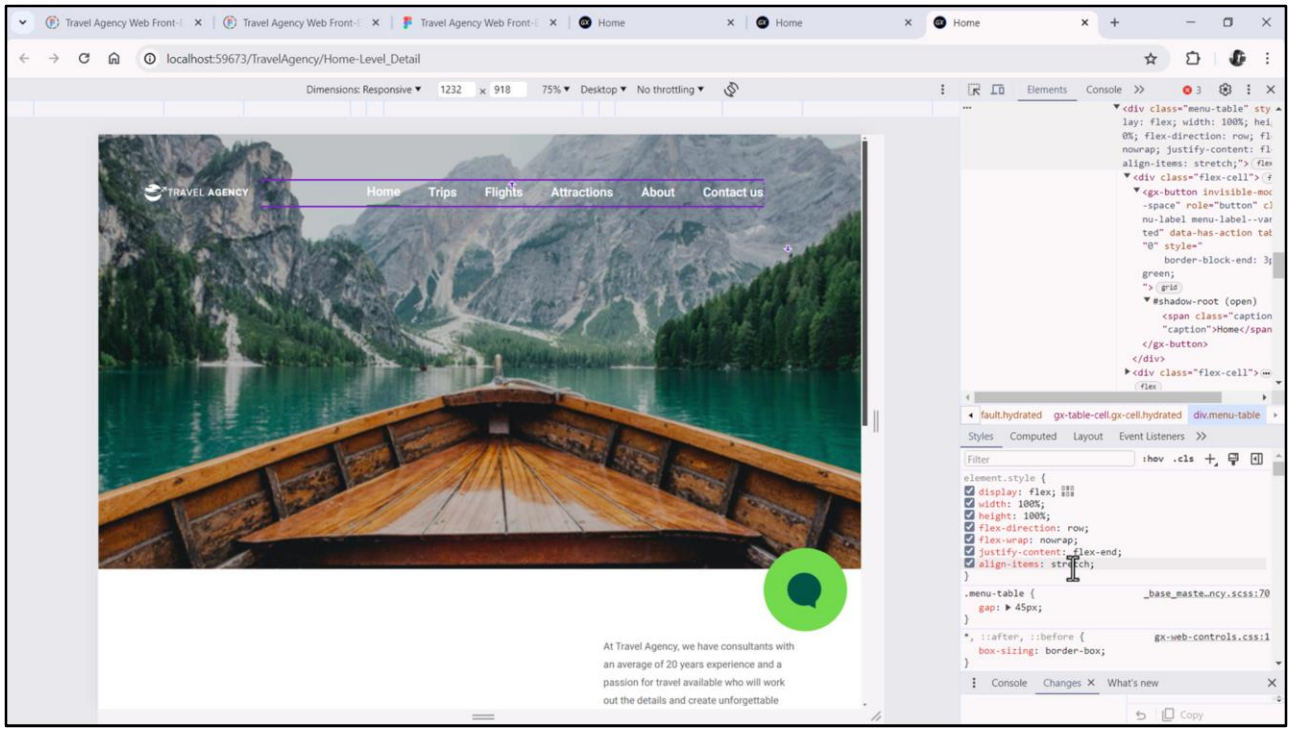
Será suficiente que as bordas de cima e de baixo tenham um valor positivo, mas sejam transparentes para todos os botões, enquanto a borda de baixo tenha cor apenas para o botão selecionado.

Vamos testar com este elemento... a propriedade é a border, e utilizarei a lógica, não a física, então será border-block, com o que estarei me referindo às duas bordas em direção vertical: a de início e a de fim. Se só quero a de fim, então coloco end. E ali indico que quero que seja de 3 pixels, por exemplo, com linha sólida, e o terceiro parâmetro seria a cor, por exemplo, green.

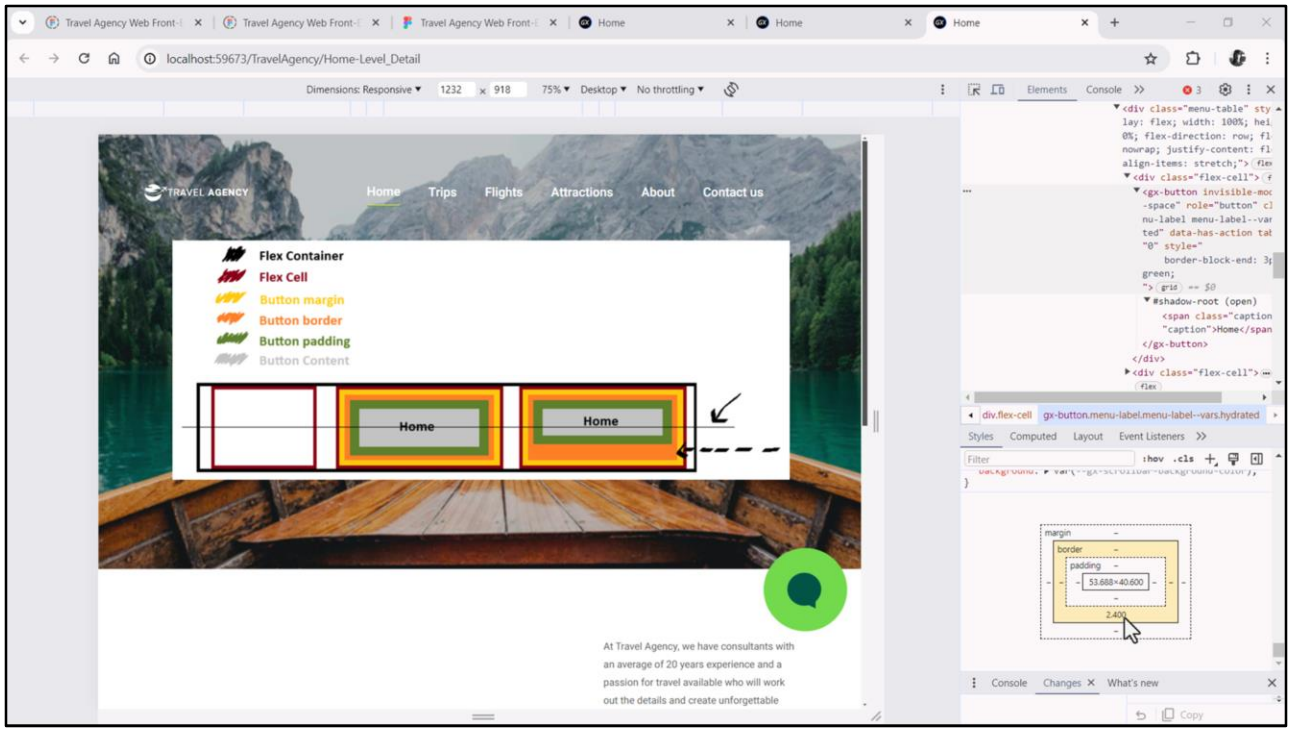



Este 2.4 em vez de 3 é um erro de precisão de ponto flutuante. Isso geralmente ocorre se for feito zoom na tela em um múltiplo diferente de 100%, o que é o caso. Não vamos prestar atenção nisso agora.

Sim, observemos que dando 2.4 para a borda, a altura do botão será desse valor, de modo que juntos somam os 43 pixels da altura do flex.

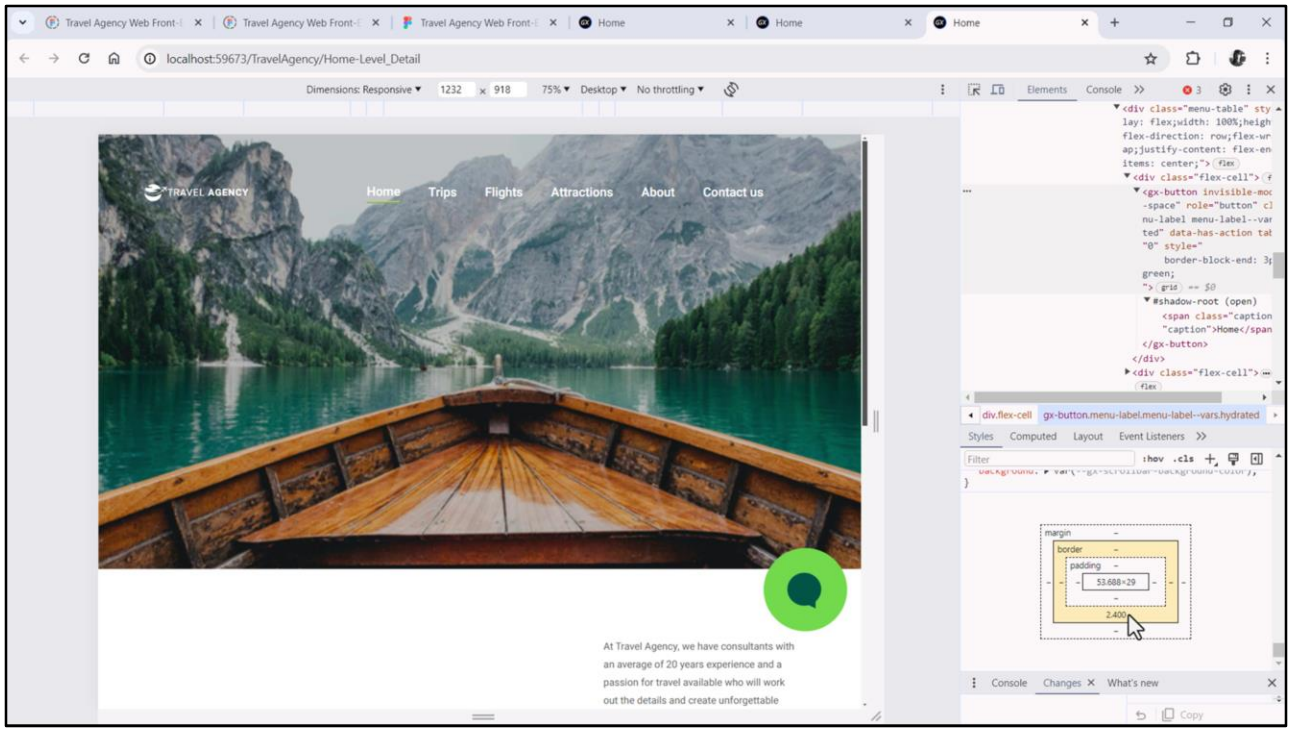



Qual é o problema com isso? Vemos que o flex tem a propriedade align-items com o valor default que é Stretch. Isso faz com que cada item se estique para ocupar verticalmente toda a altura de sua célula, que ocupará por sua vez toda a altura do flex. Neste caso 43 pixels.

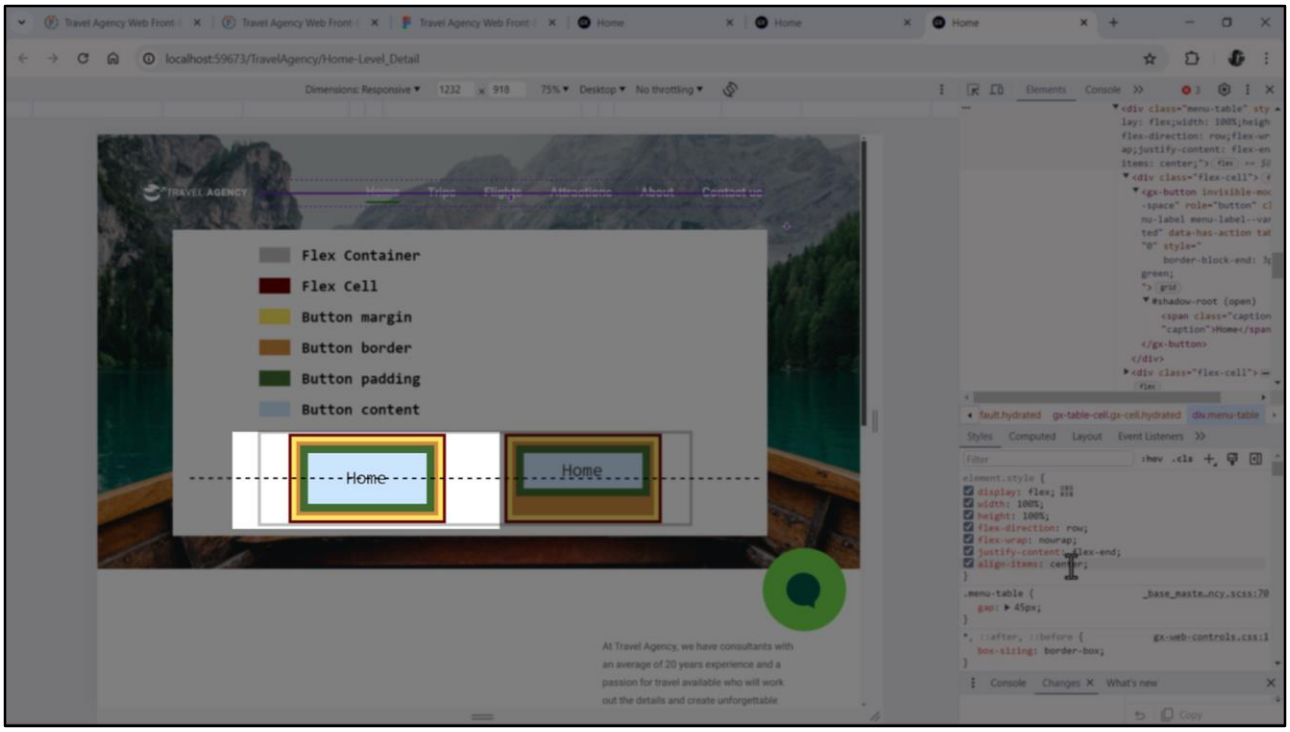



Mas cada caixa do botão inclui não apenas o conteúdo do botão, mas também padding, border e margin. Então, se deixarmos borda inferior de 3 pixels, o que veremos é que o texto do botão não ficará centralizado pela linha média, e portanto, não ficará centralizado em relação ao logotipo de Travel Agency.

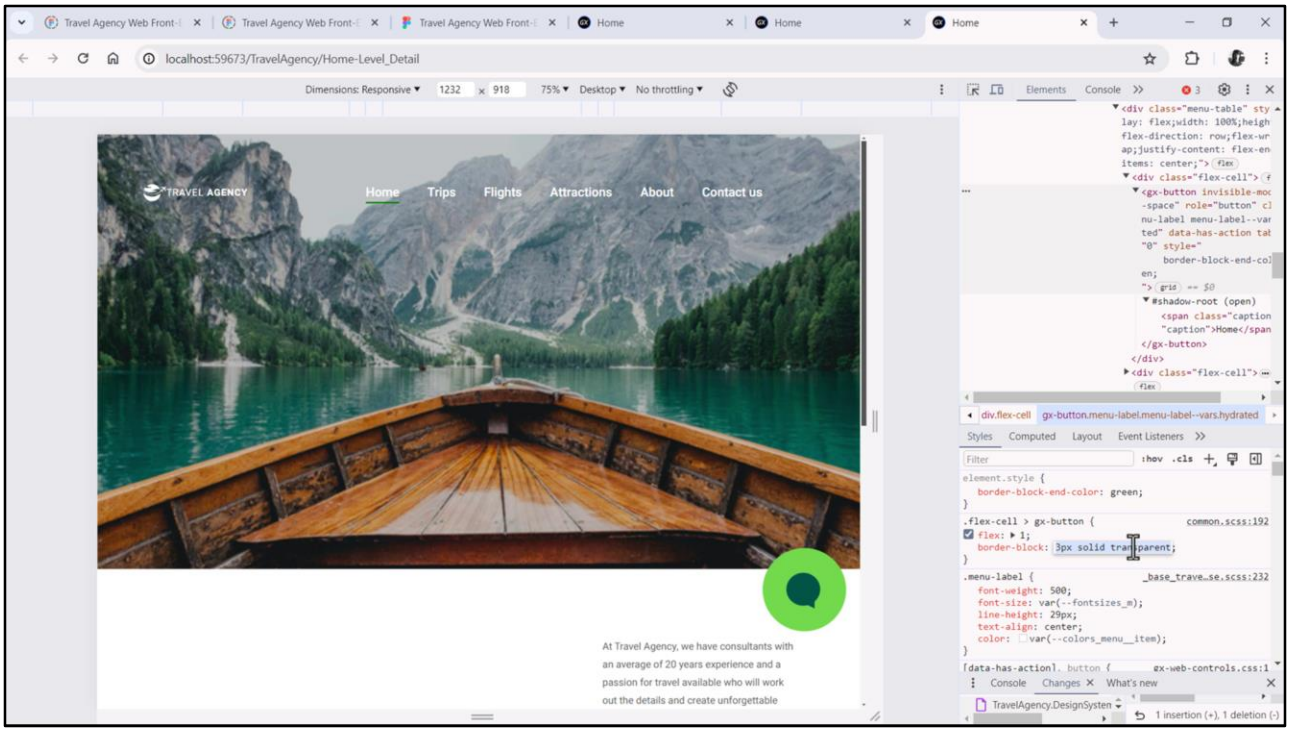
Neste desenho um tanto mal feito estou representando com o botão da direita esta situação em que estou deixando uma borda inferior maior que a de cima (que no nosso caso é 0: a de baixo de 3 pixels, a de cima de 0). E ao fazer isso, não ficará centralizado o conteúdo do botão. Esse é o problema.




Alterar o alinhamento dos itens do flex para center não resolve o problema. Fazer isso, a única coisa que consegue é que a altura do elemento não seja a altura do flex, mas que corresponda à altura do texto, do botão, mais padding, border e margin... Então será de 32 pixels em vez de 43. Mas no que diz respeito ao alinhamento pela linha média, estamos na mesma situação.



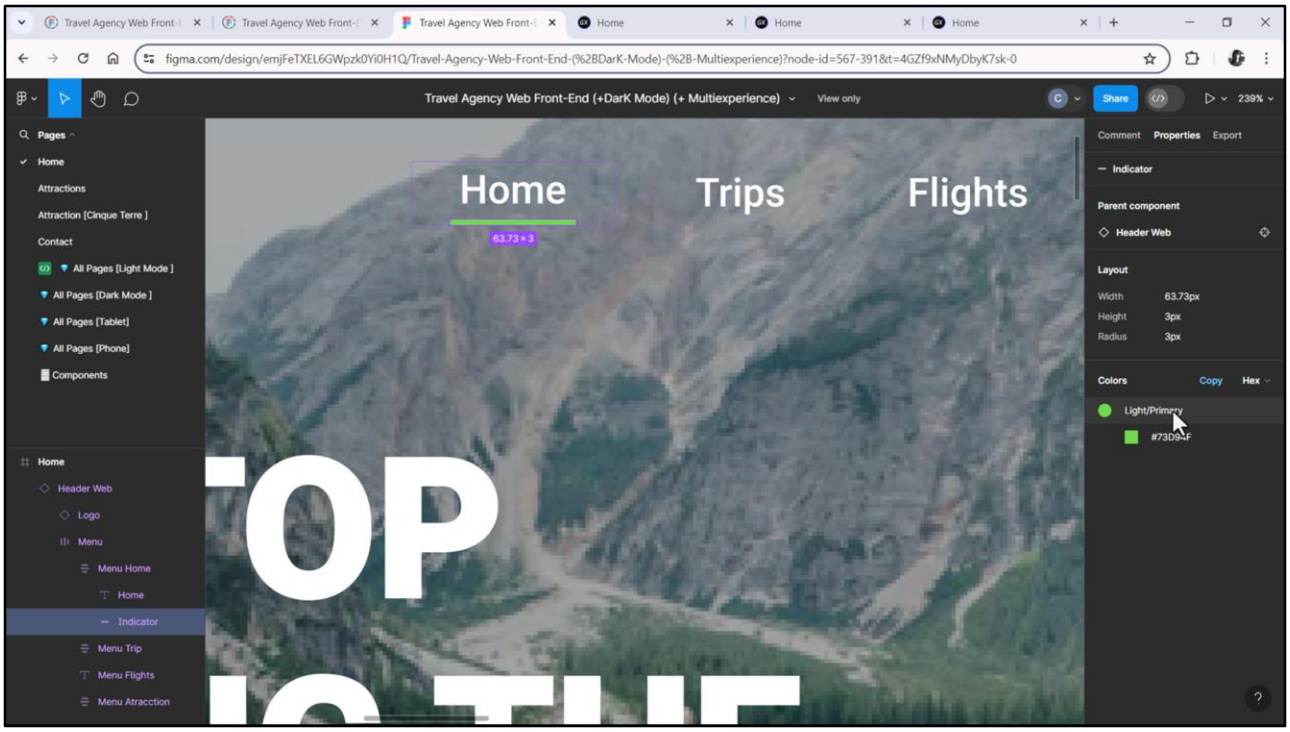

Uma solução será dar a mesma espessura a ambas as bordas, como no exemplo da esquerda. Ali o texto ficará centralizado verticalmente. E à de cima associar como cor transparente.



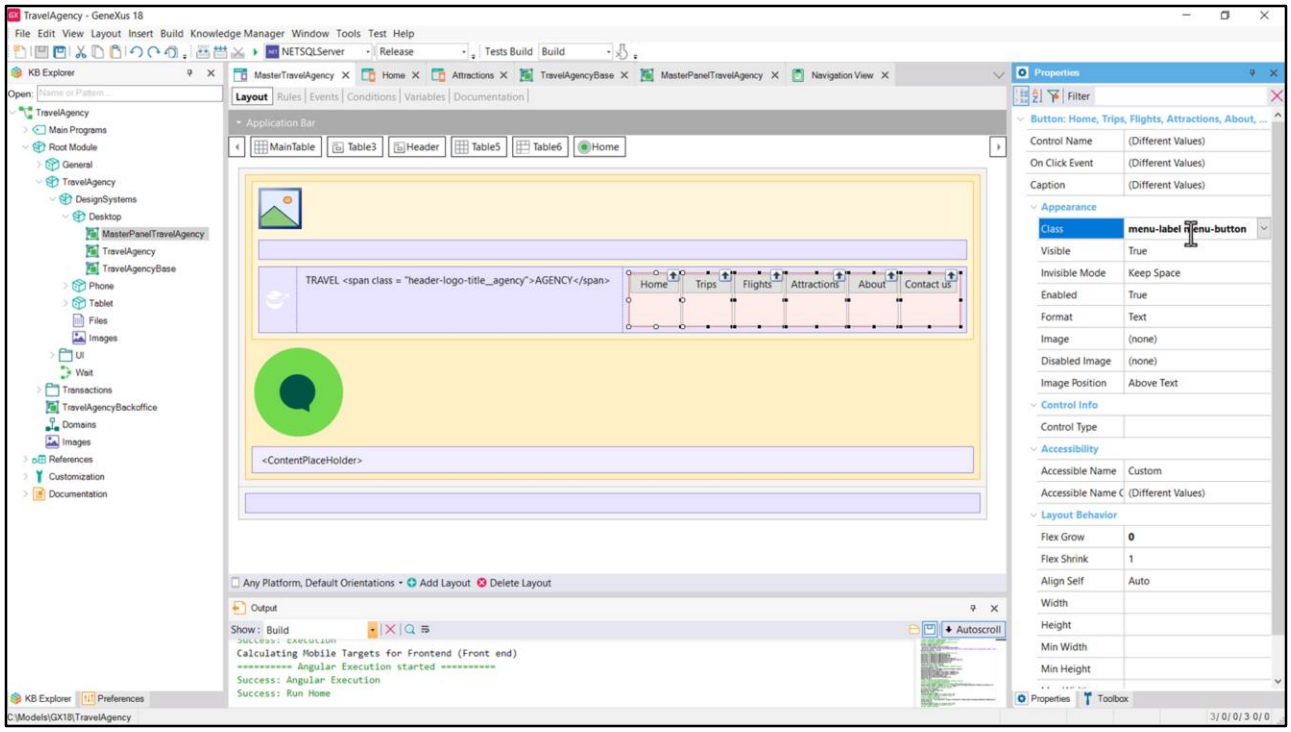

Para testar aqui, vamos nos posicionar sobre o elemento que corresponde ao botão Home. Essa propriedade está comandando o estilo desse elemento em particular. Neste outro seletor, temos tudo o que se aplica a todos os botões das células do flex, então vamos tentar colocar lá border-block (ou seja, valerá para cima e para baixo), de 3 pixels, solid e cor transparente.

E então, para o elemento selecionado, não precisamos repetir isso, apenas dar a ele cor green. Então escolhemos esta propriedade… e o valor green.

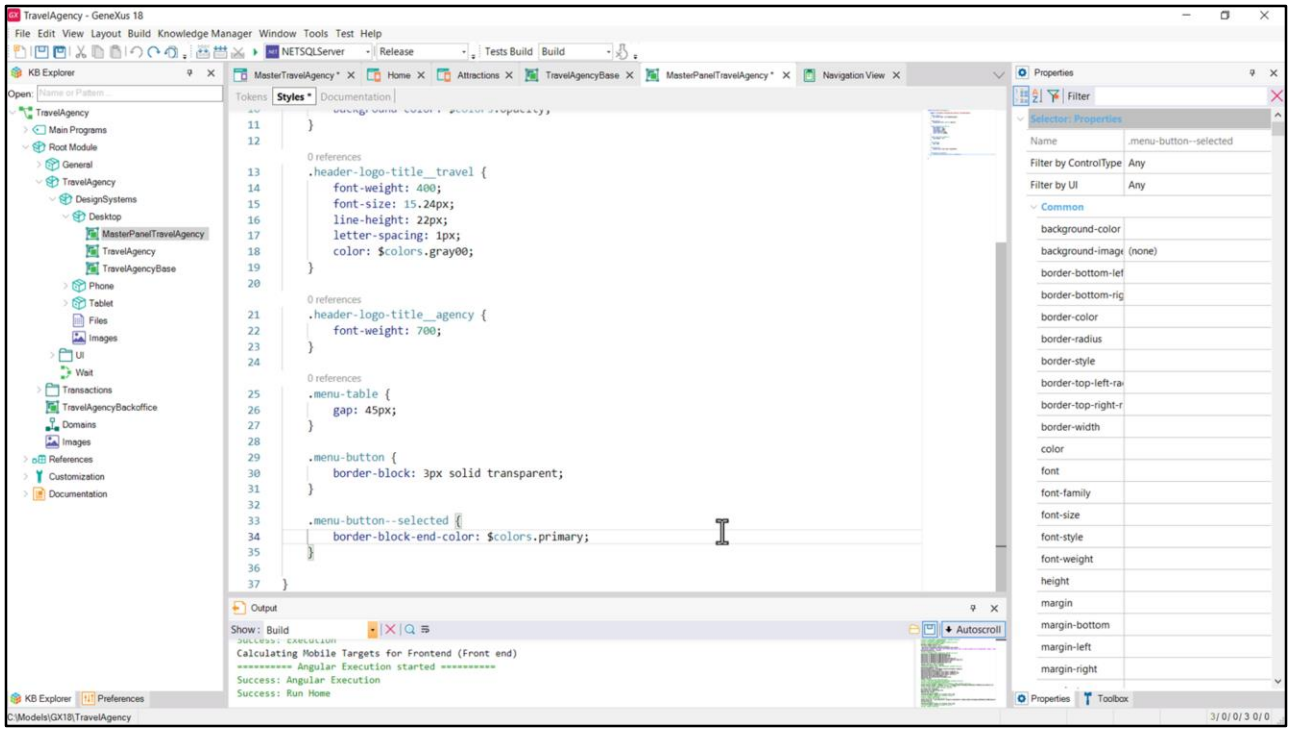
Com isso, teremos então aplicada essa propriedade para todos os botões, e em especial para cada um deles, será alterada a cor da borda inferior de transparente para este verde; para cada um que corresponde, no momento em que corresponde.



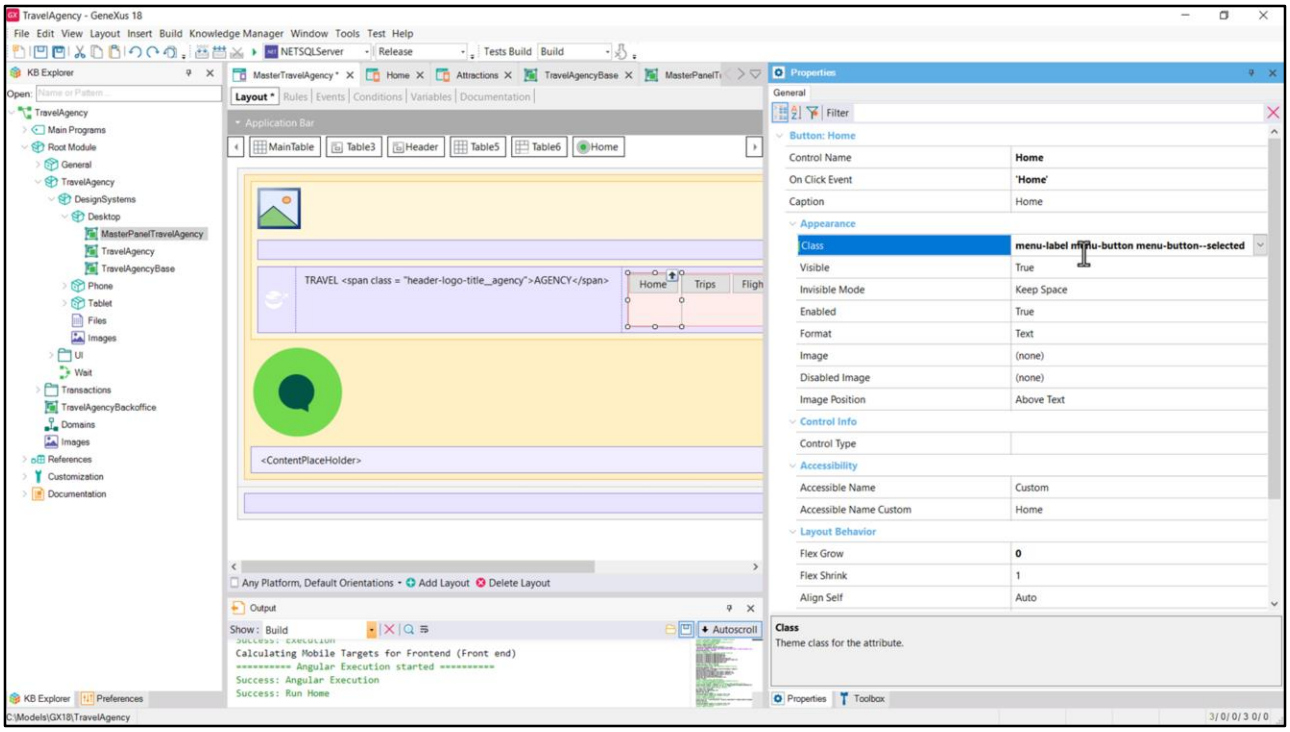

Na verdade, se formos ao Figma... a cor será a primary, e está certo isso dos 3 pixels. A este radius não daremos atenção nesta ocasião.




Então, à classe que todos os botões têm, que controla o estilo tipográfico dos rótulos, vamos adicionar outra classe, que chamarei de menu-button…

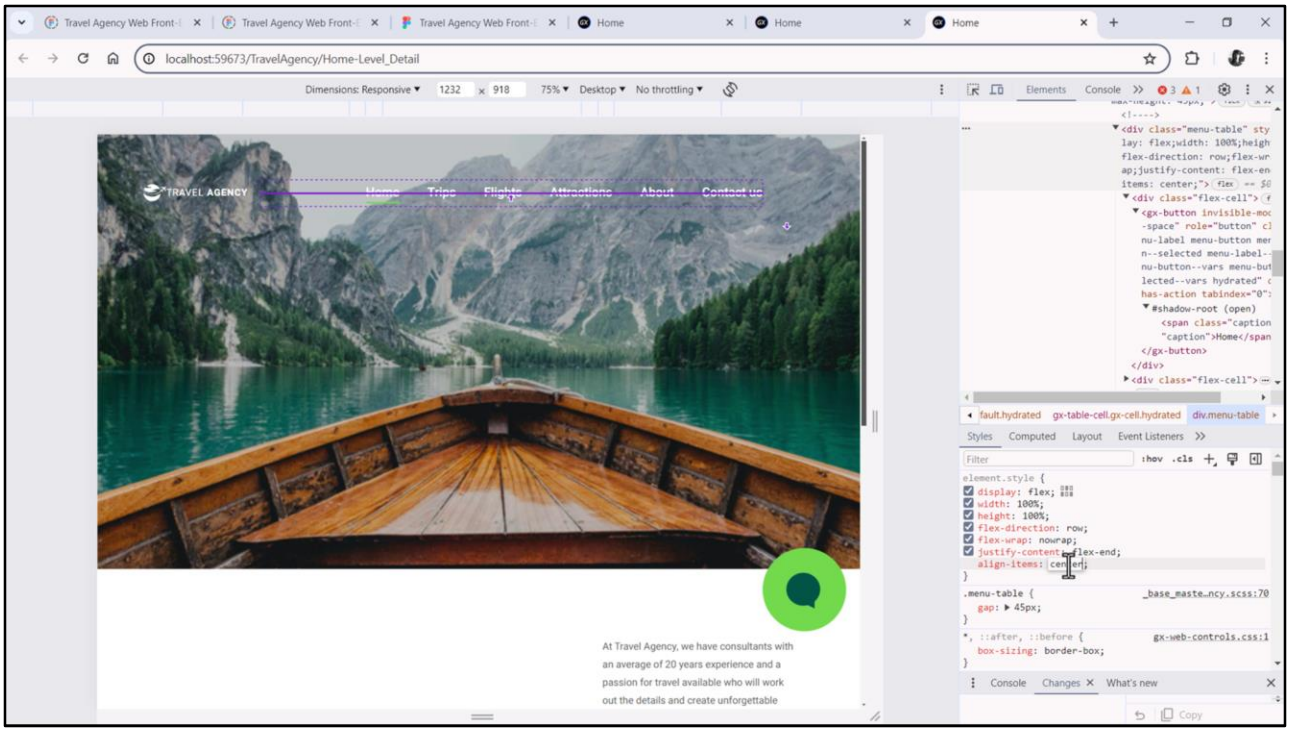



….À qual atribuirei a borda transparente de baixo e de cima… E então criarei outra classe para alterar a cor para a primary da borda de baixo.

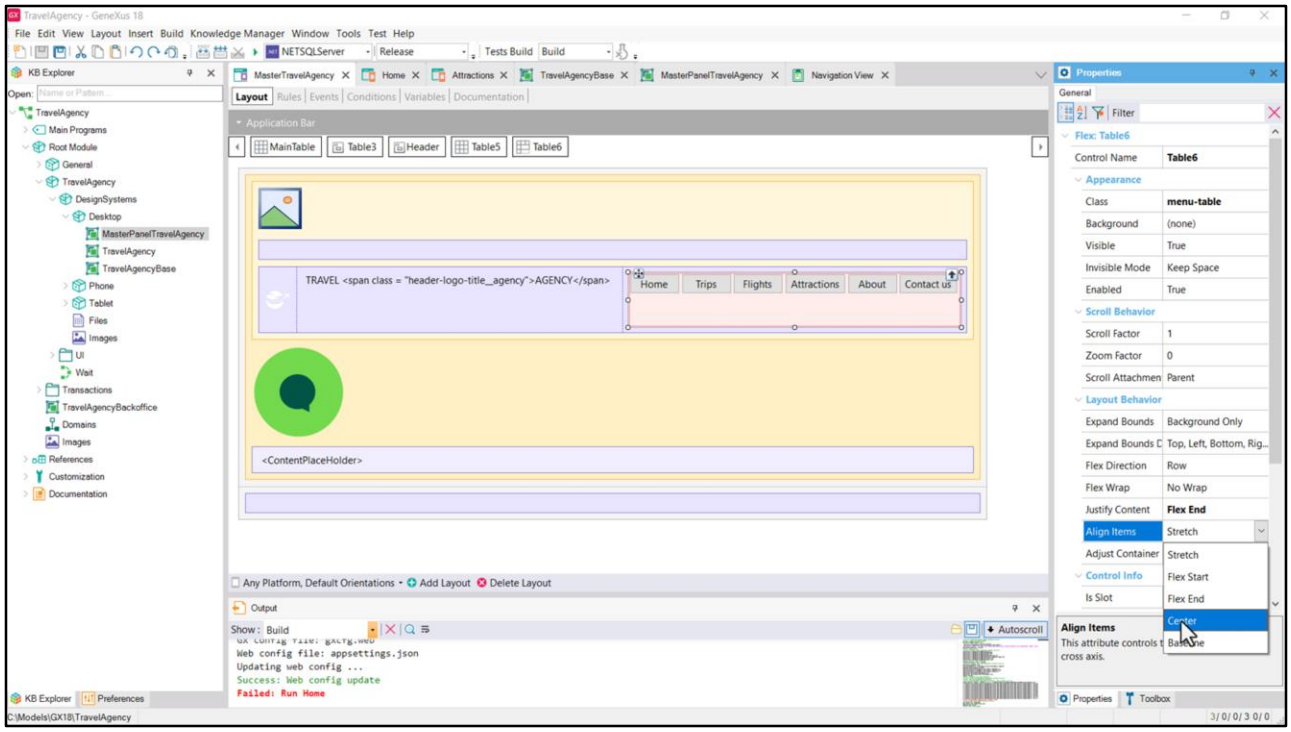



Por enquanto, vou atribuí-la ao botão Home para que vejamos o estilo.

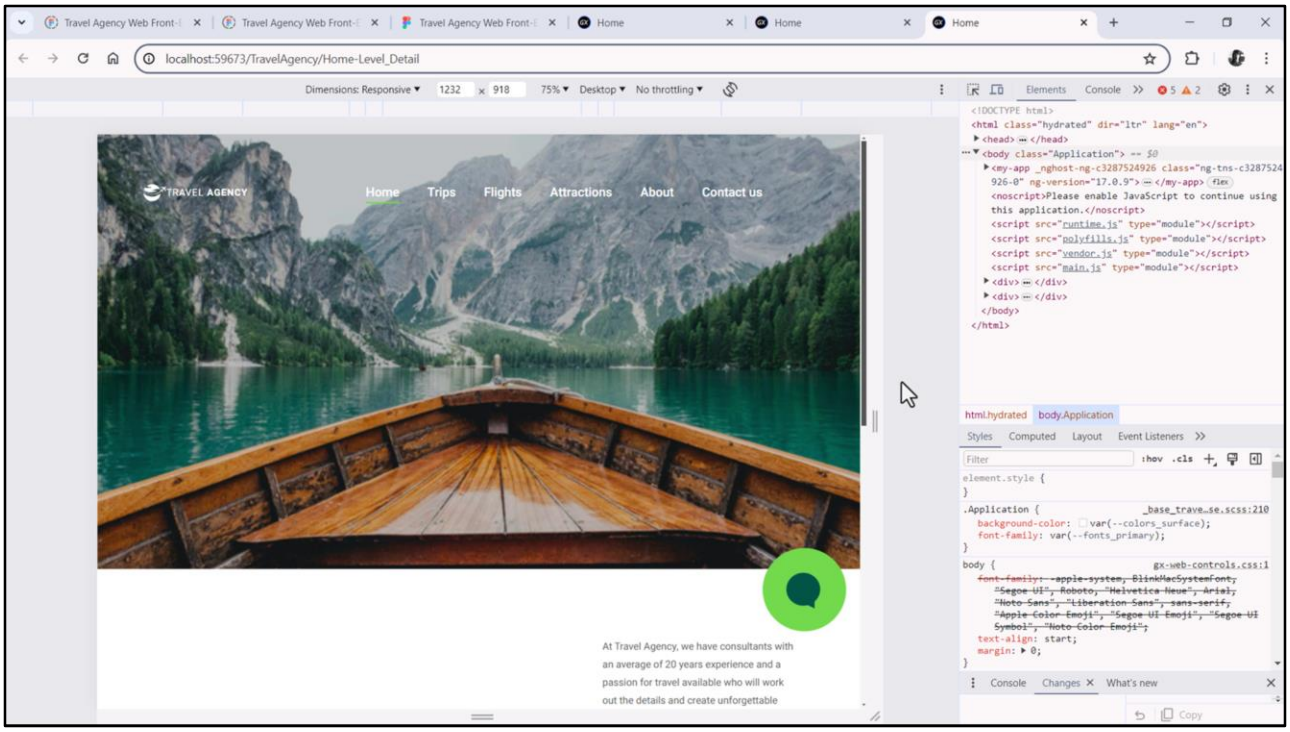
Quando estudarmos as navegações, veremos como alterá-la dinamicamente de acordo com quem está sendo carregado no contentplaceholder.



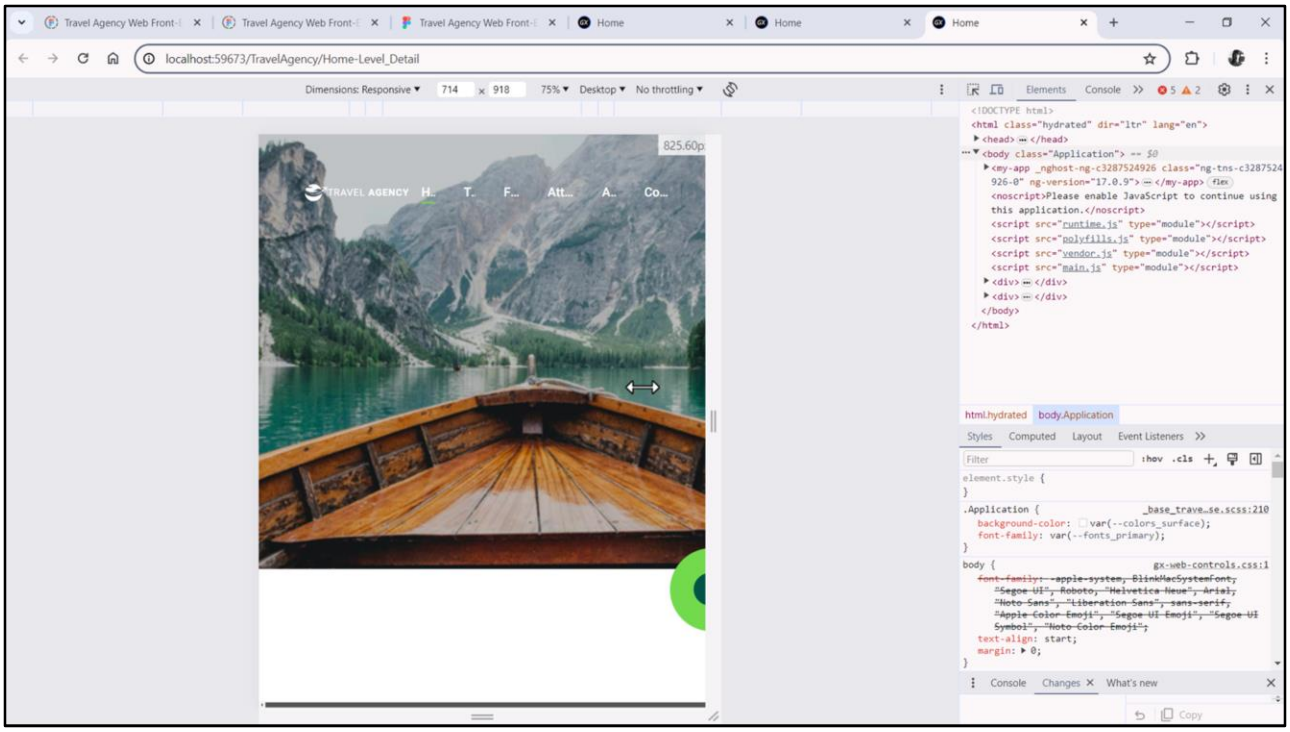

Pode parecer-nos que há muita distância entre o texto e a linha da borda. É que, lembremos, o botão está ocupando toda a altura da célula, que está ocupando toda a altura do flex, porque deixamos o valor default para a propriedade align-items, que era stretch. Se mudarmos para center, conseguimos que a altura seja o mínimo necessário para conter seu conteúdo mais padding, borda e margem. E também que seja centralizado pela linha média.



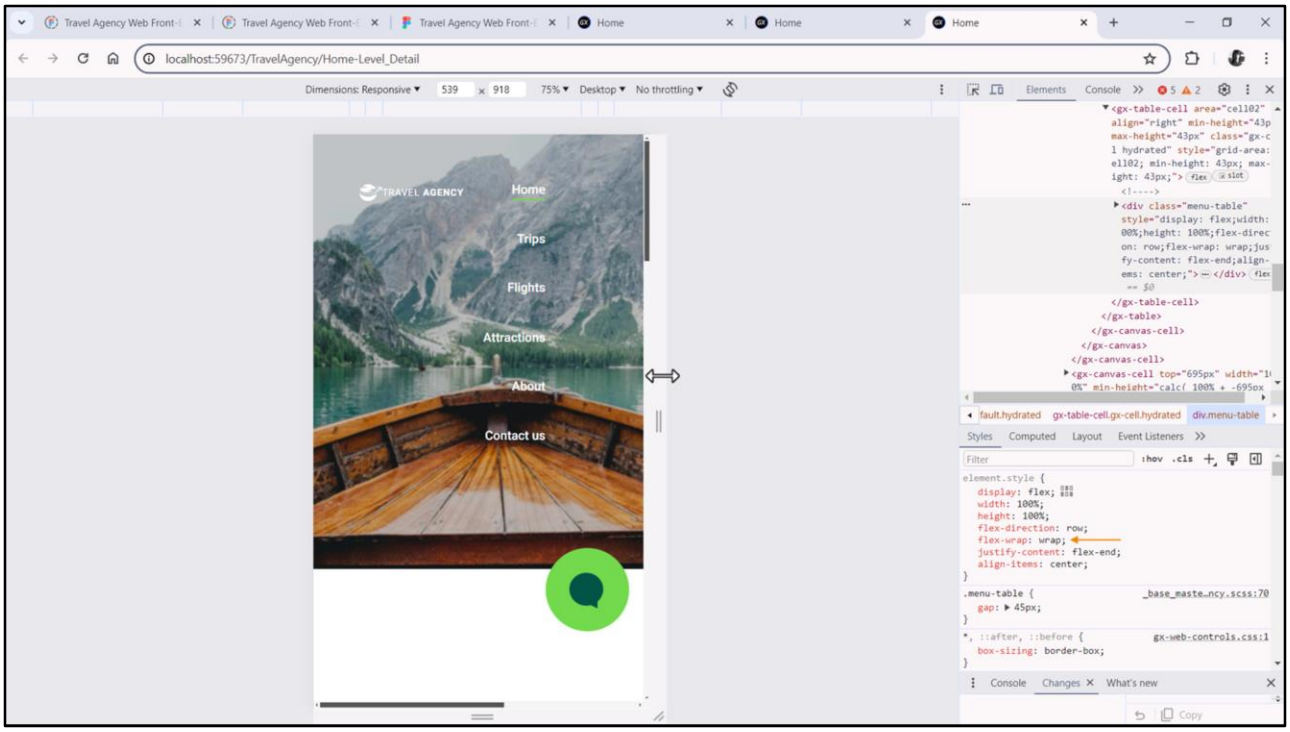

Conseguimos isso com a propriedade estática no nível do flex… (há um bug no editor abstrato que impede que sejam vistos os botões, mas estão lá. É apenas um problema visual. No editor de GeneXus web já foi corrigido)




Na verdade, se executarmos… veremos tudo exatamente como esperamos.

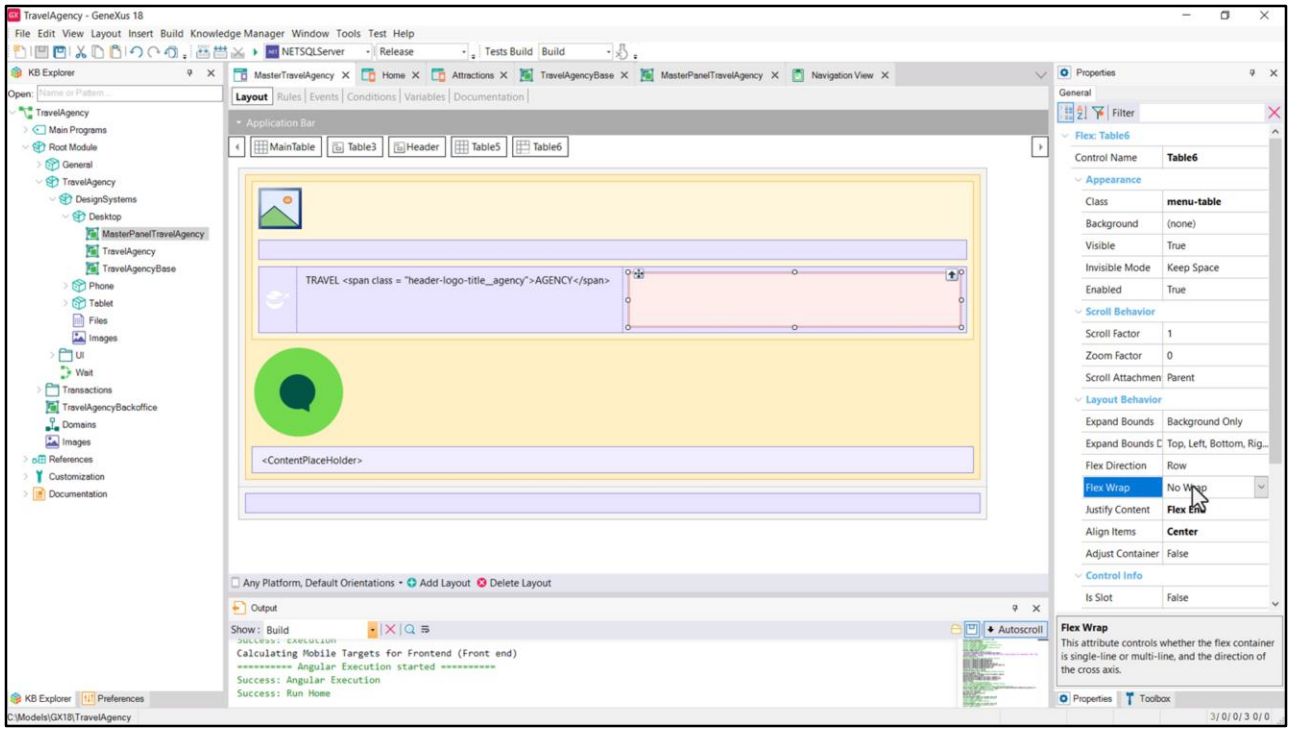



Uma última coisa sobre o menu: vejam o que acontece se diminuirmos a largura da tela. Os botões começam a reduzir para caber no espaço que eles têm, deixando o gap que indicamos, claro.

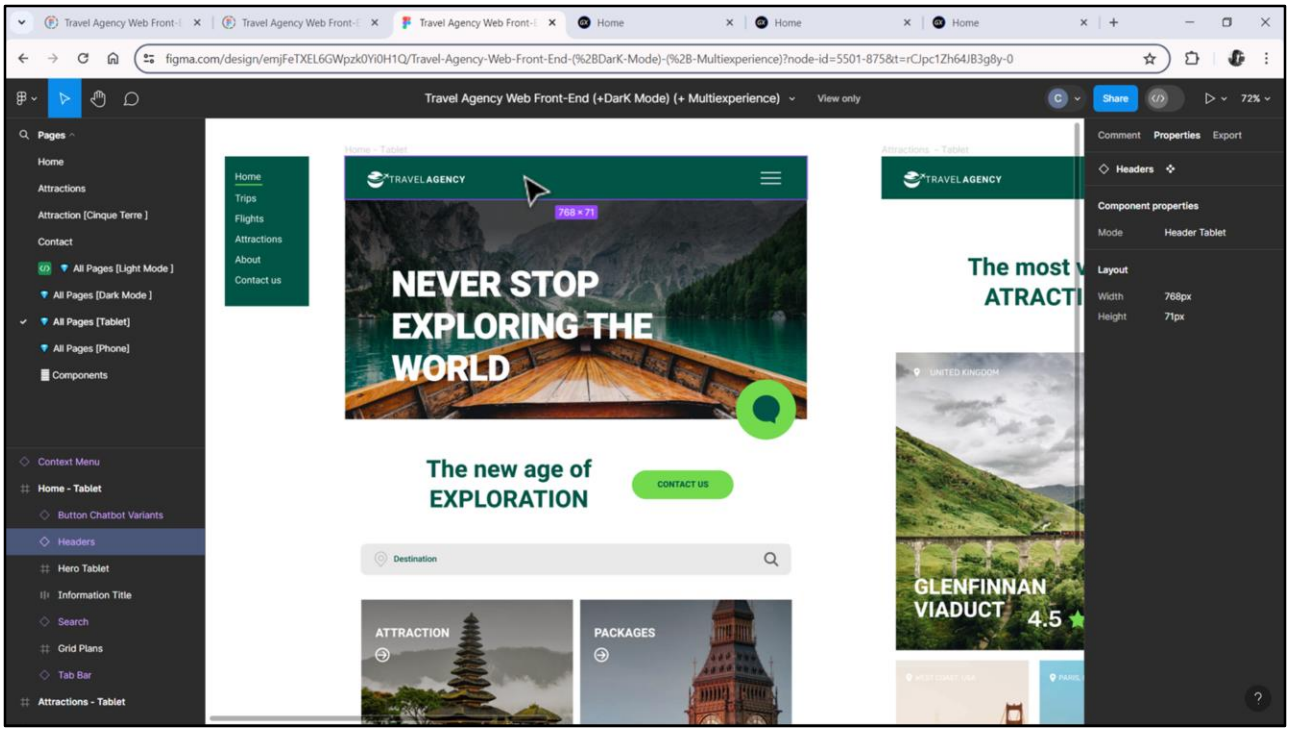



Uma das características que o container flex oferece, diferentemente da tabela, é que ele permite o wrap, de modo que os itens que não cabem mais na linha sejam colocados em outra.

Não será do nosso interesse para o nosso caso, mas eu queria mostrar isso.

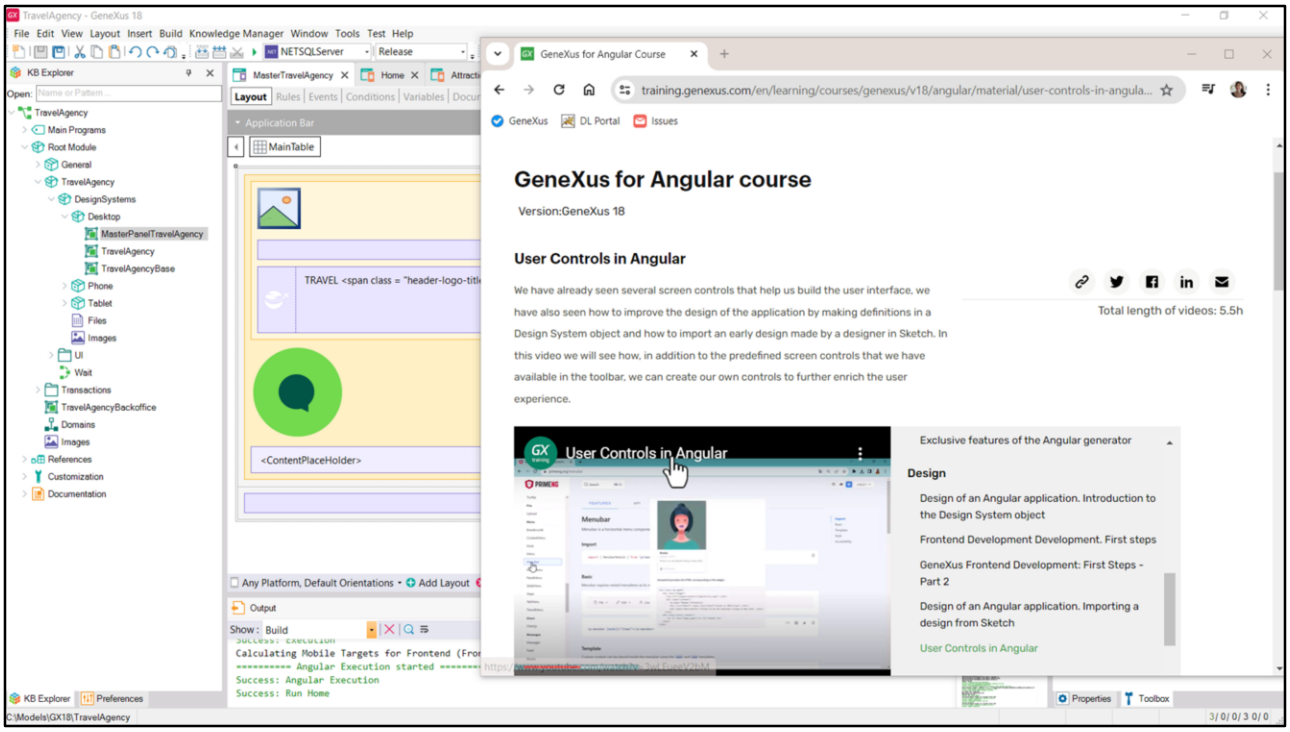



E vejam onde fica a propriedade.



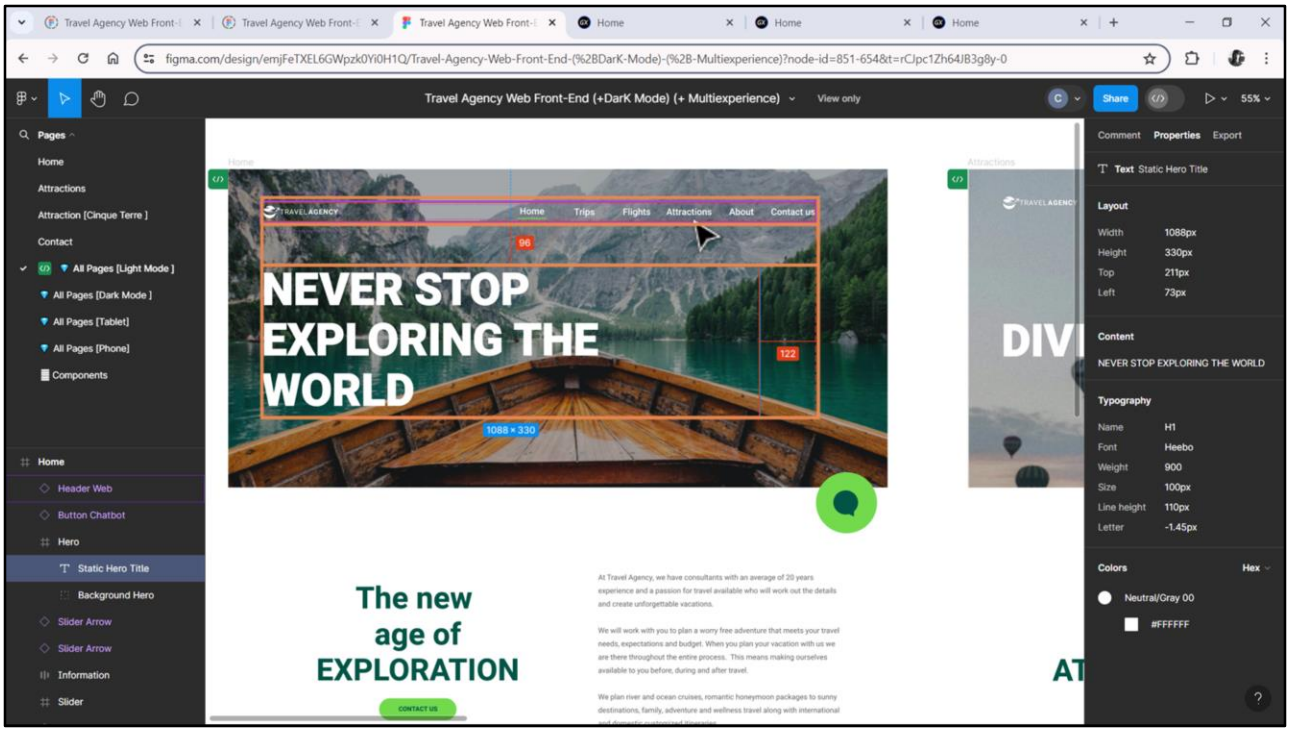

Lembremos que Chechu criou outros designs para tamanho de tela de Tablet... onde o menu será o clássico hambúrguer, e para tamanho Phone também.

Voltaremos a esses breakpoints quando analisarmos a multiexperiência, no próximo módulo.




Aproveito para mencionar que se precisarmos de controles mais sofisticados que incluam design e comportamento, podemos incluí-los no GeneXus definindo **User controls**, ou seja, objetos que são chamados assim, user controls, nos quais podemos estabelecer o html e intervir minimamente para que possa ser utilizado em qualquer objeto GeneXus com interface. Ou seja, o incorporamos à KB e já está disponível para ser utilizado.

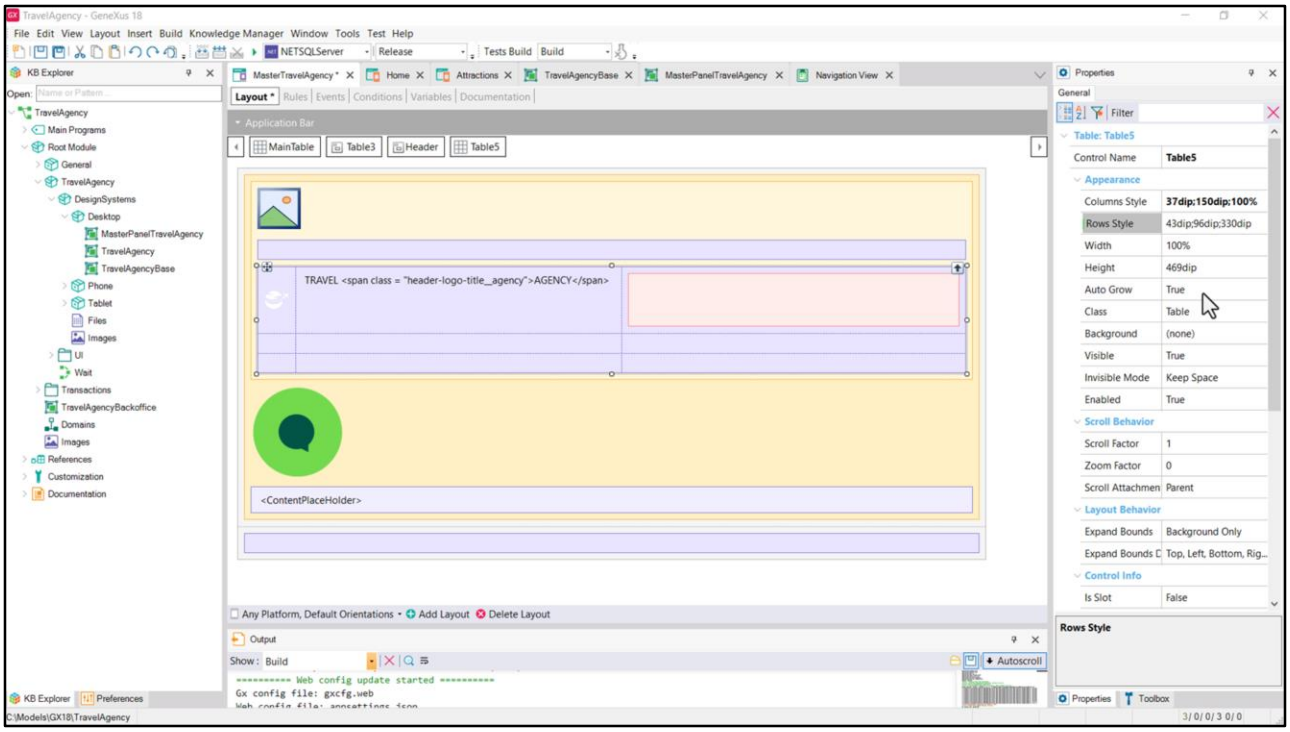
Isto valerá para a plataforma Web (tanto Angular quanto Net e Java).



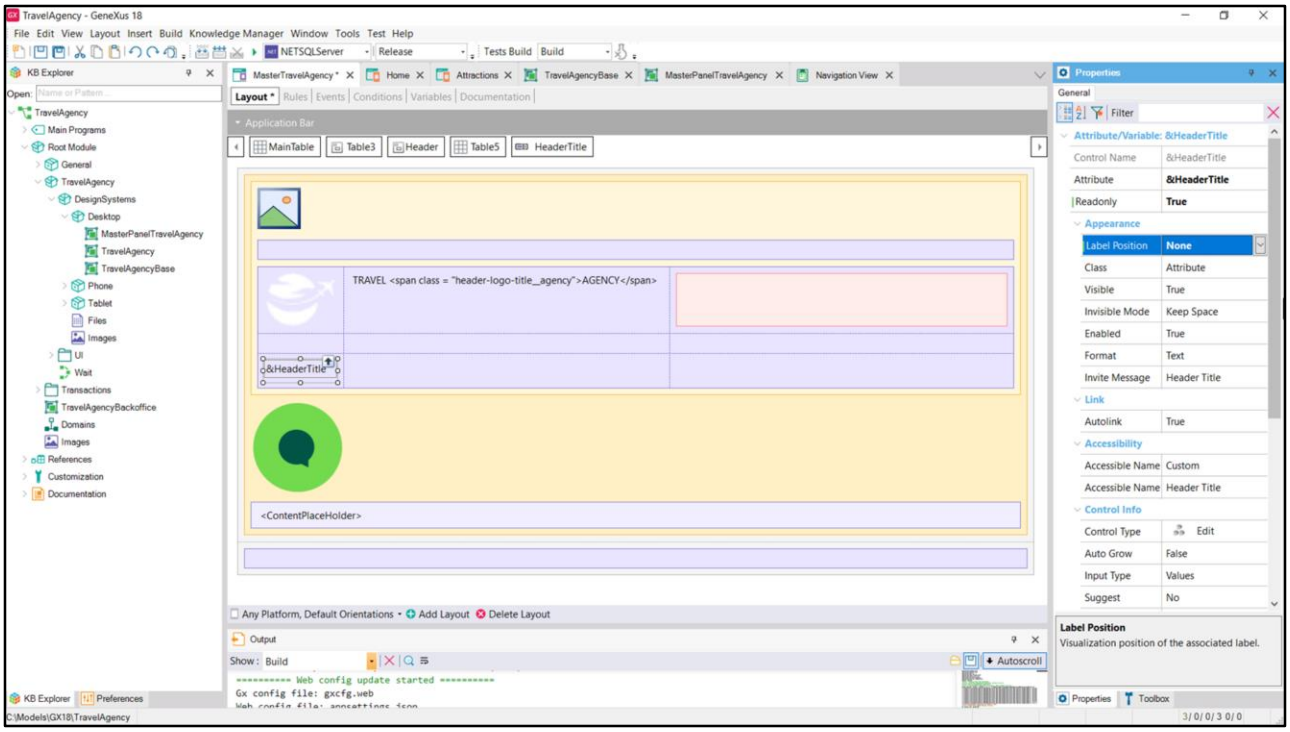

Vamos agora adicionar rapidamente o que está faltando para o Header para terminar de implementá-lo: o texto que se destaca sobre a imagem. Lembremos que seu estilo tipográfico era o H1. E vamos copiar seu conteúdo para a área de transferência.

Se alinha à esquerda junto com o ícone de travel agency, e por isso pensamos em utilizar a tabela. Vamos extrair as medidas: a altura da linha será de 330 pixels, e estará do menu a esta outra distância, a 96 pixels…

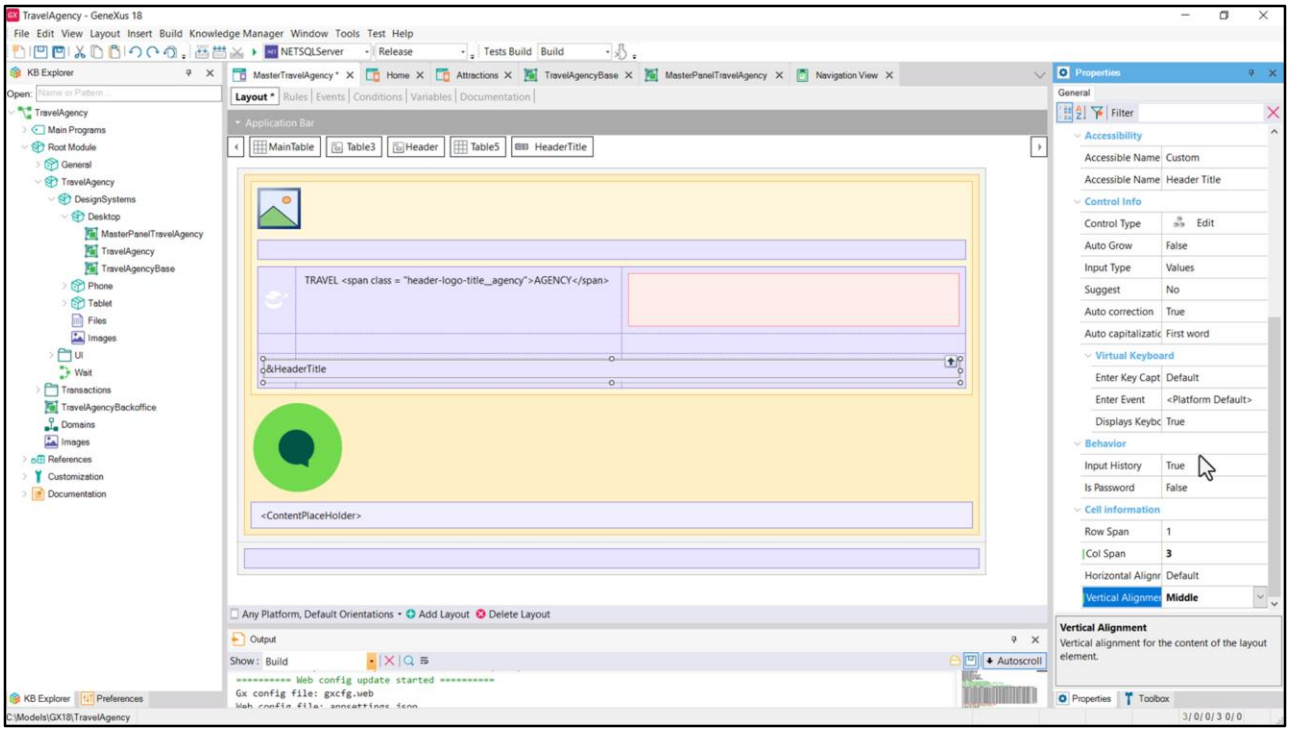
Claramente, escolheremos implementar o texto não como um textblock, mas como uma variável, já que seu conteúdo irá variar entre as telas.



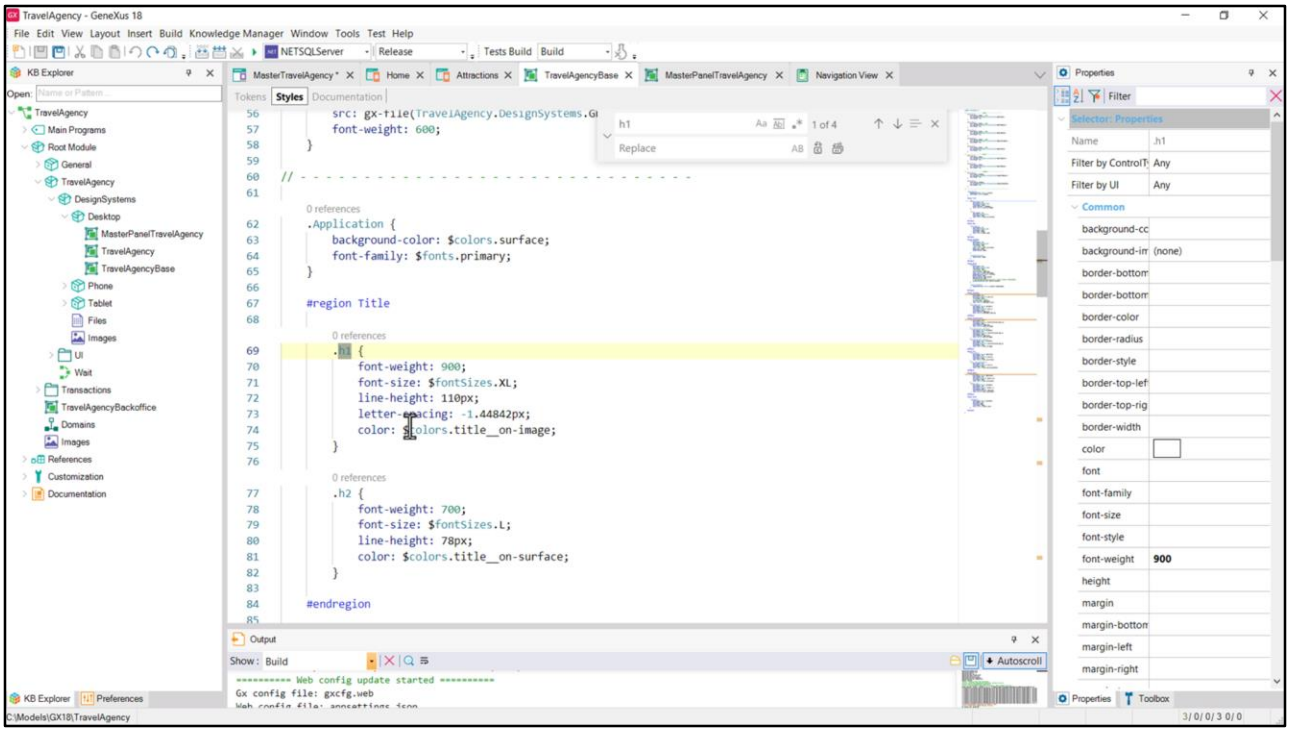

Inserimos então duas linhas na tabela. A segunda será de 96 pixels e a terceira de 330.



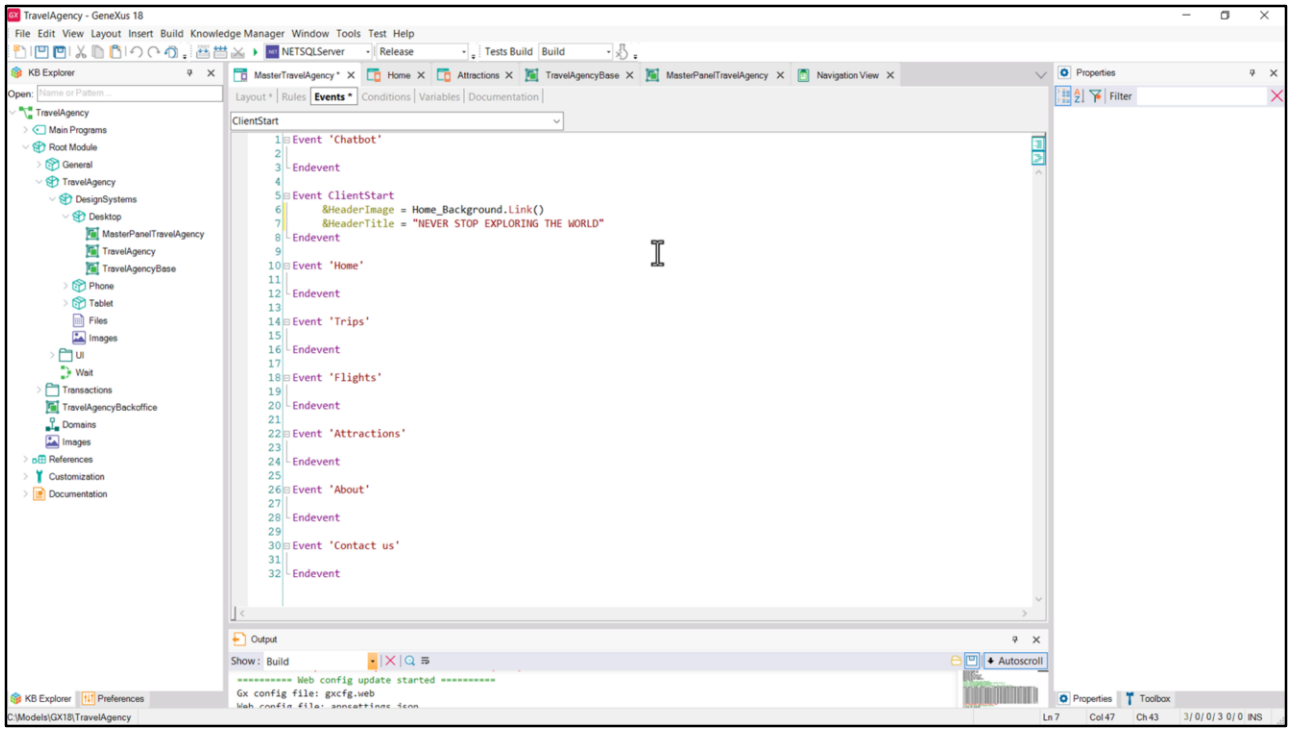

Na terceira, inserimos um controle variável… chamado HeaderTitle…. E tipo de dados varchar. Será readonly e sem mostrar o rótulo.



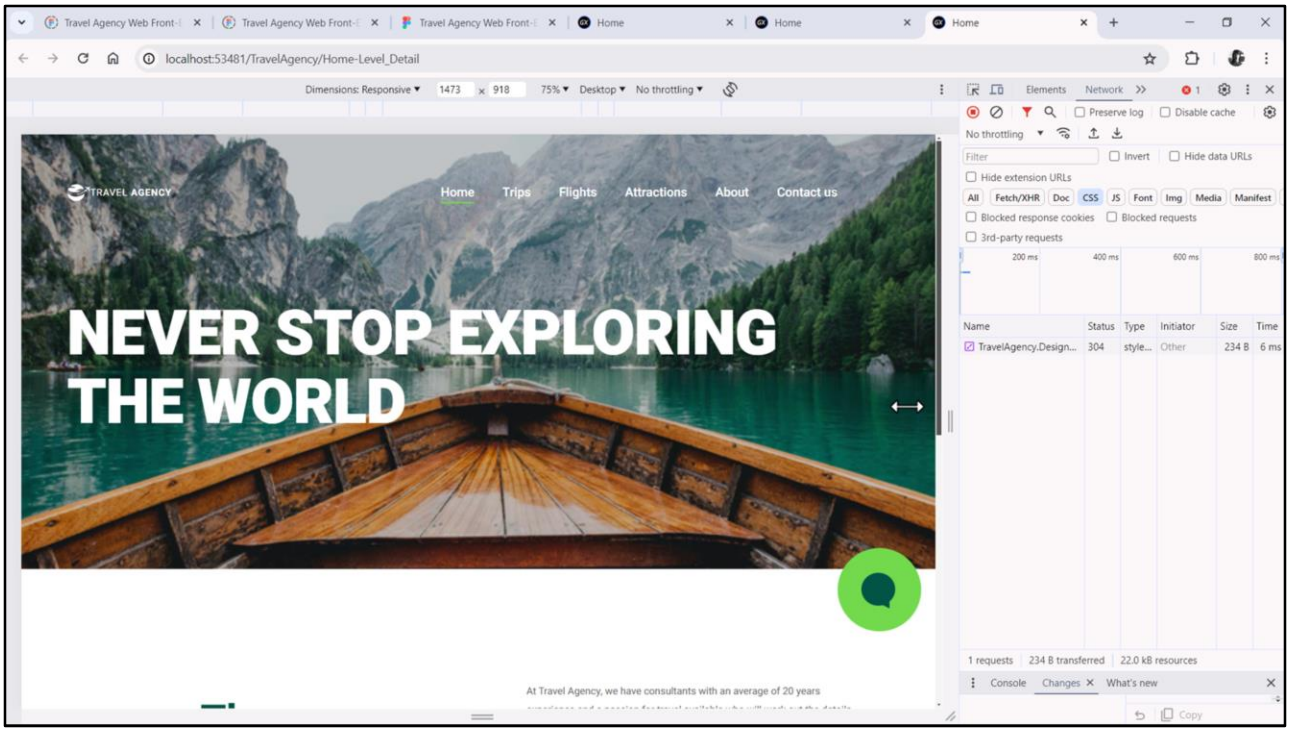

Mas também queremos que se expanda para ocupar as três colunas. E queremos que seu conteúdo seja alinhado verticalmente pelo meio.



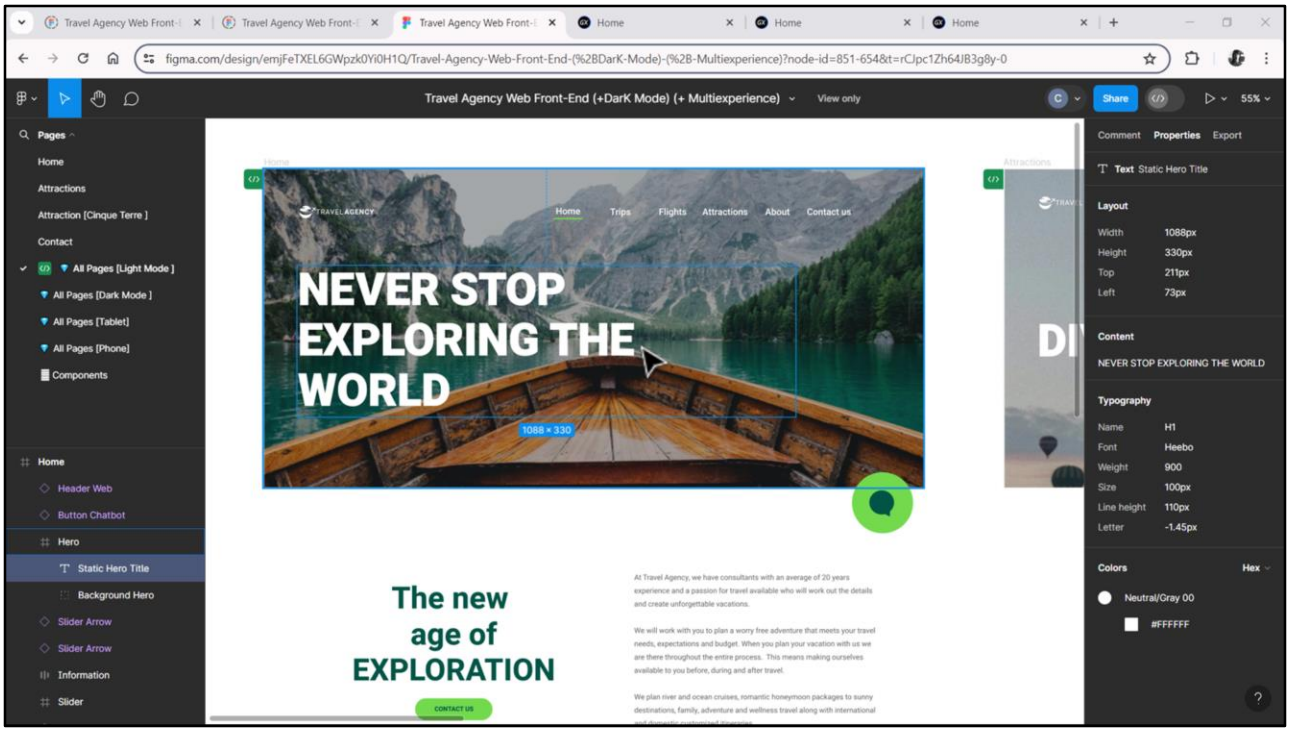

Por outro lado, sua classe será aquela que chamamos de h1 quando introduzimos no módulo de preparação todas as classes para a tipografia.



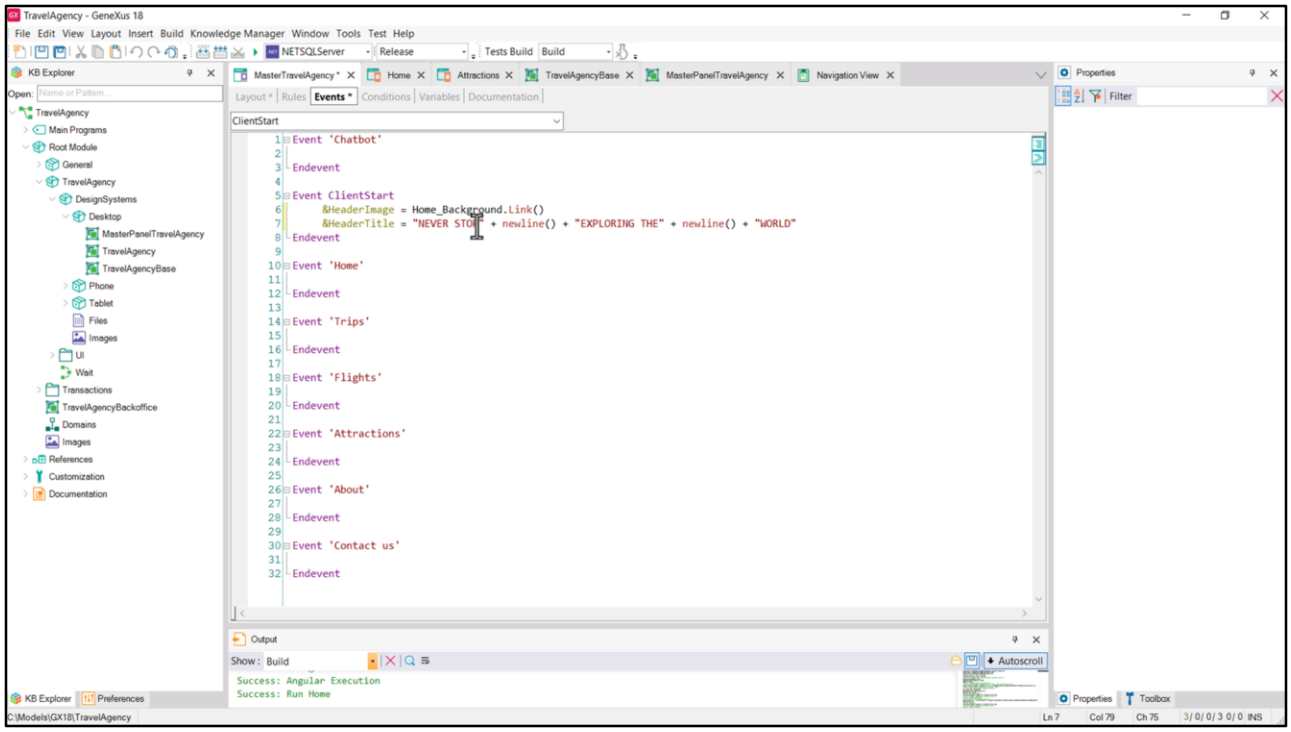

No evento ClientStart, vamos atribuir valor a ele. Por enquanto vamos deixar assim, estático.



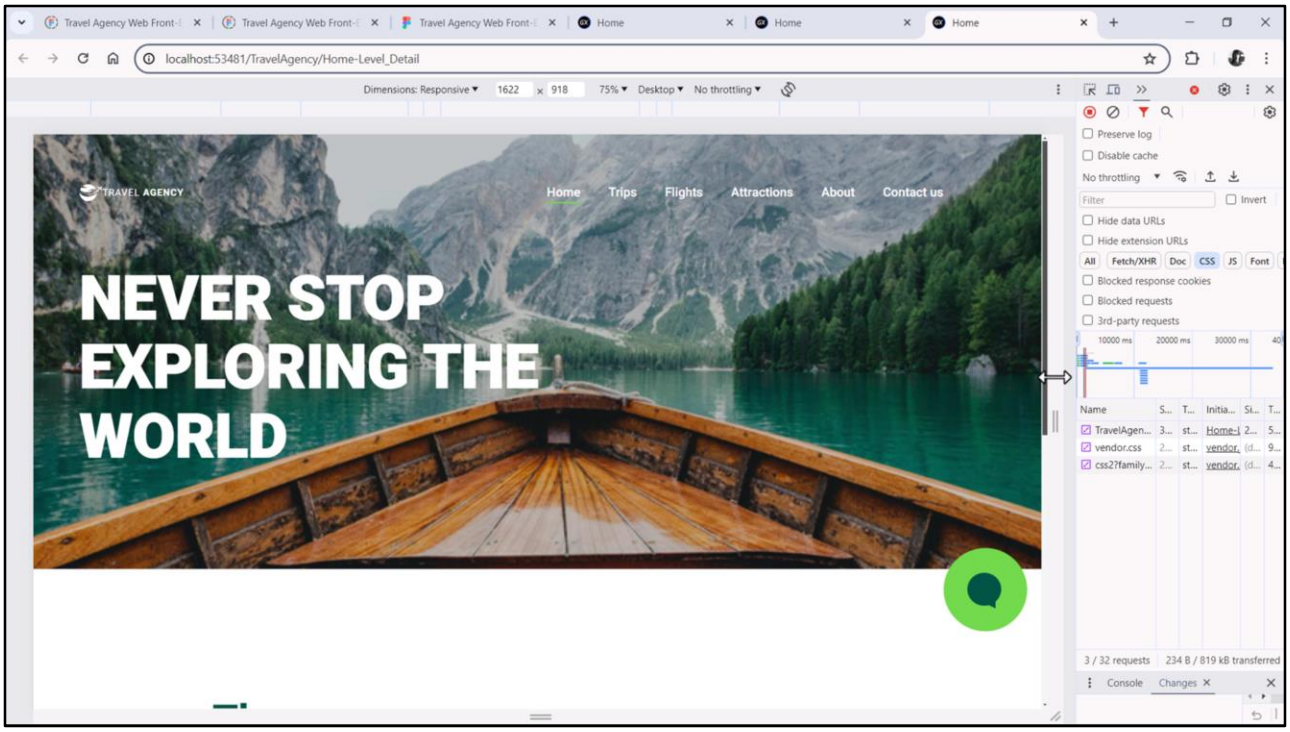

Vamos testar. Se quisermos que sempre fique em exatamente 3 linhas…




...como aqui, e exatamente estas...




… basta adicionar newlines().




Bem, agora estamos prontos para implementar as navegações, conseguindo mudar o Header de acordo com quem está sendo carregado no contentplaceholder em cada oportunidade.

Bom, continuamos no próximo vídeo.


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